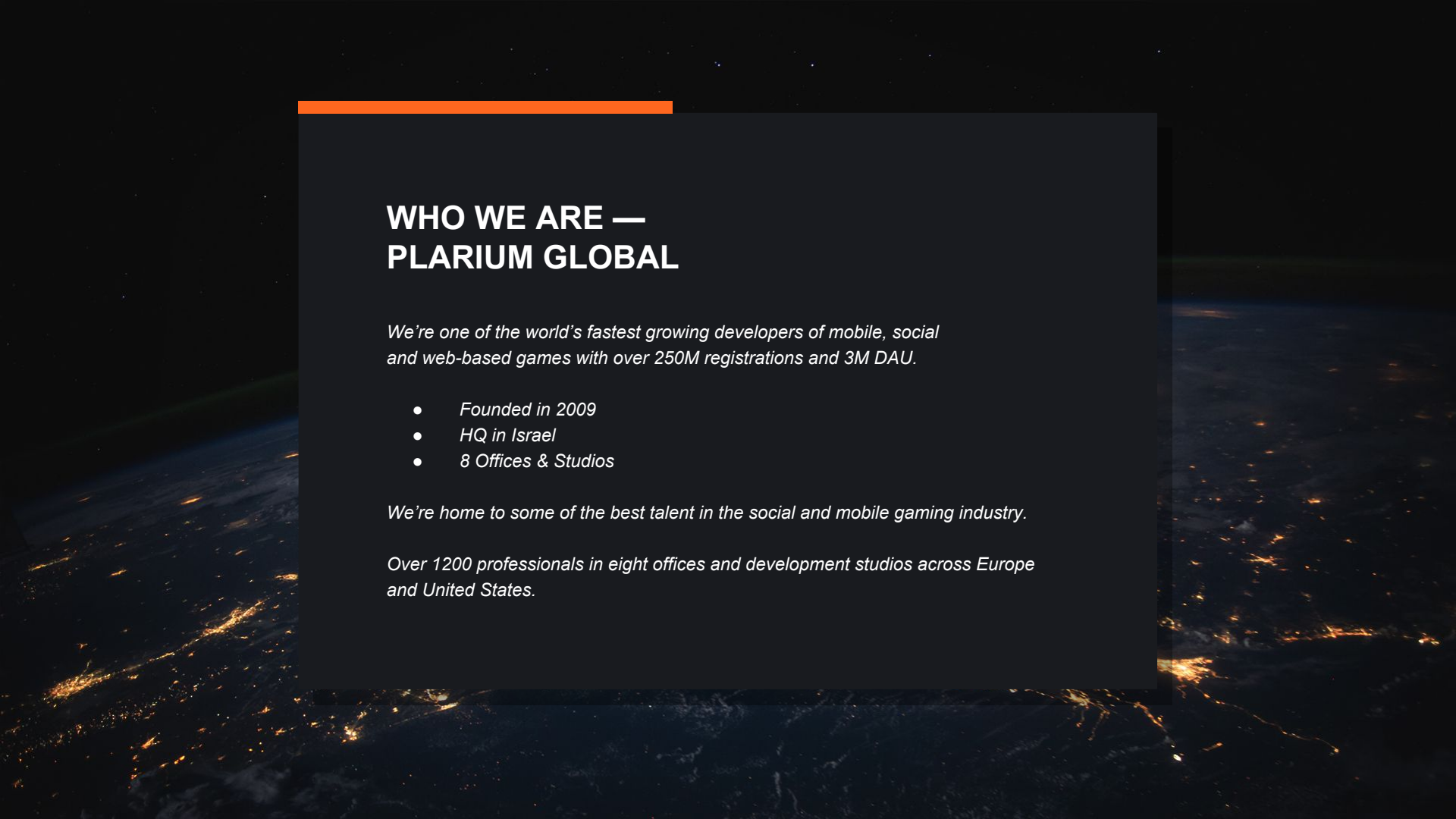

PLARIUM



WHO WE ARE — PLARIUM GLOBAL

We’re one of the world’s fastest growing developers of mobile, social and web-based games with over 250M registrations and 3M DAU.

- *Founded in 2009*
- *HQ in Israel*
- *8 Offices & Studios*

We’re home to some of the best talent in the social and mobile gaming industry.

Over 1200 professionals in eight offices and development studios across Europe and United States.

Martin Fowler

Martin Fowler

Any fool can write code that a computer can understand.

Martin Fowler

Any fool can write code that a computer can understand.

Good programmers write code that humans can understand.

Martin Fowler

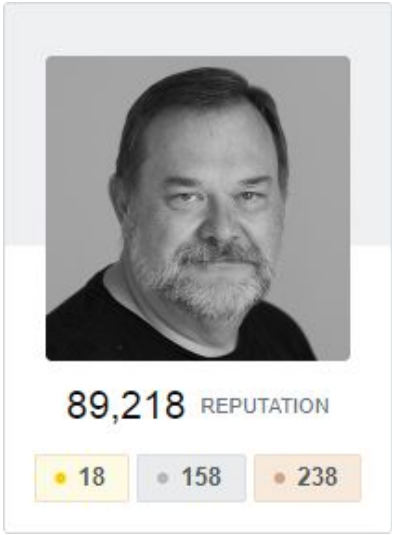
Any fool can write code that a computer can understand.

Good programmers write code that humans can understand.

89,218 REPUTATION

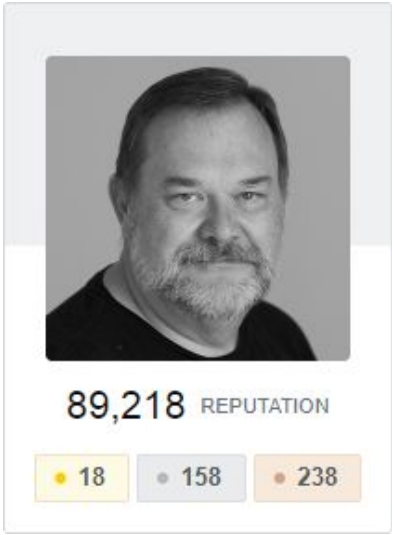
18 158 238

Charlie Martin



Charlie Martin

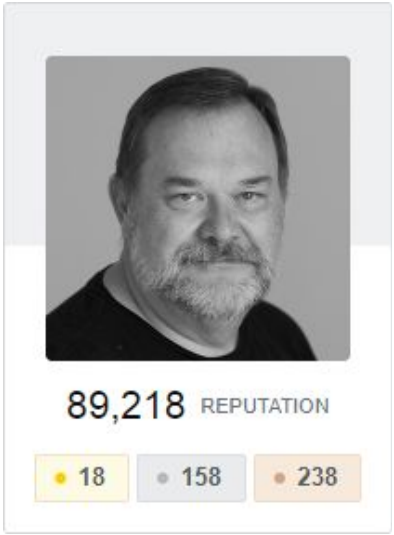
If the computer doesn't run it, it's broken.



Charlie Martin

If the computer doesn't run it, it's broken.

If people can't read it, it will be broken.



Charlie Martin

If the computer doesn't run it, it's broken.

If people can't read it, it will be broken. Soon.

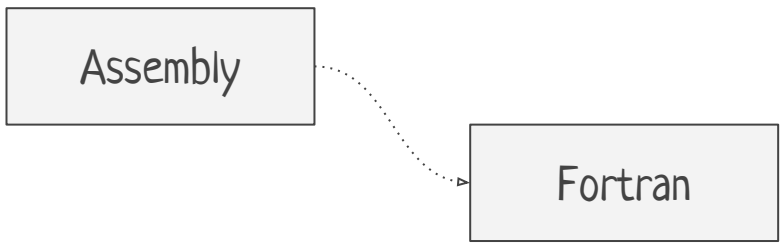
Код пишется для людей

Assembly

Assembly

```
MEMSCAN:INC     CX
        JZ      SETEND
        MOV     DS,CX
        MOV     AL,[BX]
        NOT     AL
        MOV     [BX],AL
        CMP     AL,[BX]
        NOT     AL
        MOV     [BX],AL
        JZ      MEMSCAN
SETEND:
        MOV     [MEMORY_SIZE],CX
        ENDIF

        IF      IBMVER OR IBMJAPVER
        MOV     CX,[MEMORY_SIZE]
        ENDIF
```

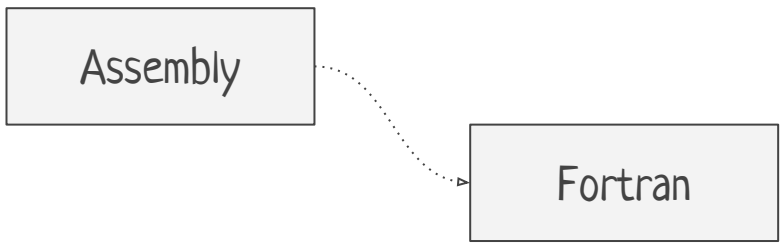
Assembly

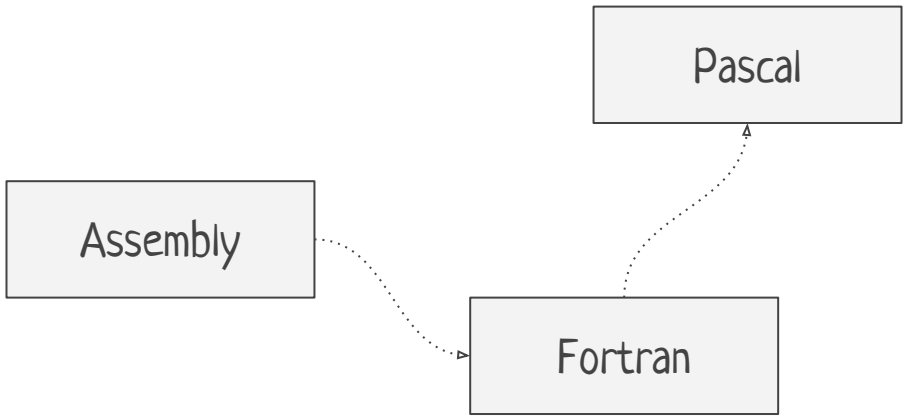


Assembly

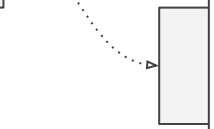
Fortran

```
c.....................................................................
c
c     now transform ss to contracted basis, then set up transformation
c     matrix to o.n. contracted basis.
c.....................................................................
         call trafsd(nsym,nbas,ndim,ss,nbc,cont,nstrt,dc)
         call trafsd(nsym,nbas,ndim,fc,nbc,cont,nstrt,dc)
c...
         nstep1 = 1
         nstep2 = 1
         do i = 1 , nsym
crz      to surpress problems when there is one type of shell missing...
            if (nbc(i).ne.0) then
               call shalfd(fc(nstep1),transf(nstep2),nbc(i))
               call starcd(cc(nstep2),ss(nstep1),nbc(i),ncsh(i),
     +                     nosh(i))
               nstep1 = nstep1 + nbc(i)*(nbc(i)+1)/2
               nstep2 = nstep2 + nbc(i)**2
            end if
         enddo
```

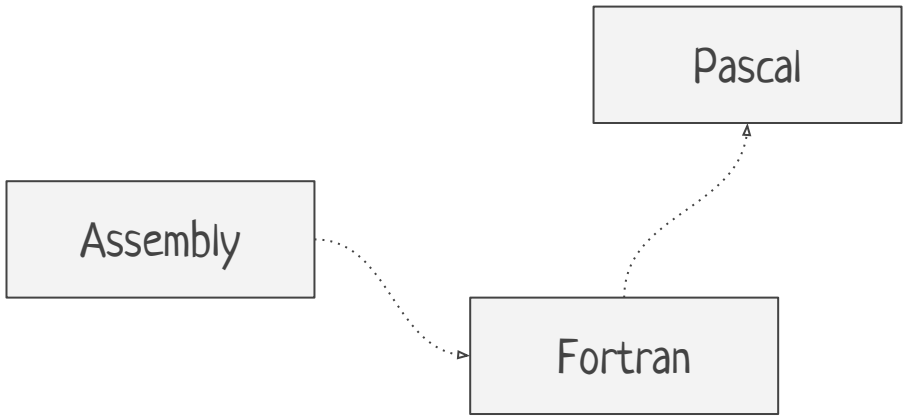


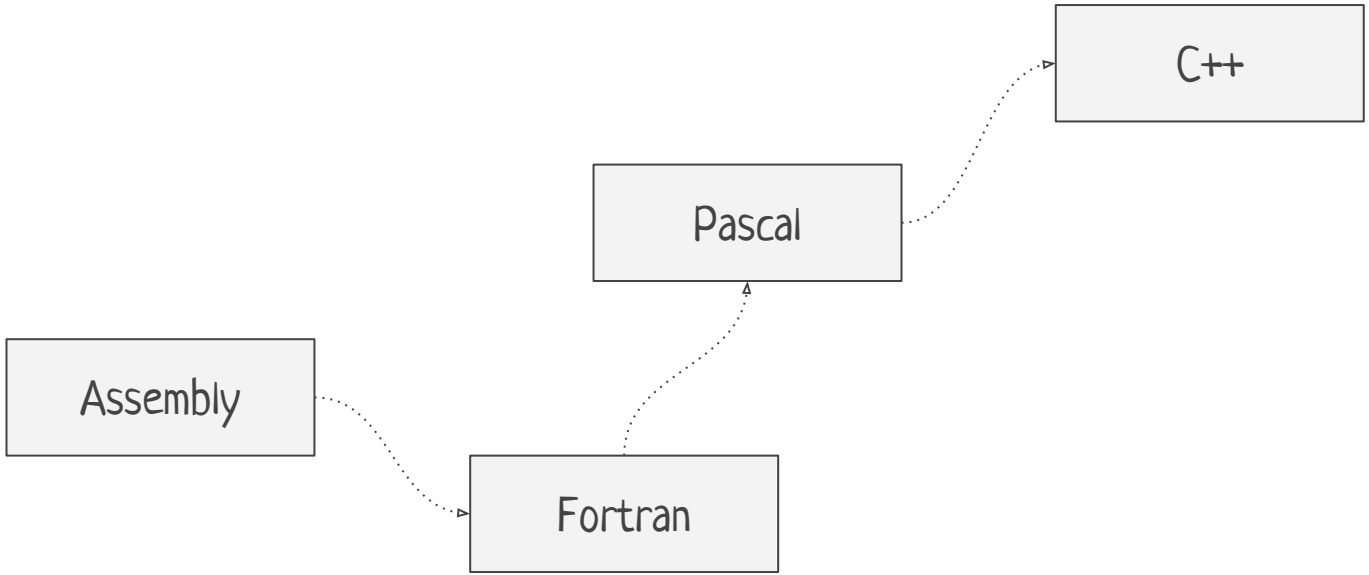


Assembly



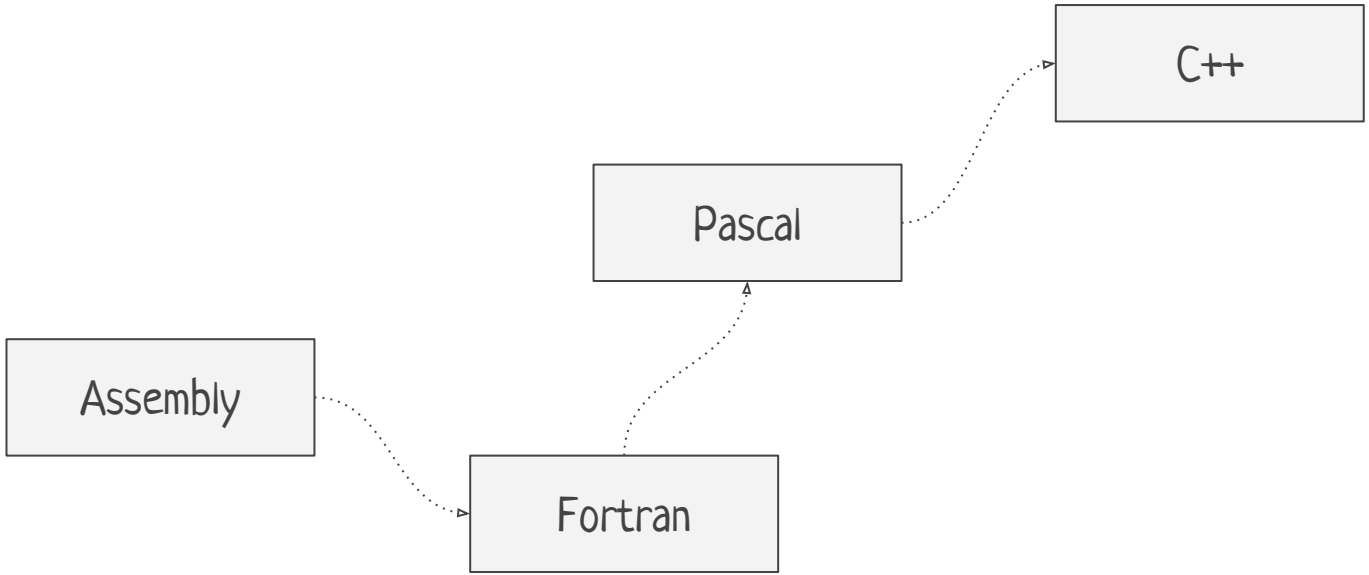
```
begin
  filesize:=filedata.Size; // getfilesize(hprocess,@ignore);
  lpbuffer.AllocationBase:=pointer(filebaseaddress);
  lpbuffer.AllocationProtect:=PAGE_EXECUTE_READWRITE;
  lpbuffer.RegionSize:=filesize-ptruint(lpBuffer.BaseAddress-lpbuffer.AllocationBase);
  if (lpbuffer.RegionSize mod 4096)>0 then
    lpbuffer.RegionSize:=lpbuffer.RegionSize+($1000-lpbuffer.RegionSize mod $1000);

  lpbuffer.State:=mem_commit;
  lpbuffer.Protect:=PAGE_EXECUTE_READWRITE;
  lpbuffer._Type:=MEM_PRIVATE;

  if (ptrUint(lpAddress)>filesize+filebaseaddress) //bigger than the file
  then
  begin
    zeromemory(@lpbuffer,dwlength);
    result:=0
  end
  else
    result:=dwlength;
end;
```

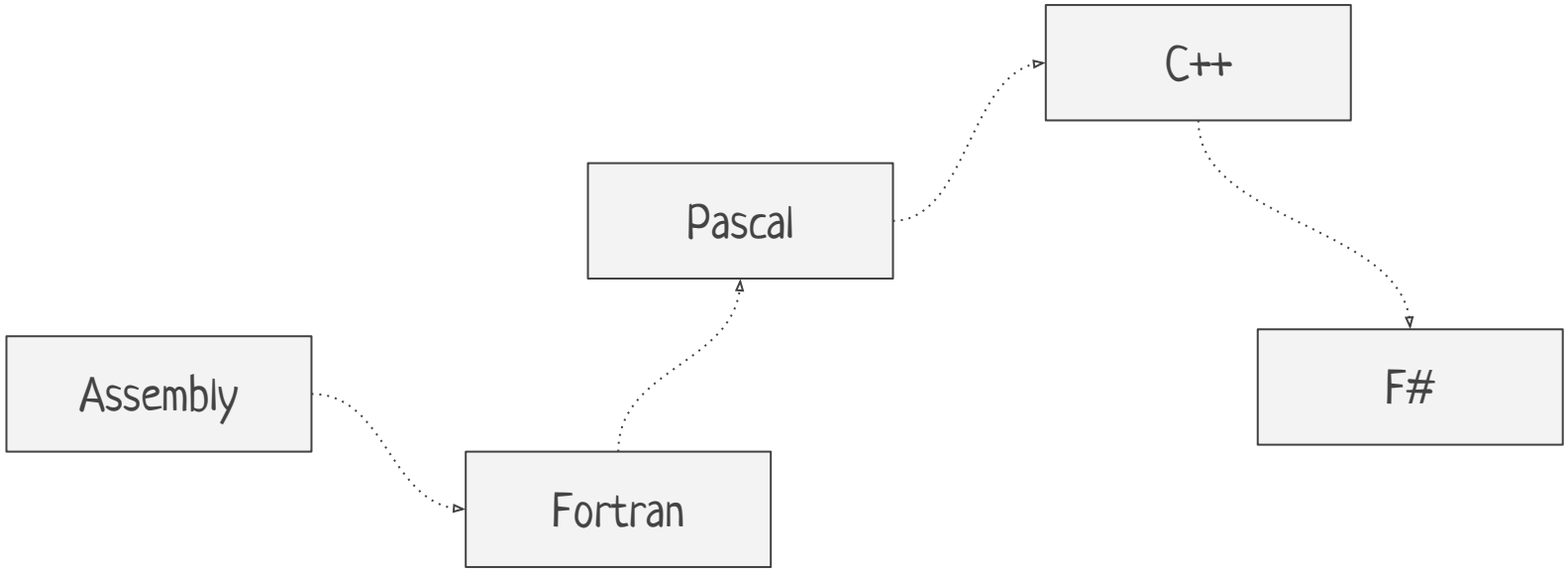


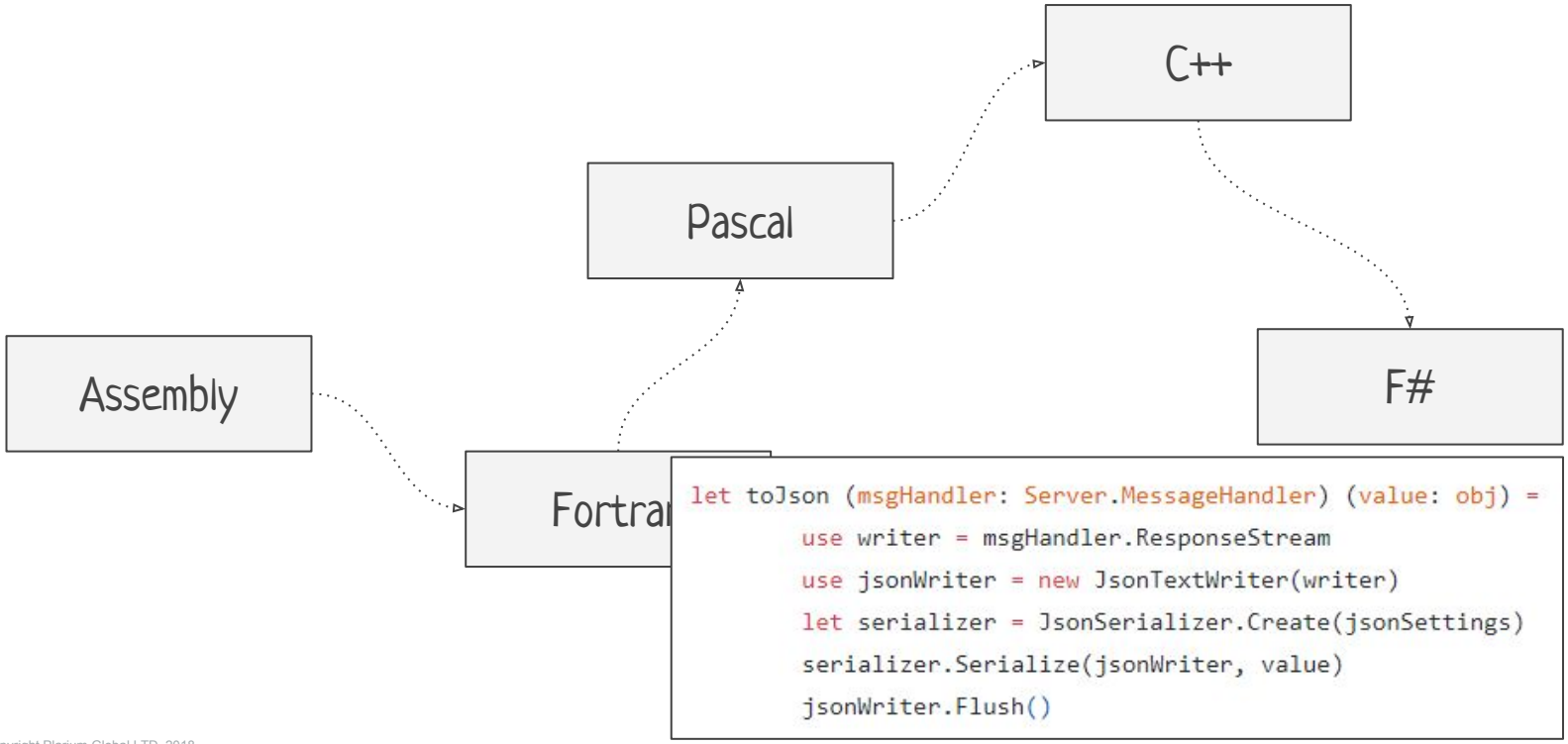


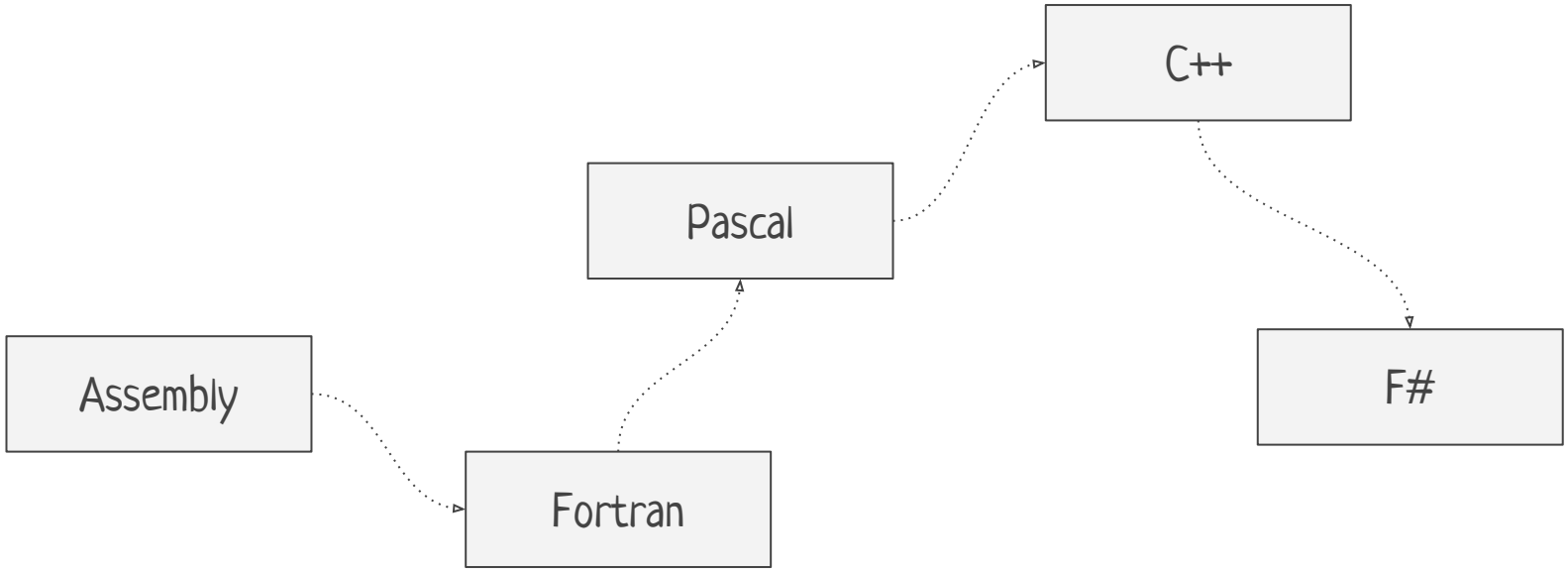
Assembly

```
TagNameParserOutput parseTagName(DOMString tagname)
{
    TagNameParserOutput output;
    char* splitterRef = (char*)tagname.findChar(AFTER_PREFIX_CHAR);
    if(splitterRef == 0)
    {
        output.hasPrefix = false;
        output.name = tagname;
    } else
    {
        unsigned long splitterIndex = splitterRef - tagname.raw();
        output.hasPrefix = true;
        output.prefix = tagname.substring(0, splitterIndex - 1);
        output.name = tagname.substring(splitterIndex + 1, tagname.size() - splitterIndex - 1).toLower_();
    }
    return output;
}
```









```
MEMSCAN:INC     CX
        JZ      SETEND
        MOV     DS,CX
        MOV     AL,[BX]
        NOT     AL
        MOV     [BX],AL
        CMP     AL,[BX]
        NOT     AL
        MOV     [BX],AL
        JZ      MEMSCAN
SETEND:
        MOV     [MEMORY_SIZE],CX
        ENDIF

        IF      IBMVER OR IBMJAPVER
        MOV     CX,[MEMORY_SIZE]
        ENDIF
```

```
MEMSCAN:INC     CX
        JZ      SETEND
        MOV     DS,CX
        MOV     AL,[BX]
        NOT     AL
        MOV     [BX],AL
        CMP     AL,[BX]
        NOT     AL
        MOV     [BX],AL
        JZ      MEMSCAN
SETEND:
        MOV     [MEMORY_SIZE],CX
        ENDIF

        IF      IBMVER OR IBMJAPVER
        MOV     CX,[MEMORY_SIZE]
        ENDIF
```

```
let toJson (msgHandler: Server.MessageHandler) (value: obj) =
        use writer = msgHandler.ResponseStream
        use jsonWriter = new JsonTextWriter(writer)
        let serializer = JsonSerializer.Create(jsonSettings)
        serializer.Serialize(jsonWriter, value)
        jsonWriter.Flush()
```

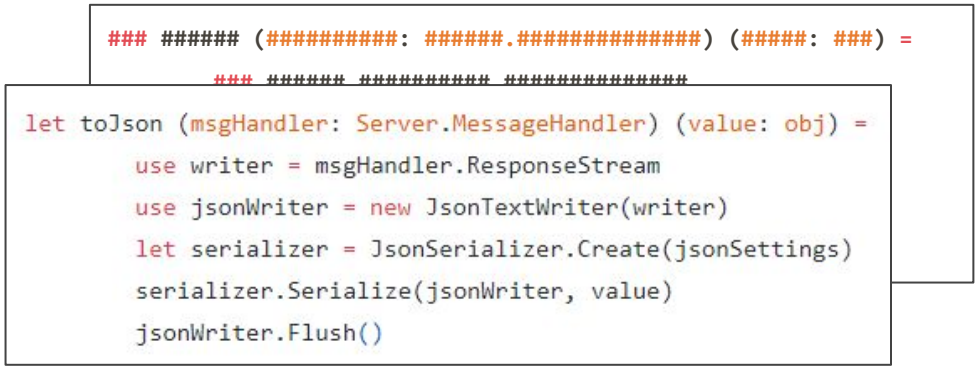
```
MEMSCAN:INC     CX
        JZ      SETEND
        MOV     DS,CX
        MOV     AL,[BX]
        NOT     AL
        MOV     [BX],AL
        CMP     AL,[BX]
        NOT     AL
        MOV     [BX],AL
        JZ      MEMSCAN
SETEND:
        MOV     [MEMORY_SIZE],CX
        ENDIF

        IF      IBMVER OR IBMJAPVER
        MOV     CX,[MEMORY_SIZE]
        ENDIF
```

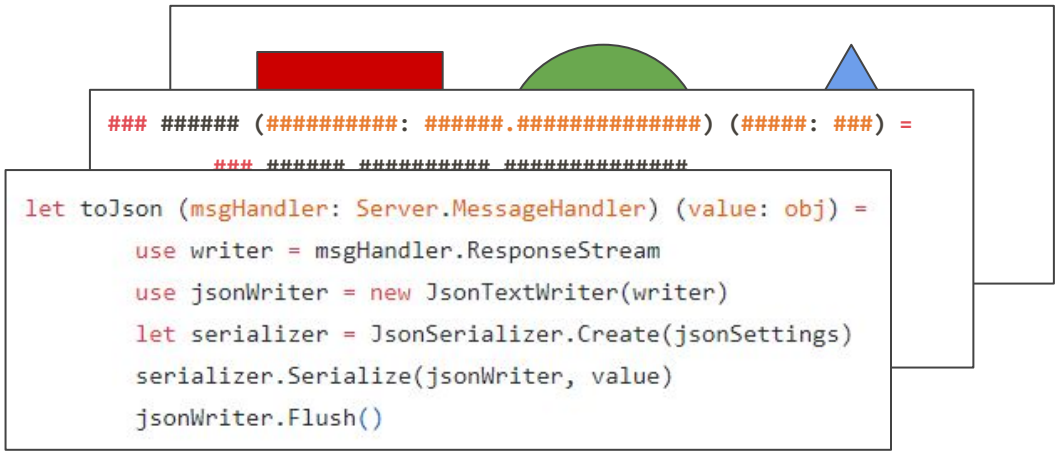
```
let toJson (msgHandler: Server.MessageHandler) (value: obj) =
        use writer = msgHandler.ResponseStream
        use jsonWriter = new JsonTextWriter(writer)
        let serializer = JsonSerializer.Create(jsonSettings)
        serializer.Serialize(jsonWriter, value)
        jsonWriter.Flush()
```

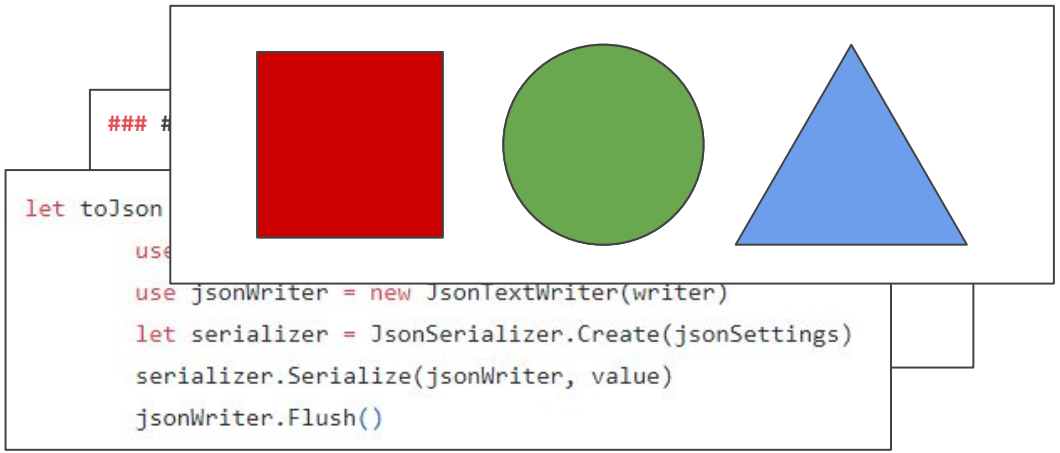
Assembly → Fortran → Pascal → C++ → F#

```
let toJson (msgHandler: Server.MessageHandler) (value: obj) =
        use writer = msgHandler.ResponseStream
        use jsonWriter = new JsonTextWriter(writer)
        let serializer = JsonSerializer.Create(jsonSettings)
        serializer.Serialize(jsonWriter, value)
        jsonWriter.Flush()
```

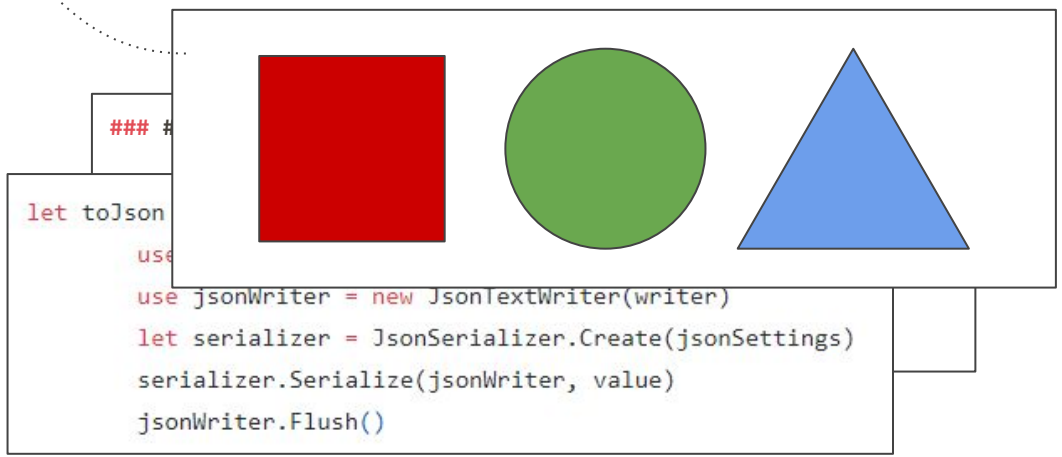
```
let toJson (msgHandler: Server.MessageHandler) (value: obj) =
        use writer = msgHandler.ResponseStream
        use jsonWriter = new JsonTextWriter(writer)
        let serializer = JsonSerializer.Create(jsonSettings)
        serializer.Serialize(jsonWriter, value)
        jsonWriter.Flush()
```



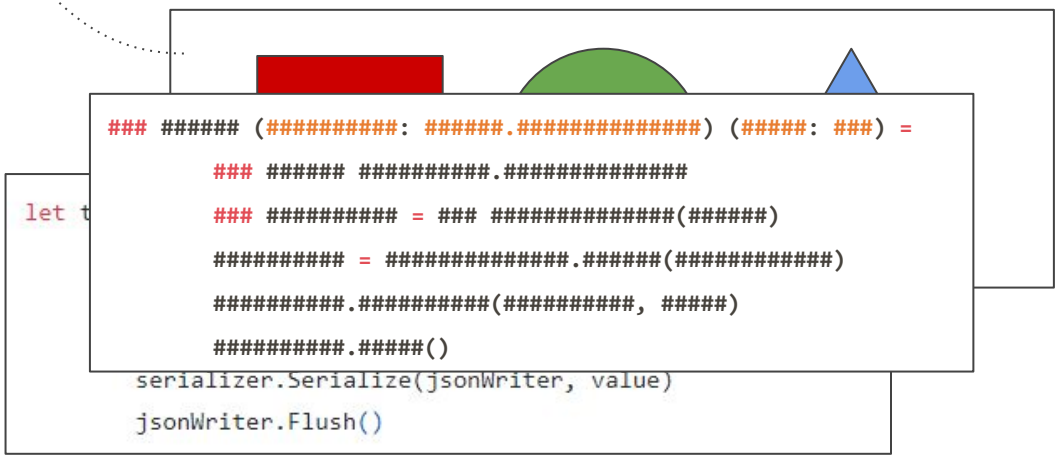




Парадигмы

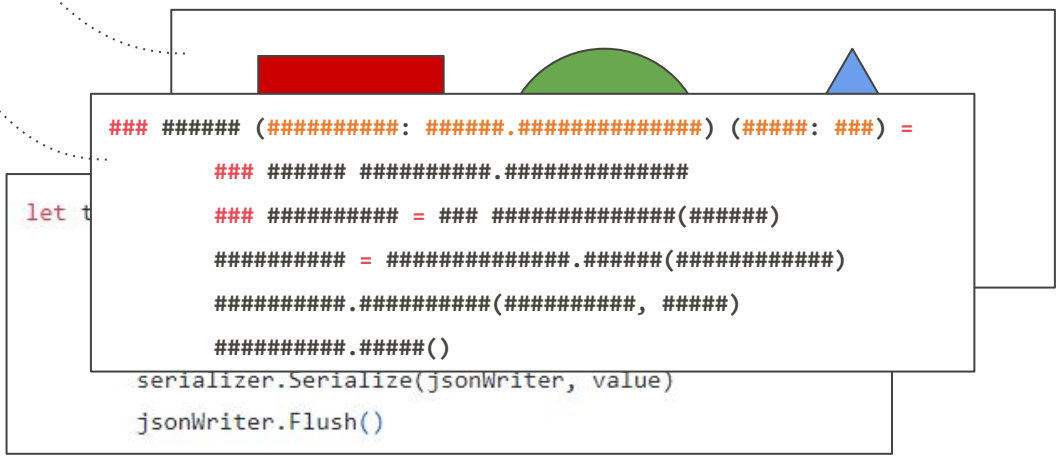


Парадигмы

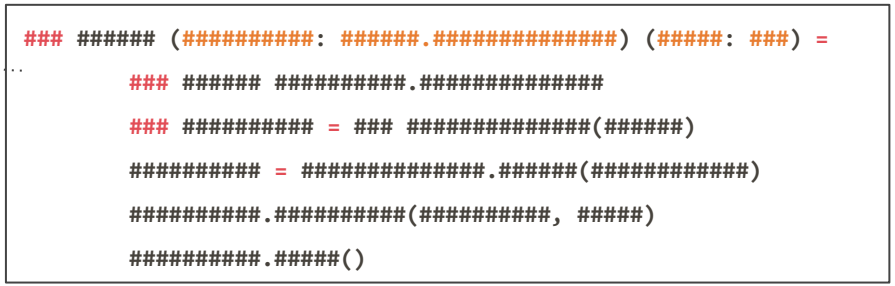


Парадигмы

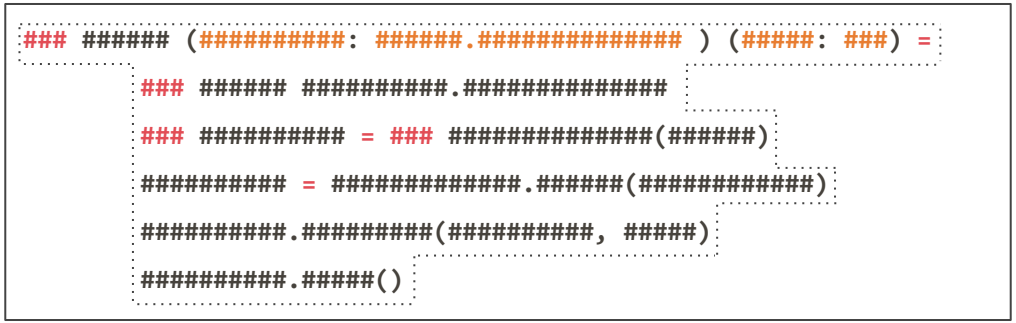
Структура



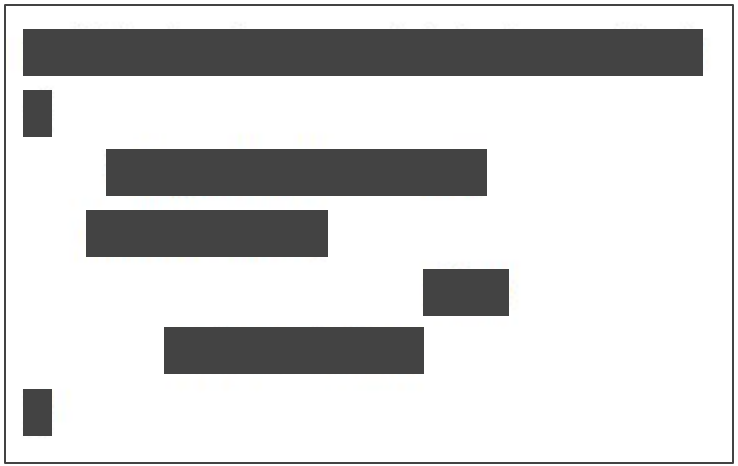
Структура

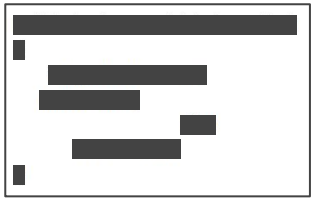


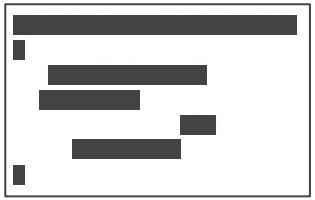
```
### ###### (##########: ######.############### ) (#####: ###) =
        ### ###### ##########.###############
        ### ########## = ### ###############(######)
        ########## = ##############.######(#############)
        ##########.#########(##########, #####)
        ##########.#####()
```

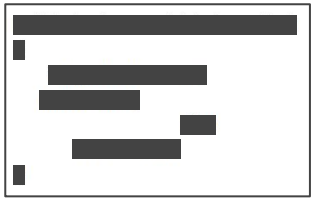


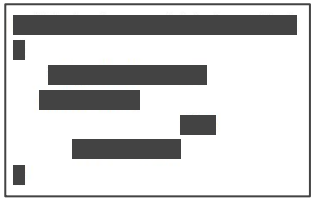
Форма

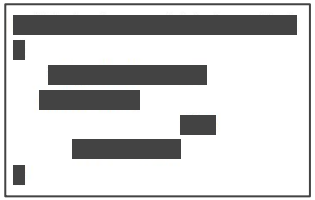


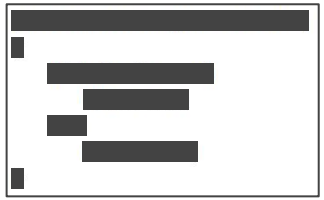


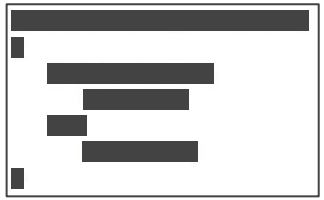






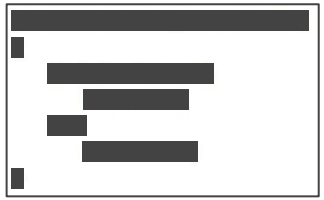




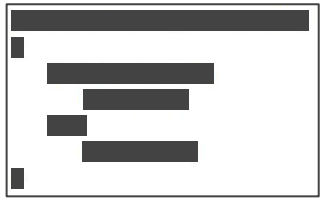


```
public bool IsEven(this int value)
{
    if (value % 2 == 0)
        return true;
    else
        return false;
}
```

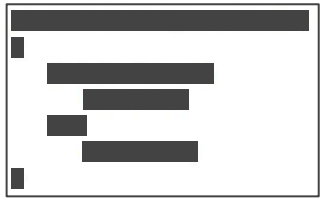




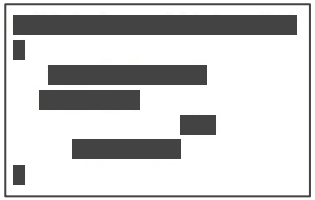
```
public bool IsEven(this int value)
{
    if (value % 2 == 0)
        return true;
    else
        return false;
}
```

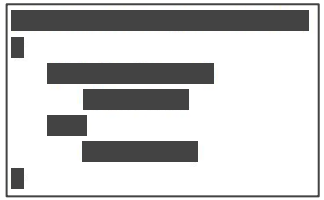


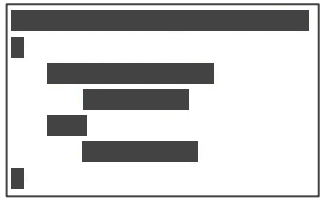
```
public bool IsEven(this int value)
{
    if (value % 2 == 0)
        return true;
    else
        return false;
}
```



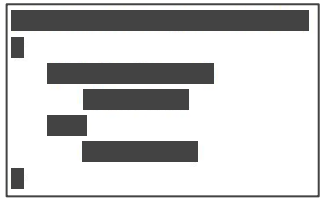
```
public bool IsEven(this int value)
{
    if (value % 2 == 0)
        return true;
    else
        return false;
}
```



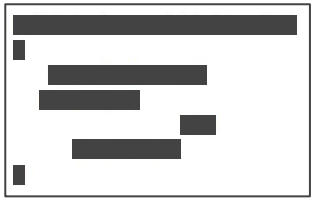


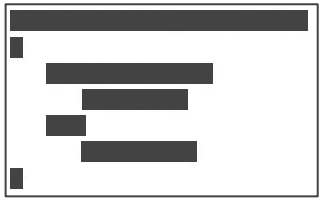


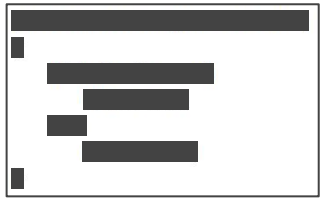
```
public bool IsEven(this int value)
{
    if (value % 2 == 0)
        return true;
    else
        return false;
}
```



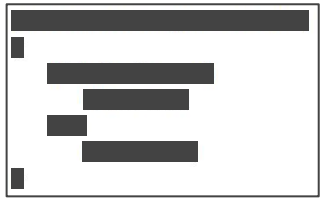
```
public bool IsEven(this int value)
{
  if (value % 2 == 0)
    return true;
  else
    return false;
}
```



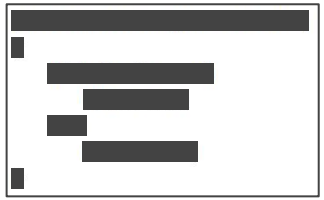




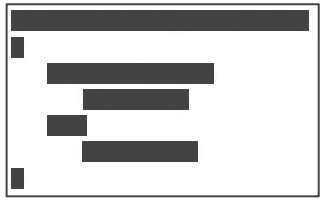
```
public bool IsEven(this int value)
{
    if (value % 2 == 0)
        return true;
    else
        return false;
}
```



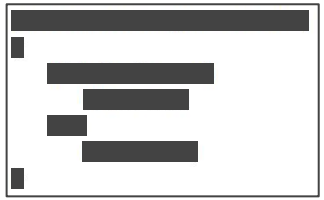
```
public bool IsEven(this int value)
{
    if (value % 2 == 0)
        return true;

    return false;
}
```



```
public bool IsEven(this int value)
{
    [redacted]
        [redacted]

    return false;
}
```





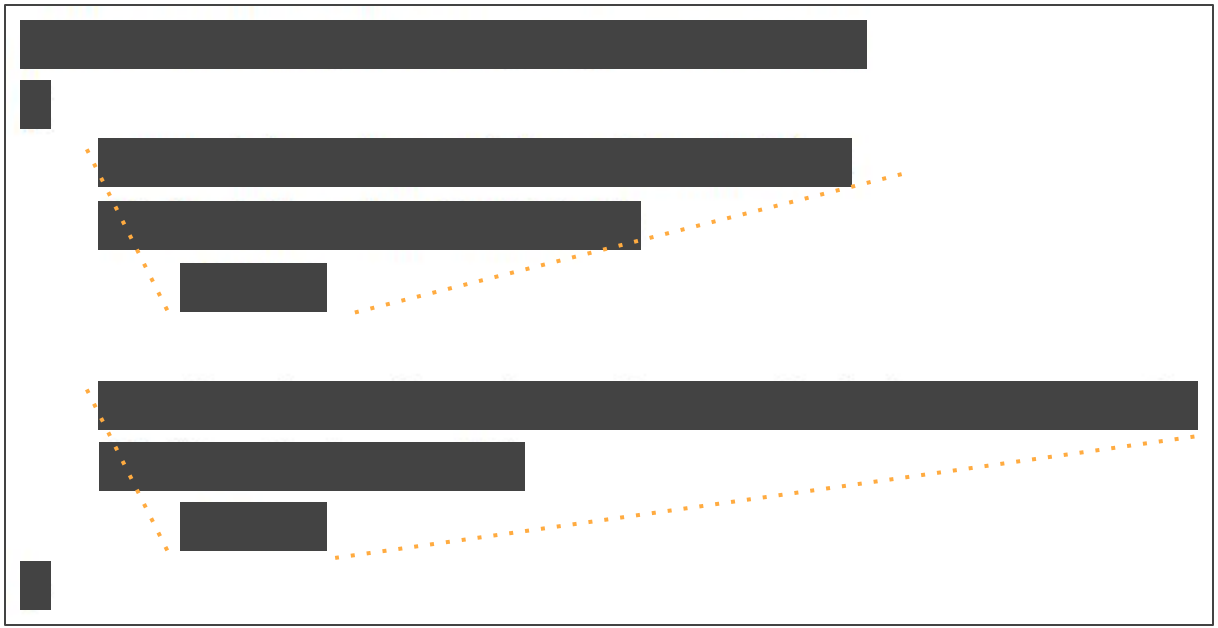


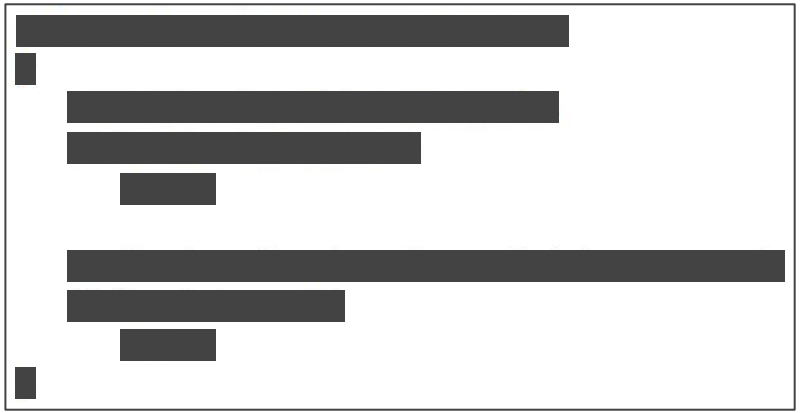
```
public bool IsEvenOrDivisibleBy3(this int value)
{
    if (value % 2 == 0)
        return true;
    if (value % 3 == 0)
        return true;

    return false;
}
```

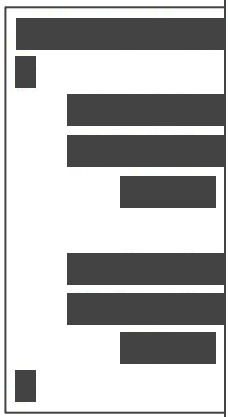
```
public void Put(this IBag bag, IItem item)
{
    var weight = bag.Weight + item.Weight;
    if (weight > bag.MaxWeight)
        return;

    var freeSlot = bag.Slots.FirstOrDefault(x => x.IsFree);
    if (freeSlot == null)
        return;

    // ...
}
```

```
public void Put(this IBag bag, IItem item)
{
    var weight = bag.Weight + item.Weight;
    if (weight > bag.MaxWeight)
        return;

    var freeSlot = bag.Slots.FirstOrDefault(x => x.IsFree);
    if (freeSlot == null)
        return;

    // ...
}
```

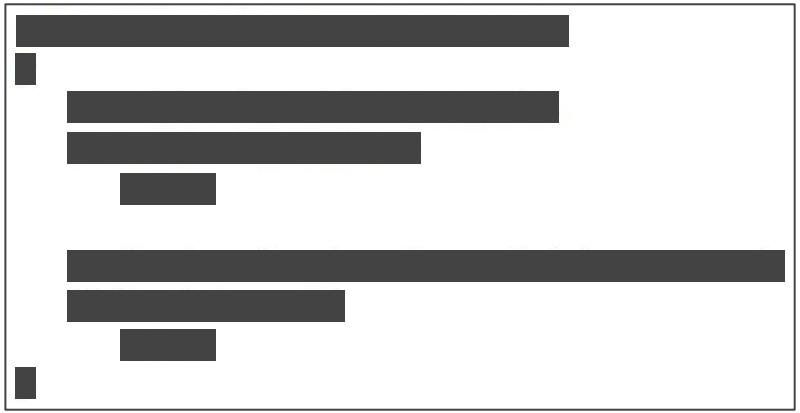
```
public void Put(this IBag bag, IItem item)
{
    var weight = bag.Weight + item.Weight;
    if (weight > bag.MaxWeight)
        return;

    var freeSlot = bag.Slots.FirstOrDefault(x => x.IsFree);
    if (freeSlot == null)
        return;
}
```

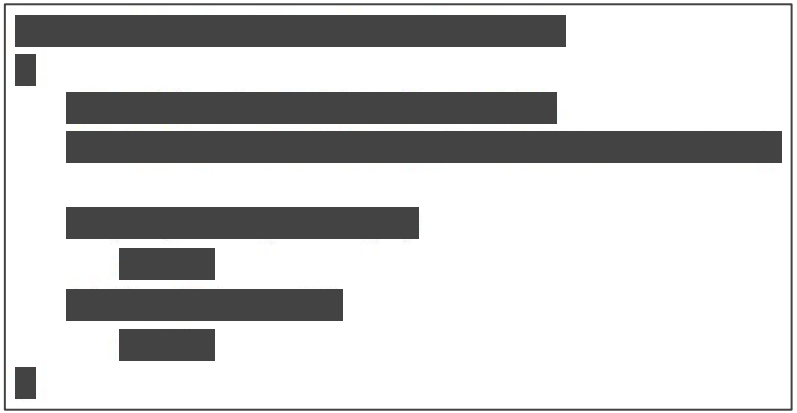






```
public void Put(this IBag bag, IItem item)
{
    var weight = bag.Weight + item.Weight;
    var freeSlot = bag.Slots.FirstOrDefault(x => x.IsFree);

    if (weight > bag.MaxWeight)
        return;
    if (freeSlot == null)
        return;
}
```

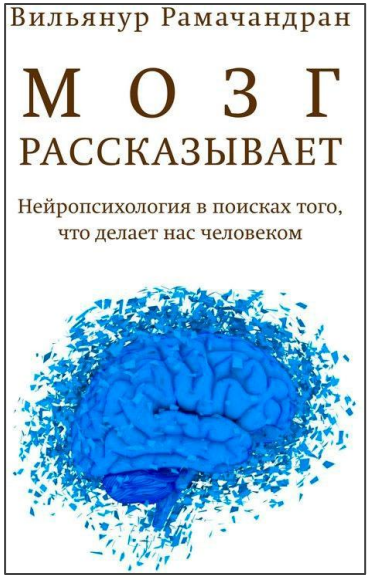


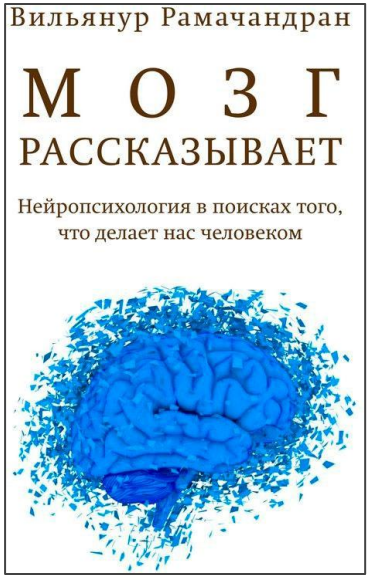


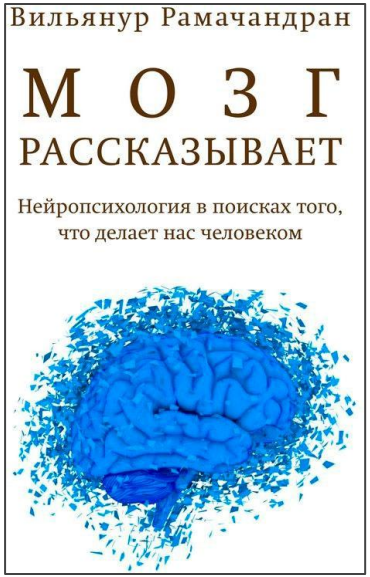
[redacted]

[redacted]

Как это работает?



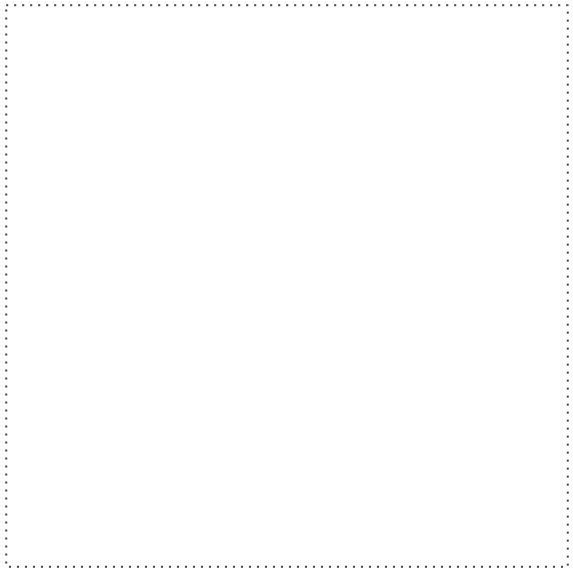


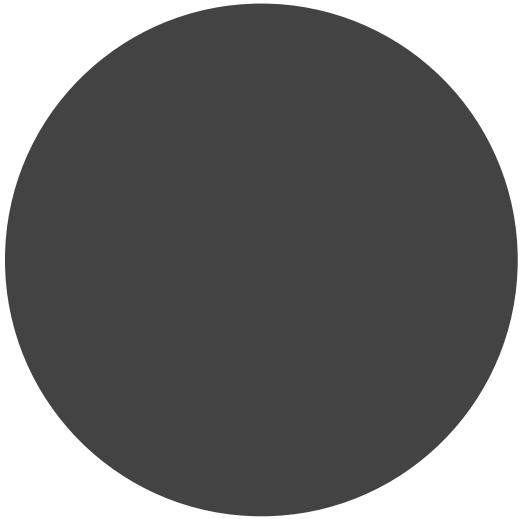


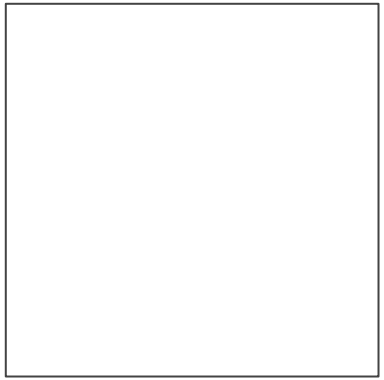
Как это работает?

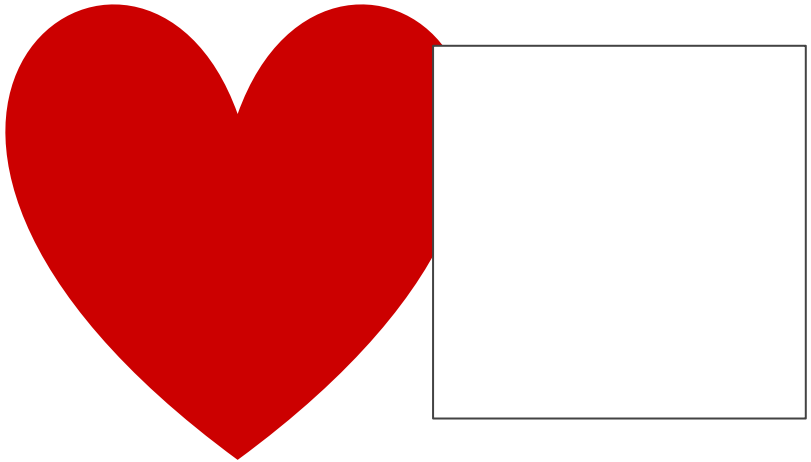
Как это работает?

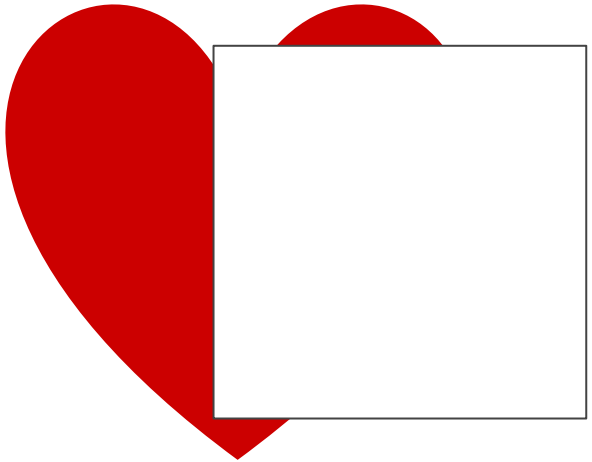
- Различение.

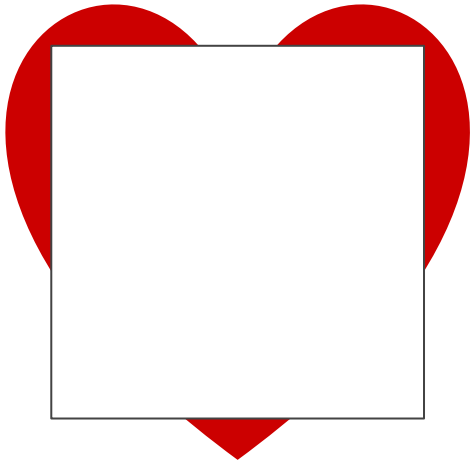












Как это работает?

- Различение.

Как это работает?

- Различение.
- Память.

когдавморскомпутитоскагрызетматросовонидосужийчас
желаяскоротатьбеспечныхловятптицогромныхальбатрос
овкоторыесудатаклюбятпровожатьиводкогдацарялюбим
оголазуринапалубекладутонснежныхдвакрылаумевшихта
клегкопаритьнавстречубуризастенчивовлачиткакдваболь
шихвеслабыстрейшийизгонцовкакгрузноонступаеткрасав
оздушныхстранкаксталонвдругсмешондразнятотвклювем
утабачныйдымпускаеттотвеселиттолпухромаякакионпоэ
твотобразтвойлытакжебезусильялетаешьвоблакахсредь
молнийигромовноисполинскиетебемешаюткрыльявнизух
одитьвтолпесредьшиканьяглупцов

йирьжвжухсзхплдеду.
ываопрджухсцъёджввапдажмфвафывафывасжелаяскорке
йцуотатьбеспеивфывчныхловтлджфолджятптыумевшихт
аклегкухэиэвэййуъввчсяссэиопааитьнавстрваываййечубу
ризастфываыфвпыкцукйцвклтфывлджлдцуолджколывфол
даполдчсолджфясмювемутаборывфящачвсмфыныйдфвм
ымпуьлодскаетвфымтотвеселиттовыфмлпухрафомаякак
фывполджйцолджионпоэтвотобразввпфйлдчсдлилдзхэъх
уйлзхвдфвжизидщлтвойлытакжебцуккпмитфвйцмисяьяв
нпщзизухчсмодполдиолджйцукитьвтолпесредмифиьшик
вйапййуцуыаньяглвааппцолвсцццпцовцукйцупзховпрсьтй
джсмппжитд

когдавморскомпутитоскагрызетматросовонидосужийчас
желаяскоротатьбеспечныхловятптицогромныхальбатрос
овкоторыесудатаклюбятпровожатьиводкогдацарялюбим
оголазуринапалубекладутонснежныхдвакрылаумевшихта
клегкопаритьнавстречубуризастенчивовлачиткакдваболь
шихвеслабыстрейшийизгонцовкакгрузноонступаеткрасав
оздушныхстранкаксталонвдругсмешондразнятотвклювем
утабачныйдымпускаеттотвеселиттолпухромаякакионпоэ
твотобразтвойлытакжебезусильялетаешьвоблакахсредь
молнийигромовноисполинскиетебемешаюткрыльявнизух
одитьвтолпесредьшиканьяглупцов

Когдавморскомпутитоскагрызетматросов Онидосужийча
сжелаяскоротатьБеспечныхловятптицогромныхальбатрос
овКоторыесудатаклюбятпровожатьИвоткогдацаря
любимоголазуриНапалубекладутонснежныхдвакрылаУме
вшихтаклегкопаритьнавстречубуриЗастенчивовлачиткак
двабольшихвеслаБыстрейшийизгонцовкакгрузноонступае
тКрасавоздушныхстранкаксталонвдругсмешонДразнятот
вклювемутабачныйдымпускаетТотвеселиттолпухромаяк
акионПоэтвтобразтвойТытакжебезусильяЛетаешь
воблакахсредьмолнийигромовНоисполинскиетебемешаю
ткрыльяВнизуходитьвтолпесредьшиканьяглупцов

Когдавморскомпутитоскагрызетматросов,Они,
досужийчасжелаяскоротать,Беспечныхловятптиц,
огромныхальбатросов,Которыесудатаклюбятпровожать.
Ивот,когдацаря любимоголазуриНапалубекладут,
онснежныхдвакрыла,
Умевшихтаклегкопаритьнавстречубури,
Застенчивовлачит,как двабольшихвесла.
Быстрейшийизгонцов,какгрузноонступает!
Красавоздушныхстран,каксталонвдругсмешон!Дразня,
тотвклювемутабачныйдымпускает,Тотвеселиттолпу,
хромая,какион.Поэт,вотобразтвой!
ТытакжебезусильяЛетаешь воблаках,
средьмолнийигромовНоисполинскиетебемешаюткрыльяВ
низуходить,втолпе,средьшиканьяглупцов.

Когда в морском пути тоска грызет матросов, Они,
досужий час желая скоротать, Беспечных ловят птиц,
огромных альбатросов, Которые суда так любят
провожать. И вот, когда царя любимого лазури На
палубе кладут, он снежных два крыла, Умевших так
легко парить навстречу бури, Застенчиво влачит, как
два больших весла. Быстрейший из гонцов, как грузно он
ступает! Краса воздушных стран, как стал он вдруг
смешон! Дразня, тот в клюв ему табачный дым пускает,
Тот веселит толпу, хромая, как и он. Поэт, вот образ
твой! Ты также без усилья Летаешь в облаках, средь
молний и громов, Но исполинские тебе мешают крылья
Внизу ходить, в толпе, средь шиканья глупцов.

Когда в морском пути тоска грызет матросов,
Они, досужий час желая скоротать,
Беспечных ловят птиц, огромных альбатросов,
Которые суда так любят провожать.
И вот, когда царя любимого лазури
На палубе кладут, он снежных два крыла,
Умевших так легко парить навстречу бури,
Застенчиво влачит, как два больших весла.
Быстрейший из гонцов, как грузно он ступает!
Краса воздушных стран, как стал он вдруг смешон!
Дразня, тот в клюв ему табачный дым пускает,
Тот веселит толпу, хромая, как и он.
Поэт, вот образ твой! Ты также без усилья
Летаешь в облаках, средь молний и громов,
Но исполинские тебе мешают крылья
Внизу ходить, в толпе, средь шиканья глупцов.

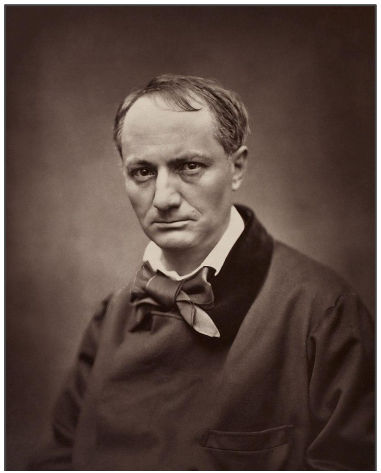
Когда в морском пути тоска грызет матросов,
Они, досужий час желая скоротать,
Беспечных ловят птиц, огромных альбатросов,
Которые суда так любят провожать.
И вот, когда царя любимого лазури
На палубе кладут, он снежных два крыла,
Умевших так легко парить навстречу бури,
Застенчиво влачит, как два больших весла.
Быстрейший из гонцов, как грузно он ступает!
Краса воздушных стран, как стал он вдруг смешон!
Дразня, тот в клюв ему табачный дым пускает,
Тот веселит толпу, хромая, как и он.
Поэт, вот образ твой! Ты также без усилья
Летаешь в облаках, средь молний и громов,
Но исполинские тебе мешают крылья
Внизу ходить, в толпе, средь шиканья глупцов.

Когда в морском пути тоска грызет матросов,
Они, досужий час желая скоротать,
Беспечных ловят птиц, огромных альбатросов,
Которые суда так любят провожать.

И вот, когда царя любимого лазури
На палубе кладут, он снежных два крыла,
Умевших так легко парить навстречу бури,
Застенчиво влачит, как два больших весла.

Быстрейший из гонцов, как грузно он ступает!
Краса воздушных стран, как стал он вдруг смешон!
Дразня, тот в клюв ему табачный дым пускает,
Тот веселит толпу, хромая, как и он.

Поэт, вот образ твой! Ты также без усилья
Летаешь в облаках, средь молний и громов,
Но исполинские тебе мешают крылья
Внизу ходить, в толпе, средь шиканья глупцов.



Шарль Бодлер

“Альбатрос”

Когда в морском пути тоска грызет матросов,
Они, досужий час желая скоротать,
Беспечных ловят птиц, огромных альбатросов,
Которые суда так любят провожать.

И вот, когда царя любимого лазури
На палубе кладут, он снежных два крыла,
Умевших так легко парить навстречу бури,
Застенчиво влачит, как два больших весла.

Быстрейший из гонцов, как грузно он ступает!
Краса воздушных стран, как стал он вдруг смешон!
Дразня, тот в клюв ему табачный дым пускает,
Тот веселит толпу, хромая, как и он.

Поэт, вот образ твой! Ты также без усилья
Летаешь в облаках, средь молний и громов,
Но исполинские тебе мешают крылья
Внизу ходить, в толпе, средь шиканья глупцов.

Когда в морском пути тоска грызет матросов,
Они, досужий час желая скоротать,
Беспечных ловят птиц, огромных альбатросов,
Которые суда так любят провожать.

И вот, когда царя любимого лазури
На палубе кладут, он снежных два крыла,
Умевших так легко парить навстречу бури,
Застенчиво влачит, как два больших весла.

Быстрейший из гонцов, как грузно он ступает!
Краса воздушных стран, как стал он вдруг смешон!
Дразня, тот в клюв ему табачный дым пускает,
Тот веселит толпу, хромая, как и он.

Поэт, вот образ твой! Ты также без усилья
Летаешь в облаках, средь молний и громов,
Но исполинские тебе мешают крылья
Внизу ходить, в толпе, средь шиканья глупцов.

Когда в морском пути тоска грызет матросов,
Они, досужий час желая скоротать,
Беспечных ловят птиц, огромных альбатросов,
Которые суда так любят провожать.

И вот, когда царя любимого лазури
На палубе кладут, он снежных два крыла,
Умевших так легко парить навстречу бури,
Застенчиво влачит, как два больших весла.

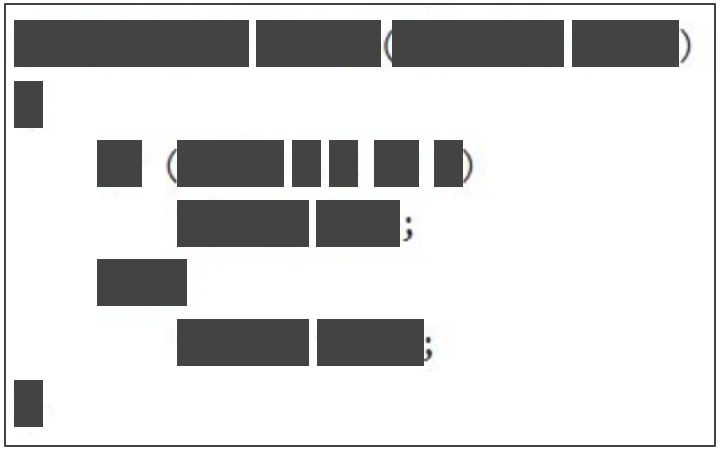
Быстрейший из гонцов, как грузно он ступает!
Краса воздушных стран, как стал он вдруг смешон!
Дразня, тот в клюв ему табачный дым пускает,
Тот веселит толпу, хромая, как и он.

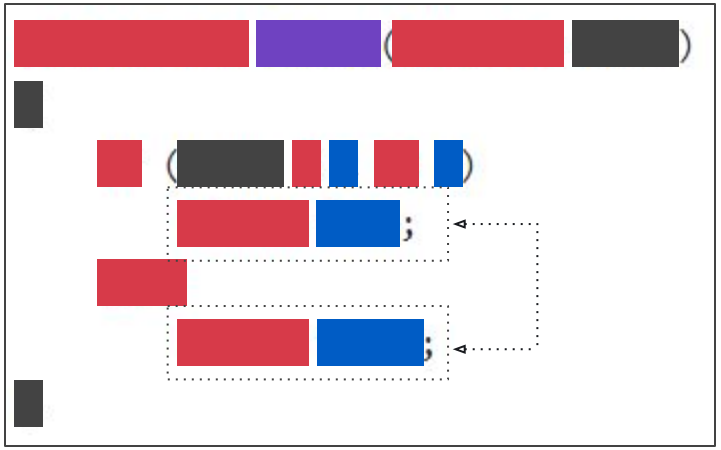
Поэт, вот образ твой! Ты также без усилья
Летаешь в облаках, средь молний и громов,
Но исполинские тебе мешают крылья
Внизу ходить, в толпе, средь шиканья глупцов.

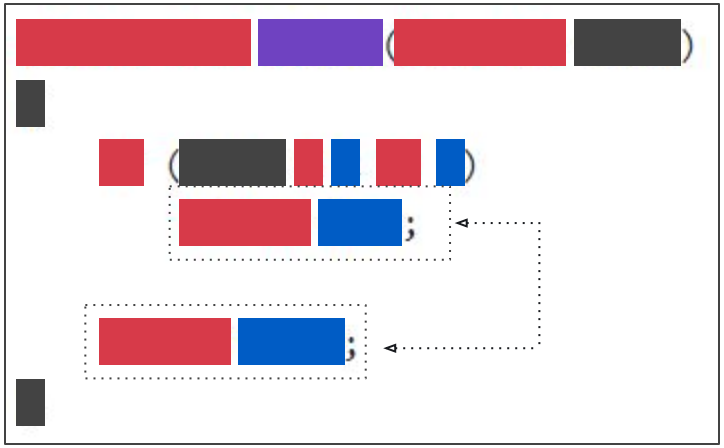
когдавморскомпутитоскагрызетматросовонидосужийчас
желаяскоротатьбеспечныхловятптицогромныхальбатрос
овкоторыесудатаклюбятпровожатьивоткогдацарялюбим
оголазуринапалубекладутонснежныхдвакрылаумевшихта
клегкопаритьнавстречубуризастенчивовлачжткакдвабол
ьшихвеславыстрейшийизгонцовкакгрузноонступаеткраса
воздушныхстранкакталонвдругсмешондразнятотвклюве
мутабачныйдымпускаеттотвеселиттолпухромаякакионпо
этвотобразтвойлытакжебезусильялетаешьвоблакахсредь
молнийигромовноисполинскиетебемешаюткрыльявнизух
одитьвтолпесредьшиканьяглупцов

```
public bool IsEven(this int value)
{
    if (value % 2 == 0)
        return true;
    else
        return false;
}
```

```
public bool IsEven(this int value)
{
    if (value % 2 == 0)
        return true;
    else
        return false;
}
```

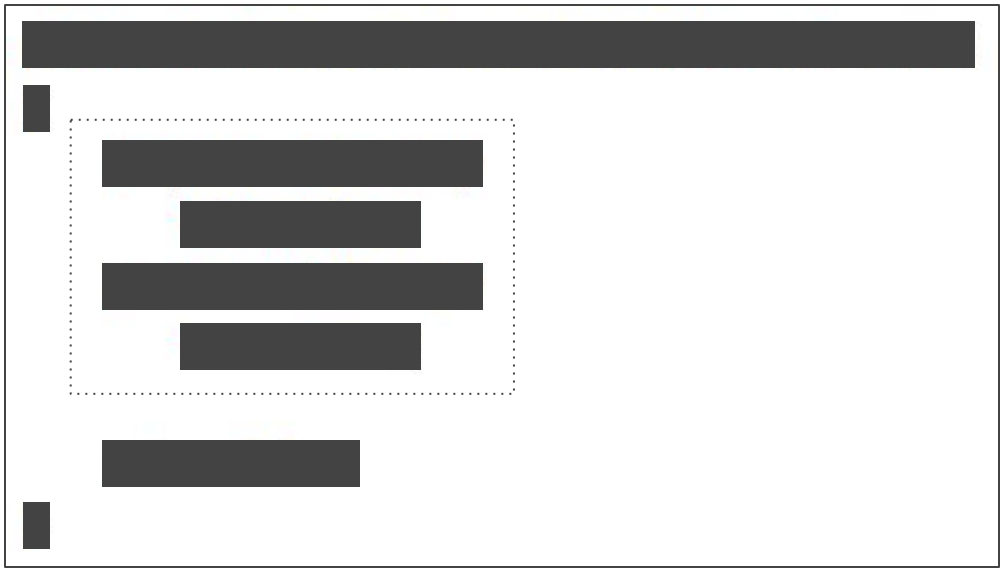


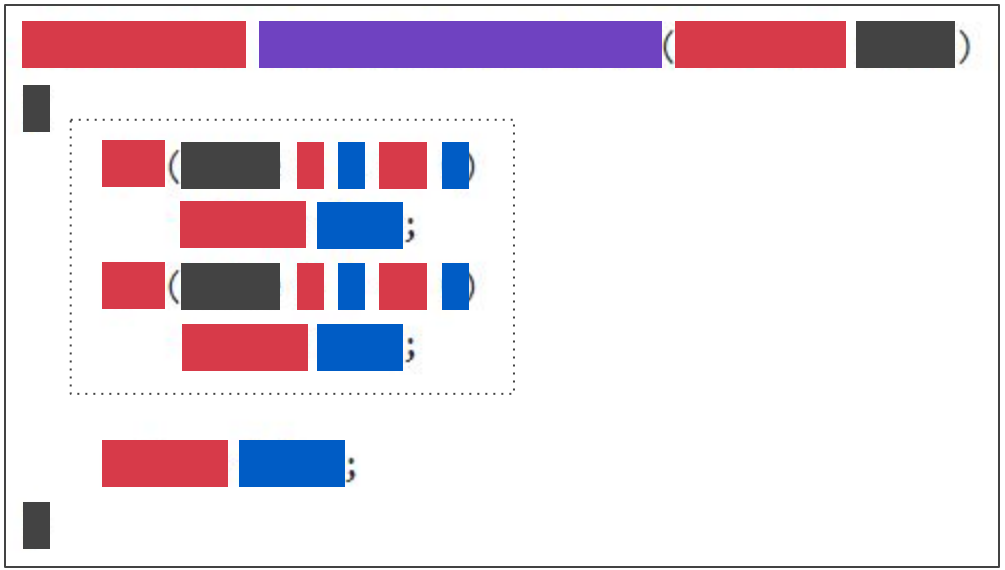




```
public bool IsEvenOrDivisibleBy3(this int value)
{
    if (value % 2 == 0)
        return true;
    if (value % 3 == 0)
        return true;

    return false;
}
```

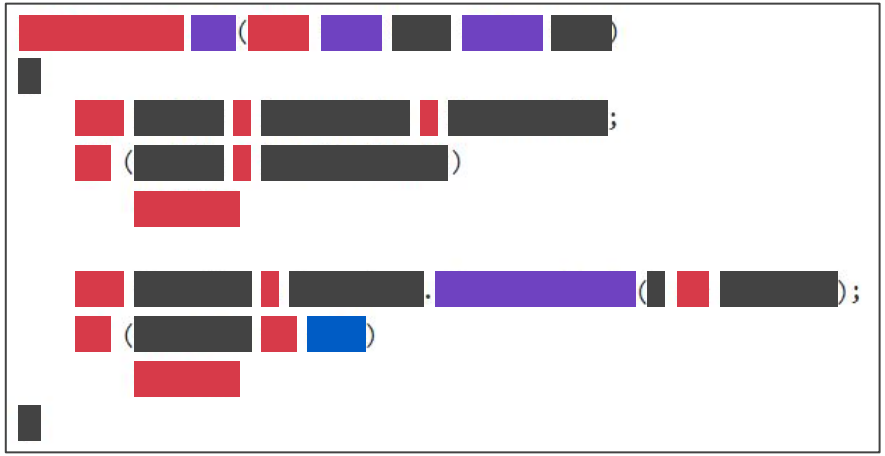
```
public bool IsEvenOrDivisibleBy3(this int value)
{
    if (value % 2 == 0)
        return true;
    if (value % 3 == 0)
        return true;

    return false;
}
```

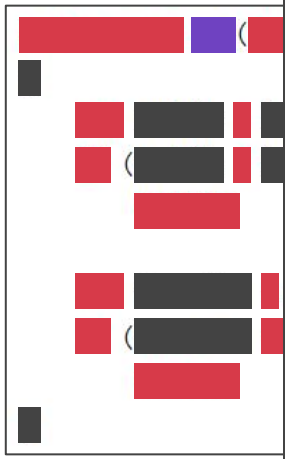




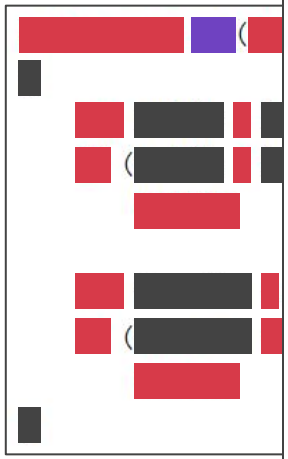
```
public void Put(this IBag bag, IItem item)
{
    var weight = bag.Weight + item.Weight;
    if (weight > bag.MaxWeight)
        return;

    var freeSlot = bag.Slots.FirstOrDefault(x => x.IsFree);
    if (freeSlot == null)
        return;
}
```

```
public void Put(this IBag bag, IItem item)
{
    var weight = bag.Weight + item.Weight;
    if (weight > bag.MaxWeight)
        return;

    var freeSlot = bag.Slots.FirstOrDefault(x => x.IsFree);
    if (freeSlot == null)
        return;
}
```

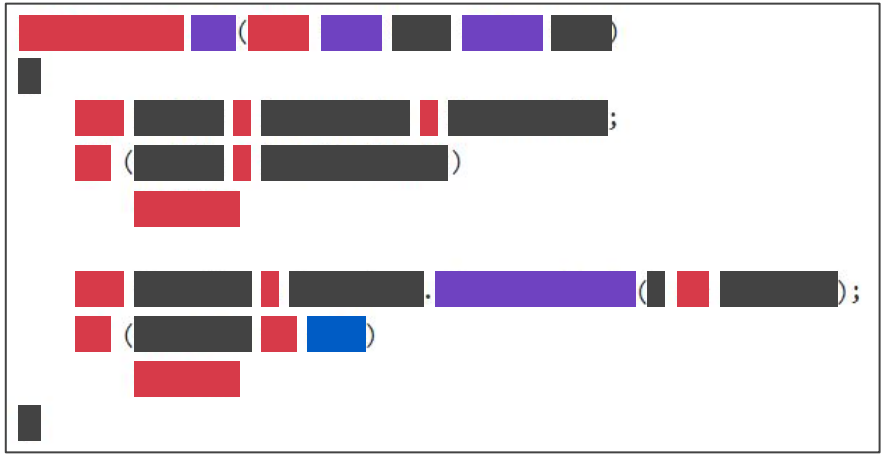


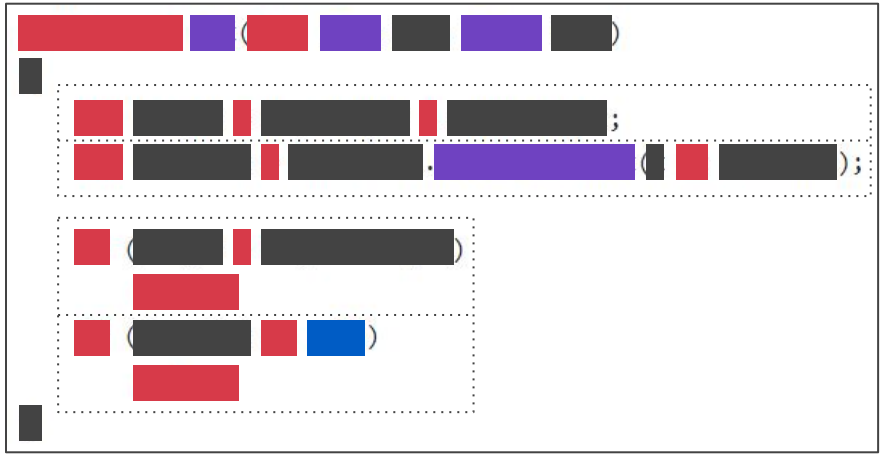


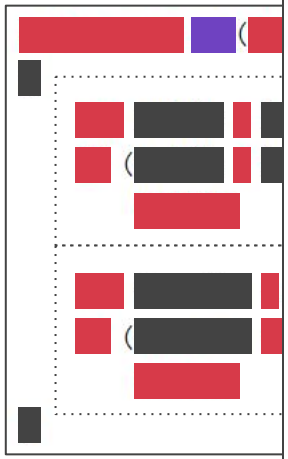
```
public void Put(this IBag bag, IItem item)
{
    var weight = bag.Weight + item.Weight;
    var freeSlot = bag.Slots.FirstOrDefault(x => x.IsFree);

    if (weight > bag.MaxWeight)
        return;
    if (freeSlot == null)
        return;
}
```



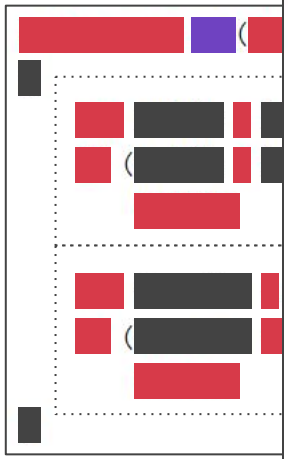
```
public void Put(this IBag bag, IItem item)
{
    var weight = bag.Weight + item.Weight;
    var freeSlot = bag.Slots.FirstOrDefault(x => x.IsFree);

    if (weight > bag.MaxWeight)
        return;
    if (freeSlot == null)
        return;
}
```



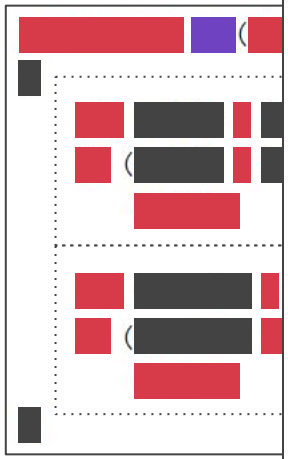




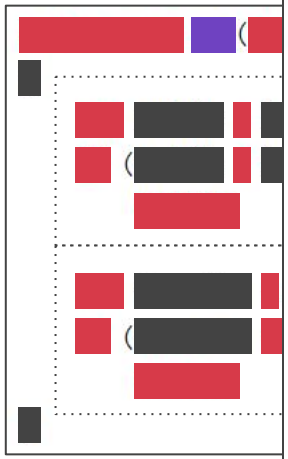
```
public void Put(this IBag bag, IItem item)
{
    var weight = bag.Weight + item.Weight;
    var freeSlot = bag.Slots.FirstOrDefault(x => x.IsFree);

    if (weight > bag.MaxWeight)
        return;
    if (freeSlot == null)
        return;
}
```



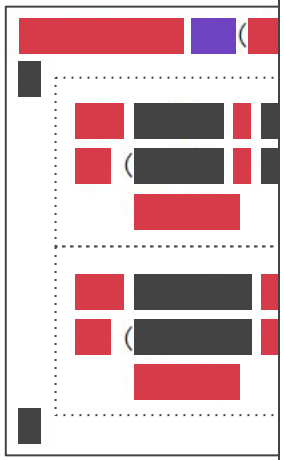
```
public void Put(this IBag bag, IItem item)
{
    var weight = bag.Weight + item.Weight;
    var freeSlot = bag.Slots.FirstOrDefault(x => x.IsFree);

    if (weight > bag.MaxWeight)
        return;
    if (freeSlot == null)
        return;
}
```



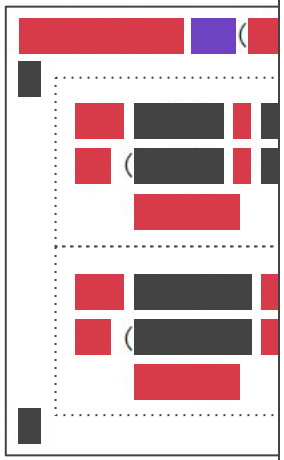
```
public void Put(this IBag bag, IItem item)
{
    var weight = bag.Weight + item.Weight;
    var freeSlot = bag.Slots.FirstOrDefault(x => x.IsFree);

    if (weight > bag.MaxWeight)
        return;
    if (freeSlot == null)
        return;
}
```



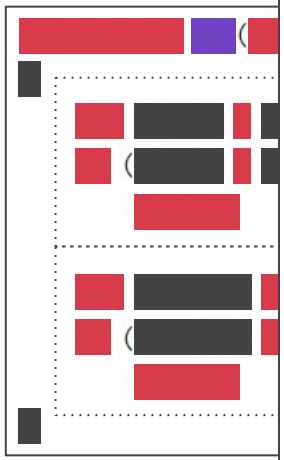
```
public void Put(this IBag bag, IItem item)
{
    var weight = bag.Weight + item.Weight;
    var freeSlot = bag.Slots.FirstOrDefault(x => x.IsFree);

    if (weight > bag.MaxWeight)
        return;
    if (freeSlot == null)
        return;
}
```



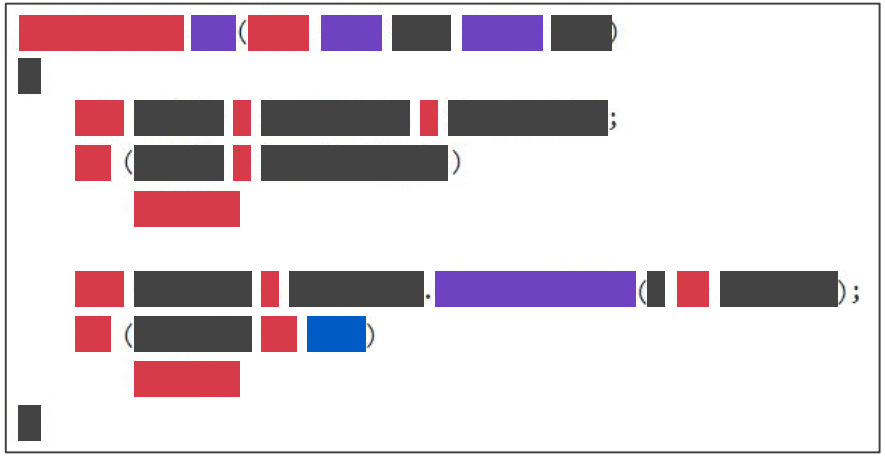
```
public void Put(this IBag bag, IItem item)
{
    var weight = bag.Weight + item.Weight;
    var freeSlot = bag.Slots.FirstOrDefault(x => x.IsFree);
    var isEthereal = item.Effects.Any(x => x.IsEthereal);

    if (weight > bag.MaxWeight)
        return;
    if (freeSlot == null)
        return;
    if (isEthereal)
        return;
}
```

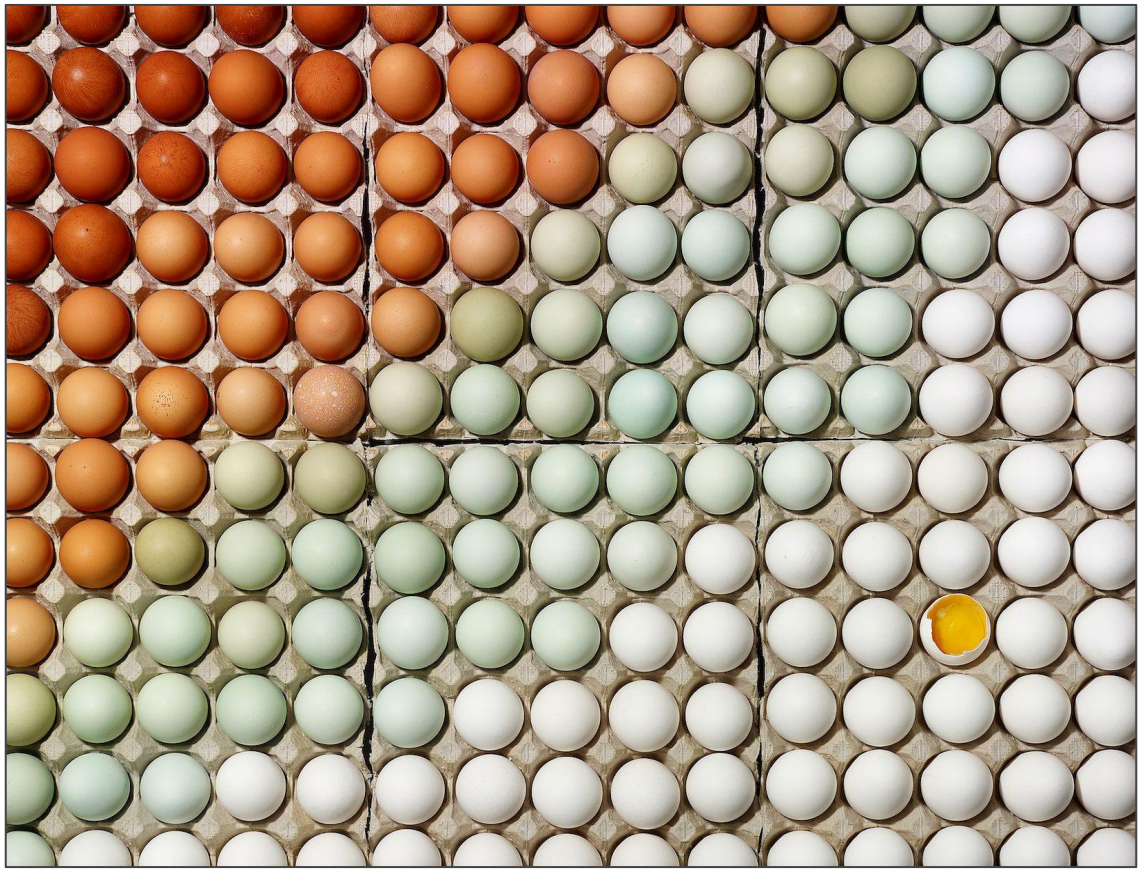


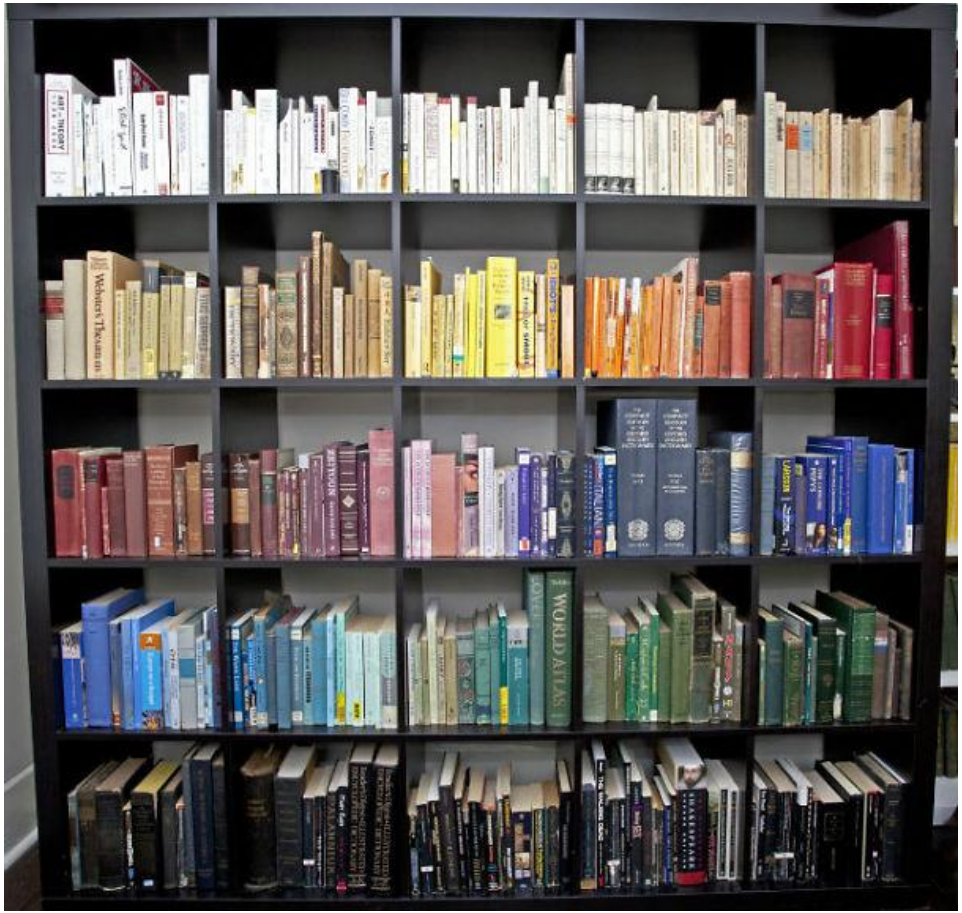
```
public void Put(this IBag bag, IItem item)
{
    var weight = bag.Weight + item.Weight;
    var freeSlot = bag.Slots.FirstOrDefault(x => x.IsFree);
    var isEthereal = item.Effects.Any(x => x.IsEthereal);

    if (weight > bag.MaxWeight)
        return;
    if (freeSlot == null)
        return;
    if (isEthereal)
        return;
}
```

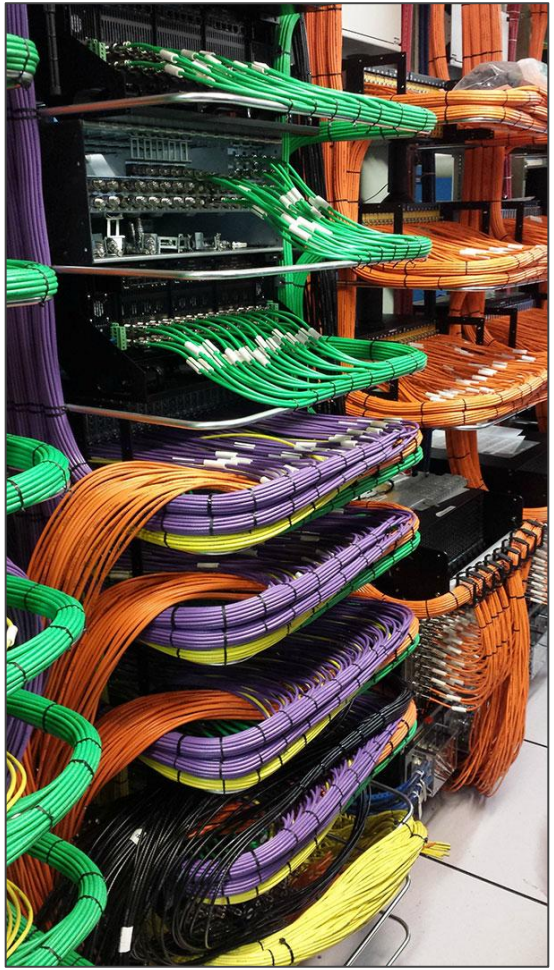


```
public void Put(this IBag bag, IItem item)
{
    var weight = bag.Weight + item.Weight;
    var freeSlot = bag.Slots.FirstOrDefault(x => x.IsFree);
    var isEthereal = item.Effects.Any(x => x.IsEthereal);

    if (weight > bag.MaxWeight)
        return;
    if (freeSlot == null)
        return;
    if (isEthereal)
        return;
}
```







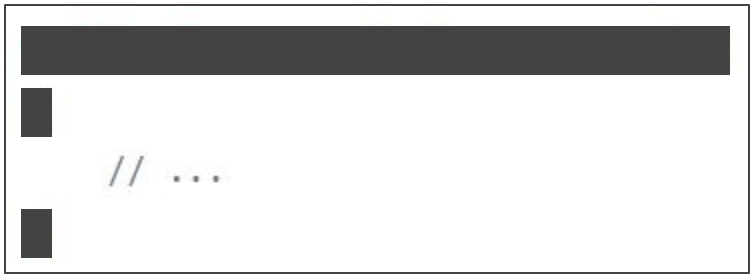


```
public class BottleOfWater
{
    // ...
}
```

```
public class BottleOfWater : Entity
{
    // ...
}
```

```
public class BottleOfWater
    : Entity
{
    // ...
}
```

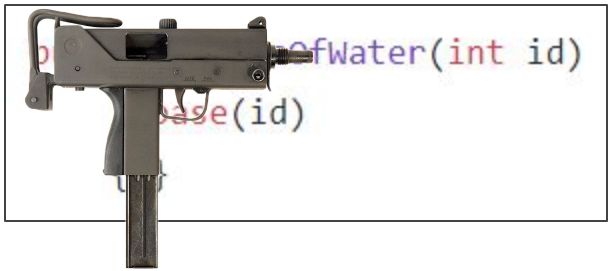
```
public class BottleOfWater : Entity
{
    // ...
}
```



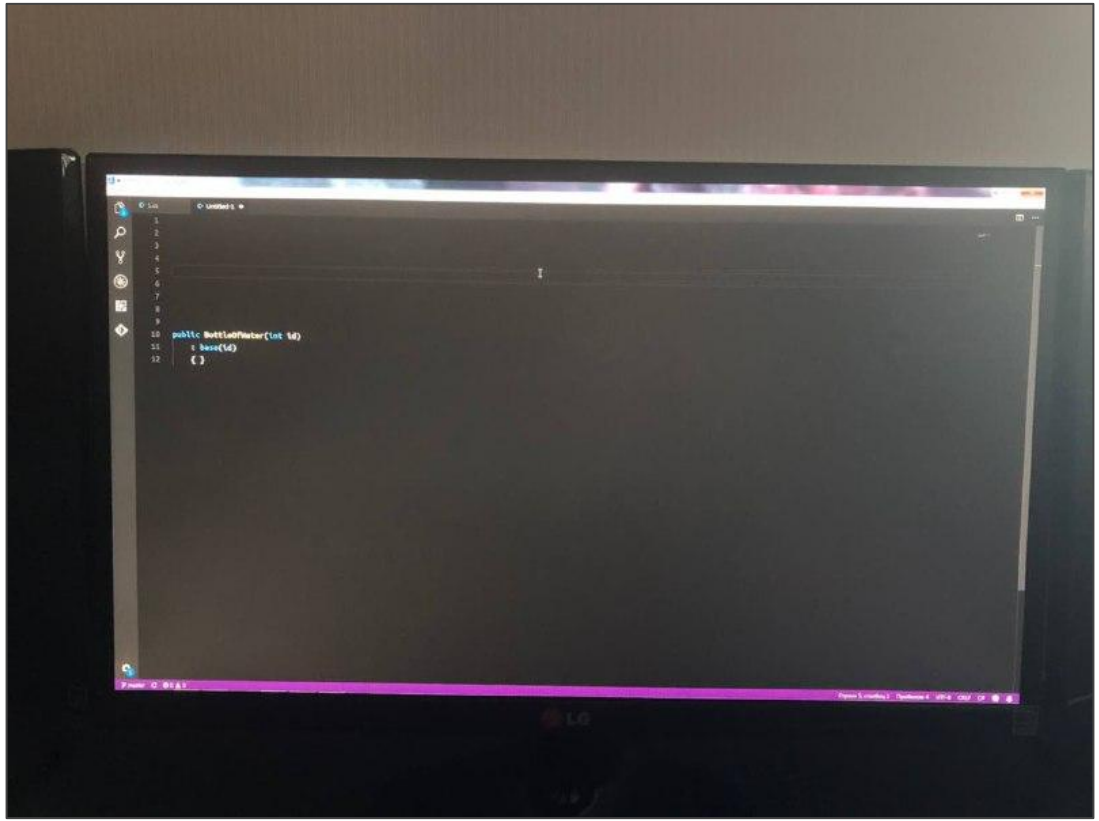
```
public BottleOfWater(int id) : base(id) { }
```

```
public BottleOfWater(int id)
    : base(id) { }
```

```
public BottleOfWater(int id)
    : base(id)
    { }
```



```
public BottleOfWater(int id)
    : base(id)
    { }
```



```
public BottleOfWater(int id)
    : base(id)
    { }
```

```
public BottleOfWater(int id)
    : base(id)
{
    if (id < 0) throw new ArgumentException($"Specified [ {id} ] id is negative");
}
```

```
public class BottleOfWater : Entity
{
    // ...
}
```

```
public class BottleOfWater : Entity, ILiquidContainer
{
    // ...
}
```

```
public class BottleOfWater :
    Entity,
    ILiquidContainer
{
    // ...
}
```

```
public class BottleOfWater :
    Entity,
    ILiquidContainer
{
    // ...
}
```

```
public class BottleOfWater : Entity, ILiquidContainer, IWeapon, IWaterElement, IMaybeAlcohol
{
    // ...
}
```

```
public class BottleOfWater :
    Entity,
    ILiquidContainer,
    IWeapon,
    IWaterElement,
    IMaybeAlcohol
{
    // ...
}
```

```
public class BottleOfWater :
             Entity,
             ILiquidContainer,
             IWeapon,
             IWaterElement,
             IMaybeAlcohol
{
    // ...
}
```

```
public class BottleOfWater :
             Entity,
             ILiquidContainer,
             IWeapon,
             IWaterElement,
             IMaybeAlcohol
{
    // ...
}
```

```
public class Rect
{
    public readonly int Left;
    public readonly int Right;
    public readonly int Top;
    public readonly int Bottom;

    public bool Overlaps(Rect other) =>
        Right > other.Left && Left < other.Right && Bottom > other.Top && Top < other.Bottom;
}
```

```
public class Rect
{
    public readonly int Left;
    public readonly int Right;
    public readonly int Top;
    public readonly int Bottom;

    public bool Overlaps(Rect other) =>
        Right > other.Left && Left < other.Right && Bottom > other.Top && Top < other.Bottom;
}
```

```
public bool Overlaps(Rect other) =>
    Right > other.Left && Left < other.Right && Bottom > other.Top && Top < other.Bottom;
```

```
public bool Overlaps(Rect other) => Right > other.Left
    && Left < other.Right
    && Bottom > other.Top
    && Top < other.Bottom;
```

```
public bool Overlaps(Rect other) => other.Left < Right
    && other.Right > Left
    && other.Top < Bottom
    && other.Bottom > Top;
```

```
public bool Overlaps(Rect other) => other.Left < Right
    && other.Right > Left
    && other.Top < Bottom
    && other.Bottom > Top;
```

```
public bool Overlaps(Rect other) =>
    other.Left < Right
    && other.Right > Left
    && other.Top < Bottom
    && other.Bottom > Top;
```

```
public bool Overlaps(Rect other) =>
    other.Left < Right &&
    other.Right > Left &&
    other.Top < Bottom &&
    other.Bottom > Top;
```

```
public class SocketCommunicator : Communicator
{
    private const float TimeOut = 10f;
    private const int MessageLength = 12000;
    byte[] m_messageHolder = new byte[MessageLength];
    int m_comPort;
    Socket m_sender;
    byte[] m_lengthHolder = new byte[4];
    CommunicatorParameters communicatorParameters;
```

```
public class SocketCommunicator : Communicator
{
    private const float TimeOut = 10f;
    private const int MessageLength = 12000;
    private byte[] m_messageHolder = new byte[MessageLength];
    private int m_comPort;
    private Socket m_sender;
    private byte[] m_lengthHolder = new byte[4];
    private CommunicatorParameters communicatorParameters;
```

```
public class SocketCommunicator : Communicator
{
    private const float TimeOut = 10f;
    private const int MessageLength = 12000;
    private byte[] _messageHolder = new byte[MessageLength];
    private int _comPort;
    private Socket _sender;
    private byte[] _lengthHolder = new byte[4];
    private CommunicatorParameters _communicatorParameters;
```

```
public class SocketCommunicator : Communicator
{
    private const float TimeOut = 10f;
    private const int MessageLength = 12000;
    private readonly byte[] _messageHolder = new byte[MessageLength];
    private int _comPort;
    private Socket _sender;
    private readonly byte[] _lengthHolder = new byte[4];
    private readonly CommunicatorParameters _communicatorParameters;
```

```
public class SocketCommunicator : Communicator
{
    private const float TimeOut = 10f;
    private const int MessageLength = 12000;
    private readonly byte[] _lengthHolder = new byte[4];
    private readonly CommunicatorParameters _communicatorParameters;
    private readonly byte[] _messageHolder = new byte[MessageLength];
    private int _comPort;
    private Socket _sender;
```

```
public class SocketCommunicator : Communicator
{
    private const float TimeOut = 10f;
    private const int MessageLength = 12000;

    private readonly byte[] _lengthHolder = new byte[4];
    private readonly CommunicatorParameters _communicatorParameters;
    private readonly byte[] _messageHolder = new byte[MessageLength];

    private int _comPort;
    private Socket _sender;
```

```
public class SocketCommunicator : Communicator
{
    private const float TimeOut = 10f;
    private const int MessageLength = 12000;

    private readonly CommunicatorParameters _communicatorParameters;
    private readonly byte[] _messageHolder = new byte[MessageLength];
    private readonly byte[] _lengthHolder = new byte[4];

    private Socket _sender;
    private int _comPort;
```

```
public class SocketCommunicator : Communicator
{
    private const float TimeOut = 10f;
    private const int MessageLength = 12000;

    private readonly CommunicatorParameters _communicatorParameters;
    private readonly byte[] _messageHolder = new byte[MessageLength];
    private readonly byte[] _lengthHolder = new byte[4];

    private Socket _sender;
    private int _comPort;
```

```
public class SocketCommunicator : Communicator
{
    private const float TimeOut = 10f;
    private const int MessageLength = 12000;

    private readonly CommunicatorParameters _communicatorParameters;
    private readonly byte[] _messageHolder = new byte[MessageLength];
    private readonly byte[] _lengthHolder = new byte[4];

    private Socket _sender;
    private int _comPort;
```

```
public class SocketCommunicator : Communicator
{
    private const float TimeOut = 10f;
    private const int MessageLength = 12000;

    private readonly CommunicatorParameters _communicatorParameters;
    private readonly byte[] _messageHolder = new byte[MessageLength];
    private readonly byte[] _lengthHolder = new byte[4];

    private Socket _sender;
    private int _comPort;
```

```
public class SocketCommunicator : Communicator
{
    private const float TimeOut = 10f;
    private const int MessageLength = 12000;

    private readonly CommunicatorParameters _communicatorParameters;
    private readonly byte[] _messageHolder = new byte[MessageLength];
    private readonly byte[] _lengthHolder = new byte[4];

    private Socket _sender;
    private int _comPort;
```

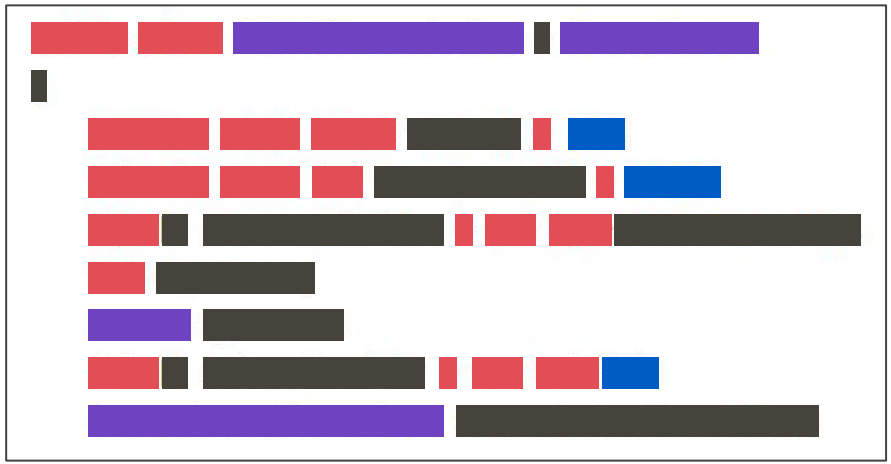
ML-Agents

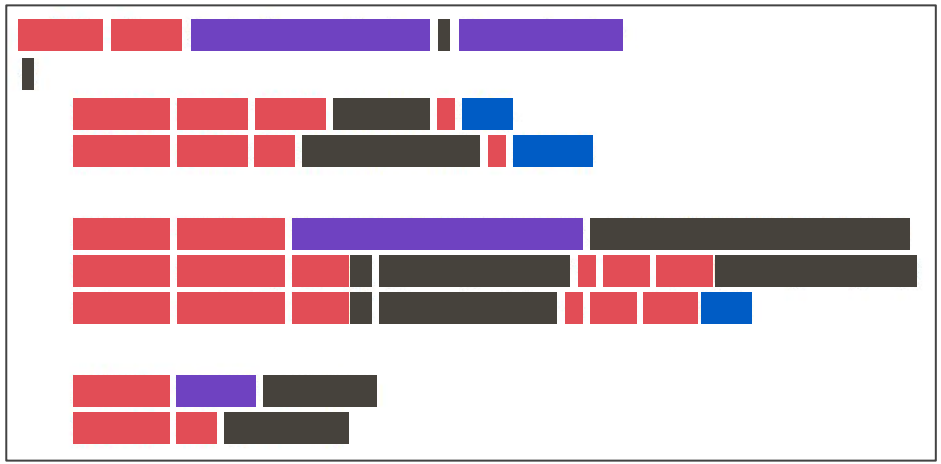
```
public class SocketCommunicator : Communicator
{
    private const float TimeOut = 10f;
    private const int MessageLength = 12000;
    byte[] m_messageHolder = new byte[MessageLength];
    int m_comPort;
    Socket m_sender;
    byte[] m_lengthHolder = new byte[4];
    CommunicatorParameters communicatorParameters;
```


```
public class SocketCommunicator : Communicator
{
    private const float TimeOut = 10f;
    private const int MessageLength = 12000;
    byte[] m_messageHolder = new byte[MessageLength];
    int m_comPort;
    Socket m_sender;
    byte[] m_lengthHolder = new byte[4];
    CommunicatorParameters communicatorParameters;
```

```
public class SocketCommunicator : Communicator
{
    private const float TimeOut = 10f;
    private const int MessageLength = 12000;

    private readonly CommunicatorParameters _communicatorParameters;
    private readonly byte[] _messageHolder = new byte[MessageLength];
    private readonly byte[] _lengthHolder = new byte[4];

    private Socket _sender;
    private int _comPort;
```

ML-Agents






```
public UnityInput Exchange(UnityOutput unityOutput)
{
    Send(WrapMessage(unityOutput, 200).ToByteArray());
    byte[] received = null;
    var task = Task.Run(() => received = Receive());
    if (!task.Wait(System.TimeSpan.FromSeconds(TimeOut)))
    {
        throw new UnityAgentsException(
            "The communicator took too long to respond.");
    }

    var message = UnityMessage.Parser.ParseFrom(received);

    if (message.Header.Status != 200)
    {
        return null;
    }
    return message.UnityInput;
}
```

ML-Agents

```
public UnityInput Exchange(UnityOutput unityOutput)
{
    Send(WrapMessage(unityOutput, 200).ToByteArray());
    byte[] received = null;
    var task = Task.Run(() => received = Receive());
    if (!task.Wait(System.TimeSpan.FromSeconds(TimeOut)))
    {
        throw new UnityAgentsException(
            "The communicator took too long to respond.");
    }

    var message = UnityMessage.Parser.ParseFrom(received);

    if (message.Header.Status != 200)
    {
        return null;
    }
    return message.UnityInput;
}
```


```
public UnityInput Exchange(UnityOutput unityOutput)
{
    Send(WrapMessage(unityOutput, 200).ToByteArray());
    byte[] received = null;
    var task = Task.Run(() => received = Receive());
    if (!task.Wait(System.TimeSpan.FromSeconds(TimeOut)))
    {
        throw new UnityAgentsException(
            "The communicator took too long to respond.");
    }

    var message = UnityMessage.Parser.ParseFrom(received);

    if (message.Header.Status != 200)
    {
        return null;
    }
    return message.UnityInput;
}
```


```
public UnityInput Exchange(UnityOutput unityOutput)
{
    Send(WrapMessage(unityOutput, 200).ToByteArray());
    byte[] received = null;
    var task = Task.Run(() => received = Receive());
    if (!task.Wait(System.TimeSpan.FromSeconds(TimeOut)))
    {
        throw new UnityAgentsException(
            "The communicator took too long to respond.");
    }

    var message = UnityMessage.Parser.ParseFrom(received);

    if (message.Header.Status != 200)
    {
        return null;
    }
    return message.UnityInput;
}
```

ML-Agents

```
public UnityInput Exchange(UnityOutput unityOutput)
{
    Send(WrapMessage(unityOutput, 200).ToByteArray());
    byte[] received = null;
    var task = Task.Run(() => received = Receive());
    if (!task.Wait(System.TimeSpan.FromSeconds(TimeOut)))
    {
        throw new UnityAgentsException(
            "The communicator took too long to respond.");
    }

    var message = UnityMessage.Parser.ParseFrom(received);

    if (message.Header.Status != 200)
    {
        return null;
    }
    return message.UnityInput;
}
```


```
public UnityInput Exchange(UnityOutput unityOutput)
{
    Send(WrapMessage(unityOutput, 200).ToByteArray());
    byte[] received = null;
    var task = Task.Run(() => received = Receive());
    if (!task.Wait(System.TimeSpan.FromSeconds(TimeOut)))
    {
        throw new UnityAgentsException(
            "The communicator took too long to respond.");
    }

    var message = UnityMessage.Parser.ParseFrom(received);

    if (message.Header.Status != 200)
    {
        return null;
    }
    return message.UnityInput;
}
```


```
public UnityInput Exchange(UnityOutput unityOutput)
{
    Send(WrapMessage(unityOutput, 200).ToByteArray());
    byte[] received = null;
    var task = Task.Run(() => received = Receive());
    if (!task.Wait(System.TimeSpan.FromSeconds(TimeOut)))
    {
        throw new UnityAgentsException(
            "The communicator took too long to respond.");
    }

    var message = UnityMessage.Parser.ParseFrom(received);

    if (message.Header.Status != 200)
    {
        return null;
    }
    return message.UnityInput;
}
```

ML-Agents

```
public UnityInput Exchange(UnityOutput unityOutput)
{
    Send(WrapMessage(unityOutput, 200).ToByteArray());

    byte[] received = null;
    var task = Task.Run(() => received = Receive());
    if (!task.Wait(System.TimeSpan.FromSeconds(TimeOut)))
    {
        throw new UnityAgentsException(
            "The communicator took too long to respond.");
    }

    var message = UnityMessage.Parser.ParseFrom(received);

    if (message.Header.Status != 200)
    {
        return null;
    }
    return message.UnityInput;
}
```

ML-Agents

```
public UnityInput Exchange(UnityOutput unityOutput)
{
    Send(WrapMessage(unityOutput, 200).ToByteArray());

    byte[] received = null;
    var task = Task.Run(() => received = Receive());
    if (!task.Wait(System.TimeSpan.FromSeconds(TimeOut)))
    {
        throw new UnityAgentsException(
            "The communicator took too long to respond.");
    }

    var message = UnityMessage.Parser.ParseFrom(received);

    if (message.Header.Status != 200)
    {
        return null;
    }
    return message.UnityInput;
}
```

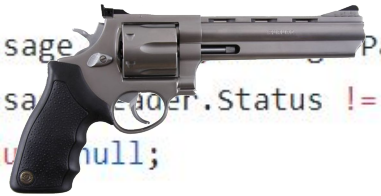

```
public UnityInput Exchange(UnityOutput unityOutput)
{
    Send(WrapMessage(unityOutput, 200).ToByteArray());

    byte[] received = null;
    var task = Task.Run(() => received = Receive());
    if (!task.Wait(System.TimeSpan.FromSeconds(TimeOut)))
        throw new UnityAgentsException(
            "The communicator took too long to respond.");

    var message = UnityMessage.Parser.ParseFrom(received);

    if (message.Header.Status != 200)
        return null;
    return message.UnityInput;
}
```


```
public UnityInput Exchange(UnityOutput unityOutput)
{
    Send(WrapMessage(unityOutput, 200).ToByteArray());

    byte[] received = null;
    var task = Task.Run(() => received = Receive());
    if (!task.Wait(System.TimeSpan.FromSeconds(TimeOut)))
        throw new UnityAgentsException(
            "The communicator took too long to respond.");

    var message = UnityMessage.Parser.ParseFrom(received);

    if (message.Header.Status != 200)
        return null;
    return message.UnityInput;
}
```


```
public UnityInput Exchange(UnityOutput unityOutput)
{
    Send(WrapMessage(unityOutput, 200).ToByteArray());

    byte[] received = null;
    var task = Task.Run(() => received = Receive());
    if (!task.Wait(System.TimeSpan.FromSeconds(TimeOut)))
        throw new UnityAgentsException(
            "The communicator took too long to respond.");

    var message = UnityMessage.Parser.ParseFrom(received);
    if (message.Header.Status != 200)
        return null;

    return message.UnityInput;
}
```

ML-Agents

```
public UnityInput Exchange(UnityOutput unityOutput)
{
    Send(WrapMessage(unityOutput, 200).ToByteArray());

    byte[] received = null;
    var task = Task.Run(() => received = Receive());
    if (!task.Wait(System.TimeSpan.FromSeconds(TimeOut)))
        throw new UnityAgentsException(
            "The communicator took too long to respond.");

    var message = CommunicatorObjects.UnityMessage.Parser.ParseFrom(received);
    if (message.Header.Status != 200)
        return null;

    return message.UnityInput;
}
```




```
public UnityInput Exchange(UnityOutput unityOutput)
{
    Send(WrapMessage(unityOutput, 200).ToByteArray());

    byte[] received = null;
    var task = Task.Run(() => received = Receive());
    if (!task.Wait(System.TimeSpan.FromSeconds(TimeOut)))
        throw new UnityAgentsException(
            "The communicator took too long to respond.");

    var message = UnityMessage.Parser.ParseFrom(received);
    if (message.Header.Status != 200)
        return null;

    return message.UnityInput;
}
```

ML-Agents

```
public UnityInput Exchange(UnityOutput unityOutput)
{
    Send(WrapMessage(unityOutput, 200).ToByteArray());

    byte[] received = null;
    var task = Task.Run(() => received = Receive());
    if (!task.Wait(System.TimeSpan.FromSeconds(TimeOut)))
        throw new UnityAgentsException(
            "The communicator took too long to respond.");

    var message = UnityMessage.Parser.ParseFrom(received);
    if (message.Header.Status != 200)
        return null;

    return message.UnityInput;
}
```


```
public UnityInput Exchange(UnityOutput unityOutput)
{
    Send(WrapMessage(unityOutput, 200).ToByteArray());

    byte[] received = null;
    var task = Task.Run(() => received = Receive());
    if (!task.Wait(System.TimeSpan.FromSeconds(TimeOut)))
        throw new UnityAgentsException(
            "The communicator took too long to respond.");

    var message = UnityMessage.Parser.ParseFrom(received);
    if (message.Header.Status != 200)
        return null;

    return message.UnityInput;
}
```

ML-Agents

```
public UnityInput Exchange(UnityOutput unityOutput)
{
    Send(WrapMessage(unityOutput, 200).ToByteArray());

    var received = Task.Run(() => Receive());
    if (!received.Wait(System.TimeSpan.FromSeconds(TimeOut)))
        throw new UnityAgentsException(
            "The communicator took too long to respond.");

    var message = UnityMessage.Parser.ParseFrom(received.Result);
    if (message.Header.Status != 200)
        return null;

    return message.UnityInput;
}
```

ML-Agents

```
public UnityInput Exchange(UnityOutput unityOutput)
{
    Send(WrapMessage(unityOutput, 200).ToByteArray());

    var received = Task.Run(() => Receive());
    if (!received.Wait(System.TimeSpan.FromSeconds(TimeOut)))
        throw new UnityAgentsException(
            "The communicator took too long to respond.");

    var message = UnityMessage.Parser.ParseFrom(received.Result);
    if (message.Header.Status != 200)
        return null;

    return message.UnityInput;
}
```

```
public UnityInput Exchange(UnityOutput unityOutput)
{
    Send(WrapMessage(unityOutput, 200).ToByteArray());

    var received = Task.Run(() => Receive());
    if (!received.Wait(System.TimeSpan.FromSeconds(TimeOut)))
        throw new UnityAgentsException(
            "The communicator took too long to respond.");

    var message = UnityMessage.Parser.ParseFrom(received.Result);
    if (message.Header.Status != 200)
        return null;

    return message.UnityInput;
}
```

ML-Agents

```
public UnityInput Exchange(UnityOutput unityOutput)
{
    Send(WrapMessage(unityOutput, 200).ToByteArray());

    var received = Task.Run(() => Receive());
    if (!received.Wait(System.TimeSpan.FromSeconds(TimeOut)))
        throw new UnityAgentsException(
            "The communicator took too long to respond.");

    var message = UnityMessage.Parser.ParseFrom(received.Result);
    if (message.Header.Status != 200)
        return null;

    return message.UnityInput;
}
```

ML-Agents

```
public UnityInput Exchange(UnityOutput unityOutput)
{
    Send(WrapMessage(unityOutput, 200).ToByteArray());

    var received = Task.Run(() => Receive());
    if (!received.Wait(System.TimeSpan.FromSeconds(TimeOut)))
        throw new UnityAgentsException(
            "The communicator took too long to respond.");

    var message = UnityMessage.Parser.ParseFrom(received.Result);
    if (message.Header.Status != 200)
        return null;

    return message.UnityInput;
}
```

ML-Agents

```
public UnityInput Exchange(UnityOutput unityOutput)
{
    Send(WrapMessage(unityOutput, 200).ToByteArray());

    var received = Task.Run(() => Receive());
    if (!received.Wait(TimeOut * 1000))
        throw new UnityAgentsException(
            "The communicator took too long to respond.");

    var message = UnityMessage.Parser.ParseFrom(received.Result);
    if (message.Header.Status != 200)
        return null;

    return message.UnityInput;
}
```

ML-Agents

```
public UnityInput Exchange(UnityOutput unityOutput)
{
    Send(WrapMessage(unityOutput, 200).ToByteArray());

    var received = Task.Run(() => Receive());
    if (!received.Wait(TimeOut * 1000))
        throw new UnityAgentsException(
            "The communicator took too long to respond.");

    var message = UnityMessage.Parser.ParseFrom(received.Result);
    if (message.Header.Status != 200)
        return null;

    return message.UnityInput;
}
```

ML-Agents

```
public UnityInput Exchange(UnityOutput unityOutput)
{
    Send(WrapMessage(unityOutput, 200).ToByteArray());

    var received = Task.Run(() => Receive());
    if (!received.Wait(TimeOut * 1000))
        throw new UnityAgentsException(
            "The communicator took too long to respond.");

    var message = UnityMessage.Parser.ParseFrom(received.Result);
    if (message.Header.Status != 200)
        return null;

    return message.UnityInput;
}
```

ML-Agents

```
public UnityInput Exchange(UnityOutput unityOutput)
{
    Send(WrapMessage(unityOutput, 200).ToByteArray());

    var received = Task.Run(() => Receive());
    if (received.Wait(TimeOut * 1000))
    {
        var message = UnityMessage.Parser.ParseFrom(received.Result);
        if (message.Header.Status != 200)
            return null;

        return message.UnityInput;
    }

    throw new UnityAgentsException("The communicator took too long to respond.");
}
```

ML-Agents

```
public UnityInput Exchange(UnityOutput unityOutput)
{
    Send(WrapMessage(unityOutput, 200).ToByteArray());

    var received = Task.Run(() => Receive());
    if (received.Wait(TimeOut * 1000))
    {
        var message = UnityMessage.Parser.ParseFrom(received.Result);
        if (message.Header.Status != 200)
            return null;

        return message.UnityInput;
    }

    throw new UnityAgentsException("The communicator took too long to respond.");
}
```

ML-Agents

```
public UnityInput Exchange(UnityOutput unityOutput)
{
    Send(WrapMessage(unityOutput, 200).ToByteArray());

    var received = Task.Run(() => Receive());
    if (received.Wait(TimeOut * 1000))
    {
        var message = UnityMessage.Parser.ParseFrom(received.Result);
        if (message.Header.Status != 200)
            return null;

        return message.UnityInput;
    }

    throw new UnityAgentsException("The communicator took too long to respond.");
}
```

```
public UnityInput Exchange(UnityOutput unityOutput)
{
    Send(WrapMessage(unityOutput, 200).ToByteArray());

    var received = Task.Run(() => Receive());
    if (received.Wait(TimeOut * 1000))
    {
        var message = UnityMessage.Parser.ParseFrom(received.Result);
        if (message.Header.Status != 200)
            return null;

        return message.UnityInput;
    }

    throw new UnityAgentsException("The communicator took too long to respond.");
}
```

ML-Agents

```
public UnityInput Exchange(UnityOutput unityOutput)
{
```

```
public byte[] Exchange(byte[] bytes)
{
    Send(bytes);

    var received = Task.Run(() => Receive());
    if (received.Wait(TimeOut * 1000))
        return received.Result;

    throw new UnityAgentsException("The communicator took too long to respond.");
}
```

```
    throw new UnityAgentsException("The communicator took too long to respond.");
}
```

```
public byte[] Exchange(byte[] bytes)
{
    Send(bytes);

    var received = Task.Run(() => Receive());
    if (received.Wait(TimeOut * 1000))
        return received.Result;

    throw new UnityAgentsException("The communicator took too long to respond.");
}
```

```
public byte[] Exchange(byte[] bytes)
{
    Send(bytes);

    var received = Task.Run(() => Receive());
    if (received.Wait(TimeOut * 1000))
        return received.Result;

    throw new UnityAgentsException("The communicator took too long to respond.");
}
```

```
public byte[] Exchange(byte[] bytes)
{
    Send(bytes);

    return Receive(TimeOut * 1000);




}
```


```
public byte
{
    Send(by

    return Receive(TimeOut * 1000);



}
```

```
public byte[] Receive(int timeout)
{
    var received = Task.Run(() => Receive());
    if (received.Wait(timeout))
        return received.Result;

    throw new UnityAgentsException("The communicator took too long to respond.");
}
```

```
public UnityInput Exchange(UnityOutput unityOutput)
{



}
```

```
public UnityInput Exchange(UnityOutput unityOutput)
{
    var output = WrapMessage(unityOutput, 200).ToByteArray();




}
```

```
public UnityInput Exchange(UnityOutput unityOutput)
{
    var output = WrapMessage(unityOutput, 200).ToByteArray();
    var input = Exchange(output);




}
```

```
public UnityInput Exchange(UnityOutput unityOutput)
{
    var output = WrapMessage(unityOutput, 200).ToByteArray();
    var input = Exchange(output);
    var parsedInput = UnityMessage.Parser.ParseFrom(input);




}
```

```
public UnityInput Exchange(UnityOutput unityOutput)
{
    var output = WrapMessage(unityOutput, 200).ToByteArray();
    var input = Exchange(output);
    var parsedInput = UnityMessage.Parser.ParseFrom(input);
    if (parsedInput.Header.Status != 200)
        return null;

    return parsedInput.UnityInput;
}
```


```
public UnityInput Exchange(UnityOutput unityOutput)
{
    Send(WrapMessage(unityOutput, 200).ToByteArray());
    byte[] received = null;
    var task = Task.Run(() => received = Receive());
    if (!task.Wait(System.TimeSpan.FromSeconds(TimeOut)))
    {
        throw new UnityAgentsException(
            "The communicator took too long to respond.");
    }

    var message = UnityMessage.Parser.ParseFrom(received);

    if (message.Header.Status != 200)
    {
        return null;
    }
    return message.UnityInput;
}
```

ML-Agents

```
public UnityInput Exchange(UnityOutput unityOutput)
{
    Send(WrapMessage(unityOutput, 200).ToByteArray());
    byte[] received = null;
    var task = Task.Run(() => received = Receive());
    if (!task.Wait(System.TimeSpan.FromSeconds(TimeOut)))
    {
        throw new UnityAgentsException(
            "The communicator took too long to respond.");
    }

    var message = UnityMessage.Parser.ParseFrom(received);

    if (message.Header.Status != 200)
    {
        return null;
    }
    return message.UnityInput;
}
```

ML-Agents

```
public UnityInput Exchange(UnityOutput unityOutput)
{
    var output = WrapMessage(unityOutput, 200).ToByteArray();
    var input = Exchange(output);
    var parsedInput = UnityMessage.Parser.ParseFrom(input);
    if (parsedInput.Header.Status != 200)
        return null;

    return parsedInput.UnityInput;
}
```

```
public UnityInput Excha
{
    Send(WrapMessage(un
    byte[] received = n
    var task = Task.Run
    if (!task.Wait(Syst
    {
        throw new Unity
            "The commun
    }

    var message = Unity

    if (message.Header.Status != 200)
    {
        return null;
    }
    return message.UnityInput;
}
```

ML-Agents

```
public UnityInput Exchange(UnityOutput unityOutput)
{
    Send(WrapMessage(unityOutput, 200).ToByteArray());
    byte[] received = null;
    var task = Task.Run(() => received = Receive());
    if (!task.Wait(System.TimeSpan.FromSeconds(TimeOut)))
    {
        throw new UnityAgentsException(
            "The communicator took too long to respond.");
    }

    var message = UnityMessage.Parser.ParseFrom(received);

    if (message.Header.Status != 200)
    {
        return null;
    }
    return message.UnityInput;
}
```

```
public UnityInput Exchange(UnityOutput unityOutput)
{
    var output = WrapMessage(unityOutput, 200).ToByteArray();
    var input = Exchange(output);
    var parsedInput = UnityMessage.Parser.ParseFrom(input);
    if (parsedInput.Header.Status != 200)
        return null;

    return parsedInput.UnityInput;
}
```


```
public UnityInput Exchange(UnityOutput unityOutput)
{
    Send(WrapMessage(unityOutput, 200).ToByteArray());
    byte[] received = null;
    var task = Task.Run(() => received = R
    if (!task.Wait(System.TimeSpan.FromSec
    {
        throw new UnityAgentsException(
            "The communicator took too lon
    }

    var message = UnityMessage.Parser.Pars

    if (message.Header.Status != 200)
    {
        return null;
    }
    return message.UnityInput;
}
```

```
public UnityInput Exchange(UnityOutput unityOutput)
{
    var output = WrapMessage(unityOutput, 200).ToByteArray();
    var input = Exchange(output);
    var parsedInput = UnityMessage.Parser.ParseFrom(input);
    if (parsedInput.Header.Status != 200)
        return null;
```

```
public byte[] Exchange(byte[] bytes)
{
    Send(bytes);

    return Receive(TimeOut * 1000);
}
```

ML-Agents

```
public UnityInput Exchange(UnityOutput unityOutput)
{
    Send(WrapMessage(unityOutput, 200).ToByteArray());
    byte[] received = null;
    var task = Task.Run(() => received = Receive());
    if (!task.Wait(System.TimeSpan.FromSeconds(TimeOut)))
    {
        throw new UnityAgentsException(
            "The communicator took too long to respond.");
    }

    var message = UnityMessage.Parser.ParseFrom(received);

    if (message.Header.Status != 200)
    {
        return null;
    }
    return message.UnityInput;
}
```

```
public UnityInput Exchange(UnityOutput unityOutput)
{
    var output = WrapMessage(unityOutput, 200).ToByteArray();
    var input = Exchange(output);
    var parsedInput = UnityMessage.Parser.ParseFrom(input);
    if (parsedInput.Header.Status != 200)
        return null;

    return parsedInput.UnityInput;
}
```

```
public byte[] Exchange(byte[] bytes)
{
    Send(bytes);

    return Receive(TimeOut * 1000);
}
```

ML-Agents

```
public UnityInput Exchange(UnityOutput unityOutput)
{
    var output = WrapMessage(unityOutput, 200).ToByteArray();
    var input = Exchange(output);
    var parsedInput = UnityMessage.Parser.ParseFrom(input);
    if (parsedInput.Header.Status != 200)
        return null;
```

```
public UnityInput Exchange(UnityOutput unityOutput)
{
    Send(WrapMessage(unityOutput, 200).ToByteArray());
    byte[]
    var tas
    if (!ta
    {
        thr

    }

    var mes

    if (mes
    {
        ret
    }
    return message.UnityInput;
}
```

```
public byte[] Receive(int timeout)
{
    var received = Task.Run(() => Receive());
    if (received.Wait(timeout))
        return received.Result;

    throw new UnityAgentsException("The communicator took too long to respond.");
}
```

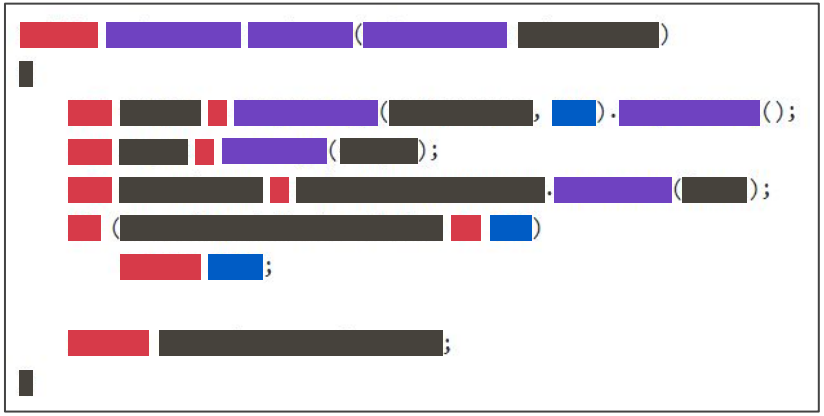

```
public UnityInput Exchange(UnityOutput unityOutput)
{
    Send(WrapMessage(unityOutput, 200).ToByteArray());
    byte[] received = null;
    var task = Task.Run(() => received = Receive());
    if (!task.Wait(System.TimeSpan.FromSeconds(TimeOut)))
    {
        throw new UnityAgentsException(
            "The communicator took too long to respond.");
    }

    var message = UnityMessage.Parser.ParseFrom(received);

    if (message.Header.Status != 200)
    {
        return null;
    }
    return message.UnityInput;
}
```

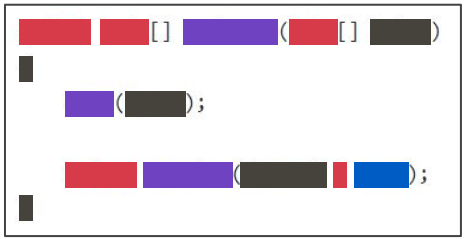
```
public UnityInput Exchange(UnityOutput unityOutput)
{
    var output = WrapMessage(unityOutput, 200).ToByteArray();
    var input = Exchange(output);
    var parsedInput = UnityMessage.Parser.ParseFrom(input);
    if (parsedInput.Header.Status != 200)
        return null;

    return parsedInput.UnityInput;
}
```

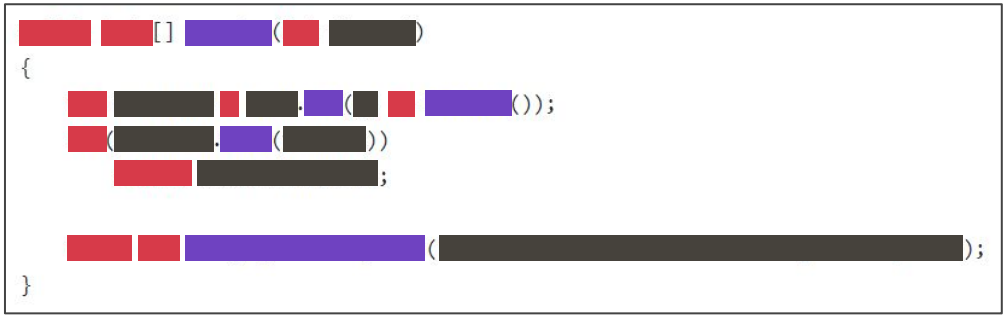
```
public byte[] Exchange(byte[] bytes)
{
    Send(bytes);

    return Receive(TimeOut * 1000);
}
```

```
public byte[] Receive(int timeout)
{
    var received = Task.Run(() => Receive());
    if (received.Wait(timeout))
        return received.Result;

    throw new UnityAgentsException("The communicator took too long to respond.");
}
```

ML-Agents







Когда в морском пути тоска грызет матросов,
Они, досужий час желая скоротать,
Беспечных ловят птиц, огромных альбатросов,
Которые суда так любят провожать.

И вот, когда царя любимого лазури
На палубе кладут, он снежных два крыла,
Умевших так легко парить навстречу бури,
Застенчиво влачит, как два больших весла.

Быстрейший из гонцов, как грузно он ступает!
Краса воздушных стран, как стал он вдруг смешон!
Дразня, тот в клюв ему табачный дым пускает,
Тот веселит толпу, хромая, как и он.

Поэт, вот образ твой! Ты также без усилья
Летаешь в облаках, средь молний и громов,
Но исполинские тебе мешают крылья
Внизу ходить, в толпе, средь шиканья глупцов.

когдавморскомпутитоскагрызетматросовонидосужийчас
желаяскоротатьбеспечныхловятптицогромныхальбатрос
овкоторыесудатаклюбятпровожатьивоткогдацарялюбим
оголазуринапалубекладутонснежныхдвакрылаумевшихта
клегкопаритьнавстречубуризастенчивовлачжткакдвабол
ьшихвеславыстрейшийизгонцовкакгрузноонступаеткраса
воздушныхстранкакталонвдругсмешондразнятотвклюве
мутабачныйдымпускаеттотвеселиттолпухромаякакионпо
этвотобразтвойлытакжебезусильялетаешьвоблакахсредь
молнийигромовноисполинскиетебемешаюткрыльявнизух
одитьвтолпесредьшиканьяглупцов

Когда в морском пути тоска грызет матросов,
Они, досужий час желая скоротать,
Беспечных ловят птиц, огромных альбатросов,
Которые суда так любят провожать.

И вот, когда царя любимого лазури
На палубе кладут, он снежных два крыла,
Умевших так легко парить навстречу бури,
Застенчиво влачит, как два больших весла.

Быстрейший из гонцов, как грузно он ступает!
Краса воздушных стран, как стал он вдруг смешон!
Дразня, тот в клюв ему табачный дым пускает,
Тот веселит толпу, хромая, как и он.

Поэт, вот образ твой! Ты также без усилья
Летаешь в облаках, средь молний и громов,
Но исполинские тебе мешают крылья
Внизу ходить, в толпе, средь шиканья глупцов.

когдавморскомпутитоскагрызетматросовонидосужийчас
желаяскоротатьбеспечныхловятптицогромныхальбатрос
овкоторыесудатаклюбятпровожатьиводкогдацарялюбим
оголазуринапалубекладутонснежныхдвакрылаумевшихта
клегкопаритьнавстречубуризастенчивовлачжткакдвабол
ьшихвеславыстрейшийизгонцовкакгрузноонступаеткраса
воздушныхстранкакталонвдругсмешондразнятотвклюве
мутабачныйдымпускаеттотвеселиттолпухромаякакионпо
этвотобразтвойлытакжебезусильялетаешьвоблакахсредь
молнийигромовноисполинскиетебемешаюткрыльявнизух
одитьвтолпесредьшиканьяглупцов

Когда в морском пути тоска грызет матросов,
Они, досужий час желая скоротать,
Беспечных ловят птиц, огромных альбатросов,
Которые суда так любят провожать.

И вот, когда царя любимого лазури
На палубе кладут, он снежных два крыла,
Умевших так легко парить навстречу бури,
Застенчиво влачит, как два больших весла.

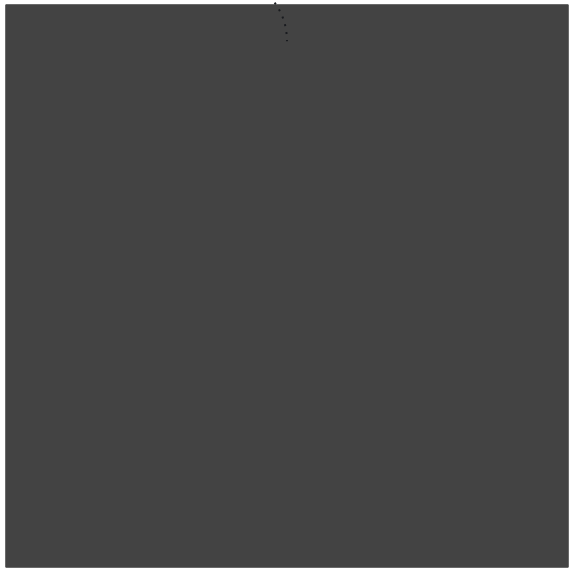
Быстрейший из гонцов, как грузно он ступает!
Краса воздушных стран, как стал он вдруг смешон!
Дразня, тот в клюв ему табачный дым пускает,
Тот веселит толпу, хромая, как и он.

Поэт, вот образ твой! Ты также без усилья
Летаешь в облаках, средь молний и громов,
Но исполинские тебе мешают крылья
Внизу ходить, в толпе, средь шиканья глупцов.

когдавморскомпутитоскагры влачжт ужийчас
желаяскоротатьбеспечныхло батрос
овкоторыесудатаклюбятпровожатьивоткогдацарялюбим
оголазуринапалубекладутонснежныхдвакрылаумевшихта
клегкопаритьнавстречубуризастенчивовлачжткакдвабол
ьшихвеславыстрейшийизгонцовкакгрузноонступаеткраса
воздушныхстранкакталонвдругсмешондразнятотвклюве
мутабачныйдымпускаеттотвеселиттолпухромаякакионпо
этвотобразтвойлытакжебезусильялетаешьвоблакахсредь
молнийигромовноисполинскиетебемешаюткрыльявнизух
одитьвтолпесредьшиканьяглупцов

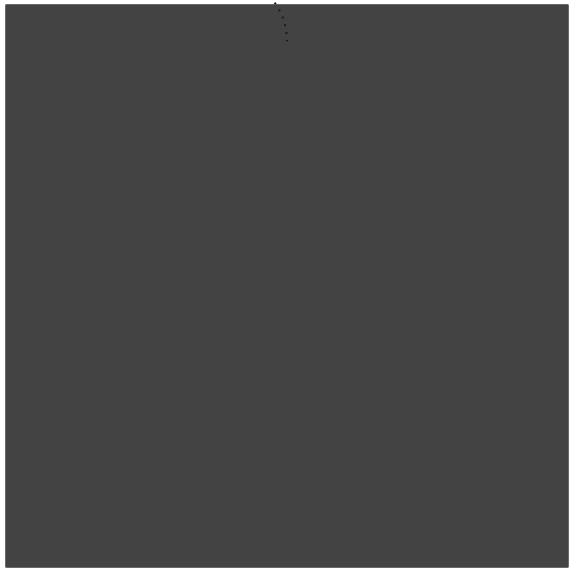
Код пишется для людей

Квадрат

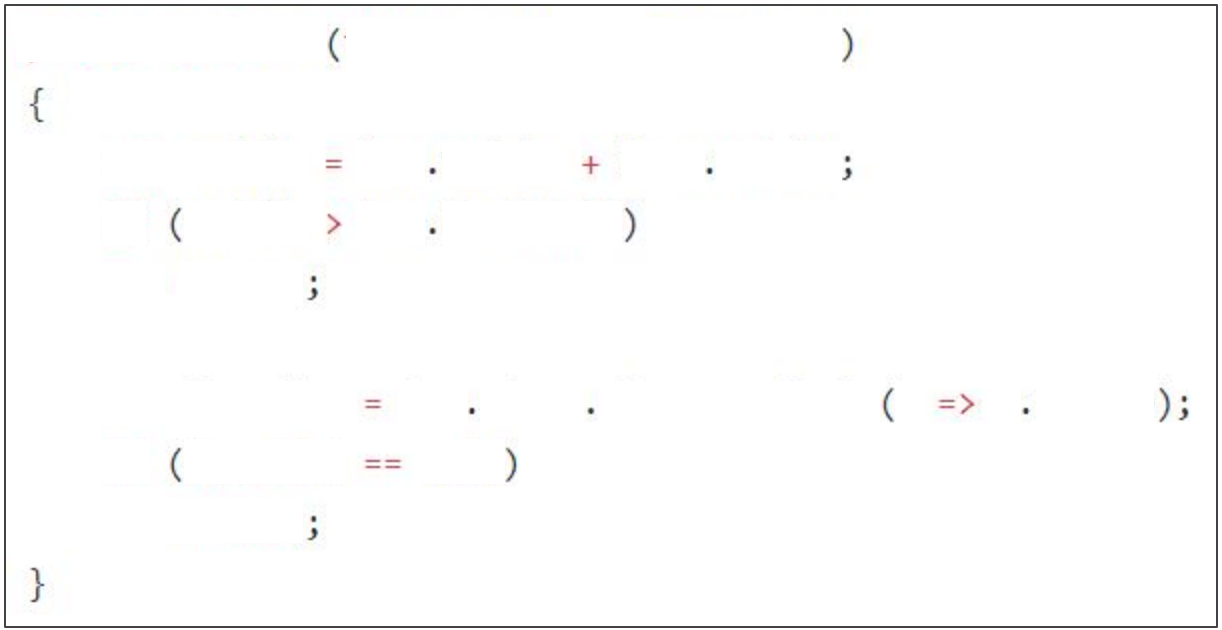


Слово

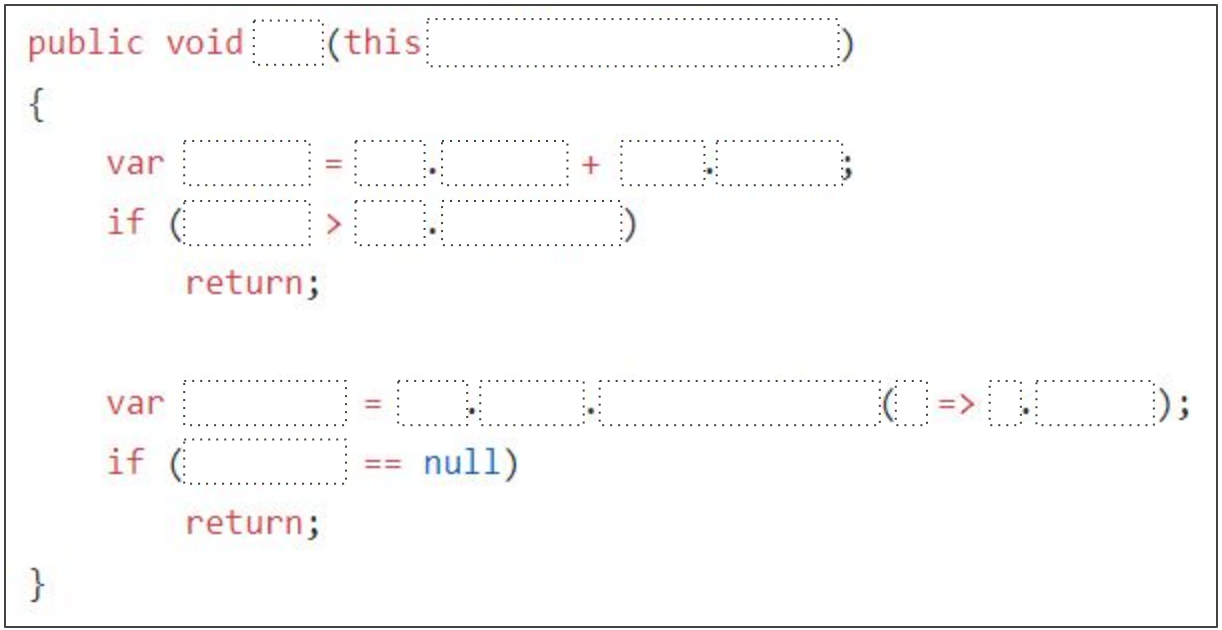
Квадрат



```
public void Put(this IBag bag, IItem item)
{
    var weight = bag.Weight + item.Weight;
    if (weight > bag.MaxWeight)
        return;

    var freeSlot = bag.Slots.FirstOrDefault(x => x.IsFree);
    if (freeSlot == null)
        return;
}
```



```
public void    (this                    )
{
    var        =    .       +    .      ;
    if (       >    .         )
        return;

    var          =    .     .             (  =>  .      );
    if (         == null)
        return;
}
```



```
public void Put(this IBag bag, IItem item)
{
    var weight = bag.Weight + item.Weight;
    if (weight > bag.MaxWeight)
        return;

    var freeSlot = bag.Slots.FirstOrDefault(x => x.IsFree);
    if (freeSlot == null)
        return;
}
```

```
public void Put(this IBag bag, IItem item)
{
    var weight = bag.Weight + item.Weight;
    if (weight > bag.MaxWeight)
        return;

    var freeSlot = bag.Slots.FirstOrDefault(x => x.IsFree);
    if (freeSlot == null)
        return;
}
```

Фрустрация — психическое состояние, возникающее в ситуации реальной или предполагаемой невозможности удовлетворения тех или иных потребностей, или, проще говоря, в ситуации несоответствия желаний имеющимся возможностям. Такая ситуация может рассматриваться как до некоторой степени травмирующая.

Фрустрация — психическое состояние, возникающее в ситуации реальной или предполагаемой невозможности удовлетворения тех или иных потребностей, или, проще говоря, в ситуации несоответствия желаний имеющимся возможностям. Такая ситуация может рассматриваться как до некоторой степени травмирующая.

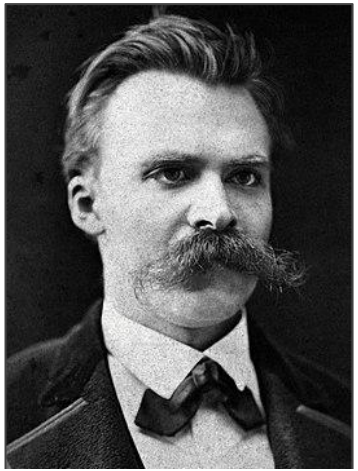
Фрустрация — что-то выходит не так, как хочется.

Фрустрация — что-то выходит не так, как хочется.

Хочется спать.

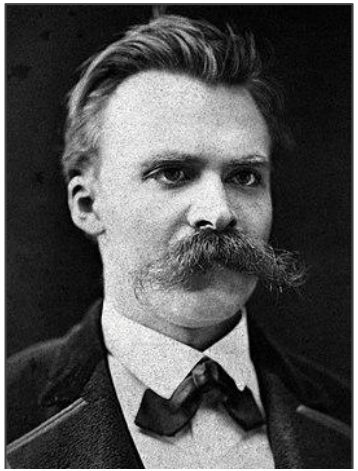
Тяжелеют веки.

“Сон колотит меня по глазам.”



Фридрих Ницше

“Сон колотит меня по глазам.”



Фридрих Ницше

“Сон колотит меня по глазам.”

В комнате было много свободного места.

“Между предметами обозначились огромные пустоты.”

Жан-Поль Сартр

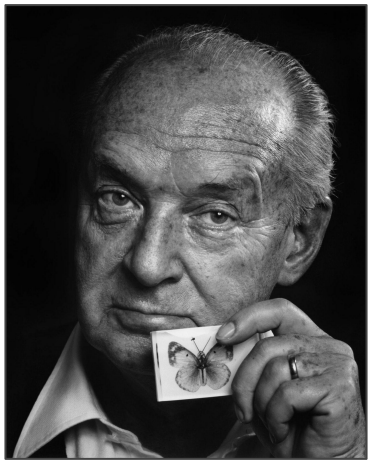
“Между предметами обозначились огромные пустоты.”

Жан-Поль Сартр

“Между предметами обозначились огромные пустоты.”

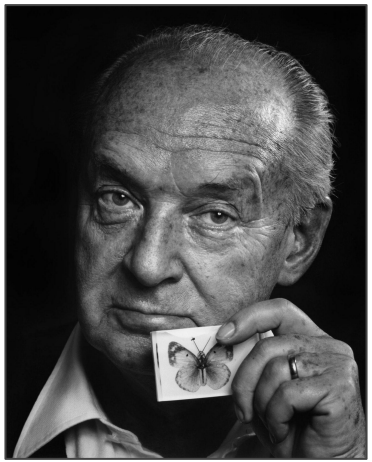
Я был обеспокоен.

“Я чувствовал, будто мое сердце бьётся всюду одновременно”



Владимир Набоков

“Я чувствовал, будто мое сердце бьётся всюду одновременно”



Владимир Набоков

“Я чувствовал, будто мое сердце бьётся всюду одновременно”

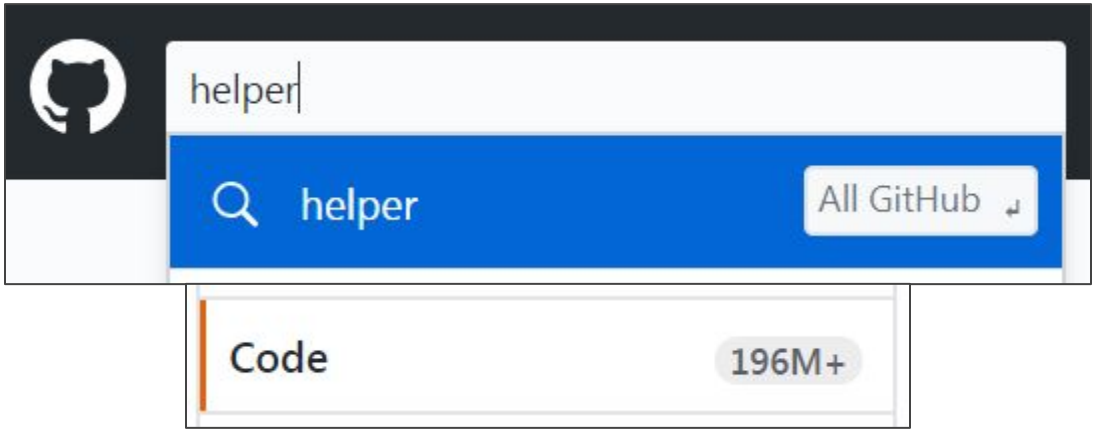
```
public interface IEmailManager
{
    // ...
}
```

```
public interface IMailbox
{
    // ...
}
```

```
public interface IMailbox
{
    // ...
}
```

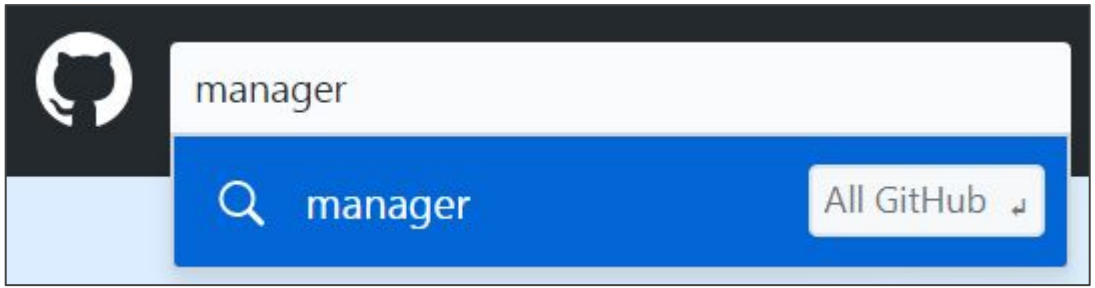
```
public interface IMailbox
{
    // ...
}
```

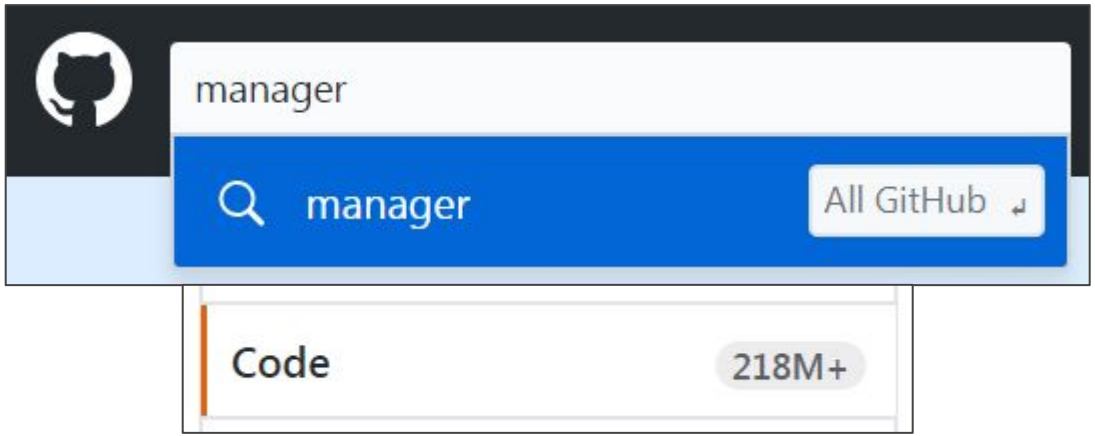
?



```
DEF_HELPER_2(raise_exception_err, void, i32, i32)
DEF_HELPER_1(raise_exception, void, i32)
DEF_HELPER_3(tw, void, tl, tl, i32)
#if defined(TARGET_PPC64)
DEF_HELPER_3(td, void, tl, tl, i32)
```

```
ExcelHelper excelHelper = new ExcelHelper();
excelHelper.Create();
excelHelper.CreateSheet("条码导出");
excelHelper.SetCellValue(0, 0, "条码");
excelHelper.SetCellValue(0, 1, "工单号");
excelHelper.SetCellValue(0, 2, "订单类型");
excelHelper.SetCellValue(0, 3, "物料编号");
```





```
virtual OpenViBE::Kernel::IAlgorithmManager& getAlgorithmManager(void) const;
virtual OpenViBE::Kernel::IConfigurationManager& getConfigurationManager(void) const;
virtual OpenViBE::Kernel::IKernelObjectFactory& getKernelObjectFactory(void) const;
virtual OpenViBE::Kernel::IPlayerManager& getPlayerManager(void) const;
virtual OpenViBE::Kernel::IPluginManager& getPluginManager(void) const;
virtual OpenViBE::Kernel::IScenarioManager& getScenarioManager(void) const;
virtual OpenViBE::Kernel::ITypeManager& getTypeManager(void) const;
virtual OpenViBE::Kernel::ILogManager& getLogManager(void) const;
virtual OpenViBE::Kernel::IVisualisationManager& getVisualisationManager(void) const;
```

```
public abstract class IntRangeManager
```

```
/**
 * Clients can enable reception of SMS-CB messages for specific ranges of
 * message identifiers (channels). This class keeps track of the currently
 * enabled message identifiers and calls abstract methods to update the
 * radio when the range of enabled message identifiers changes.
 *
 * An update is a call to {@link #startUpdate} followed by zero or more
 * calls to {@link #addRange} followed by a call to {@link #finishUpdate}.
 * Calls to {@link #enableRange} and {@link #disableRange} will perform
 * an incremental update operation if the enabled ranges have changed.
 * A full update operation (i.e. after a radio reset) can be performed
 * by a call to {@link #updateRanges}.
 *
 * Clients are identified by String (the name associated with the User ID
 * of the caller) so that a call to remove a range can be mapped to the
 * client that enabled that range (or else rejected).
 */
public abstract class IntRangeManager
```

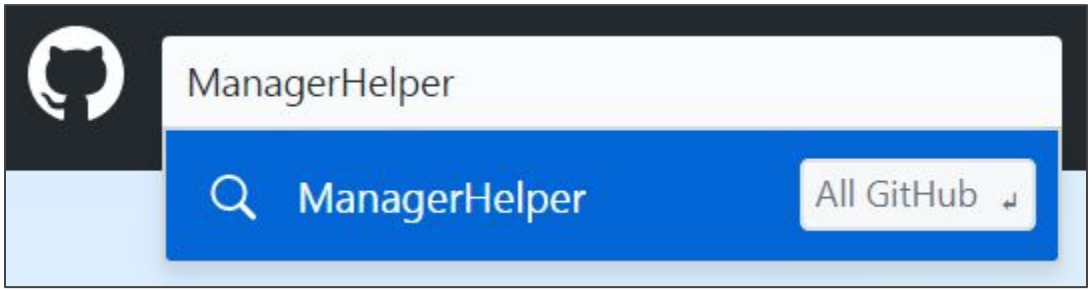
```
@Component
@Transactional
public class AaaUpdateTimeManager extends BaseManager<NeoString,java.lang.Integer>{
```

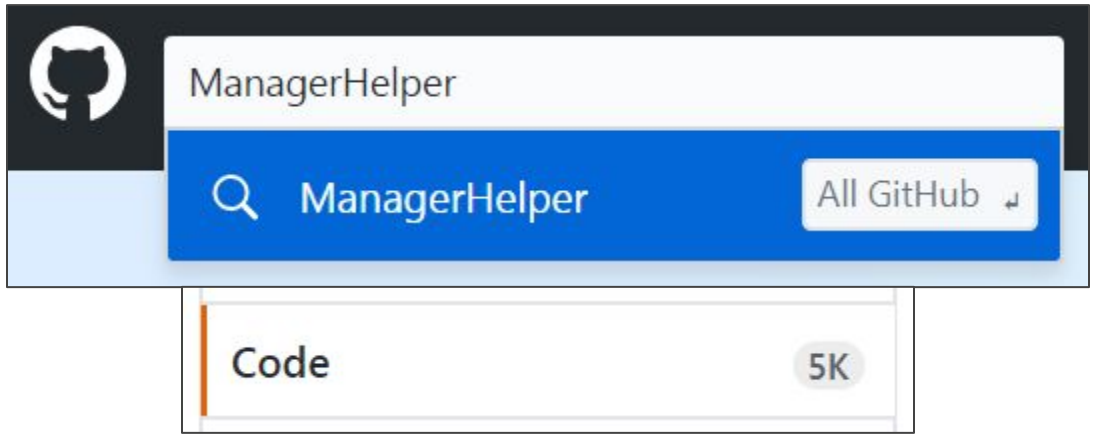
```
@Component
@Transactional
public class AaaUpdateTimeManager extends BaseManager<NeoString,java.lang.Integer>{
```

```java
@Component
@Transactional
public class AaaUpdateTimeManager extends BaseManager<NeoString,java.lang.Integer>{
```

```
@Component
@Transactional
public class AaaUpdateTimeManager extends BaseManager<NeoString,java.lang.Integer>{
```

```java
@Component
@Transactional
public class AaaUpdateTimeManager extends BaseManager<NeoString,java.lang.Integer>{

        //设置需要的Dao
        private E288PowerManager e288PowerManager;
        private ELineBasicinfoManager eLineBasicinfoManager;
        private EMaxminPowerManager eMaxminPowerManager;
        private ENowEnergyManager eNowEnergyManager;
        private EPowerCountManager ePowerCountManager;
        private EPowerDayManager ePowerDayManager;
        private EPowerHourManager ePowerHourManager;
        private ERealtimePowerManager eRealtimePowerManager;
```





```
namespace
{
    public class ManagerHelper
    {
        // ...

    }
}
```

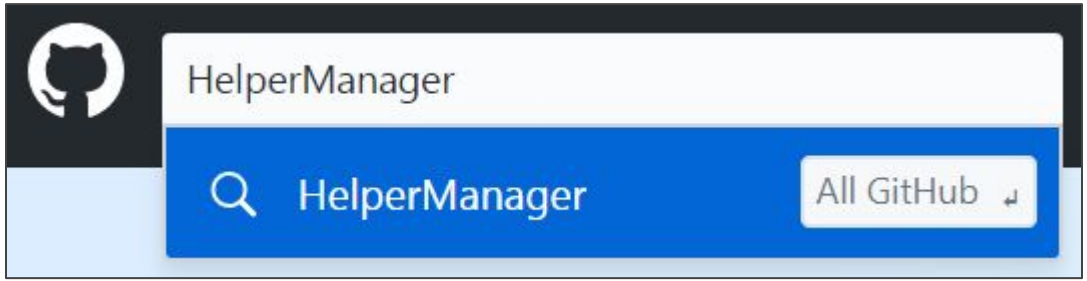
```
namespace
{
    public class ManagerHelper
    {
        // ...

        public ManagerHelper(String address)
        {
            this.monitor = new Monitor(address);
            this.initialTime = 0;
            this.interfacesTotalNumber = 0;
            this.bulkOIDs = new OID[6];
            this.ifNumOIDs = new OID[2];
            this.registry = new IfTableRegistry();
        }
    }
}
```

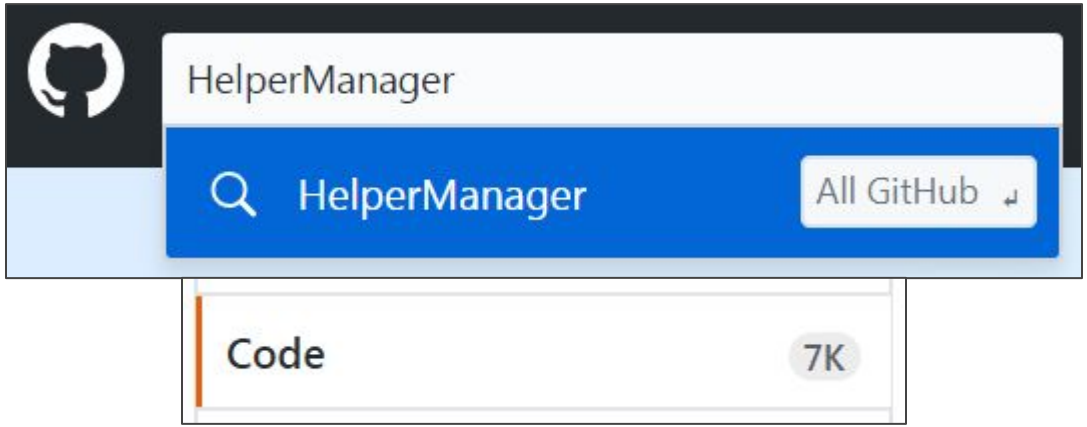
```
namespace Company.BusinessLayer.Threads
{
    public class ManagerHelper
    {
        // ...

        public ManagerHelper(String address)
        {
            this.monitor = new Monitor(address);
            this.initialTime = 0;
            this.interfacesTotalNumber = 0;
            this.bulkOIDs = new OID[6];
            this.ifNumOIDs = new OID[2];
            this.registry = new IfTableRegistry();
        }
    }
}
```

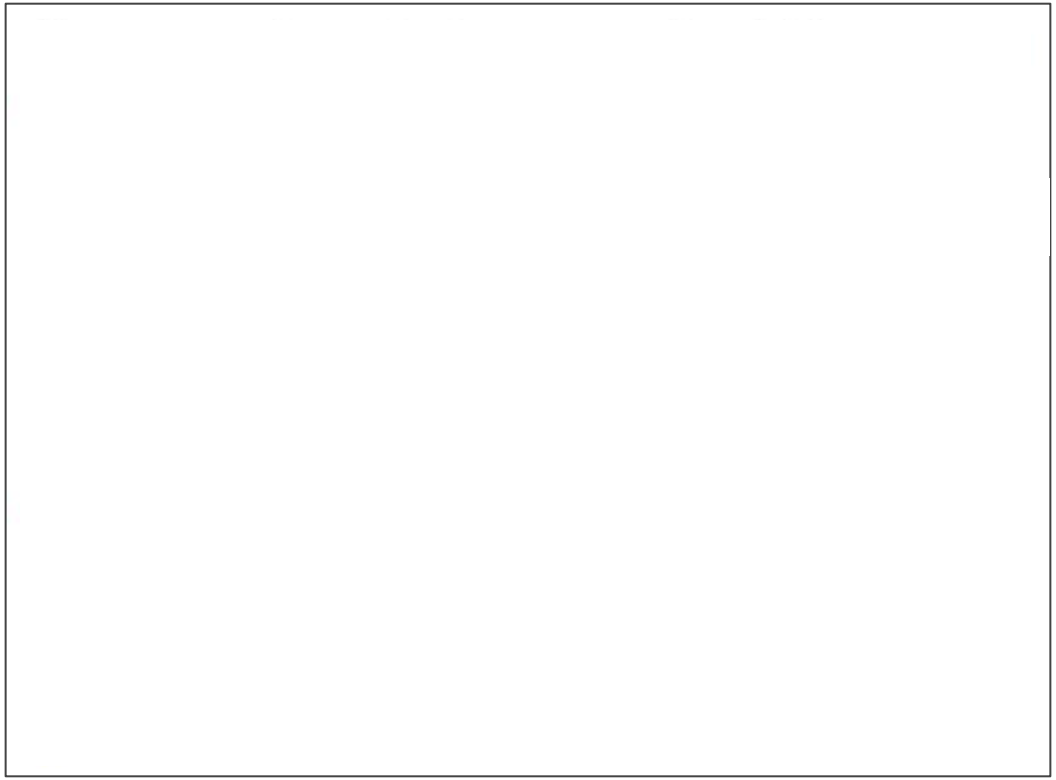
```
class ManagerHelper {
```

```
/**
 * ManagerHelper
 * This class is intented to facilitate getting current status information to
 * the ServerManager class.
 * @author Tim Vidas
 * @author Chris Eagle
 * @version 0.4.0, August 2012
 */

class ManagerHelper {
```

```
/**
 * ManagerHelper
 * This class is intented to facilitate getting current status information to
 * the ServerManager class.
 * @author Tim Vidas
 * @author Chris Eagle
 * @version 0.4.0, August 2012
 */

class ManagerHelper {
```

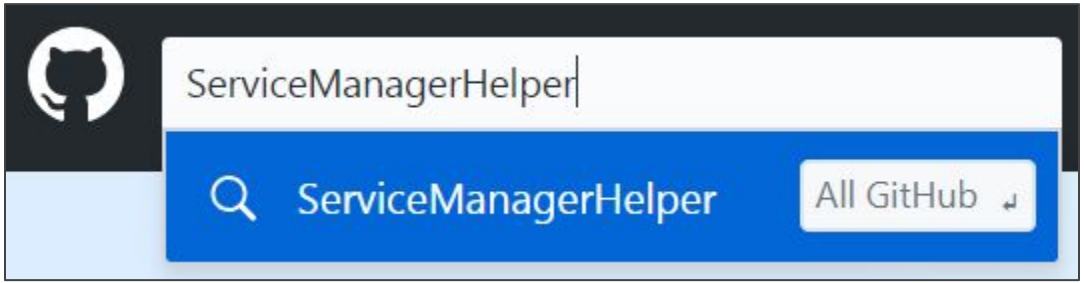


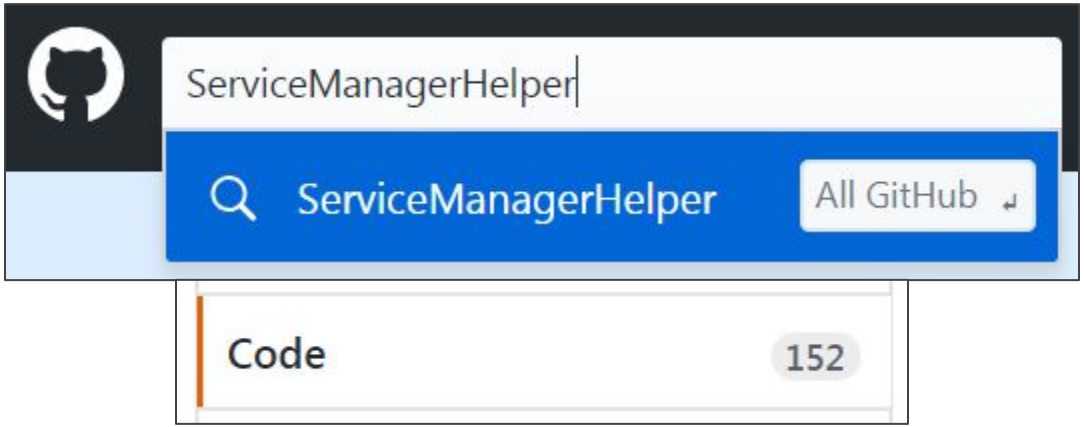




```
// Gets run when end-of-game scoreboard first appears
void HelperManager::onGameOver()
{
   if(mHelperStack.contains(&mTeamShuffleHelper))
      exitHelper(&mTeamShuffleHelper);
}
```

```
// Gets run when end-of-game scoreboard first appears
void HelperManager::onGameOver()
{
   if(mHelperStack.contains(&mTeamShuffleHelper))
      exitHelper(&mTeamShuffleHelper);
}


void HelperManager::onTextInput(char ascii)
{
   if(mHelperStack.size() > 0)
      mHelperStack.last()->onTextInput(ascii);
}
```





```php
/**
 * Class ServiceManagerHelper
 *
 * @package Phpro\SmartCrud\Console\Helper
 */
class ServiceManagerHelper extends Helper
{
    protected $serviceManager;
```

```php
/**
 * Class ServiceManagerHelper
 *
 * @package Phpro\SmartCrud\Console\Helper
 */
class ServiceManagerHelper extends Helper
{
    protected $serviceManager;
```

```
class ServiceManagerHelper
{
    private static final String TAG = "nf_job_svcmgr_helper";
    private final ManagerStatusListener mManagerStatusListener;
    private ServiceManager mServiceManager;
    private final ServiceManagerHelper$ServiceManagerHelperListener mServiceManagerHelperListener;
    private ServiceManagerHelper$ServiceManagerState mState;
```

```
class ServiceManagerHelper
{
    private static final String TAG = "nf_job_svcmgr_helper";
    private final ManagerStatusListener mManagerStatusListener;
    private ServiceManager mServiceManager;
    private final ServiceManagerHelper$ServiceManagerHelperListener mServiceManagerHelperListener;
    private ServiceManagerHelper$ServiceManagerState mState;
```

```
class ServiceManagerHelper
{
    private static final String TAG = "nf_job_svcmgr_helper";
    private final ManagerStatusListener mManagerStatusListener;
    private ServiceManager mServiceManager;
    private final ServiceManagerHelper$ServiceManagerHelperListener mServiceManagerHelperListener;
    private ServiceManagerHelper$ServiceManagerState mState;
```

Код пишется для людей

?

```
namespace Namespace
{
    public class Class
    {
        public FieldType Field;
        public PropertyType Property => someValue;

        public MethodReturnType Method(ArgType arg1, ArgType arg2)
        {
            var localVariable = arg1 + arg2;
        }
    }
}
```

```
namespace Namespace
{
    public class Class
    {
        public FieldType Field;
        public PropertyType Property => someValue;

        public MethodReturnType Method(ArgType arg1, ArgType arg2)
        {
            var localVariable = arg1 + arg2;
        }
    }
}
```

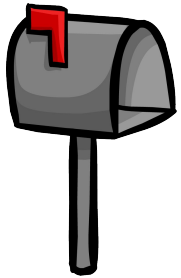
```
public interface IEmailManager
{
    // ...
}
```

```
public interface IEmailManager
{
    // ...
}
```

```
public interface IMailbox
{
    // ...
}
```

```
public interface IEmailManager
{
    // ...
}
```

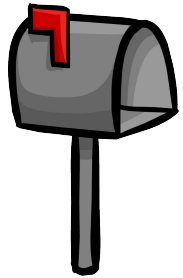
```
public interface IMailbox
{
    // ...
}
```



```
public interface IEmailManager
{
    // ...
}
```

```
public interface IMailbox
{
    // ...
}
```

Про имена объектов:

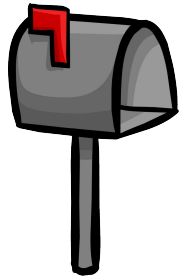


```
ilManager
```

```
public interface IMailbox
{
    // ...
}
```

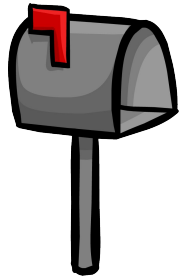
Про имена объектов:

1. Сильные существительные:



```
ilManager
```

```
public interface IMailbox
{
    // ...
}
```

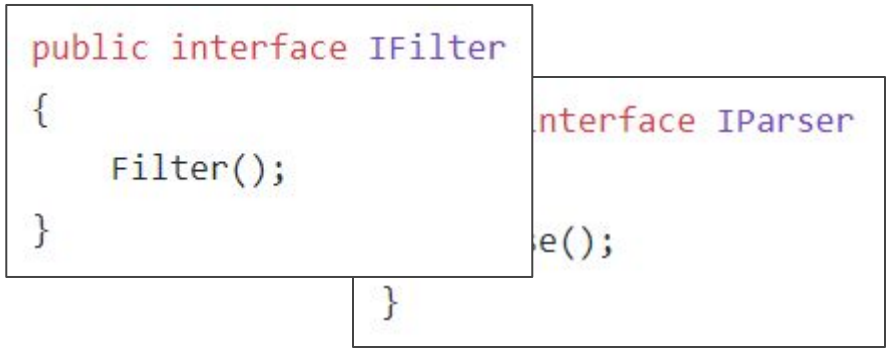


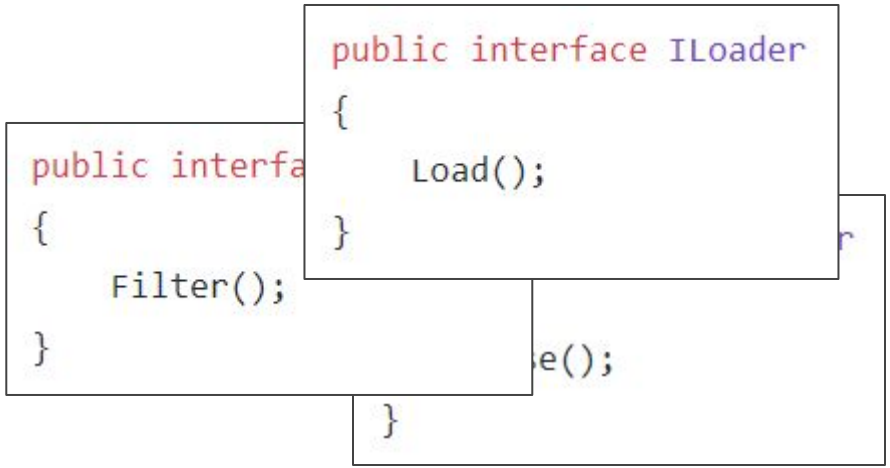
```
public interface IEmailManager
{
    // ...
}
```

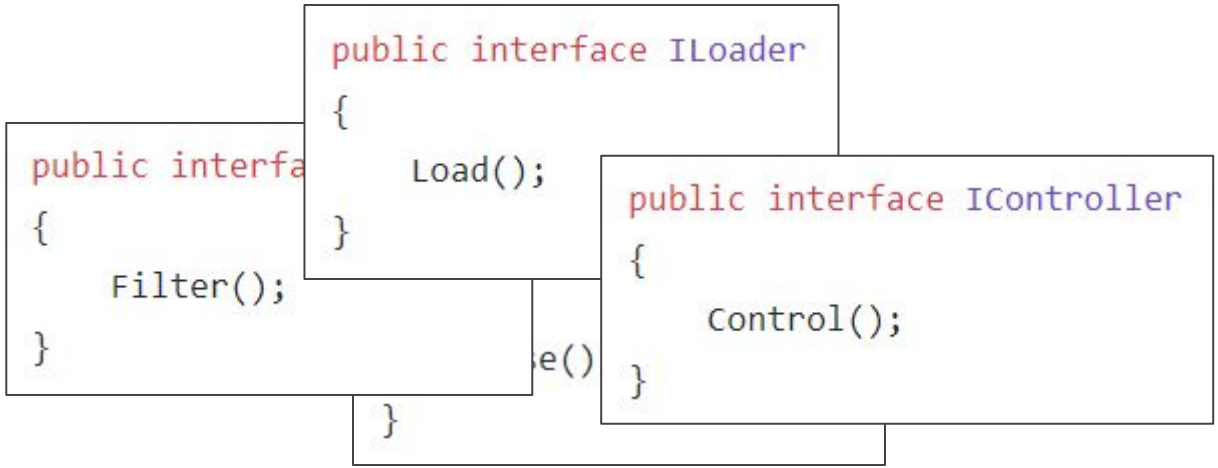
```
public interface IMailbox
{
    // ...
}
```

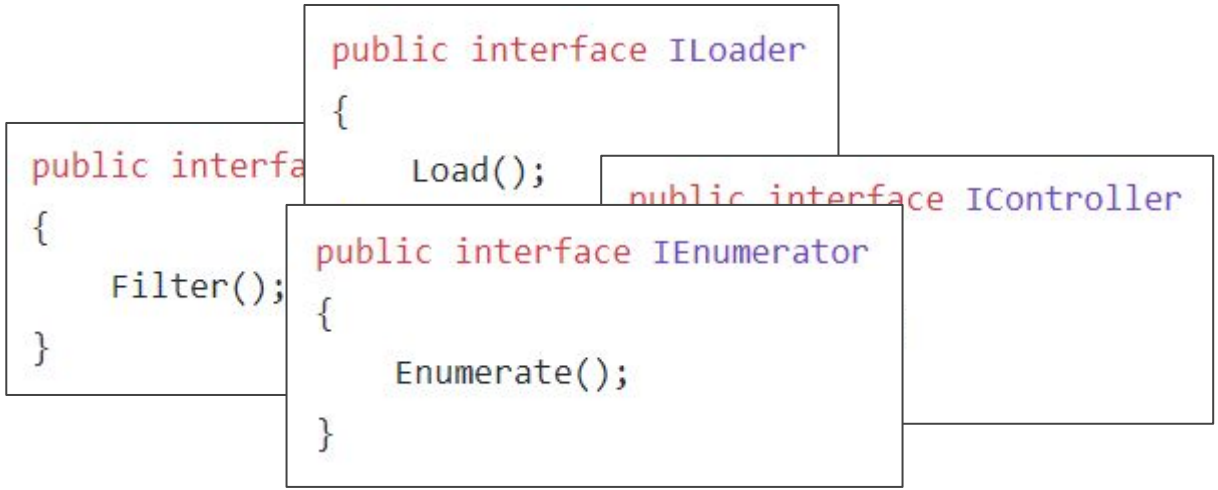
```
public interface IParser
{
    // ...
}
```

```
public interface IParser
{
    Parse();
}
```



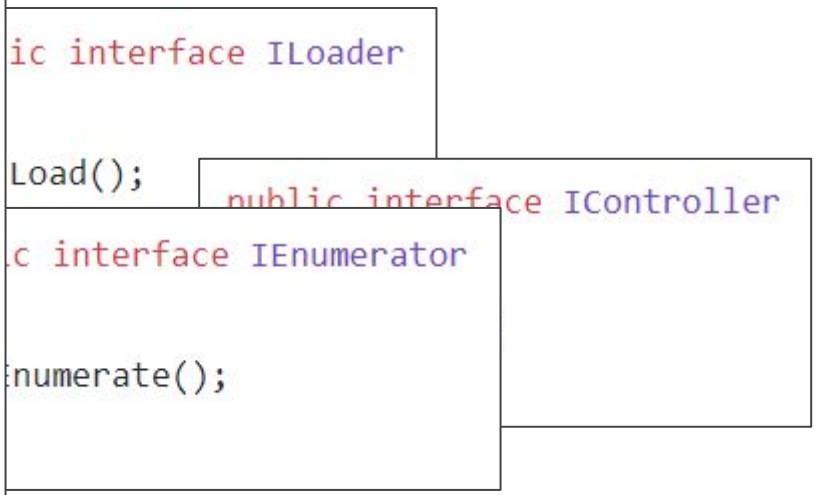






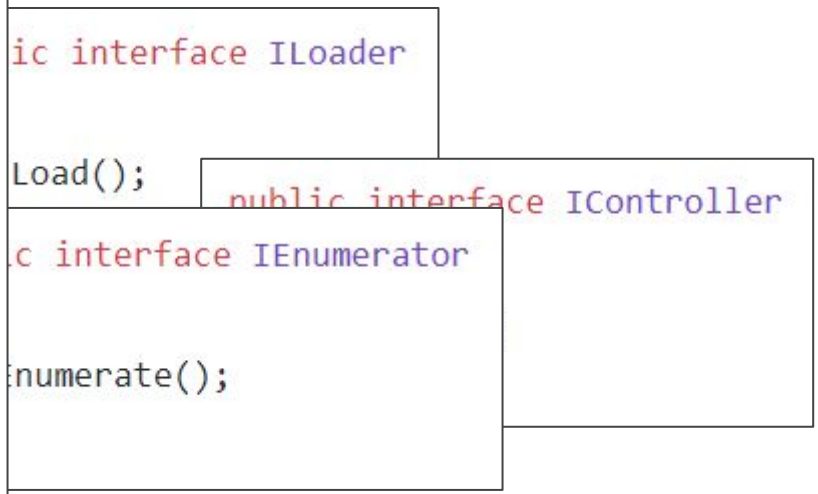
Про имена объектов:

1. Сильные существительные:



Про имена объектов:

1. Сильные существительные:
   - не отглагольные;



```
public class IHelper
{
    // ???
}
```

```
public class IHelper
{
    // ???
}
```

```
public class IManager
{
    // ???
}
```

```
public class IHelper
{
    // ???
}
```

```
public class IManager
{
    // ???
}
```

```
public class IService
{
    // ???
}
```

```
public interface IEmailManager
{
    // ...
}
```

```
public interface IMailbox
{
    // ...
}
```

```
public interface IEmailManager
{
    // ...
}
```

```
public interface IMailbox
{
    // ...
}
```

Про имена объектов:

1. Сильные существительные:
   - не отглагольные;

```
ilManager
```

```
public interface IMailbox
{
    // ...
}
```

Про имена объектов:

1. Сильные существительные:
   - не отглагольные;
   - не составные.

```
ilManager
```

```
public interface IMailbox
{
    // ...
}
```

Про имена объектов:

1. Сильные существительные:
   - не отглагольные;
   - односоставные.

```
ilManager
```

```
public interface IMailbox
{
    // ...
}
```

Про имена объектов:

1. Сильные существительные:
   - не отглагольные;
   - не составные.

```
ilManager
```

```
public interface IMailbox
{
    // ...
}
```

Про имена объектов:

1. Сильные существительные:
   - не отглагольные;
   - не составные.

```
ilManager
```

```
public interface IMailbox
{
    // ...
}
```

```
public interface IEmailManager
{
    // ...
}
```

```
public interface IMailbox
{
    // ...
}
```

```
public interface IGitService
{
    // ...
}
```

```
public interface IRepository
{
    // ...
}
```

```
public interface IGitService
{
    // ...
}
```

```
public interface IRepository
{
    // ...
}
```

```
public interface IGitService
{
    // ...
}
```

```
public interface ICommit
{
    // ...
}
```

```
public interface IGitService
{
    // ...
}
```

```
public interface IRepository
{
    // ...
}
```

```
public interface ICommit
{
    // ...
}
```

```
public interface IBranch
{
    // ...
}
```

Про имена объектов:

1. Сильные существительные:
   - не отглагольные;
   - не составные.

```
public interface IRepository
{
    // ...
}
```

```
public interface ICommit
{
    // ...
}
```

```
public interface IBranch
{
    // ...
}
```

Про имена объектов:

1. Сильные существительные:
   - не отглагольные;
   - не составные.
2. Из предметной области.

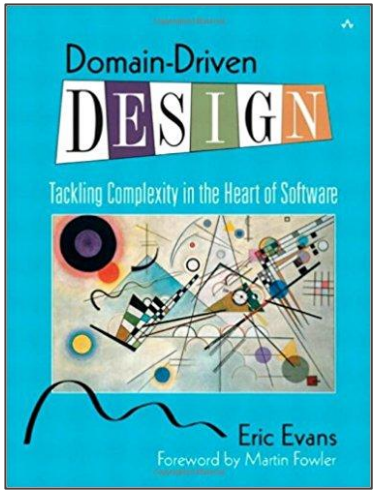
```
public interface IRepository
{
    // ...
}
```

```
public interface ICommit
{
    // ...
}
```

```
public interface IBranch
{
    // ...
}
```

Про имена объектов:

1. Сильные существительные:
   - не отглагольные;
   - не составные.
2. Из предметной области.



```
public interface IRepository
{
    // ...
}
```

```
public interface ICommit
{
    // ...
}
```

```
public interface IBranch
{
    // ...
}
```

```
public interface IGitService
{
    // ...
}
```

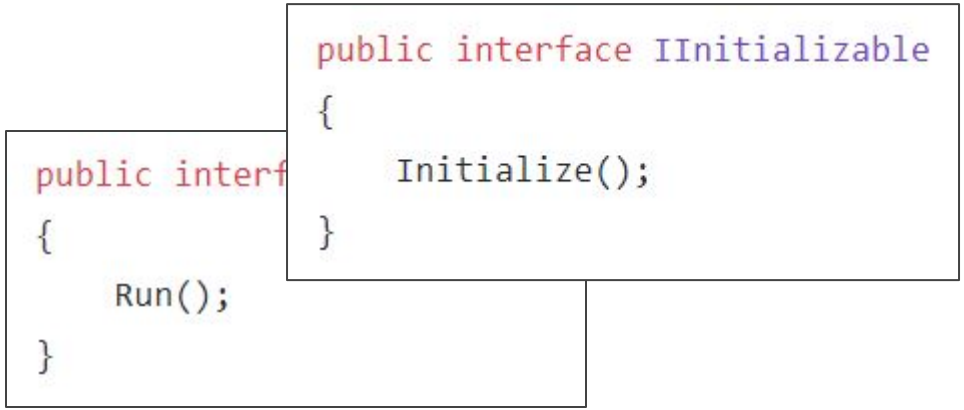
```
public interface IRepository
{
    // ...
}
```

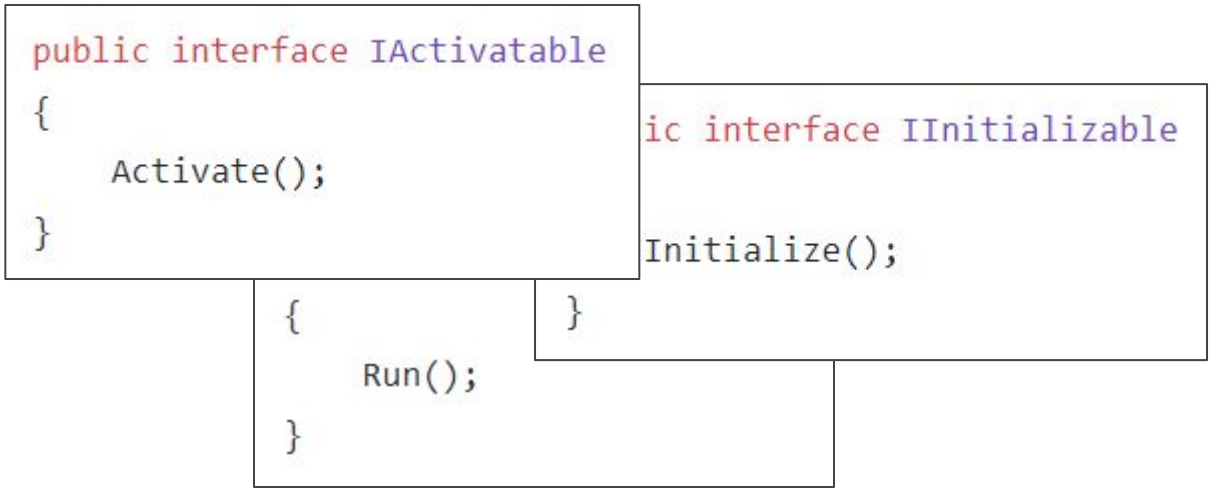
```
public interface ICommit
{
    // ...
}
```

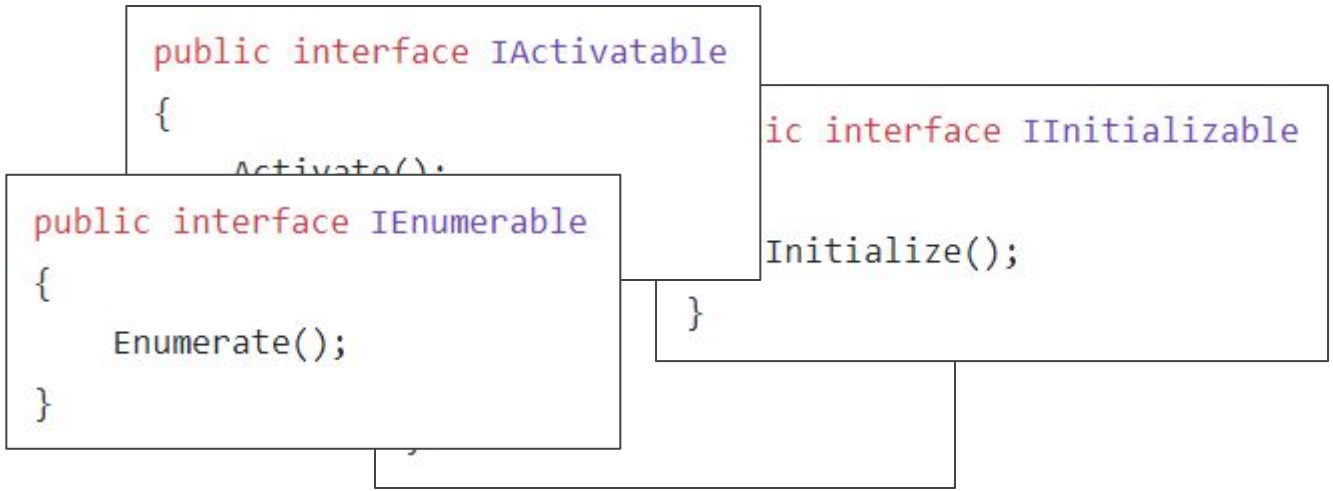
```
public interface IBranch
{
    // ...
}
```

```
public interface IRunner
{
    Run();
}
```

```
public interface IRunnable
{
    Run();
}
```

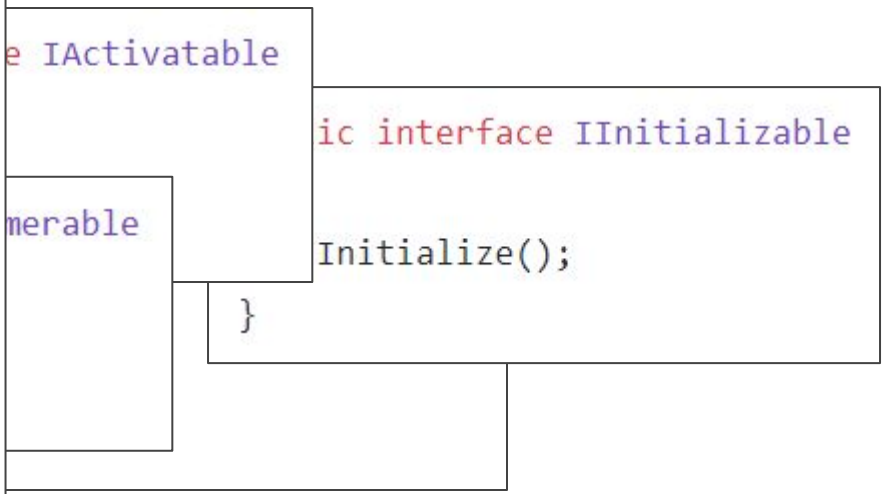






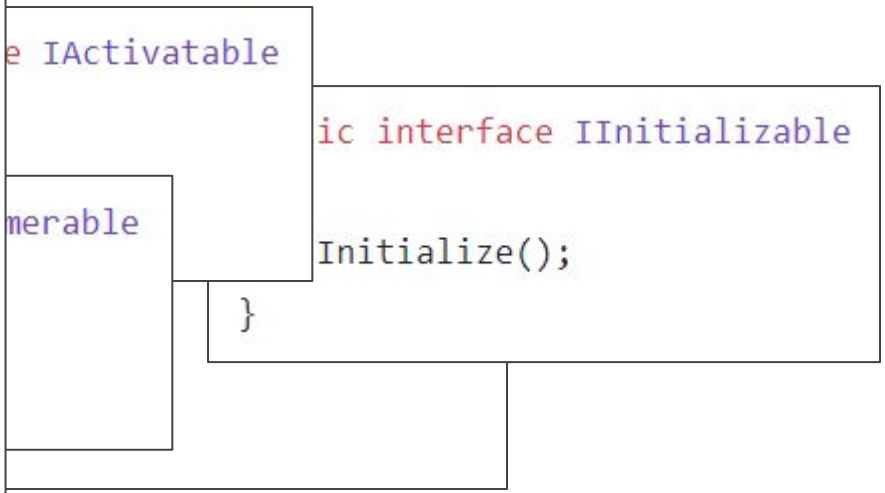
Про имена объектов:

1. Сильные существительные:
   - не отглагольные;
   - не составные.
2. Из предметной области.



Про имена объектов:

1. Сильные существительные:
   - не отглагольные;
   - не составные.
2. Из предметной области.
3. Не отглагольные прилагательные.

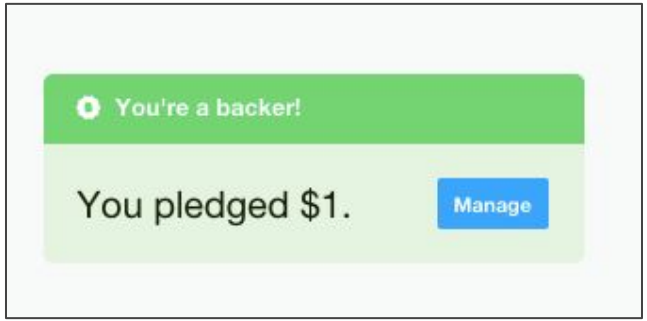


```
// Обещание.
public interface IPromise
{
  // ...
}
```

```
// Обещание.
public interface IPledge
{
   // ...
}
```


```
// Обещание.
public interface IPledge
{
   // ...
}
```


```
// Обещание.
public interface IPledge
{
   // ...
}
```



```
// Обещание.
public interface IPledge
{
   // ...
}
```

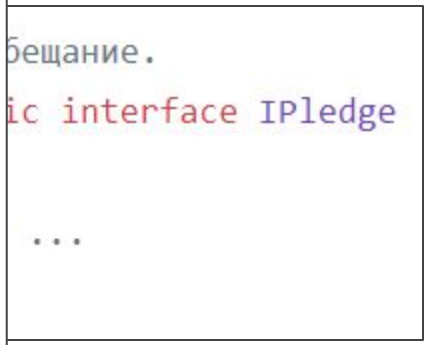
Про имена объектов:

1. Сильные существительные:
   - не отглагольные;
   - не составные.
2. Из предметной области.
3. Не отглагольные прилагательные.

```
бещание.
ic interface IPledge

...
```

Про имена объектов:

1. Сильные существительные:
   - не отглагольные;
   - не составные.
2. Из предметной области.
3. Не отглагольные прилагательные.
4. Не вычурные слова.



Про имена объектов:

1. Сильные существительные:
   - не отглагольные;
   - не составные.
2. Из предметной области.
3. Не отглагольные прилагательные.
4. Не вычурные слова.

```
бещание.
ic interface IPledge

...
```

Про имена объектов:

1. Сильные существительные:
   - не отглагольные;
   - не составные.
2. Из предметной области.
3. Не отглагольные прилагательные.
4. Простые слова.

```
бещание.
ic interface IPledge

...
```

```
// Обещание.
public interface IPledge
{
   // ...
}
```

```
namespace Namespace
{
    public class Class
    {
        public FieldType Field;
        public PropertyType Property => someValue;

        public MethodReturnType Method(ArgType arg1, ArgType arg2)
        {
            var localVariable = arg1 + arg2;
        }
    }
}
```

```
namespace Namespace
{
    public class Class
    {
        public FieldType Field;
        public PropertyType Property => someValue;

        public MethodReturnType Method(ArgType arg1, ArgType arg2)
        {
            var localVariable = arg1 + arg2;
        }
    }
}
```

1

2

```
public interface IUserRepository
{
    IEnumerable<IUser> GetUsers();
}
```

```
public interface IUserRepository
{
    IEnumerable<IUser> Users();
}
```

```
public interface IUserRepository
{
    IEnumerable<IUser> TakeUsers();
}
```

```
public interface IUserRepository
{
    IEnumerable<IUser> FetchUsers();
}
```

```
public interface IUserRepository
{
    IEnumerable<IUser> LoadUsers();
}
```

```
public interface IUserRepository
{
    IEnumerable<IUser> Users();
}
```

Про имена методов:

```
interface IUserRepository

numerable<IUser> Users();
```

Про имена методов:

1. Запросы — существительные:

```
interface IUserRepository

numerable<IUser> Users();
```

Про имена методов:

1. Запросы — не глаголы:

```
interface IUserRepository

numerable<IUser> Users();
```

Про имена методов:

1. Запросы — не глаголы:
   - существительные;

```
interface IUserRepository

numerable<IUser> Users();
```

```
public interface IUserRepository
{
    IEnumerable<IUser> Users();
}
```

```
public interface IUserRepository
{
    IEnumerable<IUser> Users();

    IUser User(int id);
    IUser User(string name);
}
```

```
users.User("Kierkegaard");
```

```
public interface IUserRepository
{
    IEnumerable<IUser> Users();

    IUser User(int id);
    IUser User(string name);
}
```

```
users.User("Kierkegaard");
```

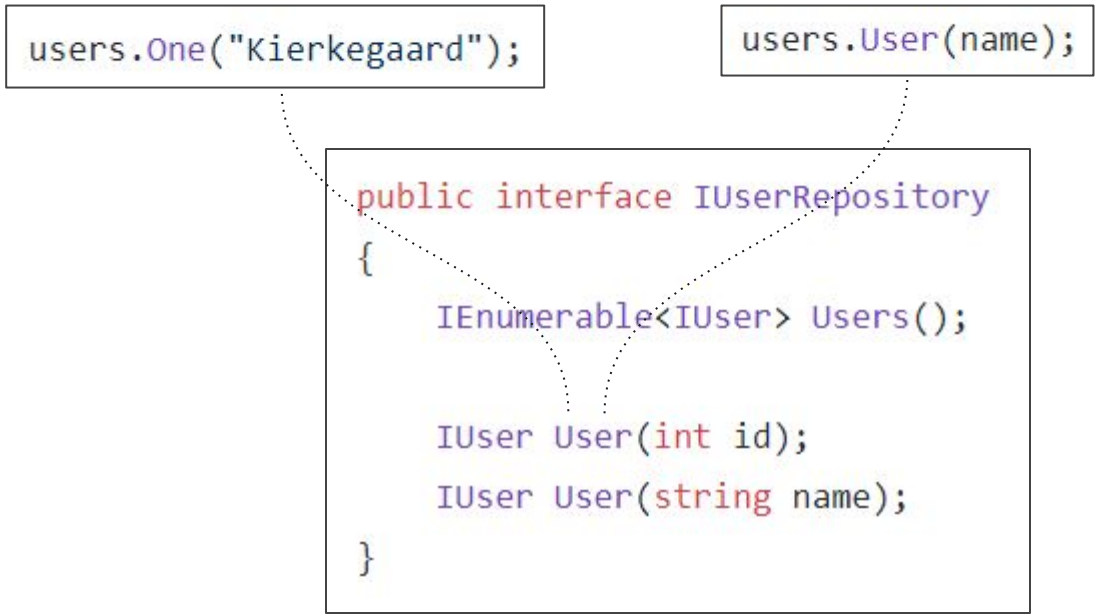
```
public interface IUserRepository
{
    IEnumerable<IUser> Users();

    IUser User(int id);
    IUser User(string name);
}
```

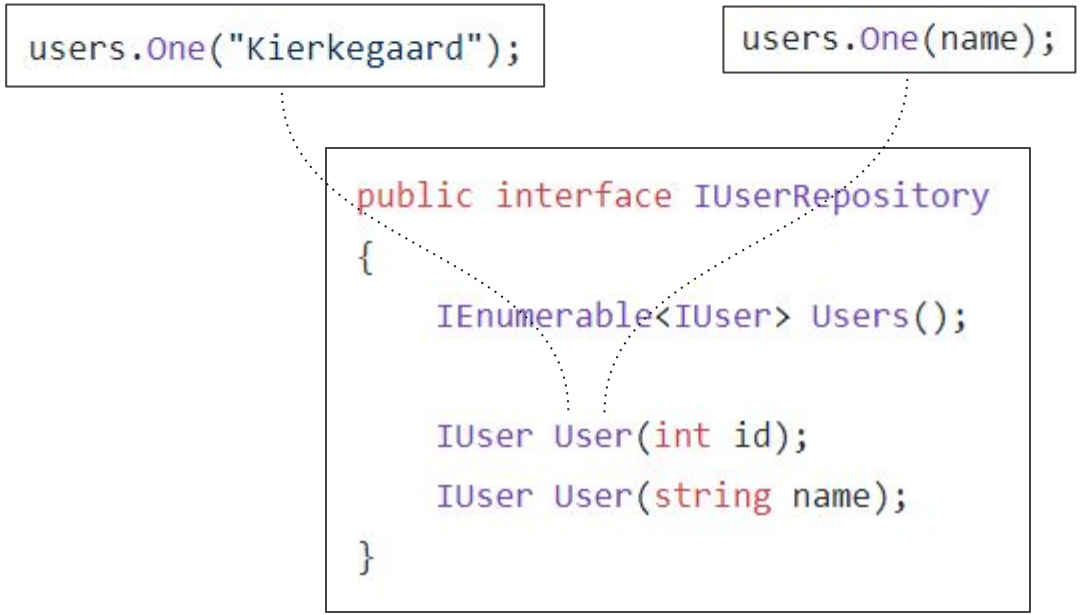
```
users.OneWithName("Kierkegaard");
```

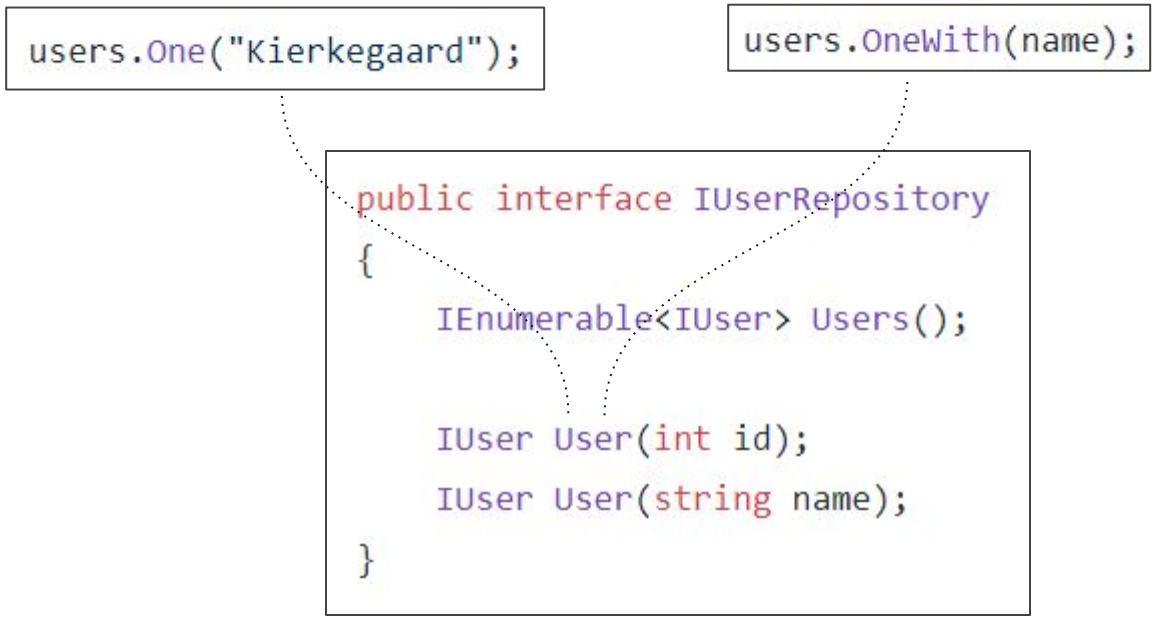
```
public interface IUserRepository
{
    IEnumerable<IUser> Users();

    IUser User(int id);
    IUser User(string name);
}
```

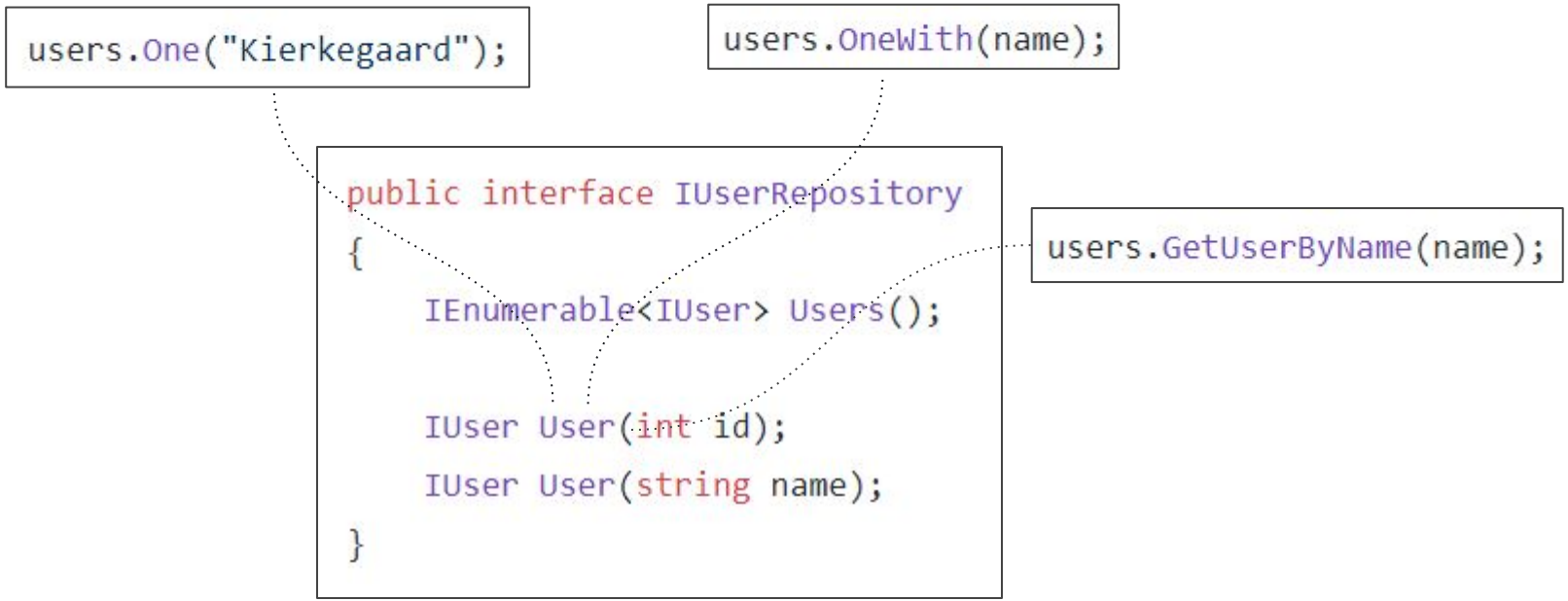
```
users.One("Kierkegaard");
```

```
public interface IUserRepository
{
    IEnumerable<IUser> Users();

    IUser User(int id);
    IUser User(string name);
}
```



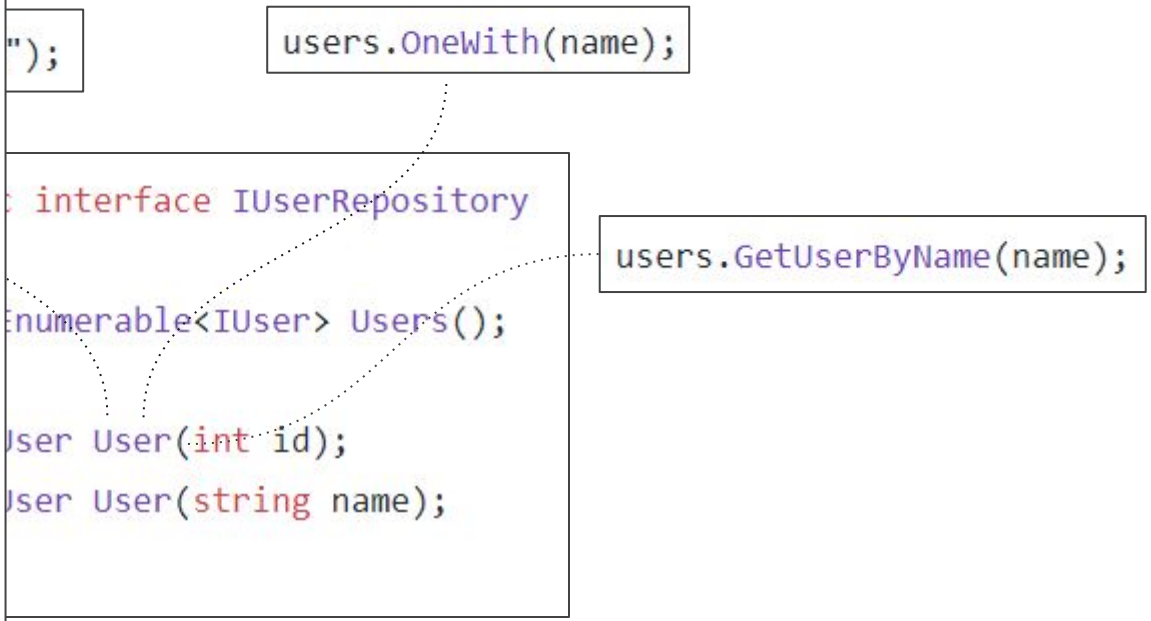






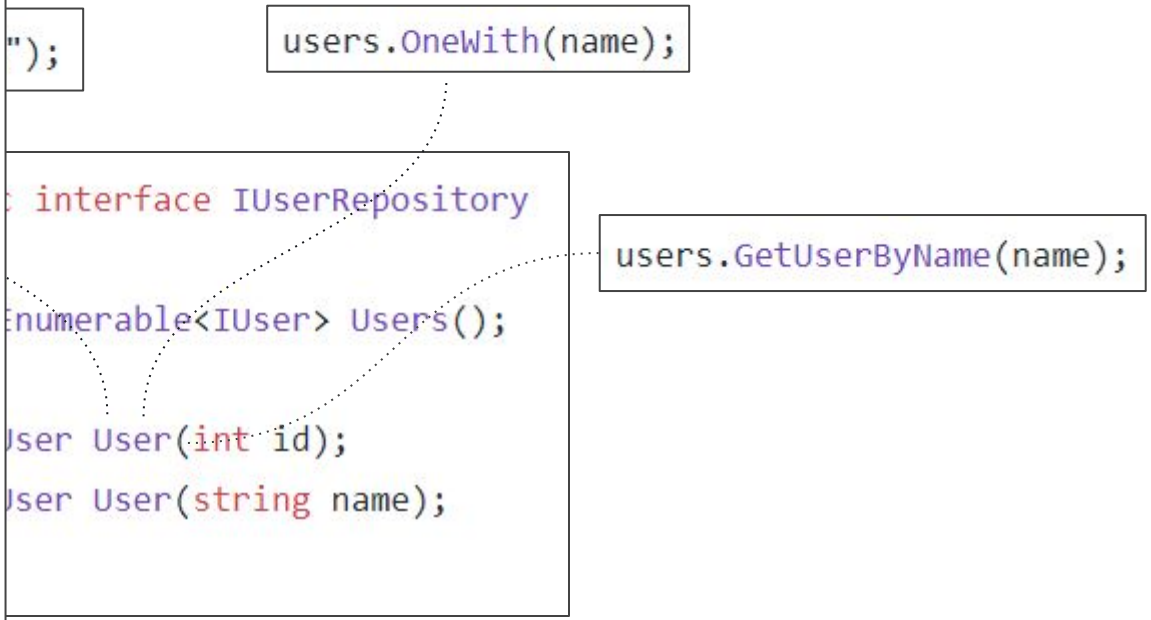
Про имена методов:

1. Запросы — не глаголы:
   - существительные;



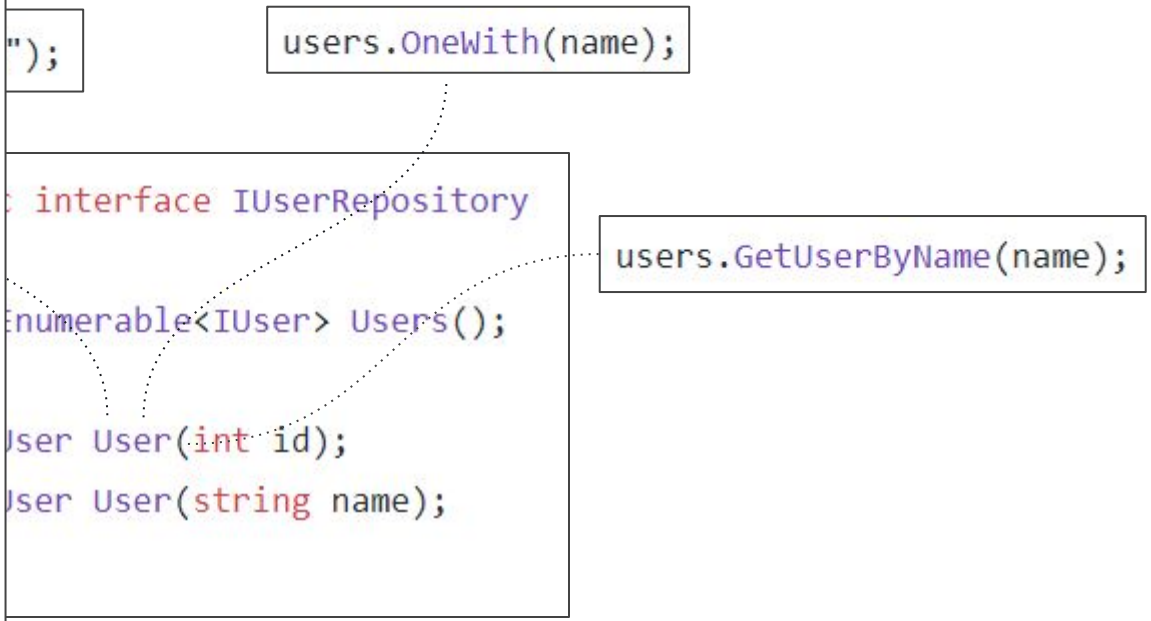
Про имена методов:

1. Запросы — не глаголы:
   - существительные;
   - местоимения;



Про имена методов:

1. Запросы — не глаголы:
   - существительные;
   - местоимения;
   - и т.д.



```
public interface IBank
{
    void Purchase(int productId);
}
```

Про имена методов:

1. Запросы — не глаголы:
   - существительные;
   - местоимения;
   - и т.д.

```
 interface IBank

id Purchase(int productId);
```

Про имена методов:

1. Запросы — не глаголы:
   - существительные;
   - местоимения;
   - и т.д.
2. Действия — глаголы:

```
interface IBank

id Purchase(int productId);
```

```
public interface IBank
{
    void Purchase(int productId);
}
```

```
public interface IBank
{
    void PerformPurchase(int productId);
}
```

Совершить покупку

```
public interface IBank
{
    void PerformPurchase(int productId);
}
```

Совершить покупку → купить

```
public interface IBank
{
    void PerformPurchase(int productId);
}
```

Совершить покупку → купить
Произвести атаку

```
public interface IBank
{
    void PerformPurchase(int productId);
}
```

Совершить покупку → купить
Произвести атаку → атаковать

```
public interface IBank
{
    void PerformPurchase(int productId);
}
```

Совершить покупку → купить
Произвести атаку → атаковать
Одержать победу

```
public interface IBank
{
    void PerformPurchase(int productId);
}
```

Совершить покупку → купить
Произвести атаку → атаковать
Одержать победу → победить

```
public interface IBank
{
    void PerformPurchase(int productId);
}
```

Совершить покупку → купить
Произвести атаку → атаковать
Одержать победу → победить
Допустить ошибку

```
public interface IBank
{
    void PerformPurchase(int productId);
}
```

Совершить покупку → купить
Произвести атаку → атаковать
Одержать победу → победить
Допустить ошибку → ошибиться

```
public interface IBank
{
    void PerformPurchase(int productId);
}
```

Совершить покупку → купить
Произвести атаку → атаковать
Одержать победу → победить
Допустить ошибку → ошибиться

```
public interface IBank
{
    void PerformPurchase(int productId);
}
```

Совершить покупку → купить
Произвести атаку → атаковать
Одержать победу → победить
Допустить ошибку → ошибиться

```
public interface IBank
{
    void PerformPurchase(int productId);
}
```

Про имена методов:

1. Запросы — не глаголы:
   - существительные;
   - местоимения;
   - и т.д.
2. Действия — глаголы:

ить покупку → купить
ести атаку → атаковать
ать победу → победить
ить ошибку → ошибиться

```
terface IBank

PerformPurchase(int productId);
```

Про имена методов:

1. Запросы — не глаголы:
   - существительные;
   - местоимения;
   - и т.д.
2. Действия — глаголы:
   - не составные;

ить покупку → купить
ести атаку → атаковать
ать победу → победить
ить ошибку → ошибиться

```
terface IBank

PerformPurchase(int productId);
```

Про имена методов:

1. Запросы — не глаголы:
   - существительные;
   - местоимения;
   - и т.д.
2. Действия — глаголы.
   - не составные;

Про имена методов:

1. Запросы — не глаголы:
   - существительные;
   - местоимения;
   - и т.д.
2. Действия — глаголы.
   - не составные;

```
public interface IBank
{
    void Buy(int productId);
}
```

Про имена методов:

1. Запросы — не глаголы:
   - существительные;
   - местоимения;
   - и т.д.
2. Действия — глаголы.
   - не составные;

Про имена методов:

1. Запросы — не глаголы:
   - существительные;
   - местоимения;
   - и т.д.
2. Действия — глаголы.
   - не составные;

```
public interface IWeapon
{
    void DealDamage(ICreature creature);
}
```

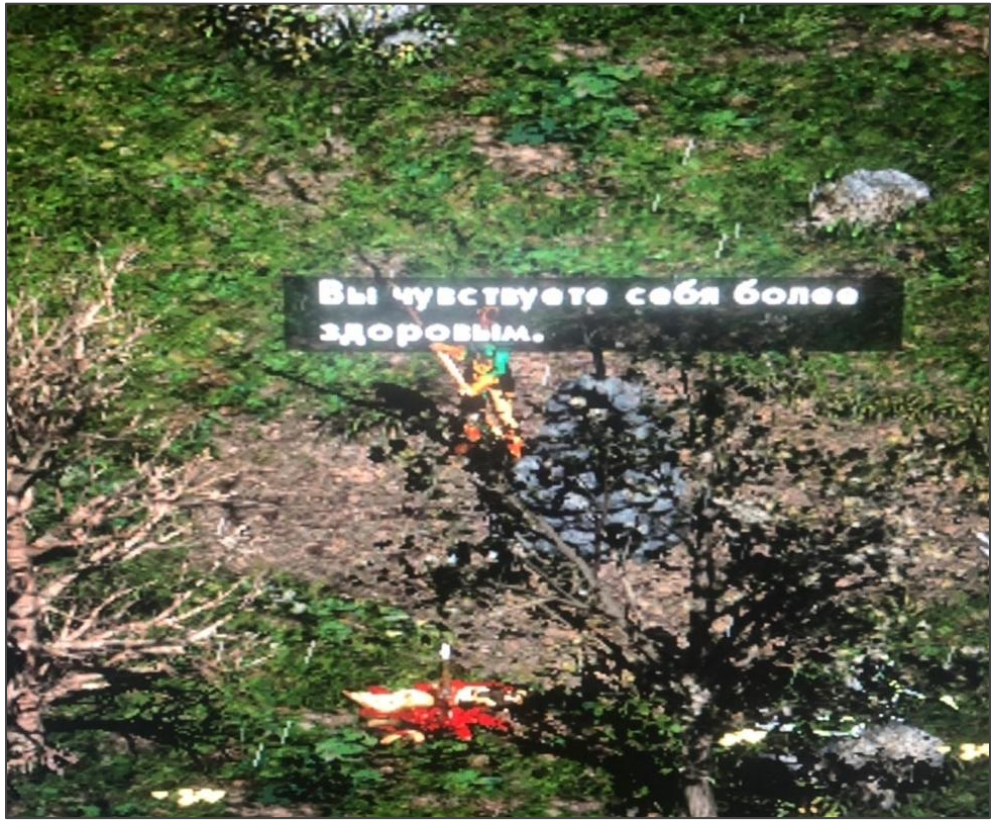
Про имена методов:

1. Запросы — не глаголы:
   - существительные;
   - местоимения;
   - и т.д.
2. Действия — глаголы.
   - не составные;

```
public interface IWeapon
{
    void Attack(ICreature creature);
}
```

```
public interface ICreature
{
    void AddHealth(int value);
}
```

```
public interface ICreature
{
    void RestoreHealth(int value);
}
```



```
public interface ICreature
{
    void RestoreHealth(int value);
}
```

Про имена методов:

1. Запросы — не глаголы:
   - существительные;
   - местоимения;
   - и т.д.
2. Действия — глаголы:
   - не составные;

```
ic interface ICreature

void RestoreHealth(int value);
```

Про имена методов:

1. Запросы — не глаголы:
   - существительные;
   - местоимения;
   - и т.д.
2. Действия — глаголы:
   - не составные;
   - сильные.

```
ic interface ICreature

void RestoreHealth(int value);
```

```
public interface IMailbox
{
    void SendLetter(ILetter letter);
}
```

```
public interface IMailbox
{
    void Send(ILetter letter);
}
```

```
public interface Inventory
{
    IEnumerable<InventoryItem> InventoryItems { get; }
}
```

```
public interface Inventory
{
    IEnumerable<InventoryItem> InventoryItems { get; }
}
```

```
public interface Inventory
{
    IEnumerable<InventoryItem> Items { get; }
}
```

```
public class IntervalTimer
{
    // ...
}
```

A **timer** is a specialized type of clock used for measuring specific time intervals.

```
public class IntervalTimer
{
    // ...
}
```

A **timer** is a specialized type of clock used for measuring specific time intervals.

```
public class IntervalTimer
{
    // ...
}
```

Programmable interval timer

From Wikipedia, the free encyclopedia

```
public class IntervalTimer
{
    // ...
}
```

Он кивнул своей головой

```
public class IntervalTimer
{
    // ...
}
```

Он кивнул

```
public class IntervalTimer
{
    // ...
}
```

Про имена методов:

1. Запросы — не глаголы:
   - существительные;
   - местоимения;
   - и т.д.
2. Действия — глаголы.
   - не составные;
   - сильные.

кивнул

```
blic class IntervalTimer

  // ...
```

Про имена методов:

1. Запросы — не глаголы:
   - существительные;
   - местоимения;
   - и т.д.
2. Действия — глаголы.
   - не составные;
   - сильные.
3. Ёмкие (не дублируют смысл).

кивнул

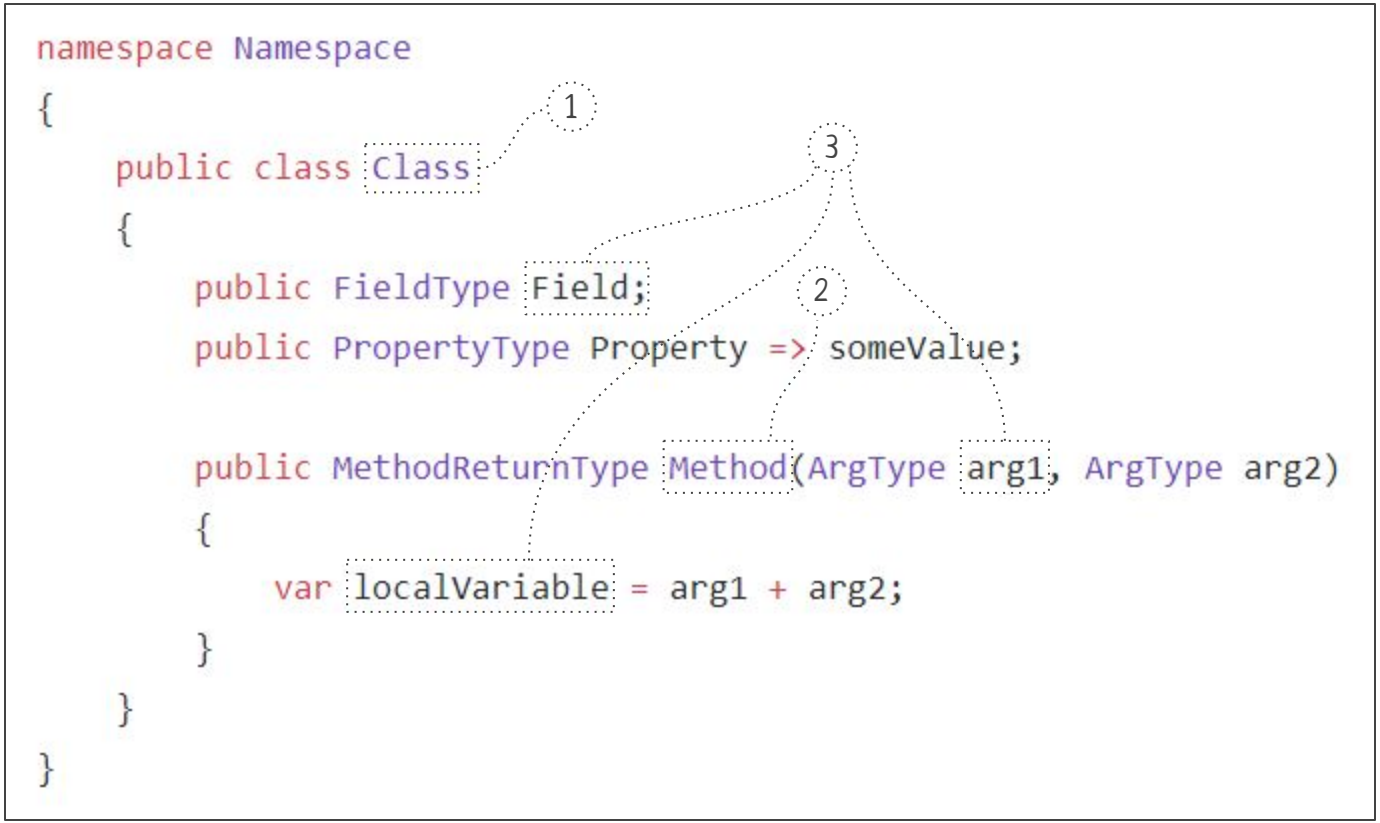
```
blic class IntervalTimer

  // ...
```

```
public class IntervalTimer
{
    // ...
}
```

```
namespace Namespace
{
    public class Class
    {
        public FieldType Field;
        public PropertyType Property => someValue;

        public MethodReturnType Method(ArgType arg1, ArgType arg2)
        {
            var localVariable = arg1 + arg2;
        }
    }
}
```

1

2



```
public class Sword : IWeapon
{
    void Attack(ICreature creature)
    {

    }
}
```

```
public class Sword : IWeapon
{
    void Attack(ICreature creature)
    {
        var damageToApply = _damage;



    }
}
```

```
public class Sword : IWeapon
{
    void Attack(ICreature creature)
    {
        var damageToApply = _damage;
        var currentCreatureHp = creature.Health;




    }
}
```

```
public class Sword : IWeapon
{
    void Attack(ICreature creature)
    {
        var damageToApply = _damage;
        var currentCreatureHp = creature.Health;
        var creatureHpAfterDamageApplied = creature.Health - currentCreatureHp;



    }
}
```

```
public class Sword : IWeapon
{
    void Attack(ICreature creature)
    {
        var damageToApply = _damage;
        var currentCreatureHp = creature.Health;
        var creatureHpAfterDamageApplied = creature.Health - currentCreatureHp;
        var newCreatureHp = Math.Max(0, creatureHpAfterDamageApplied);


    }
}
```

```
public class Sword : IWeapon
{
    void Attack(ICreature creature)
    {
        var damageToApply = _damage;
        var currentCreatureHp = creature.Health;
        var creatureHpAfterDamageApplied = creature.Health - currentCreatureHp;
        var newCreatureHp = Math.Max(0, creatureHpAfterDamageApplied);

        creature.Health = newCreatureHp;
    }
}
```

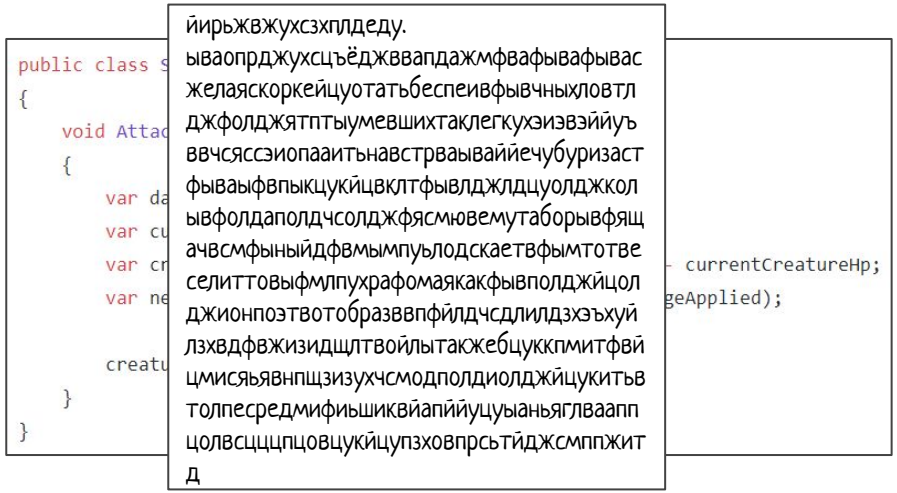
```
public class Sword : IWeapon
{
    void Attack(ICreature creature)
    {
        var damage = _damage;
        var hp = creature.Health;
        var newHp = hp - damage;
        newHp = Math.Max(0, newHp);

        creature.Health = newHp;
    }
}
```

```
public class Sword : IWeapon
{
    void Attack(ICreature creature)
    {
        var damageToApply = _damage;
        var currentCreatureHp = creature.Health;
        var creatureHpAfterDamageApplied = creature.Health - currentCreatureHp;
        var newCreatureHp = Math.Max(0, creatureHpAfterDamageApplied);

        creature.Health = newCreatureHp;
    }
}
```

```
public class Sword : IWeapon
{
    void Attack(ICreature creature)
    {
        var damageToApply = _damage;
        var currentCreatureHp = creature.Health;
        var creatureHpAfterDamageApplied = creature.Health - currentCreatureHp;
        var newCreatureHp = Math.Max(0, creatureHpAfterDamageApplied);

        creature.Health = newCreatureHp;
    }
}
```

```
public class Sword : IWeapon
{
    void Attack(ICreature creature)
    {
        var damageToApply = _damage;
        var currentCreatureHp = creature.Health;
        var creatureHpAfterDamageApplied = creature.Health - currentCreatureHp;
        var newCreatureHp = Math.Max(0, creatureHpAfterDamageApplied);

        creature.Health = newCreatureHp;
    }
}
```

```
public class Sword : IWeapon
{
    void Attack(ICreature creature)
    {
        var damage = _damage;
        var hp = creature.Health;
        var newHp = hp - damage;
        newHp = Math.Max(0, newHp);

        creature.Health = newHp;
    }
}
```



Про другие имена:

```
urrentCreatureHp;
pplied);
```

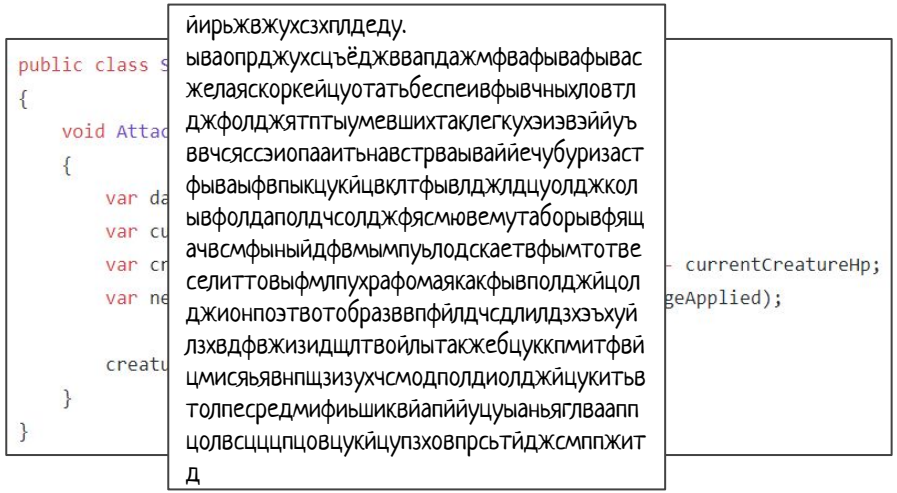
```
public class Sword : IWeapon
{
    void Attack(ICreature creature)
    {
        var damage = _damage;
        var hp = creature.Health;
        var newHp = hp - damage;
        newHp = Math.Max(0, newHp);

        creature.Health = newHp;
    }
}
```

Про другие имена:

1. Не составные.

```
urrentCreatureHp;
pplied);
```

```
public class Sword : IWeapon
{
    void Attack(ICreature creature)
    {
        var damage = _damage;
        var hp = creature.Health;
        var newHp = hp - damage;
        newHp = Math.Max(0, newHp);

        creature.Health = newHp;
    }
}
```



```
public class Sword : IWeapon
{
    void Attack(ICreature creature)
    {
        var damage = _damage;
        var hp = creature.Health;
        var newHp = hp - currentCreatureHp;
        newHp = Math.Max(0, newHp);

        creature.Health = newHp;
    }
}
```

```
public class Sword : IWeapon
{
    void Attack(ICreature l)
    {
        var a = _b;
        var c = l.D;
        var z = y - x;
        v = Math.Max(0, w);

        l.D = v;
    }
}
```

```
public void Example()
{
    var sb = new StringBuilder();

    if (WeatherIsFine)
        sb.Append("Weather is fine!");

    // ...
}
```

```
public void Example()
{
    var sb = new StringBuilder();

    sb.AppendIf(WeatherIsFine, "Weather is fine!");


    // ...
}
```

```
public void Example()
{
    var sb = new StringBuilder();

    sb.Append("Weather is fine!", condition: WeatherIsFine);


    // ...
}
```

```
public void Example()
{
    var sb = new StringBuilder();

    sb.Append("Weather is fine!", when: WeatherIsFine);


    // ...
}
```

Про другие имена:

1. Не составные.

```
ngBuilder();

r is fine!", when: WeatherIsFine);
```

Про другие имена:

1. Не составные.
2. Вообще: это могут быть любые слова.

```
ngBuilder();

r is fine!", when: WeatherIsFine);
```

Лишь бы читалось хорошо

Лишь бы читалось хорошо

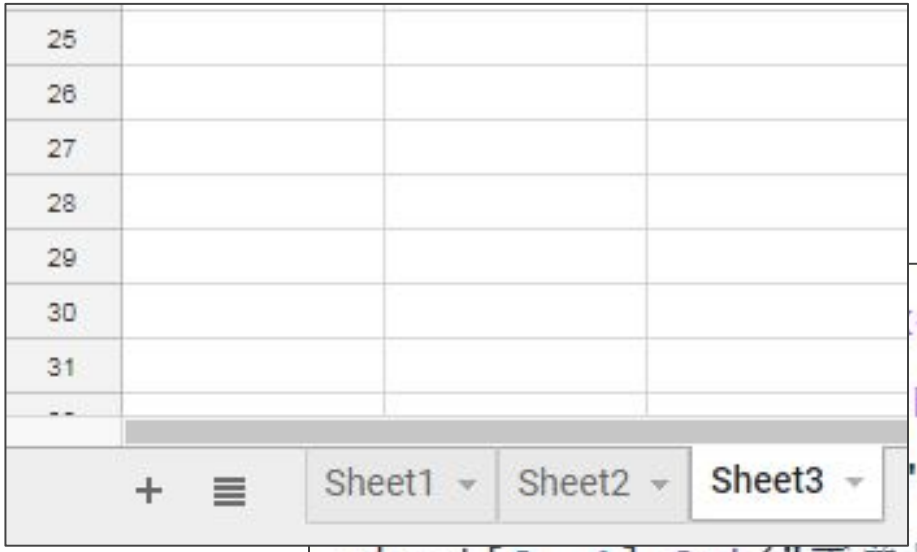
```
public interface IDirectoryWatcher
{
    IDisposable OnChange(Action action);
}
```

```
public interface IDirectoryChanges
{
    IDisposable OnNext(Action action);
}
```

```
ExcelHelper excelHelper = new ExcelHelper();
excelHelper.Create();
excelHelper.CreateSheet("条码导出");
excelHelper.SetCellValue(0, 0, "条码");
excelHelper.SetCellValue(0, 1, "工单号");
excelHelper.SetCellValue(0, 2, "订单类型");
excelHelper.SetCellValue(0, 3, "物料编号");
```

```
var document = new ExcelDocument();
var sheet = document.NewSheet("条码导出");
sheet.Cell(0, 0).Set("条码");
sheet.Cell(0, 1).Set("工单号");
sheet.Cell(0, 2).Set("订单类型");
sheet.Cell(0, 3).Set("物料编号");
```

```
var document = new ExcelDocument();
var sheet = document.NewSheet("条码导出");
sheet[0, 0].Set("条码");
sheet[0, 1].Set("工单号");
sheet[0, 2].Set("订单类型");
sheet[0, 3].Set("物料编号");
```




```
celDocument();
NewSheet("条码导出");
");
sheet[0, 1].Set("工单号");
sheet[0, 2].Set("订单类型");
sheet[0, 3].Set("物料编号");
```

Environment.TickCount Property

Namespace: System

Assemblies: System.Runtime.Extensions.dll, mscorlib.dll, netstandard.dll

Gets the number of milliseconds elapsed since the system started.

C#

```
public static int TickCount { get; }
```

Environment.TickCount Property

Namespace: System

Assemblies: System.Runtime.Extensions.dll, mscorlib.dll, netstandard.dll

Gets the number of milliseconds elapsed since the system started.

C#

```
public static int TickCount { get; }
```

```
public class ManualResetEvent : EventWaitHandle
{
    // ...
}
```

```
public class ManualResetEvent : EventWaitHandle
{
    // ...
}
```

```
public class EventWaitHandle
{



}
```

```
public class EventWaitHandle
{
    public bool Set();


}
```

```
public class EventWaitHandle
{
    public bool Set();
    public bool Reset();

}
```

```
public class EventWaitHandle
{
    public bool Set();
    public bool Reset();
    public bool WaitOne(int millisecondsTimeout);
}
```

```
public class ThreadGate
{
    public bool Set();
    public bool Reset();
    public bool WaitOne(int millisecondsTimeout);
}
```

```
public class ThreadGate
{
    public bool Open();
    public bool Reset();
    public bool WaitOne(int millisecondsTimeout);
}
```

```
public class ThreadGate
{
    public bool Open();
    public bool Close();
    public bool WaitOne(int millisecondsTimeout);
}
```

```
public class ThreadGate
{
    public bool Open();
    public bool Close();
    public bool WaitForOpen(int millisecondsTimeout);
}
```

```
public class ThreadGate
{
    public bool Open();
    public bool Close();
    public bool WaitForOpen(int milli
}
```

```
public class ThreadGate
{
    public bool Open();
    public bool Close();
    public bool WaitForOpen(int milli
}
```

```
int? nullable = ...;
var number = nullable.GetValueOrDefault();
```

```
int? nullable = ...;
var number = nullable.Or(0);
```

```
int? nullable = ...;
var number = nullable.OrDefault();
```

```
public class Sword : IWeapon
{
    void Attack(ICreature creature)
    {
        var damage = _damage;
        var hp = creature.Health;
        var newHp = hp - damage;
        newHp = Math.Max(0, newHp);

        creature.Health = newHp;
    }
}
```

```
public class Sword : IWeapon
{
    void Attack(ICreature creature)
    {

    }
}
```

```
public class Sword : IWeapon
{
    void Attack(ICreature creature)
    {
        creature.Heath = creature.Health - _damage;



    }
}
```

```
public class Sword : IWeapon
{
    void Attack(ICreature creature)
    {
        creature.Heath = creature.Health - _damage;
        creature.Health = Math.Max(0, creature.Health);


    }
}
```

```
public class Sword : IWeapon
{
    void Attack(ICreature creature)
    {
        creature.Heath = (creature.Health - _damage).Or(0).IfLess();


    }
}
```

```
Assert.That(() => 2 + 2, Is.EqualTo(4));
```

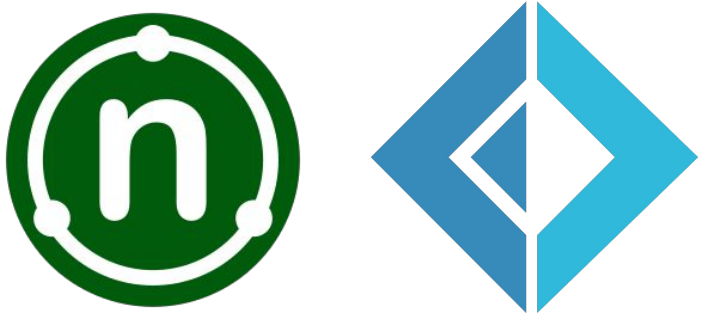
```
Assert.That(() => 2 + 2, Is.EqualTo(4));
```

```
Assert.That(() => 2 + 2, Is.EqualTo(4));
```

```
public void SomeTest()
{
    var a = 2;
    var b = 2;

    (a + b).Should().Be(4, errorMessage: "We are not in George Orwell's world.");
}
```

```
public void SomeTest()
{
    var a = 2;
    var b = 2;

    (a + b).Should().Be(4, errorMessage: "We are not in George Orwell's world.");
}
```

```
public void SomeTest()
{
    var a = 2;
    var b = 2;

    (a + b).Should().Be(4, because: "We are not in George Orwell's world.");
}
```



```
public interface IText
{
    IEnumerable<Character> Characters { get; }

    Result Write(string value, int at);
    Result Remove(int count, int at);
}
```

```
public interface IText
{
    IEnumerable<Character> Characters { get; }

    Result Write(string value, int at);
    Result Remove(int count, int at);
}
```

```
public interface IText
{
    IEnumerable<Character> Characters { get; }

    Result Write(string value, int at);
    Result Remove(int count, int at);
}
```

- Записать что-то в текст.

```
public interface IText
{
    IEnumerable<Character> Characters { get; }

    Result Write(string value, int at);
    Result Remove(int count, int at);
}
```

- Записать что-то в текст.
- Удалить что-то из текста.

```
public interface IText
{
    IEnumerable<Character> Characters { get; }

    Result Write(string value, int at);
    Result Remove(int count, int at);
}
```

- Записать что-то в текст.
- Удалить что-то из текста.
- Убедиться, что всё ок.

Когда записываешь “-” по индексу 0 в тексте “Hello” он должен стать “-Hello”

Когда записываешь “-” по индексу 0 в тексте “Hello” он должен стать “-Hello”

When write “-” at 0 in “Hello” it should equal “-Hello”

Когда записываешь “-” по индексу 0 в тексте “Hello” он должен стать “-Hello”

When write “-” at 0 in “Hello” it should equal “-Hello”

```
[<Fact>]
let ``When write "-" at 0 in "Hello" it should equal "-Hello"``() =
```

Когда записываешь “-” по индексу 0 в тексте “Hello” он должен стать “-Hello”

When write “-” at 0 in “Hello” it should equal “-Hello”

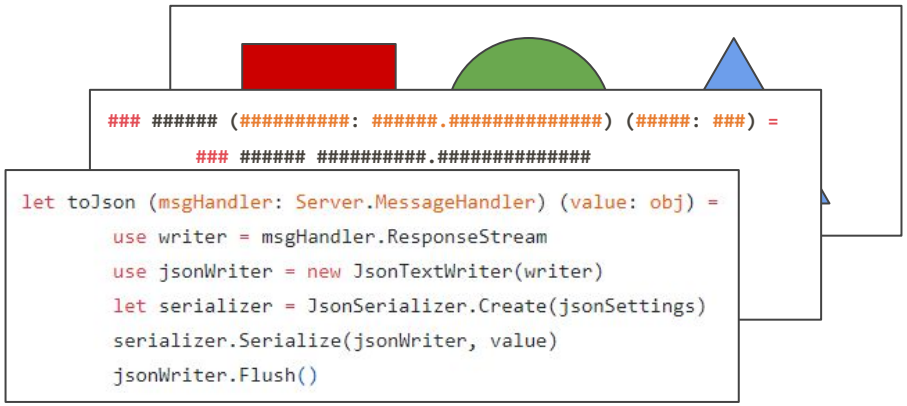
```
[<Fact>]
let ``When write "-" at 0 in "Hello" it should equal "-Hello"``() =
    write "-" <| at 0 <| in' "Hello" |> should equal "-Hello"
```

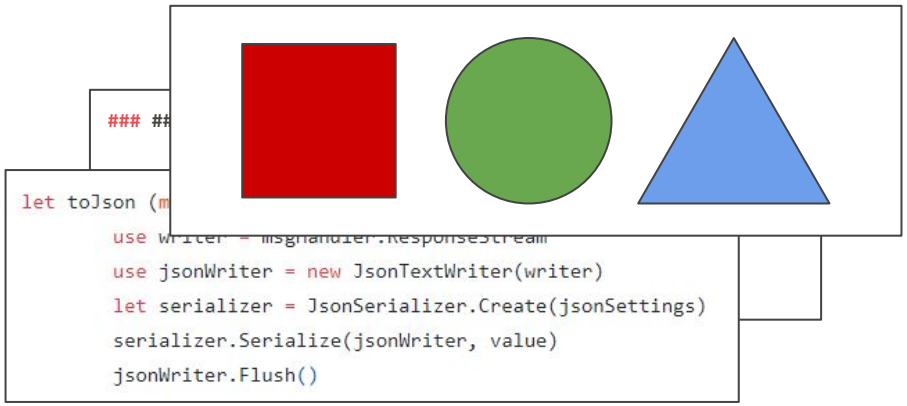
```
[<Fact>]
let ``When remove 3 characters at 0 in "Hello" it should equal "lo"``() =
```

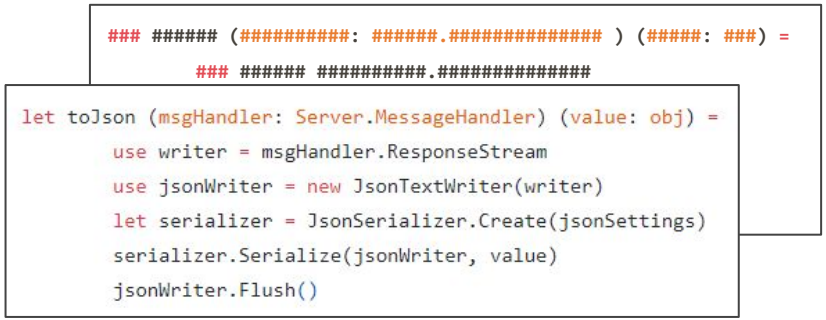
```
[<Fact>]
let ``When remove 3 characters at 0 in "Hello" it should equal "lo"``() =
    remove 3 <| at 0 <| in' "Hello" |> should equal "lo"
```

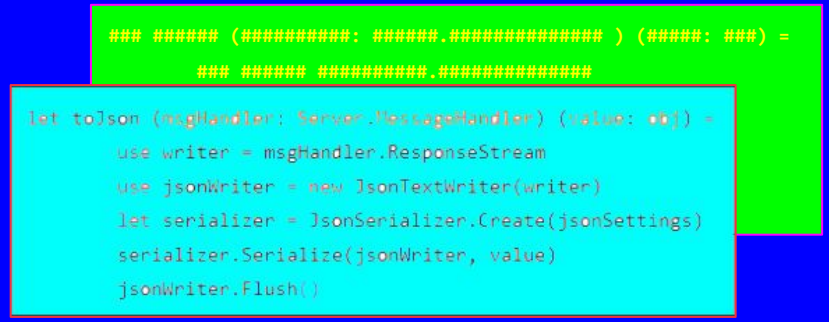
```
let at x = x
let in' x = x
```

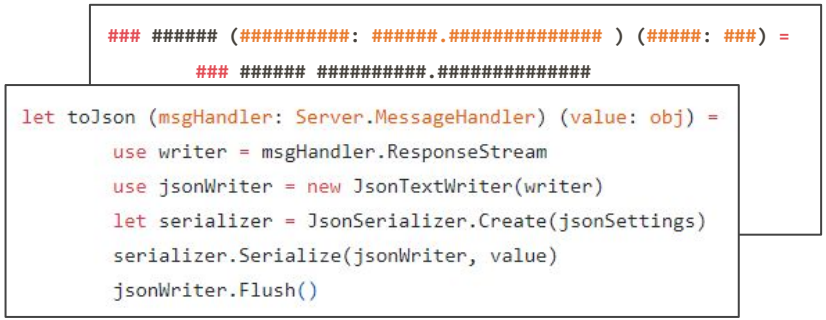
```
[<Fact>]
let ``When remove 3 characters at 0 in "Hello" it should equal "lo"``() =
    remove 3 <| at 0 <| in' "Hello" |> should equal "lo"
```











Код пишется для людей

https://github.com/JoshuaLight

https://www.linkedin.com/in/lightjoshua/

- http://www.carlopescio.com/2011/04/your-coding-conventions-are-hurting-you.html
- https://medium.com/@atherlangga/retracing-original-object-oriented-programming-f8b689c4ce50
- https://fsharpforfunandprofit.com/
- https://www.gnu.org/software/smalltalk/manual/html_node/index.html

Пишем ли мы в продакшене так?

Да

```
public static IEnumerable<T> OrEmpty<T>(this IEnumerable<T> self) =>
    self ?? Enumerable.Empty<T>();
```

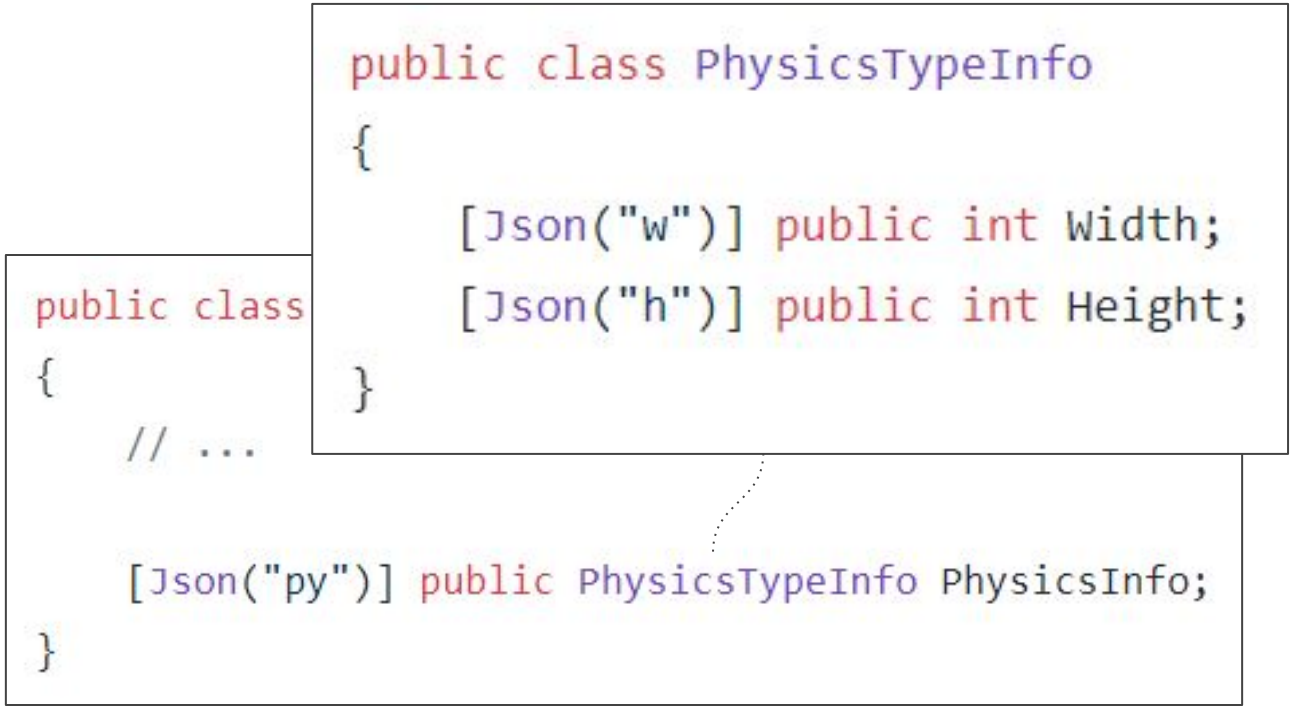
```
public class SkillsWrapper
{
    public IEnumerable<Skill> All => Data.Skills;

    public Skill One(int id) =>
        All.FirstOrDefault(x => x.Id == id);

    public Skill Required(int id) =>
       One(id).OrThrow($"Skill with '{id}' was not found.");
}
```

```
public interface IEffectsWrapperReadonly
{
    IEvent<IGameplayModelReadonly> OnChanged { get; }

    IEnumerable<Effect> All { get; }
    IEnumerable<Effect> Of(EffectTypeId id);

    bool Has(EffectTypeId id);

    Effect One(int id);
    Effect One(EffectTypeId id);

    Effect Required(int id);
    Effect Required(EffectTypeId id);
}
```

```
public interface IEffectsWrapperReadonly
{
    IEvent<IGameplayModelReadonly> OnChanged { get; }

    IEnumerable<Effect> All { get; }
    IEnumerable<Effect> Of(EffectTypeId id);

    bool Has(EffectTypeId id);

    Effect One(int id);
    Effect One(EffectTypeId id);

    Effect Required(int id);
    Effect Required(EffectTypeId id);
}
```

```
var effect = Model.User.Effects.One(EffectTypeId.EnergyConsumptionReduce);
```

```
public class SceneObjectType
{
    // ...

    [Json("py")] public PhysicsTypeInfo PhysicsInfo;
}
```



```
private static void Size(SceneObjectTypeId typeId, int width, int height)
{



}
```

```
private static void Size(SceneObjectTypeId typeId, int width, int height)
{
    var objectType = Types.First(x => x.TypeId == typeId);



}
```

```
private static void Size(SceneObjectTypeId typeId, int width, int height)
{
    var objectType = Types.First(x => x.TypeId == typeId);
    if (objectType.PhysicsInfo == null)




}
```

```
private static void Size(SceneObjectTypeId typeId, int width, int height)
{
    var objectType = Types.First(x => x.TypeId == typeId);
    if (objectType.PhysicsInfo == null)
        objectType.PhysicsInfo = new PhysicsTypeInfo();



}
```

```
private static void Size(SceneObjectTypeId typeId, int width, int height)
{
    var objectType = Types.First(x => x.TypeId == typeId);
    if (objectType.PhysicsInfo == null)
        objectType.PhysicsInfo = new PhysicsTypeInfo();

    objectType.PhysicsInfo.Width = width;

}
```

```
private static void Size(SceneObjectTypeId typeId, int width, int height)
{
    var objectType = Types.First(x => x.TypeId == typeId);
    if (objectType.PhysicsInfo == null)
        objectType.PhysicsInfo = new PhysicsTypeInfo();

    objectType.PhysicsInfo.Width = width;
    objectType.PhysicsInfo.Height = height;
}
```

```
private static void Size(SceneObjectTypeId typeId, int width, int height)
{



}
```

```
private static void Size(SceneObjectTypeId typeId, int width, int height)
{
    Ensure(typeId)
        .Has(x => x.PhysicsInfo.Width, width)
        .Has(x => x.PhysicsInfo.Height, height);



}
```

```
private static void Size(SceneObjectTypeId typeId, int width, int height)
{
    Ensure(typeId)
        .Has(x => x.PhysicsInfo.Width, width)
        .Has(x => x.PhysicsInfo.Height, height);



}
```