Карпов В.Э.

Объектно-ориентированное программирование

Смолток. Лекция 2

Задание классов

Объявление класса – это регистрация в системе спецификации экземпляра и самого класса.

- Спецификация экземпляра - объявления переменных экземпляра и его методов. Спецификация класса - объявление переменных класса и объявление методов класса.
- Экземпляры: распознают одни и те же сообщения и имеют одинаковую структуру собственной памяти.

У каждого класса существует два типа **методов**:

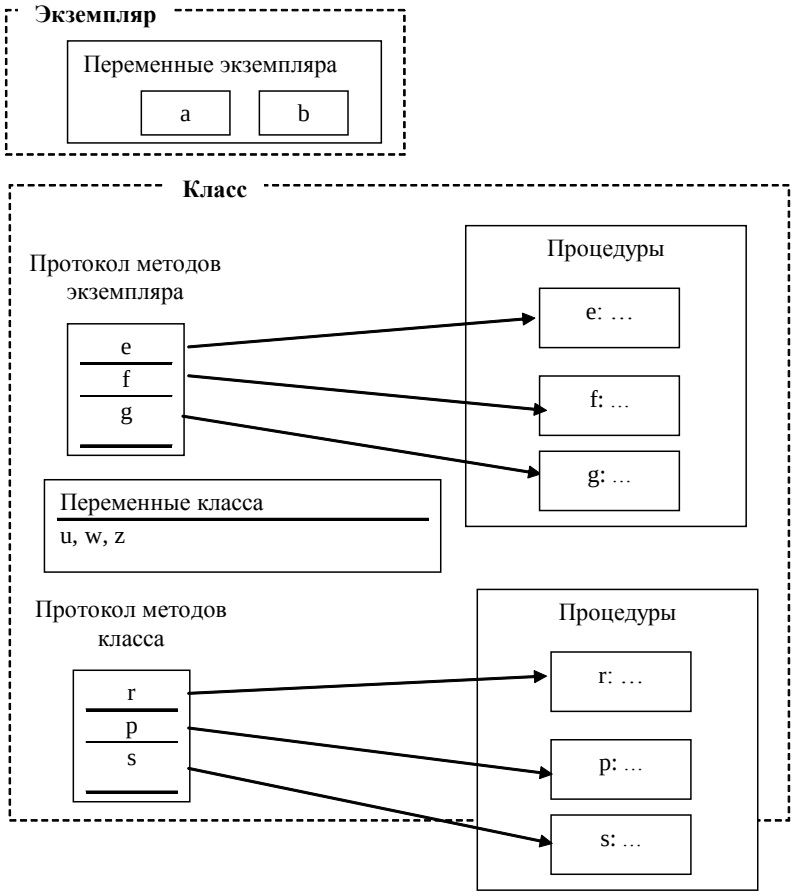
- методы класса;
- методы экземпляра.

Методы класса:

- Действия с целым классом.
- Выполняются по сообщениям, посылаемым этому классу

Объявление метода экземпляра состоит из *схемы сообщения и тела метода*.

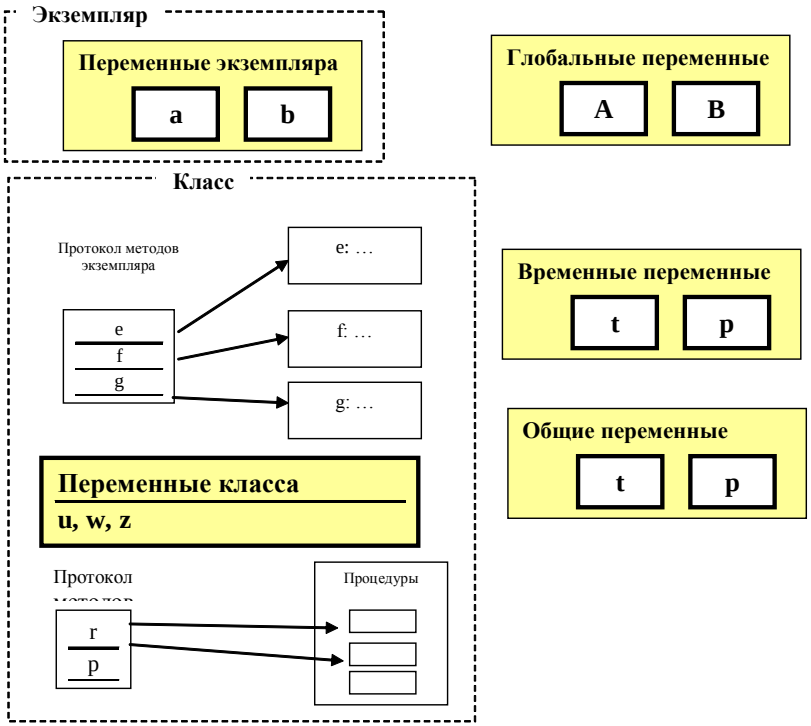
- Схема сообщения - имя сообщения и формальные параметров.
- Тело метода – программа, реакция объекта на сообщение.



Переменные

- переменные экземпляра (instance variable);
- переменные класса (class variable);
- временные переменные (temporary variable); глобальные переменные (global variable);
- общие переменные (pool variable).

Все они различаются **временем** своего существования и **областью** действия.



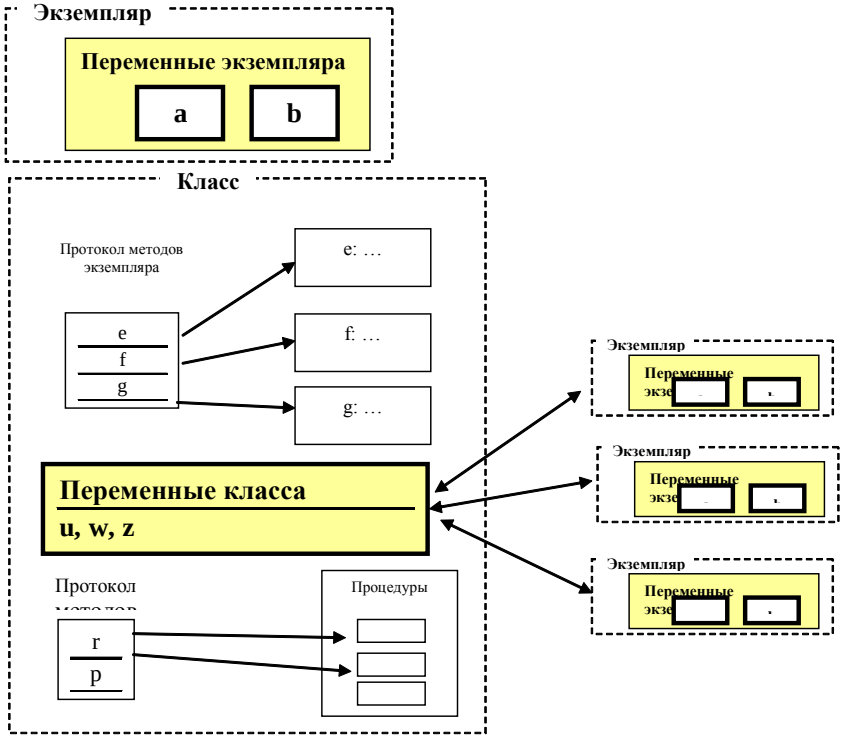
Переменные экземпляра и класса

Переменные экземпляра

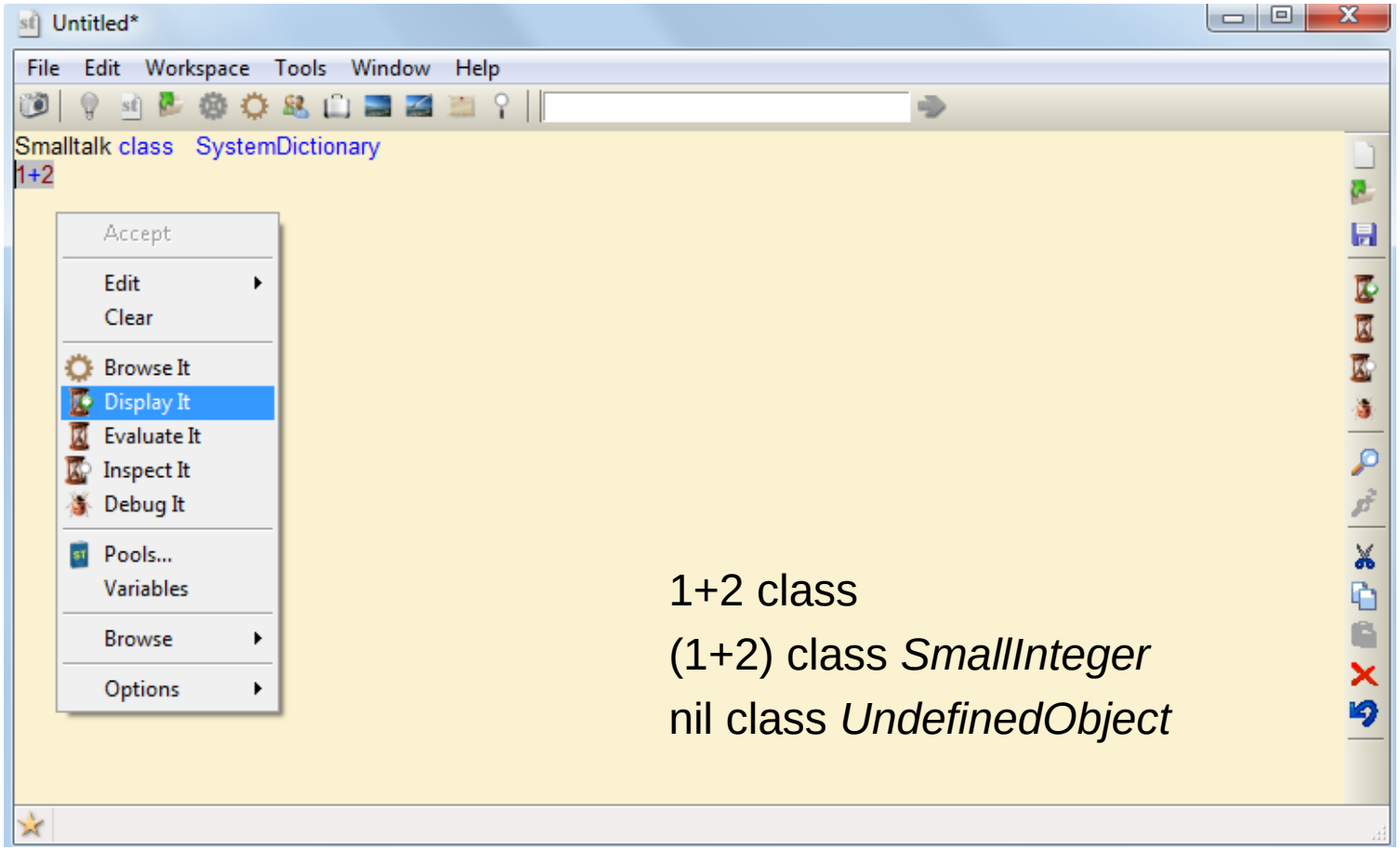
- Хранятся в памяти внутри каждого экземпляра.
- Ссылки на них допускаются только внутри данного экземпляра, и они существуют до тех пор, пока существует сам экземпляр.
- Для доступа к переменным экземпляра требуется, чтобы этот экземпляр имел соответствующие методы.

Переменные класса

- Доступны любому экземпляру этого класса по чтению и записи.
- Описываются при объявлении свойств класса.
- Для доступа к переменным класса извне требуется, чтобы у класса были соответствующие методы.



Немного о работе в среде Смолток



- DolphinSmalltalk
- SmalltalkExpress

Временные переменные

- Объявляются внутри методов.
- Создаются в момент вызова метода.
- Уничтожаются по возвращении из метода.

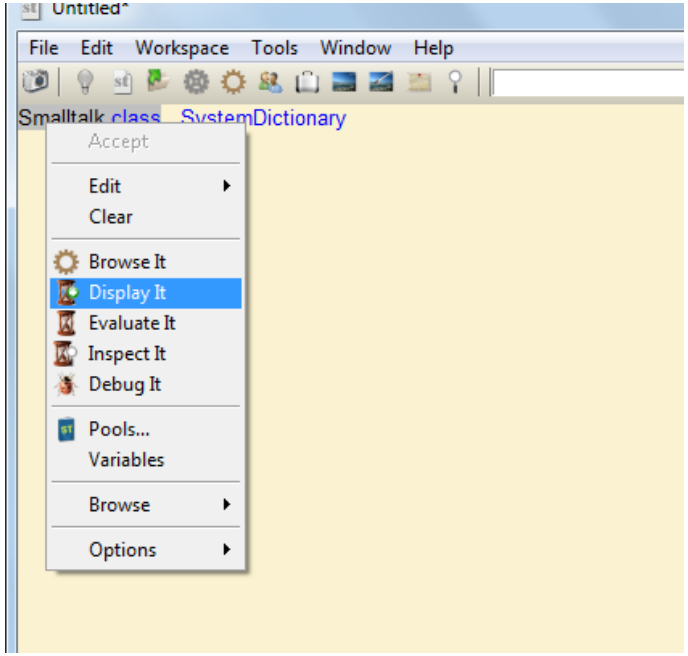
```
| path speed time |
path := 200.
speed := 5.
time := path/speed.
^time.
```

Глобальные переменные

К *переменным класса, глобальным и общим* переменным имеется доступ более чем из одного объекта. Они зарегистрированы в *общих словарях*.

Переменные классов: регистрируются в словаре по имени *Class*

Глобальные переменные. В системе есть каталог, содержащий все глобальные переменные программной среды. Он имеет имя *SmallTalk* (*Смолток*) и сам является глобальной переменной. Глобальные переменные доступны любому объекту по чтению и записи.



Smalltalk class *SystemDictionary*

1. Явное определение:

SmallTalk at: #<name> put: nil
(Смолток вПозиции: #<имя> разместить: nil)

"#"*<name>* - системное имя.

SmallTalk at: #NewVar put: nil.
NewVar := 3.1415.

или:

SmallTalk at: #NewVar put: 3.1415

2. «Интерактивное» определение

Общие переменные

- *Регистрируются в словарях*
- Для чтения и записи в общие переменные в описании класса должны быть объявлены имена словарей, содержащих эти переменные.

Например:

class name HUMAN
superclass ANIMAL
instance variable names SEX
class variable names POPULATION
shared pools BiologicalFacts -- общие переменные

Здесь словарь переменных *BiologicalFacts* будет общим для класса *Human*, и все переменные из словаря *BiologicalFacts* будут доступны по чтению и записи всем экземплярам класса *Human*.

Псевдопеременные

Переменные, которым нельзя присвоить значение (*системные псевдопеременные)*:

- *nil (нуль)* ссылка на пустой объект;
- *self (сам),*
- *super (супер)* используются в качестве адресатов при обращении из метода какого-либо объекта к самому себе или к суперклассу соответственно.

Применение данных системных псевдопеременных в левой части оператора присваивания запрещено.

true (истина), false (ложь).

Операция присваивания

Обозначение:

:= или ←

Смолток работает только с указателями на объекты ⟹

Нет типизации переменных. ⟹ *Имеет смысл лишь говорить о том, на какой объект указывает переменная.*

|*a b*| *-- Введенные переменные указывают на* **nil** *- экземпляр класса* *UndefinedObject*

```
a := 1.          -- a указывает на экземпляр класса МалоеЦелое
a := 'qwerty'.   -- a указывает на экземпляр класса Строка
a := b.          -- a вновь указывает на nil
```

При объявлении временной переменной Смолток порождает экземпляр класса ***НеопределенныйОбъект (UndefinedObject)****. У этого класса есть только один экземпляр –* ***нуль (nil)****.*

```
|a|
a class. -- Результат выдача имени класса «UndefinedObject»
a isNil. -- Проверка того, что переменная ссылается на nil
```

Псевдопеременная super

Класс A – родительский класс

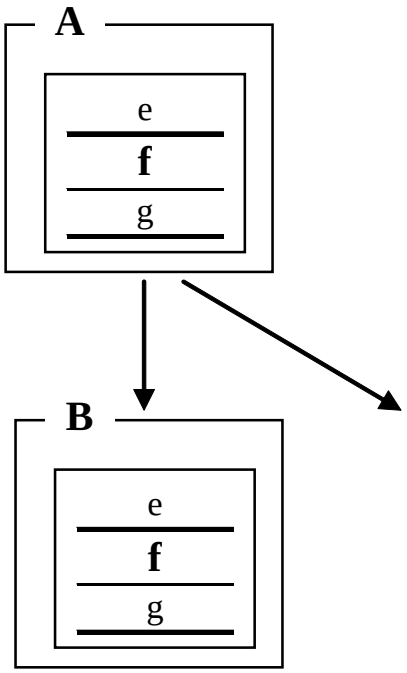
Класс B - дочерний класс.

Экземпляр b $\in$ B

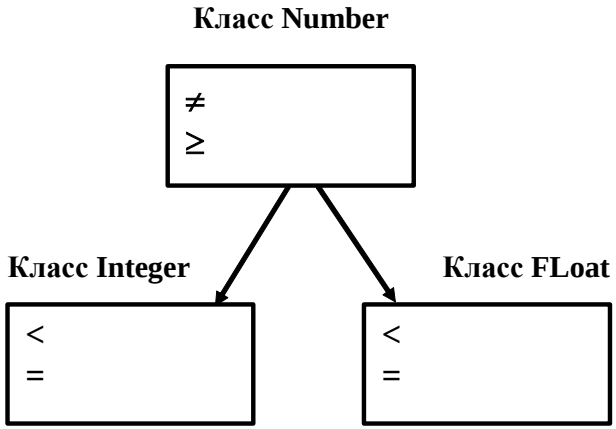
b f - вызов локального метода

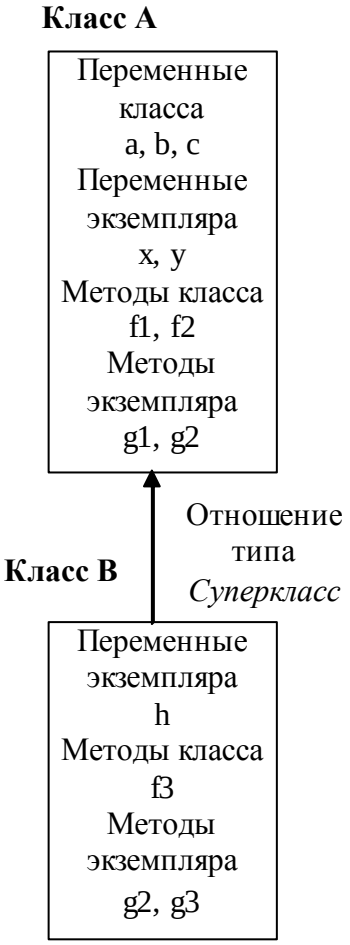
Для вызова родительского метода:

super f



Иерархия классов. Наследование и полиморфизм





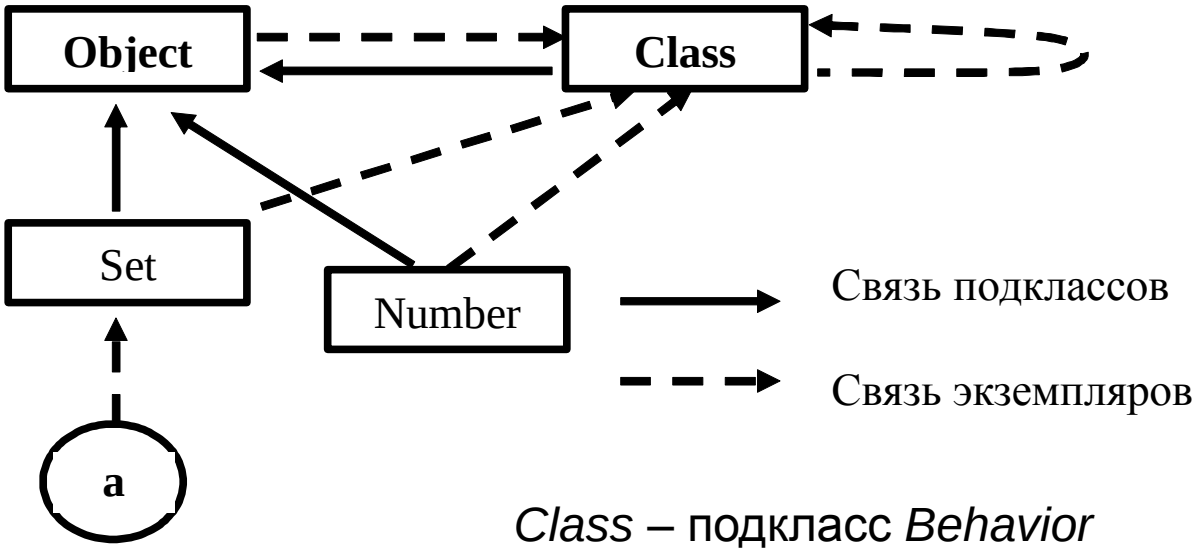
- *Number*

≥ y

^(self<y) not

Классы Object и Class

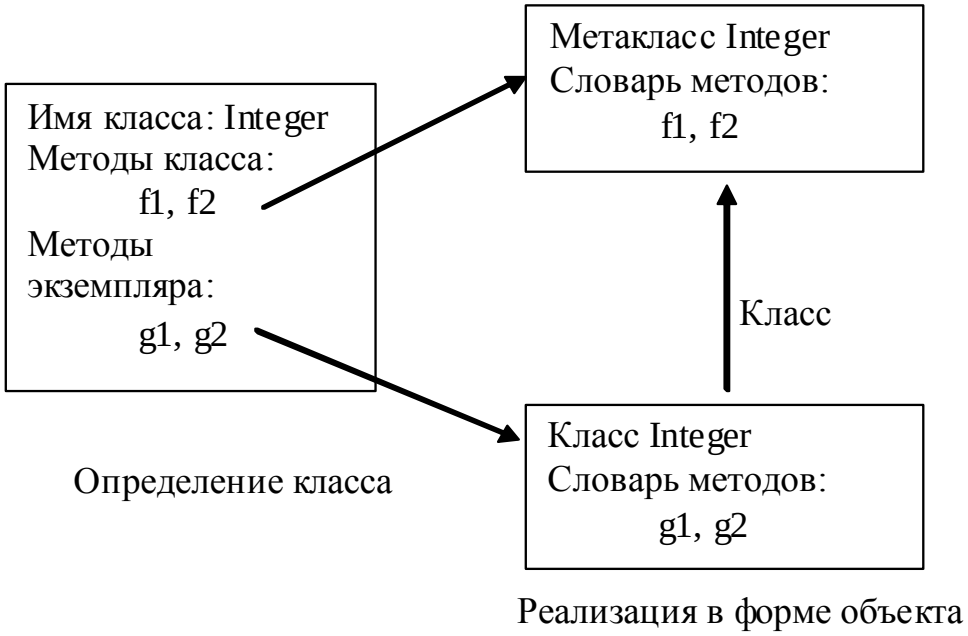
- Классы являются объектами.
- Все классы, будучи объектами, принадлежат классу *Class*.
- Класс *Object* тоже является экземпляром класса *Class*, который, в свою очередь, является подклассом корневого класса *Object*.



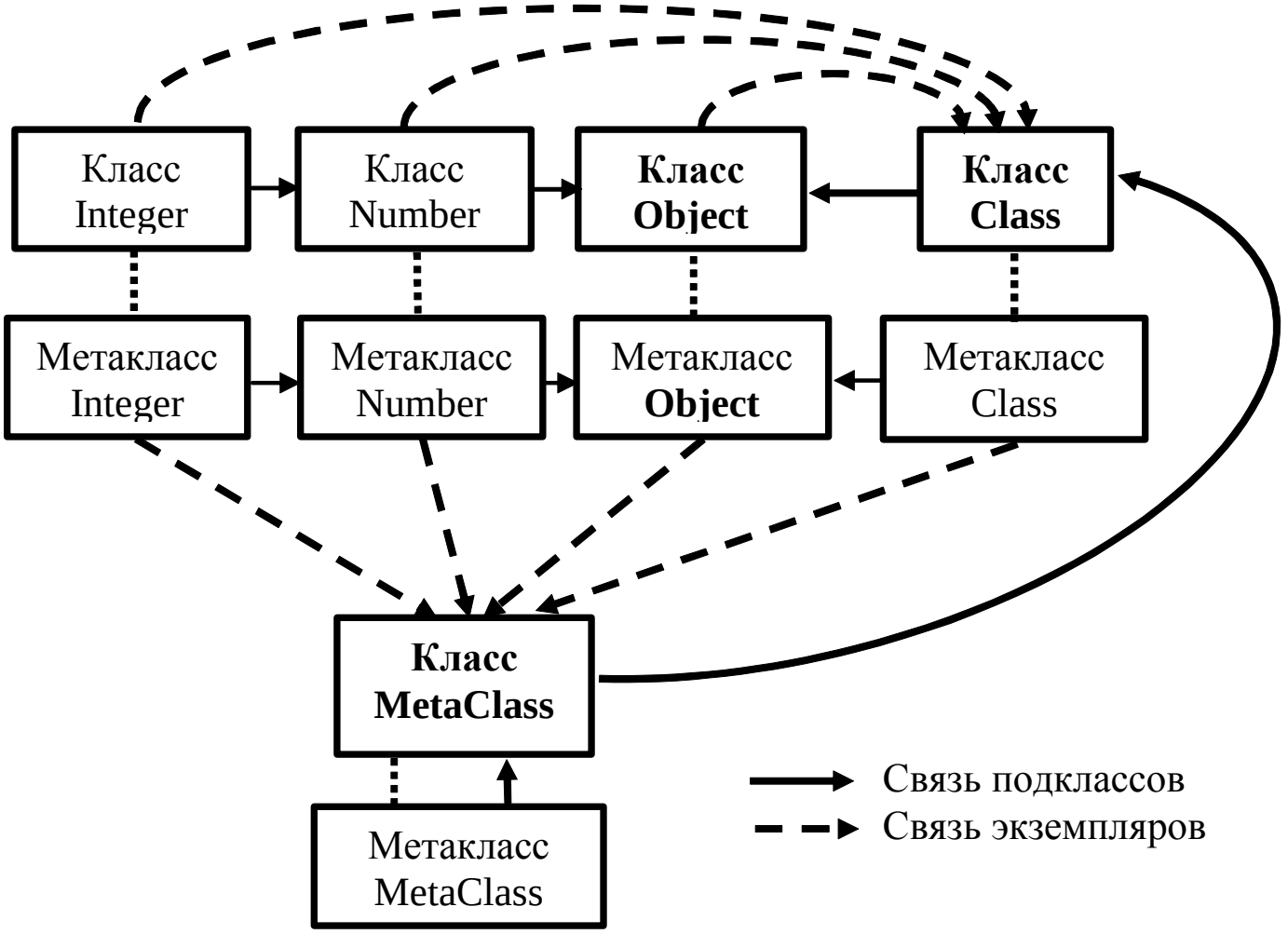
Class – подкласс *Behavior*
Добавляет и удаляет классы, обслуживает словари классов и проч.

Метаклассы

- Сообщение, посылаемое экземпляру, анализируется его классом $\Rightarrow$ Сообщение, посылаемое классу, должно анализироваться его классом (классом класса) - *метаклассом*.



Метаклассы


Класс Integer
Класс Number
Класс Object
Класс Class
Метакласс Integer
Метакласс Number
Метакласс Object
Метакласс Class
Класс MetaClass
Метакласс MetaClass
Связь подклассов
Связь экземпляров

Блоки

- Блок – это описание отложенной последовательности действий.
- Блок – экземпляр класса *Context*

[a:=1+2. c:=a*10.]

Вычисление блока:

block *value*.

| x |
x := ['строка' inPos: 5].
^x. -- результат - экзКонтекст

^x value. -- результат "к".

Блок с аргументами:

[:<аргумент>| <сообщения>] value: <значение аргумента>

^[:i | i+5] value: 5 *-- 10*
*^[:i :j| i*j] value:5 value: 3* *-- 15*

Условные конструкции и булевские величины

Операции сравнения

3 > 4
#(1 2 3)= #(1 2 3)
'привет' >'пока'
5=2+3

Условные выражения

if < условие > then <оператор 1> else <оператор 1>

или:

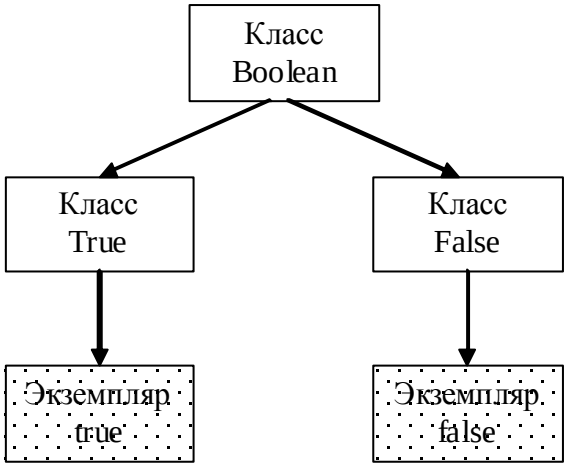
< условие >
если условие истинно: [<блок выражений 1>]
если условие ложно: [<блок выражений 2>]

На языке ООП:

<объект-адресат класса True (Истина) или False (Ложь)>
ifTrue: <объект-параметр 1>
ifFalse: <объект-параметр 2>

Организация классов

```
-- Класс Boolean (Логический)
class name Boolean
superclass Object
class methods
"Оба следующих метода выдают ошибку, т.к. значения 'истина' и 'ложь' уникальны"
new
        ^self неправильноеСообщение
new: intVal
        ^self неправильноеСообщение
instance methods
занестиВ: экзПоток
"Добавить последовательность символов приемника к заданному потоку, из которого приемник может быть создан заново"
        self печататьНа: экзПоток
печататьНа: экзПоток
"Добавить представление приемника в символьном виде к заданному потоку"
экзПоток поместитьВсеПоследующие:
        (self ifTrue: ['истина'] ifFalse:['ложь'])
```



Реализация

```
-- Класс True (Истина)
class name True
superclass Boolean
-- class methods отсутствуют
instance methods
& boolInstance
        ^boolInstance
| boolInstance
        ^true
ifTrue: Block
        ^Block value
ifTrue: Block1 ifFalse: Block2
        ^Block1 value
ifFalse: Block
        ^nil
ifFalse: Block1 ifTrue: Block2
        ^Block2 value
and: Block – И с блоком
        ^Block value
or: Block -- ИЛИ с блоком
        ^true
xor: boolInstance -- исключающее или
        ^boolInstance not
not
        ^false
equal: boolInstance -- эквивалентность
приемника и
                        --  аргумента
        ^boolInstance
```

```
-- Класс False (Ложь)
class name False
superclass Boolean
-- class methods отсутствуют
instance methods
& boolInstance
        ^false
| boolInstance
        ^boolInstance
ifTrue: Block
        ^nil
ifTrue: Block1 ifFalse: Block2
        ^Block2 value
ifFalse: Block
        ^Block value
ifFalse: Block1 ifTrue: Block2
        ^Block1 value
and: Block -- И с блоком
        ^false
or: Block -- ИЛИ с блоком
        ^Block value
xor: boolInstance -- исключающее ИЛИ
        ^boolInstance
not
        ^true
equal: boolInstance -- эквивалентность приемника
и
                        -- аргумента
        ^boolInstance not
```

Примеры

```
| maximum a b |
a:= 5 sqr.
b:= 5 factorial.
a>b
   ifTrue: [maximum:= a]
   ifFalse :[maximum:= b].
^maximum
```

Создание метода

Задача: создать метод с именем 'max'.

1. Определение спецификаци.

- Объект-адресат - это целое число ⇒ метод будет принадлежать классу целых чисел.
- Объект-параметр - целое число.
- Результат - это тоже целое число ⇒ сообщение будет выглядеть следующим образом:
  <экзЦелоеЧислоМакс> := <экзЦелоеЧислоА> maximum: <экзЦелоеЧислоВ>

2. В Смолтоке любой метод имеет следующий вид:

```
имяСообщения
" комментарии "
| временные переменные |
выражения
```

В нашем случае:

```
max: arg -- имя метода с параметром
"вычисление максимума( a^2, b!)"
| a b maximum |
a:= self sqr.              -- псевдопеременная 'self' - значение объекта-адресата
b:= arg factorial.
a>b
        ifTrue: [maximum:= a]
        ifFalse: [maximum:= b].
^maximum
```

3. Добавляем метод в класс *Integer* (суперкласс *Number* - *Величина*).

Примеры использования метода:

```
5 max: 4
10 max: 7
```