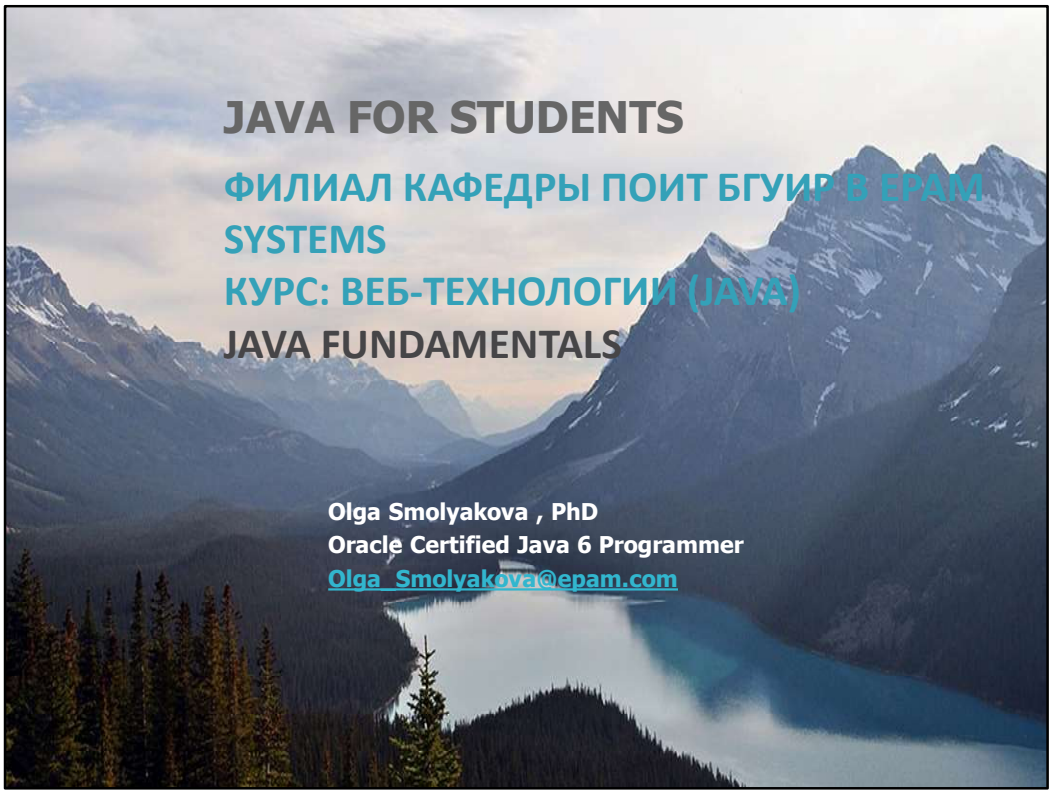

JAVA FOR STUDENTS
ФИЛИАЛ КАФЕДРЫ ПОИТ БГУИР В EPAM SYSTEMS
КУРС: ВЕБ-ТЕХНОЛОГИИ (JAVA)
JAVA FUNDAMENTALS
Olga Smolyakova , PhD
Oracle Certified Java 6 Programmer
Olga_Smolyakova@epam.com

Содержание

1. **Введение в язык Java**
2. **Типы данных, переменные**
3. **Операторы**
4. **Массивы**
5. **Элементарное про классы и объекты**

ВВЕДЕНИЕ В ЯЗЫК JAVA

<epam>

Введение в язык Java.

Что такое Java?

Java это

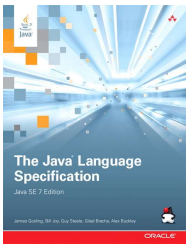
***объектно-ориентированный, платформенно-независимый* язык программирования**, используемый для разработки информационных систем, работающих в сети *Internet*

вычислительная ***платформа***

Введение в язык Java.

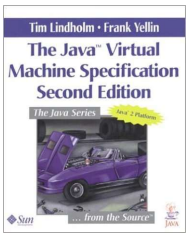
Java language specification

техническое описание синтаксиса и семантики языка программирования Java



Java Virtual Machine Specification

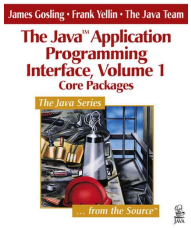
спецификация виртуальной машины Java



Введение в язык Java. API

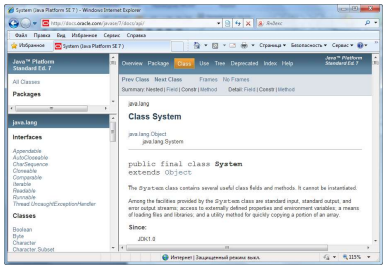
Application program interface

(API), или **library** (библиотека), **содержит предопределенные классы и интерфейсы**, используемые для разработки Java-программ.



Документация Java API

http://docs.oracle.com/javase/7/docs/api/



Введение в язык Java. Платформы Java

Java SE (Java Standart Edition)

разработка самостоятельных приложений на стороне клиента или апплетов

Java EE (Java Enterprise Edition)

разработка приложений на стороне сервера, таких как сервлеты, JSP, JSF

Java ME (Java Micro Edition)

разработка приложений для мобильных устройств

Введение в язык Java. JVM, JRE, JDK

JVM (Java Virtual Machine)

виртуальная машина Java, часть программного обеспечения Java, интерпретирующая байт-код, описываемый в класс-файлах.

JRE (Java Runtime Environment)

это Java Runtime Environment, среда выполнения Java, предназначена только для запуска готовых Java-приложений, а потому содержит лишь реализацию виртуальной машины и набор стандартных библиотек

JDK (Java Development Kit)

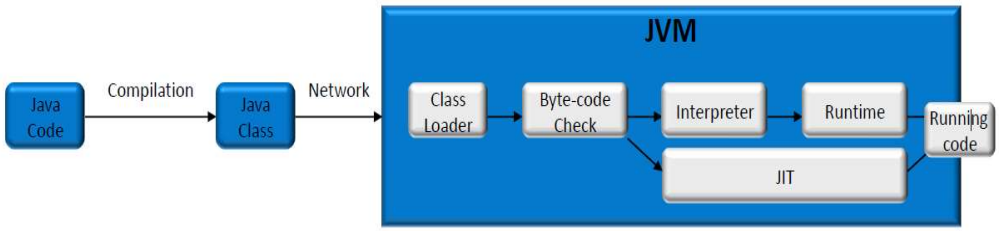
средство разработчика Java, включающее в себя набор утилит, стандартные библиотеки с их сходным кодом и набор демонстрационных примеров.

java – реализация JVM

javac – компилятор Java

javadoc – утилита для автоматической генерации документации

Введение в язык Java. JVM



<epam>

Введение в язык Java

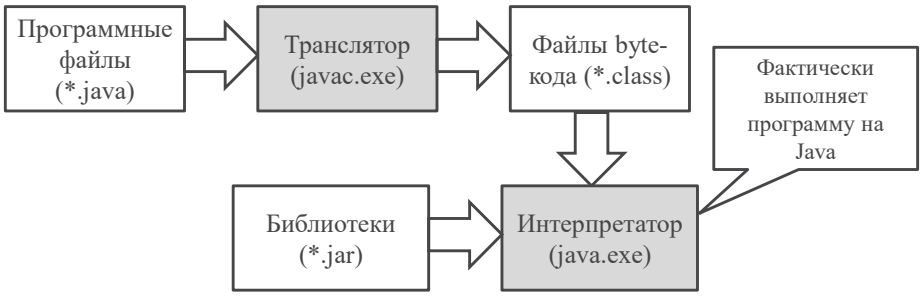
Простое линейное приложение

First.java

```
package start;

public class First {

     public static void main(String[] args){
          System.out.print("Java ");
          System.out.println("уже здесь!");
     }

}
```

Введение в язык Java.

Жизненный цикл программ на Java



Введение в язык Java.

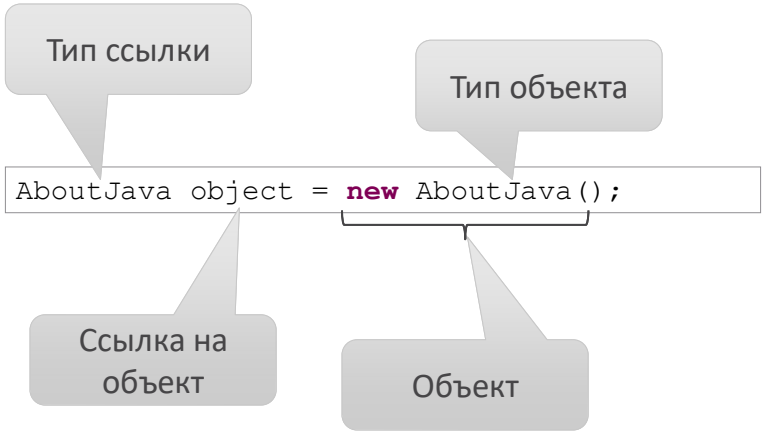
Простое объектно-ориентированное приложение

```
package firstoop;
public class AboutJava {
      public void printReleaseData(){
            System.out.println("Java уже здесь!");
      }
}
```

```
package firstoop;
public class FirstOOPProgram {
      public static void main(String[] args) {
            AboutJava object = new AboutJava();
            object.printReleaseData();
      }
}
```

Введение в язык Java

Простое объектно-ориентированное приложение



Введение в язык Java

Пакеты

```
package имя_пакета;
```

– это контейнеры классов, которые используются для разделения пространства имен классов.

```
package имя_пакета.имя_подпакета.имя_подпакета;
```

Для хранения пакетов используются *каталоги файловой системы*.

```
package one.two.three;
```

Пакеты **регулируют права доступа** к классам и подклассам.

Введение в язык Java. Import

import

Для подключения пакета используется ключевое слово **import**.

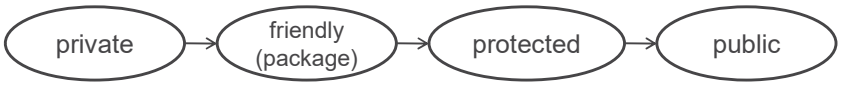
```
import имя_пакета.имя_подпакета.*;
```

```
import имя_пакета.имя_подпакета.имя_подпакета.имя_класса;
```

```
package start.mypackage.package2;

import start.mypackage.package1.Class1;

public class Class3 {
      public static void main(String[] args) {
            Class1 cl1 = new Class1();
      }
}
```

Введение в язык Java

Модификаторы доступа




```
class AboutJava{
	public String getAboutJava(){
		return info();
	}

	private String info(){
		return "About Java.";
	}
}
```

ТИПЫ ДАННЫХ, ПЕРЕМЕННЫЕ

Типы данных, переменные.

Примитивные типы

Простые типы делятся на 4 группы:

- *целые*: **int**, **byte**, **short**, **long**;
- *числа с плавающей точкой*: **float**, **double**;
- *символы*: **char**;
- *логические*: **boolean**.

Синтаксис Java позволяет создавать свои типы, получившие название ссылочных.

Типы данных, переменные.

Сравнение и особенности примитивных типов

Примитивный тип	Размер(бит)
boolean	-
char	16
byte	8
short	16
int	32
long	64
float	32
double	64
void	-

- Размер примитивных типов одинаков для всех платформ; за счет этого становится возможной переносимость кода
- Размер boolean неопределен. Указано, что он может принимать значения true или false.
- Преобразования между типом boolean и другими типами не существует.

Типы данных, переменные.

Переменные

```
тип идентификатор [ = значение];
```

```
boolean statusOn;
double javaVar = 2.34;

int itemsSold = 04;
double salary = 1.234e3;

float itemCost = 11.0f;
int i = 0xFd45, k$;

double _interestRate;
byte byteVar2 = 123;

long simpleVar = 1_000_000_000_000L;
```

В именах переменных не могут использоваться символы *арифметических* и *логических* операторов, а также символ '#'.

Применение символов '$' и '_' допустимо, в том числе и в первой позиции имени.

Типы данных, переменные.

Ключевые и зарезервированные слова языка Java

abstract	continue	for	new	switch
assert	default	goto	package	synchronized
boolean	do	if	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp	volatile
const	float	native	super	while

+ три литерала: **null**, **true**, **false**,

+ зарезервированные слова: **const**, **goto**.

Типы данных, переменные.

Преобразование примитивных типов

Java запрещает смешивать в выражениях величины разных типов, однако при числовых операциях такое часто бывает необходимо.

Различают:

- **повышающее** (разрешенное, неявное) преобразование;

```
int x = 200;
long z = x;
long value1 = 200;
```

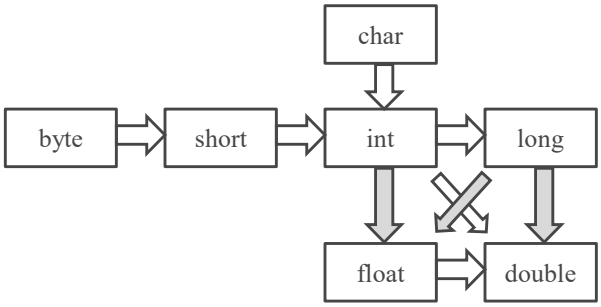
- **понижающее** (явное) приведение типа.

```
long value2 = 1000L;
int value3 = (int)value2;
```

Типы данных, переменные

Преобразование примитивных типов

Повышающее преобразование осуществляется автоматически, даже в случае потери данных.



Серыми стрелками обозначены преобразования, при которых может произойти потеря точности.

Типы данных, переменные

Приведение типов в выражении

При вычислении выражения (**a @ b**) аргументы **a** и **b** преобразовываются в числа, имеющие одинаковый тип:

- если одно из числе **double**, то в **double**;
- иначе, если одно из чисел **float**, то в **float**;
- иначе, если одно из чисел **long**, то в **long**;
- иначе оба числа преобразуются в **int**.

```
byte b1 = 50, b2 = 20, b3 = 127;
int x2 = b1 * b2 * b3;
System.out.println("x2 - " + x2);
```

Типы данных, переменные

Будьте осторожны

- Java не позволяет присваивать переменной значение более длинного типа.

```
int intVar = 1_000_000_000L;
```

- Исключение составляют операторы инкремента, декремента и операторы +=, –=, *=, /=.

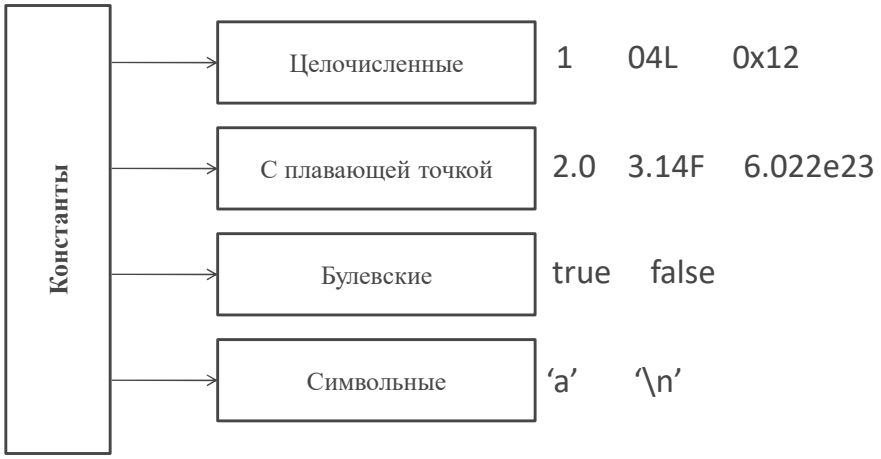
var @= expr == var = (typename)(var @ (expr))

```
int intVar = 100;
long longVar = 1000000000000L;
intVar += longVar;
```

```
intVar = intVar + longVar;
```

Типы данных, переменные

Литералы

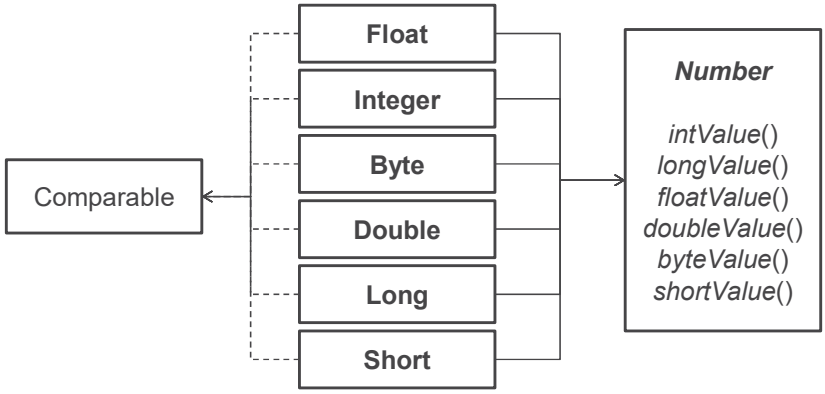


Типы данных, переменные.

Классы-оболочки

Кроме базовых типов данных широко используются соответствующие классы (wrapper классы)

Объекты этих классов являются константными.



Типы данных, переменные

Объекты классов-оболочек – константные объекты

```
public static void main(String[] args) {
      Integer i = new Integer(10);
      System.out.println("i1=" + i);
      changeInteger(i);
      System.out.println("i2=" + i);
}

public static void changeInteger(Integer x) {
      System.out.println("x1=" + x);
      x = new Integer(20);
      System.out.println("x2=" + x);
}
```

Типы данных, переменные

Автоупаковка/автораспаковка

Автоупаковка.

```
Integer  iob = 71;
```

Автораспаковка Вызовы таких методов, как **intValue()**, **doubleValue()** становятся излишними.

```
Integer j = 71; // создание объекта+упаковка
Integer k = ++j; // распаковка+операция+упаковка
int i = 2;
k = i + j + k;
System.out.println(k);
```

Типы данных, переменные

В классах **Long**, **Integer**, **Short** и **Byte** присутствует внутренний кеш ссылок на значения от -128 до 127

```
Integer i1 = 10;
Integer i2 = 10;
System.out.println(i1 == i2);

i1 = 128;
i2 = 128;
System.out.println(i1 == i2);
```

Результат:

```
true
false
```

ОПЕРАТОРЫ

Операторы

Арифметические операторы

+	/
+=	/=
–	%
–=	%=
*	++
*=	--

Битовые операторы

>> - арифметической сдвиг

>>> - логический сдвиг

|	>>
|=	>>=
&	>>>
&=	>>>=
^	<<
^=	<<=
~	

Логические операторы

Логические операции выполняются над значениями типа **boolean**

&& и || - *вычисление по сокращенной схеме*

& и | - *вычисление по полной схеме*

||, |, !=	&&, &, &=
!	^, ^=

```
if(bFalse()&&bTrue()){}
if(bFalse()&bTrue()) {}
```

Операторы отношения

<	>
<=	>=
==	!=

Instanceof – оператор принадлежности типу

[ ] – оператор доступа к элементу массива

?: – тернарный условный оператор

Операторы

Приоритет операций

№	Операция	Порядок выполнения операций в выражении при одном приоритете
1	**[ ] . () (вызов метода)**	Слева направо
2	**! ~ ++ -- +(унарный) -(унарный) () (приведение) new**	*Справа налево*
3	*** / %**	Слева направо
4	**+ - , +(конкатенация строк)**	Слева направо
5	**<< >> >>>**	Слева направо
6	**< <= > >= instanceof**	Слева направо
7	**== !=**	Слева направо
8	**&(битовое), &(логическое)**	Слева направо
9	**^(битовое), ^(логическое)**	Слева направо
10	**| (битовое), | (логическое)**	Слева направо
11	**&&**	Слева направо
12	**||**	Слева направо
13	**?:**	Слева направо
14	**= += -= *= /= %= |= ^= <<= >>= >>>=**	*Справа налево*

Операторы.

Операции над целыми числами

Аналогичны операциям большинства языков программирования.

Деление на ноль целочисленного типа **вызывает исключительную ситуацию**, **переполнение не контролируется** (исключение не выбрасывается).

Операции над числами с плавающей точкой

Все вычисления, которые проводятся над числами с плавающей точкой следуют стандарту **IEEE 754**.

По стандарту **IEEE 754** введены понятие бесконечности

+Infinity и

–Infinity и

значение **NaN** (Not a Number).

Операторы

Java вводит следующие значения в классах **Double** и **Integer**:

- **Double.MAX_VALUE;**
- **Double.MIN_VALUE;**
- **Double.POSITIVE_INFINITY;**
- **Double.NEGATIVE_INFINITY;**
- **Double.NaN;**
- **Double.isNaN().**
- **Float.MAX_VALUE;**
- **Float.MIN_VALUE;**
- **Float.POSITIVE_INFINITY;**
- **Float.NEGATIVE_INFINITY;**
- **Float.NaN;**
- **Float.isNaN().**

```
double i = 7.0;
double k;

System.out.println(i / 0);
System.out.println(-i / 0);
System.out.println(k=Math.sqrt(-i));

System.out.println(Double.isNaN(k));
```

Результат:

```
Infinity
-Infinity
NaN
true
```

Операторы

Особенности приведения вещественных чисел

- Слишком *большое дробное число* при приведении *к целому* превращается в **Integer.MAX_VALUE** или **Integer.MIN_VALUE**
- Слишком *большой double* при приведении *к float* превращается в **Float.POSITIVE_INFINITY** или **Float.NEGATIVE_INFINITY**

Операторы

Статический импорт

Ключевое слово **import** с последующим ключевым словом **static** используется для **импорта статических полей и методов классов**, в результате чего **отпадает необходимость в использовании имен классов** перед ними.

```
import static java.lang.Math.pow;
import static java.lang.Math.PI;

public class StaticImport {
	private int i = 20;

	public void staticImport() {
		double x;
		x = pow(i, 2)*PI;
		System.out.println("x=" + x);
	}
}
```

Операторы

Операторы управления

Оператор if:

```
if (boolexp) { /*операторы*/}
[else { /*операторы*/ }]
```

Циклы:

```
1. while (boolexpr) { /*операторы*/ }
2. do { /*операторы*/ }
   while (boolexp);
3. for(exp1; boolexp; exp3){ /*операторы*/ }
4. for((Тип exp1 : exp2){ /*операторы*/ }
```

break, continue

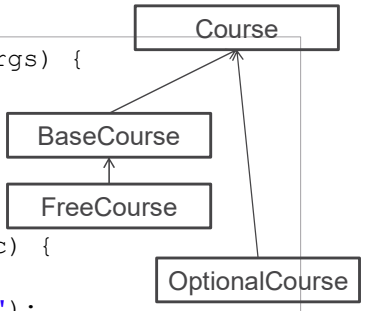
Оператор **switch**:

```
switch(exp) {
        case exp1:/*операторы, если
                  exp==exp1*/
                  break;
        case exp2:/*операторы, если
                  exp==exp2*/
                  break;
        default: /* операторы Java */
}
```

Операторы

Instanceof

Результатом действия оператора **instanceof** будет истина, если объект является объектом типа с которым идет поверка или одного из его подклассов, но не наоборот.




```
public static void main(String[] args) {
  doLogic(new BaseCourse());
  doLogic(new OptionalCourse());
  doLogic(new FreeCourse());
}

public static void doLogic(Course c) {
  if (c instanceof BaseCourse) {
    System.out.println("BaseCourse");
  } else if (c instanceof OptionalCourse) {
    System.out.println("OptionalCourse");
  } else {
    System.out.println("Что-то другое.");
  }
}
```

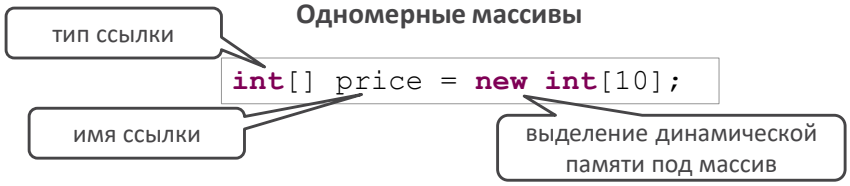
МАССИВЫ

Массивы

Определение

Массив – структура данных, для хранения однотипных значений.

Одномерные массивы




```
int[] rooms = new int[]{1,2,3};
```

```
int a[] = { 5, 10, 0, -5, 16, -2 };
```

Значения элементов **неинициализированных массивов**, для которых выделена память, устанавливается в **нуль**.

Массивы

Работа с массивами

Для доступа к i-му элементу массива **используется** оператор **[ ]**.

Индексация элементов массива начинается **с нуля**.

```
int a[] = { 5, 10, 0, -5, 16, -2 };
int max = a[0];
for (int i = 0; i < a.length; i++){
    if (max < a[i]){ max = a[i];}
}
```

Для последовательного доступа ко всех элементам массива можно использовать цикл for-each.

```
int a[] = { 5, 10, 0, -5, 16, -2 };
for(int x : a){
      System.out.print(x+" ");
}
```

Массивы

Массивы ссылок

Массивы объектов в действительности представляют собой **массивы ссылок**, проинициализированных по умолчанию значением **null**.

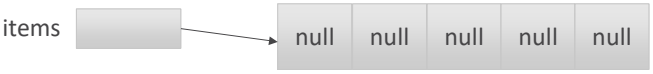
```
Item[] items = new Item[10];

Item[] definedItems = new Item[]
      { new Item(1), new Item(2), new Item(3) };
```

```
class Item {
      public Item(int i) {
      }
}
```

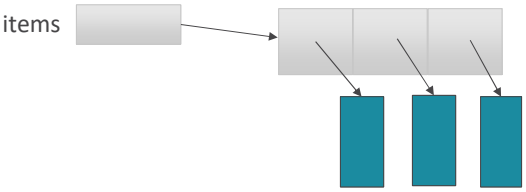
Массивы

```
int[] mas = new int[]{1,2,3,4,5};
```


```
Item[] items = new Item[5];
```



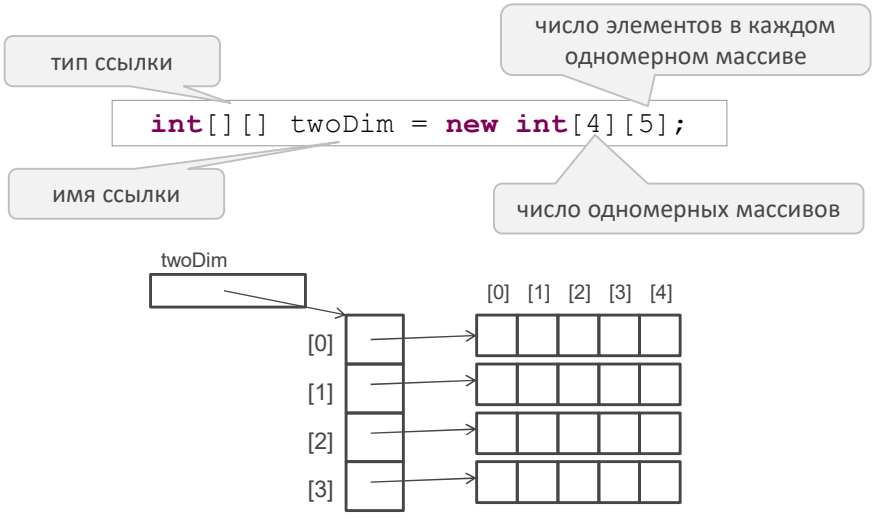

```
Item[] items = new Item[]
      { new Item(1), new Item(2), new Item(3) };
```



Массивы

Массив массивов

Двумерных массивов в Java нет. Есть только массивы массивов.



Массивы

Работа с массивами

Члены объектов-массивов:

- `public final int length` – это поле содержит длину массива
- `public Object clone()` – создает копию массива (но не объектов, ссылки на которые хранятся в массиве)
- + все методы класса `Object`.

```
int[][] twoDim = new int[4][];
…
twoDim.length; - число ссылок типа int[]
twoDim[i].length; - число элементов в одномерном
                                    массиве twoDim[i]
```

Массивы

Для доступа к элементу массива массивов используется оператор **[][].**

arr[2][0]

Каждый из массивов может иметь отличную от других длину.

```
int[][] twoDim = new int[4][];
twoDim[0] = new int [10];
twoDim[1] = new int [20];
twoDim[2] = new int [30];
twoDim[3] = new int [100];
```

```
int arr[][] = {
              { 1 },
              { 2, 3 },
              { 4, 5, 6 },
              { 7, 8, 9, 0 }
      };
```

Массивы

Ошибки времени выполнения

Обращение к несуществующему индексу массива отслеживается виртуальной машиной во время исполнения кода:

```
int array[] = new int[] { 1, 2, 3 };
System.out.println(array[3]);
```

```
java.lang.ArrayIndexOutOfBoundsException
```

Попытка поместить в массив неподходящий элемент пресекается виртуальной машиной:

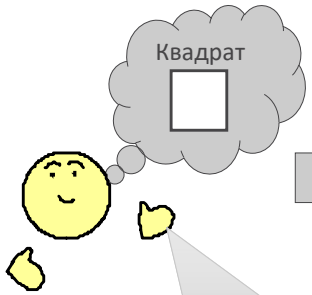
```
Object[] x = new String[3];
x[0] = new Integer(0);
```

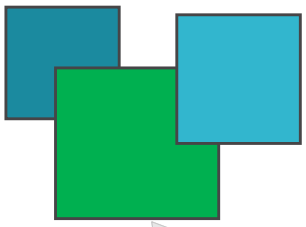
```
java.lang.ArrayStoreException
```

ЭЛЕМЕНТАРНОЕ ПРО КЛАССЫ И ОБЪЕКТЫ

Элементарное про классы и объекты

Определения





Класс – шаблон или АБСТРАКЦИЯ сущности предметной области

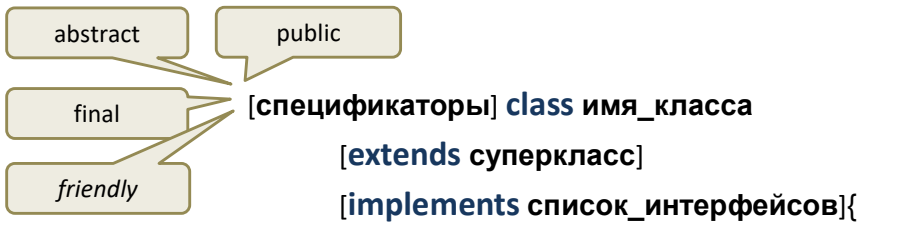
Объект – некоторая КОНКРЕТНАЯ сущность моделируемой предметной области

Классом называется *описание совокупности объектов с общими атрибутами, методами, отношениями и семантикой.*

Классы **определяют структуру и поведение** некоторого набора элементов предметной области.

Элементарное про классы и объекты

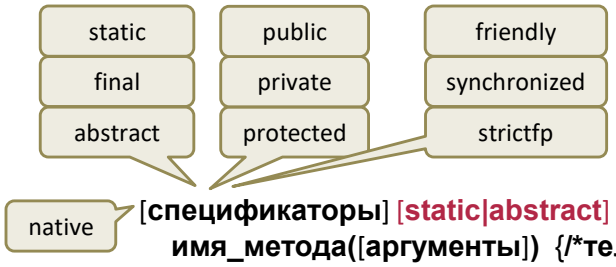
Объявление класса имеет вид:




```
public class Book {
}
```

Элементарное про классы и объекты

Объявление метода имеет вид:

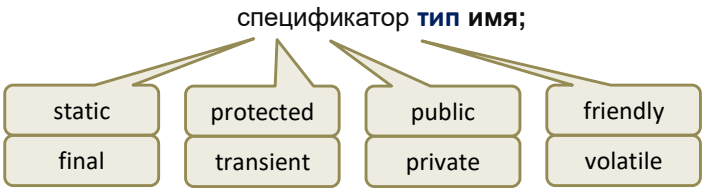


[спецификаторы] [static|abstract] возвращаемый_тип имя_метода([аргументы]) {/*тело метода*/} | ;

```
public class Book {
        public void setTitle(String title) {
                …
        }
        public String getTitle() {
                return title;
        }
}
```

Элементарное про классы и объекты

Объявление поля имеет вид:




```
public class Book {
        private String title;
        …
}
```

СПАСИБО ЗА ВНИМАНИЕ!

ВОПРОСЫ?

Java for students

Филиал кафедры ПОИТ БГУИР в Epam Systems
курс: Веб-технологии (JAVA)

Author: **Olga Smolyakova , PhD**
Oracle Certified Java 6 Programmer
Olga_Smolyakova@epam.com