

**Путь к Java ОСА.**

**Эффективное пособие для начинающих.**

# **Урок 1**



## Структура занятий

1. Краткий повтор пройденного материала

2. Проверка д/з

3. Новая Тема (Теория + Практика)

4. Подведение итогов

5. Д/з

# Java редакции



**Java Standard Edition (SE)** - это стандартная редакция, используемая для разработки стандартных приложений.

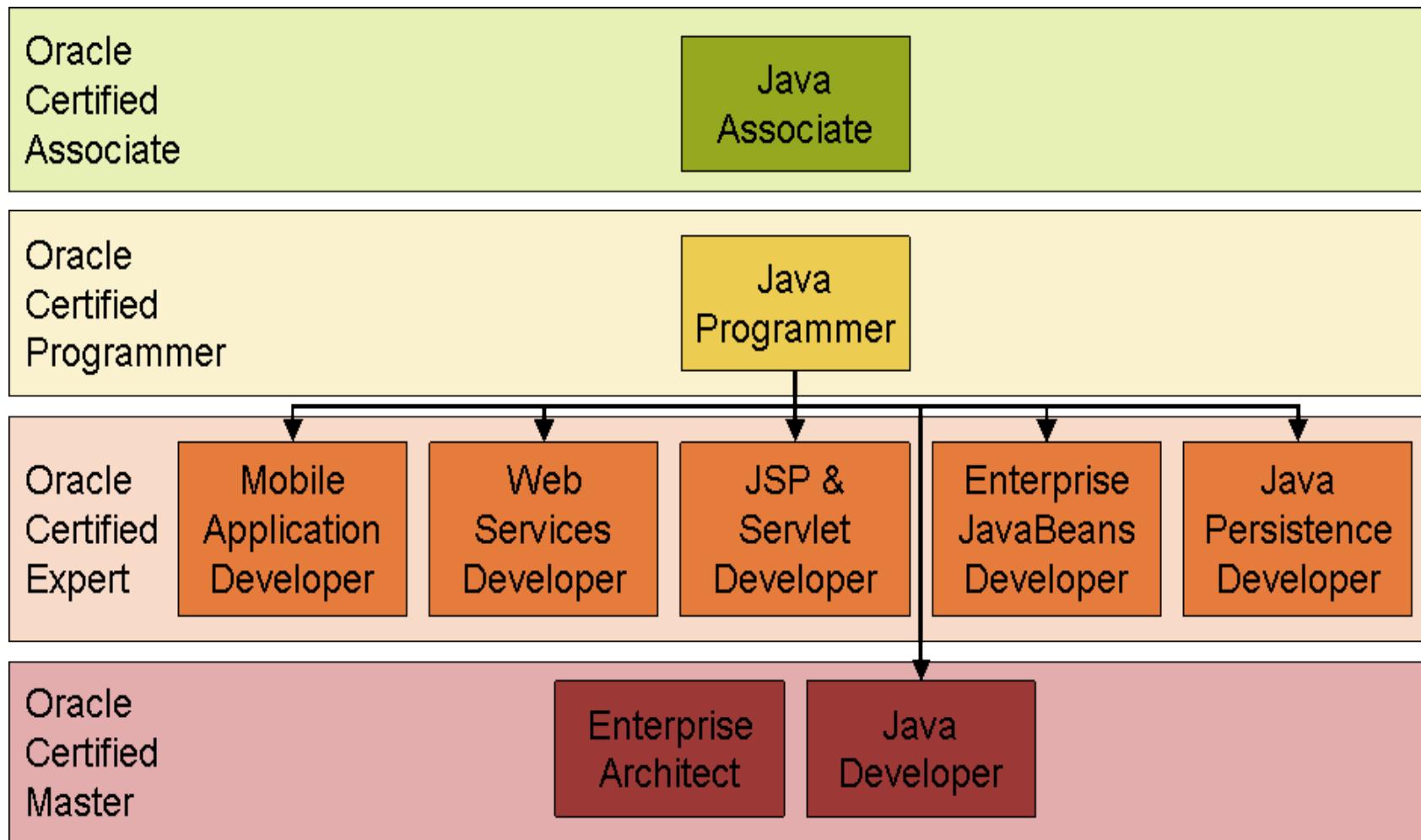


**Java Micro Edition (ME)** - редакция для разработки приложений для микрокомпьютеров – напр. мобильных телефонов.



**Java Enterprise Edition (EE)** - редакция для разработки приложений масштаба предприятия

## Иерархия экзаменов



# Версионность

Дата	Версия
23.01.1996	JDK 1.0
19.02.1997	JDK 1.1
08.12.1998	J2 SE 1.2
08.05.2000	J2 SE 1.3
06.02.2002	J2 SE 1.4
30.09.2004	J2 SE 5.0
11.12.2006	Java SE 6
07.07.2011	Java SE 7
18.03.2014	Java SE 8

PL/Sql

Delphi

Java

C#

**Язык программирования** — формальная знаковая система, предназначенная для написания компьютерных программ. Представляет собой набор слов специальных знаков и команд, «понятных» компьютеру.

Pascal

Assembler

C

Python

По возможным способам  
программирования языки можно  
поделить на 3 основные группы:

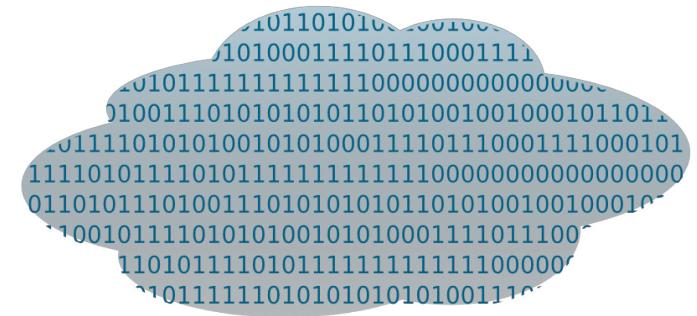
Процедурный - задачи разбиваются на шаги и решаются шаг за  
шагом

Функциональный - всё решение описывается при помощи  
функций

Объектно Ориентированный - решение проблемы производится  
при помощи функций и структур данных, описанных в классах.  
Из каждого класса можно создать объект, у которого будет  
набор свойств и методов.

# Поколения языков программирования

1-ое поколение – это машинный код = двоичный код.



2-ое поколение – это языки Ассемблера

```
mov    rRectData, eax
mov    ebx, eax
mov    bx,RECTDATA.cxClient[ebx]

mov    al,16
call   random           ; Get random number 16 bits
mov    edx,0
div    bx
mov    xLeft,edx         ; Get remainder (modulus)
```

Языки ассемблера и машинный код считаются языками низкого уровня.

## **Поколения языков программирования**

3-е поколение – это такие языки, как Fortran, Cobol, Basic, Pascal, Delphi, Python, C, C++, Java

4-е поколение – разработаны с целью упростить их изучение и использование.

5-е поколение – созданы для разработки систем искусственного интеллекта и для решения связанных с этой темой проблем.

**Данные 3 поколения считаются языками высокого уровня.**

**Определения 4-го и 5-го поколений Я.П. может существенно различаться в зависимости от литературы.**

# Уровни языков программирования

## Низкоуровневый Я.П.

Довольно  
трудный в  
изучении

Выглядит как совокупность  
байтов и мнемонических  
выражений

Написание  
программ  
занимает много  
времени

## Высокоуровневый Я.П.

Не трудный в  
изучении

Большая часть  
команд совпадает  
по значению со  
словами  
английского языка

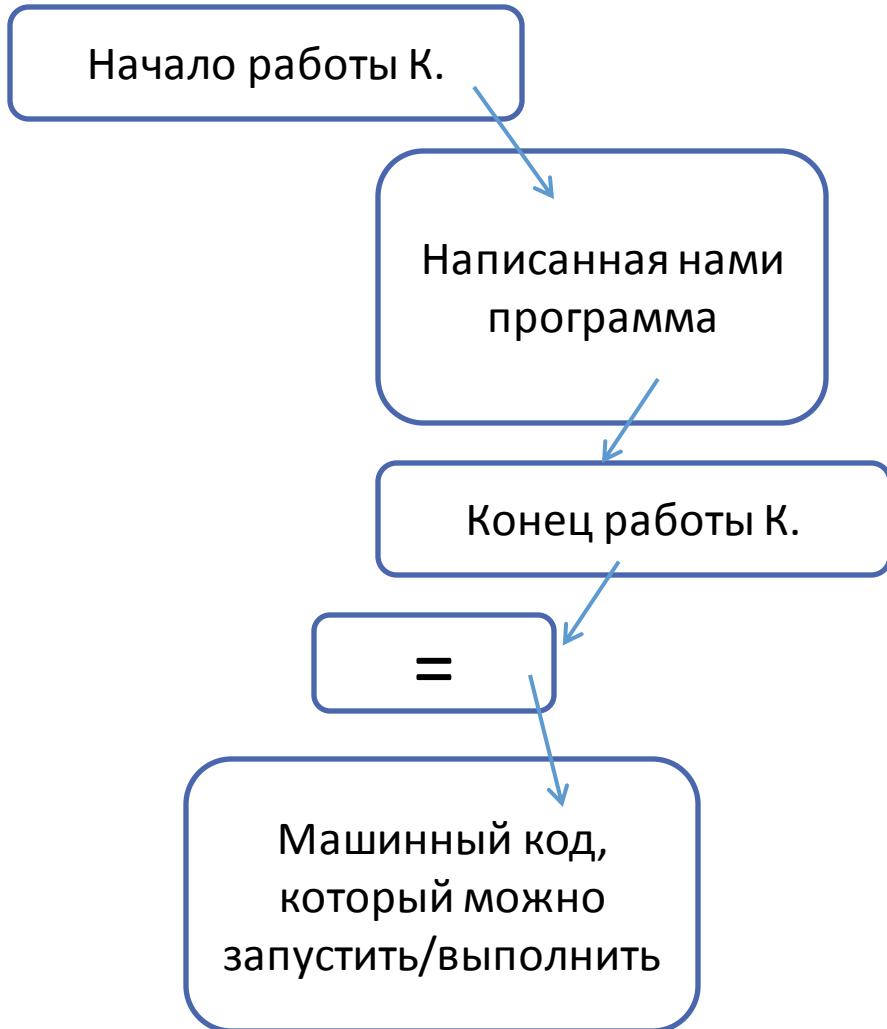
Написание  
программ  
занимает  
относительно  
немного  
времени

**Транслятор** выполняет перевод или трансляцию нашего написанного кода в машинный код – электрические сигналы

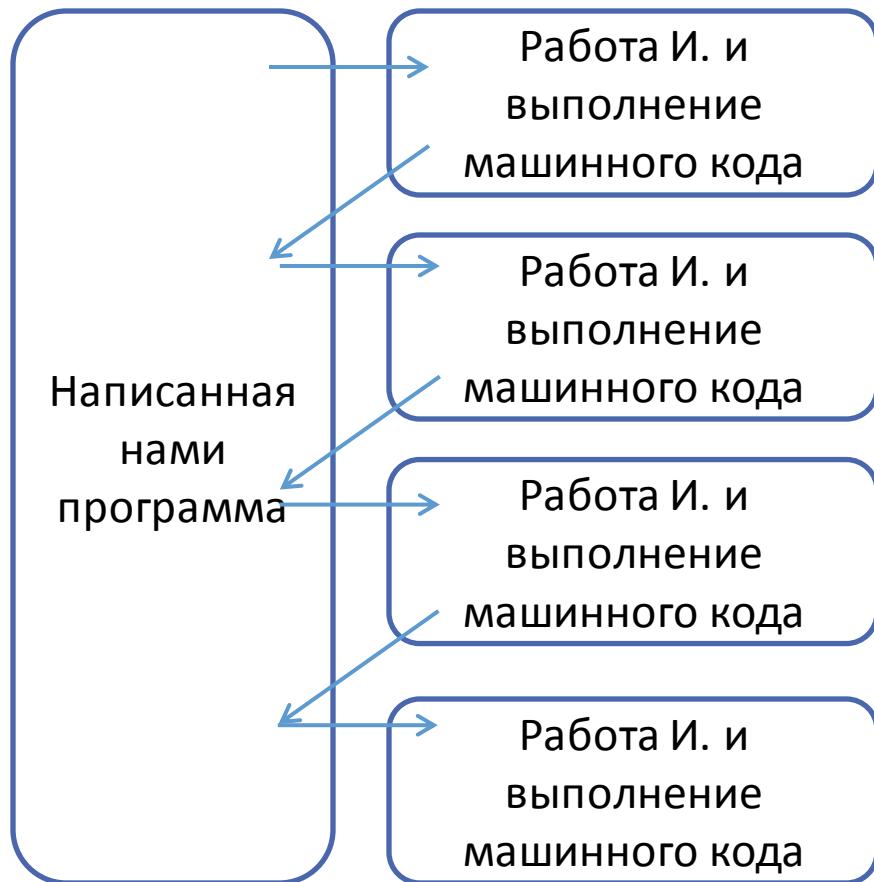
Компилятор читает текст на языке программирования от начала до конца, создавая эквивалентную программу на машинном языке.

Интерпретатор читает исходную программу по частям, сразу выполняя соответствующие действия.

## Компилятор



## Интерпритатор



# "Write Once, Run Anywhere"™



```
class Human{  
    String name;  
    int age;  
    char sex;  
  
    public String getName(){  
        return name;  
    }  
}
```

Сохраняем файл с расширением .java  
Получаем source file Human.java



Компилятор  
javac

JVM выполняет  
машичный код

На выходе имеем byte  
code, который  
находится в файле с  
расширением .class :  
Human.class

out

## Что необходимо скачать из Интернета?

JDK 1.7.0 - Java Development Kit. Это бесплатный комплект состоящий из:

- javac;
- standard library – совокупность классов;
- examples;
- documentation;
- different utilities;
- JRE - Java RuntimeEnvironment. Это минимальная реализация виртуальной машины, необходимая для исполнений Java приложений. JRE в свою очередь состоит из Virtual Machine и standard library.

IDE- Integrated development environment. Т.е. Интегрированная среда разработки - система программных средств, используемая программистами для разработки программного обеспечения. Интегрированные среды разработки были созданы для того, чтобы максимизировать производительность программиста.

## Новые методы (функции)

```
System.out.print("Welcome to the first lesson!");
```

Выводит на дисплей (экран) необходимую информацию

```
System.out.println("Добро пожаловать!");
```

Выводит на дисплей (экран) необходимую информацию и переводит курсор на следующую строку

## Подведение итогов

Java SE 7

OOP

Compiler

method  
main



OCA  
1Z0-803

High-  
level

JVM

methods  
print  
println

## Домашнее задание:

1. Вывести на дисплей рубаи (четверостишье), соблюдая все отступы в следующем виде:

### РУБАИ

Много лет размышлял я над жизнью земной.  
Непонятного нет для меня под луной.  
Мне известно, что мне ничего не известно!  
Вот последняя правда, открытая мной.

О. Хайям

*The End*

**Путь к Java ОСА.**

**Эффективное пособие для начинающих.**

# **Урок 2**

Краткий повтор пройденного материала

```
public static void main (String [] args){our code};
```

```
System.out.print("We will learn data types");
```

```
System.out.println("Успехов в обучении");
```

## Проверка д/з

1. Вывести на дисплей рубаи (четверостишье), соблюдая все отступы в следующем виде:

### РУБАИ

Много лет размышлял я над жизнью земной.

Непонятного нет для меня под луной.

Мне известно, что мне ничего не известно!

Вот последняя правда, открытая мной.

О. Хайям

Ответ: программа Homework.Lesson1

## Кухня. Контейнеры для еды.



## Проведём аналогию:

Разновидности еды – Типы данных (Data type)

Контейнеры – Переменные (Variables)

## Определения:

Тип данных – это множество допустимых значений этих данных, а также совокупность операций над ними.

Переменная – именованная область памяти, адрес которой можно использовать для осуществления доступа к данным и манипуляций над ними в ходе выполнения программы.

## Составные части переменной:

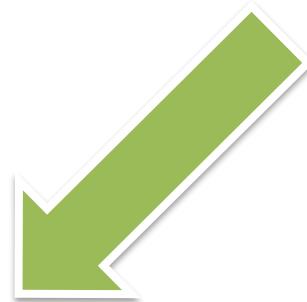
**Тип**

**Имя**

**Размер**

**Значение**

## Типы данных в Java:



**Простые**  
**(primitive)**



**Ссылочные**  
**(reference)**

# Простые типы данных (primitive data types)

Численные  
(numeric)

Символьный  
(character)

Логический  
(boolean)

Целые  
(integers)

Дробные  
(floating-point)

byte

short

int

long

float

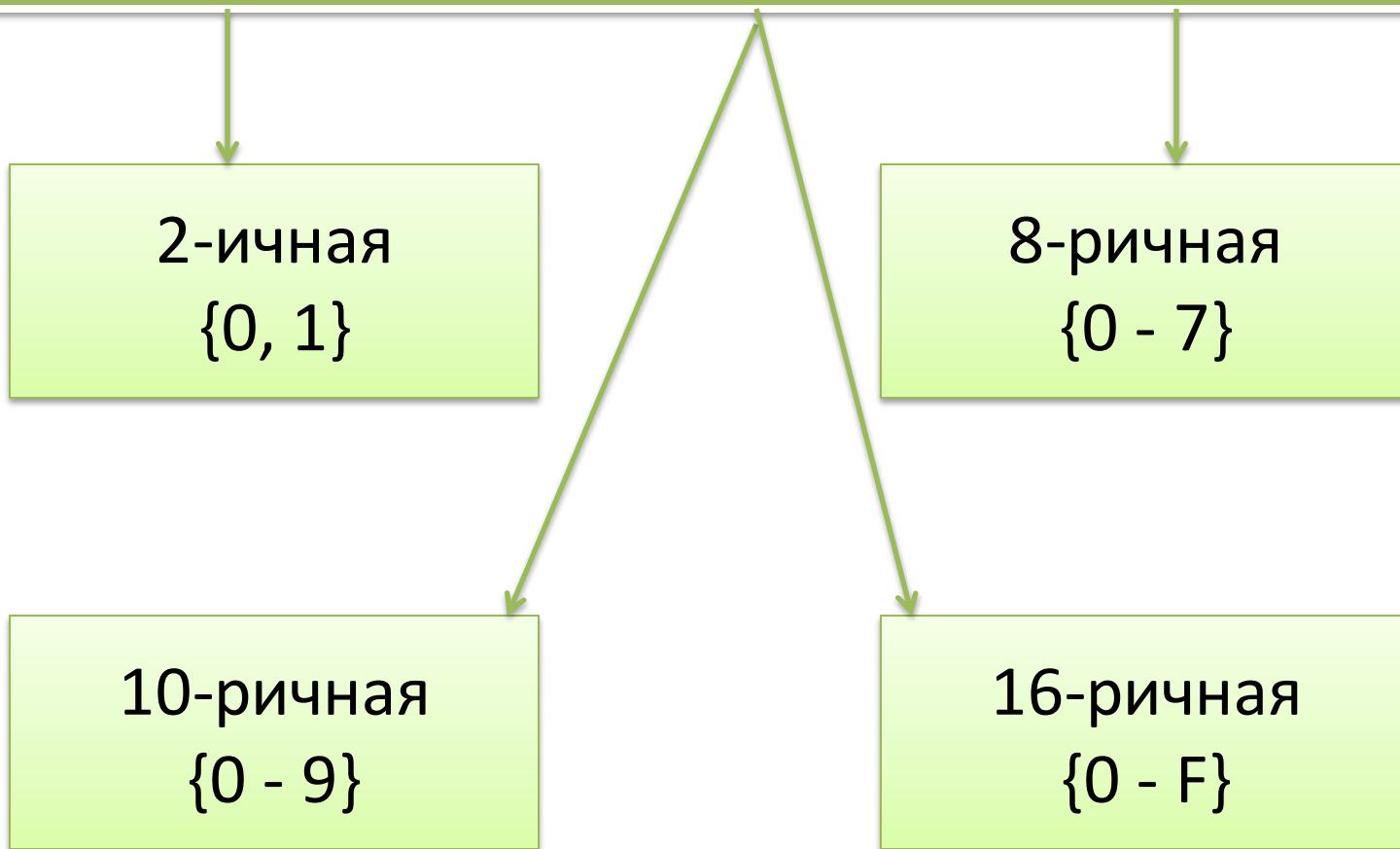
double

char

boolean

Тип данных	Размер	Диапазон
<b>byte</b>	8 бит	[-128 ; 127]
<b>short</b>	16 бит	[-32768 ; 32767]
<b>int</b>	32 бит	[-2147483648 ; 2147483647]
<b>long</b>	64 бит	[-9223372036854775808 ; 9223372036854775807]
<b>float</b>	32 бит	[3.4e-38 ; 3.4e+38]
<b>double</b>	64 бит	[1.7e-308 ; 1.7e+308]
<b>char</b>	16 бит	[0 ; 65535]
<b>boolean</b>	строго не определён	{true, false}

# Системы счислений



Где нельзя использовать символ \_

В начале и конца числа

До и после точки в дробных числах

До и после букв L, l, F, f, D, d

До, после и между 0X, 0x, 0B, 0b

Иключение: символ \_ можно использовать после «0»,  
который указывает, что число в 8-ричном формате

## Подведение итогов

variable

Logical  
d.t.

Numbering  
system

Compile  
time errors

Numeric  
d.t.

Character  
d.t.

Symbol  
—



## Домашнее задание:

1. Создать по 4 переменные всех целочисленных типов данных в следующем виде:
  - Все 4 переменные типа byte должны равняться 12 и быть записаны в разных системах счисления;
  - Все 4 переменные типа short должны равняться -1300 и быть записаны в разных системах счисления;
  - Все 4 переменные типа int должны равняться 0 и быть записаны в разных системах счисления;
  - Все 4 переменные типа long должны равняться 123456789 и быть записаны в разных системах счисления;

И вывести их на экран.
2. Создать по 2 переменные типов данных float, double и boolean. И вывести их на экран.
3. Создать n-ное количество переменных типа данных char всеми возможными способами. И вывести их на экран.

*The End*

**Путь к Java ОСА.**

**Эффективное пособие для начинающих.**

# **Урок 3**

Краткий повтор пройденного материала

byte, short, int, long, float, double, char, boolean

Binary, octal, decimal, hexadecimal  
number notation

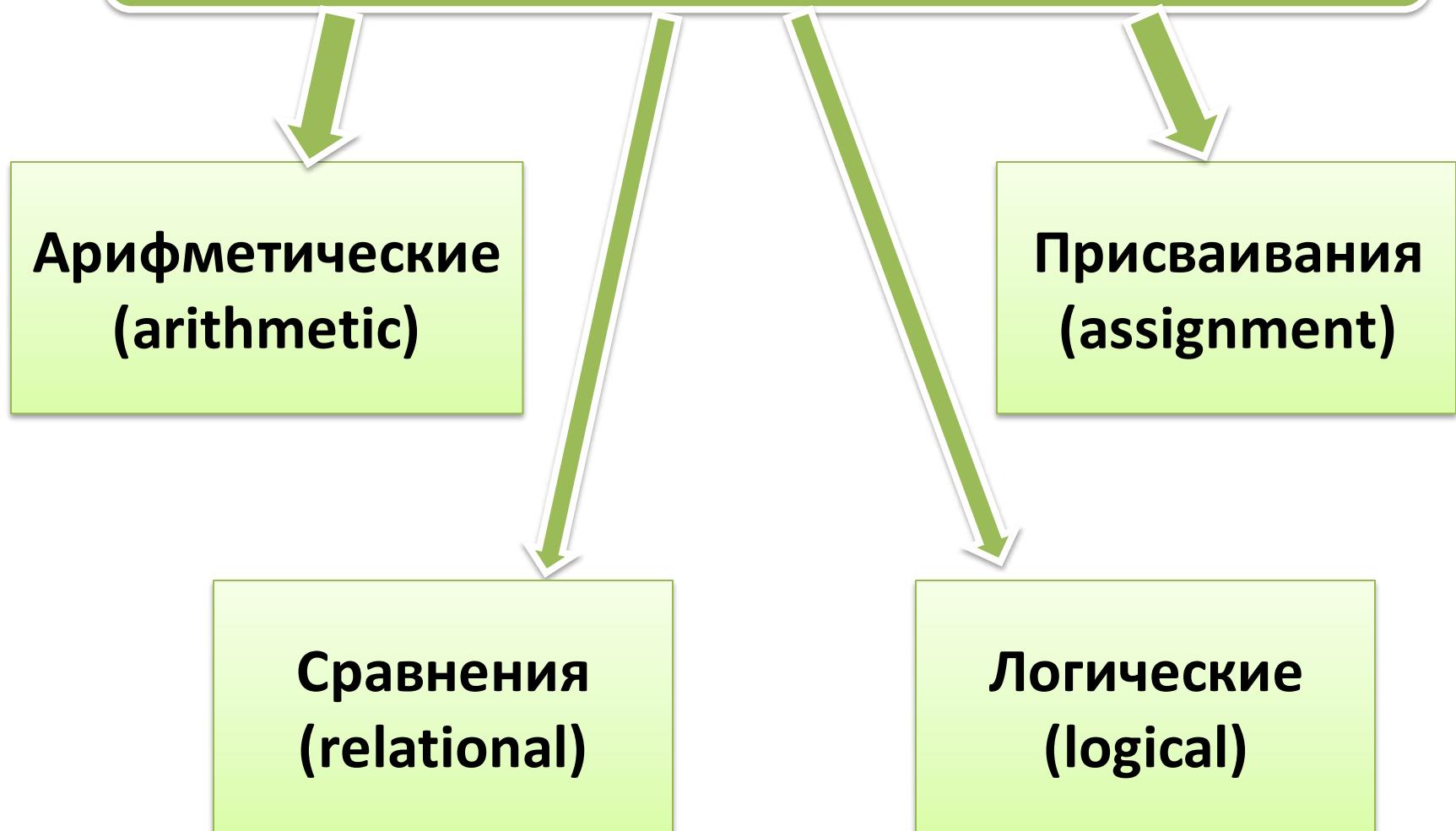
Underscore symbol

# Проверка д/з

1. Создать по 4 переменные всех целочисленных типов данных в следующем виде:
  - Все 4 переменные типа byte должны равняться 12 и быть записаны в разных системах счисления;
  - Все 4 переменные типа short должны равняться -1300 и быть записаны в разных системах счисления;
  - Все 4 переменные типа int должны равняться 0 и быть записаны в разных системах счисления;
  - Все 4 переменные типа long должны равняться 123456789 и быть записаны в разных системах счисления;И вывести их на экран.
2. Создать по 2 переменные типов данных float, double и boolean. И вывести их на экран.
3. Создать n-ное количество переменных типа данных char всеми возможными способами. И вывести их на экран.

Ответ: программа Homework.Lesson2

## Типы Операторов



# Арифметические операторы

+

-

\*

%

/

++

--

# Операторы присваивания

=

+ =

- =

\* =

/ =

Тип данных	Размер	Диапазон
<b>byte</b>	8 бит	[-128 ; 127]
<b>short</b>	16 бит	[-32768 ; 32767]
<b>int</b>	32 бит	[-2147483648 ; 2147483647]
<b>long</b>	64 бит	[-9223372036854775808 ; 9223372036854775807]
<b>float</b>	32 бит	[3.4e-38 ; 3.4e+38]
<b>double</b>	64 бит	[1.7e-308 ; 1.7e+308]
<b>char</b>	16 бит	[0 ; 65535]
<b>boolean</b>	строго не определён	{true, false}

# Операторы сравнения

>

>=

<

<=

==

!=

# Логические операторы

&&

||

!

&

|

Λ

<b>a</b>	<b>b</b>	<b>a &amp;&amp; b</b>
true	true	true
true	false	false
false	true	false
false	false	false

<b>a</b>	<b>b</b>	<b>a    b</b>
true	true	true
true	false	true
false	true	true
false	false	false

<b>a</b>	<b>! b</b>
true	false
false	true

<b>a</b>	<b>b</b>	<b><math>a \wedge b</math></b>
true	true	false
true	false	true
false	true	true
false	false	false

Prece-dence	Operator
1	<code>++ -- !</code>
2	<code>* / %</code>
3	<code>+ -</code>
4	<code>&lt; &lt;= &gt; &gt;=</code>
5	<code>== !=</code>
6	<code>&amp;&amp;   </code>
7	<code>= += -= *= /= %=</code>

## Подведение итогов

Арифметические  
операторы

Операторы  
сравнения



Операторы  
присваивания

Логические  
операторы

Приоритет  
операторов

## Домашнее задание:

1. Вычислите следующую часть кода:

```
int i1=5;  
int i2=11;  
double d1 = 5.5;  
double d2 = 1.3;  
long l = 20l;  
double result=0;  
result = i2 / d1 + d2 % i1 - l;
```

2. Чему равны выражения:

- $a-- - -a + ++a + a++ + a$ ; где  $a=5$
- $++b - b++ + ++b - --b$ ; где  $b=8$

*The End*

**Путь к Java ОСА.**

**Эффективное пособие для начинающих.**

# **Урок 4**

# Краткий повтор пройденного материала

Арифметические операторы: + ; - ; \* ; / ; % ; ++ ; --

Операторы присваивания: = ; += ; -= ; \*= ; /=

Операторы сравнения: > ; < ; >= ; <= ; == ; !=

Логические операторы : && ; || ; ! ; & ; | ; ^

Приоритет операторов

# Проверка д/з

1. Вычислите следующую часть кода:

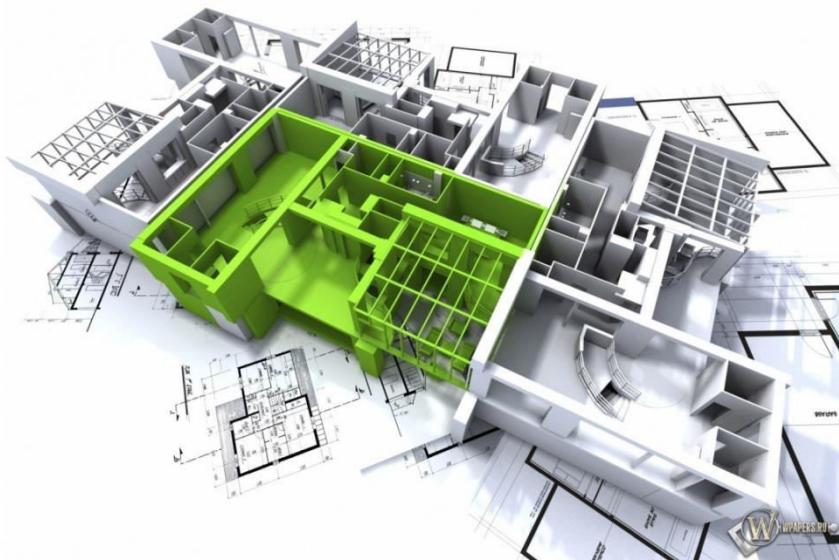
```
int i1=5;  
int i2=11;  
double d1 = 5.5;  
double d2 = 1.3;  
long l = 20l;  
double result=0;  
result = i2 / d1 + d2 % i1 - l;
```

2. Чему равны выражения:

- $a-- - -a + ++a + a++ + a$ ; где  $a=5$
- $++b - b++ + ++b - --b$ ; где  $b=8$

Ответ: программа Homework.Lesson3

# Аналогии



**Класс  
(class)**



**Объект  
(object)**

# Аналогии

**Bank account:**  
Id  
Name  
Balance

**Класс  
(class)**

**My account:**

Id: 1  
Name: Zaur  
Balance: 15\$

**Your account:**

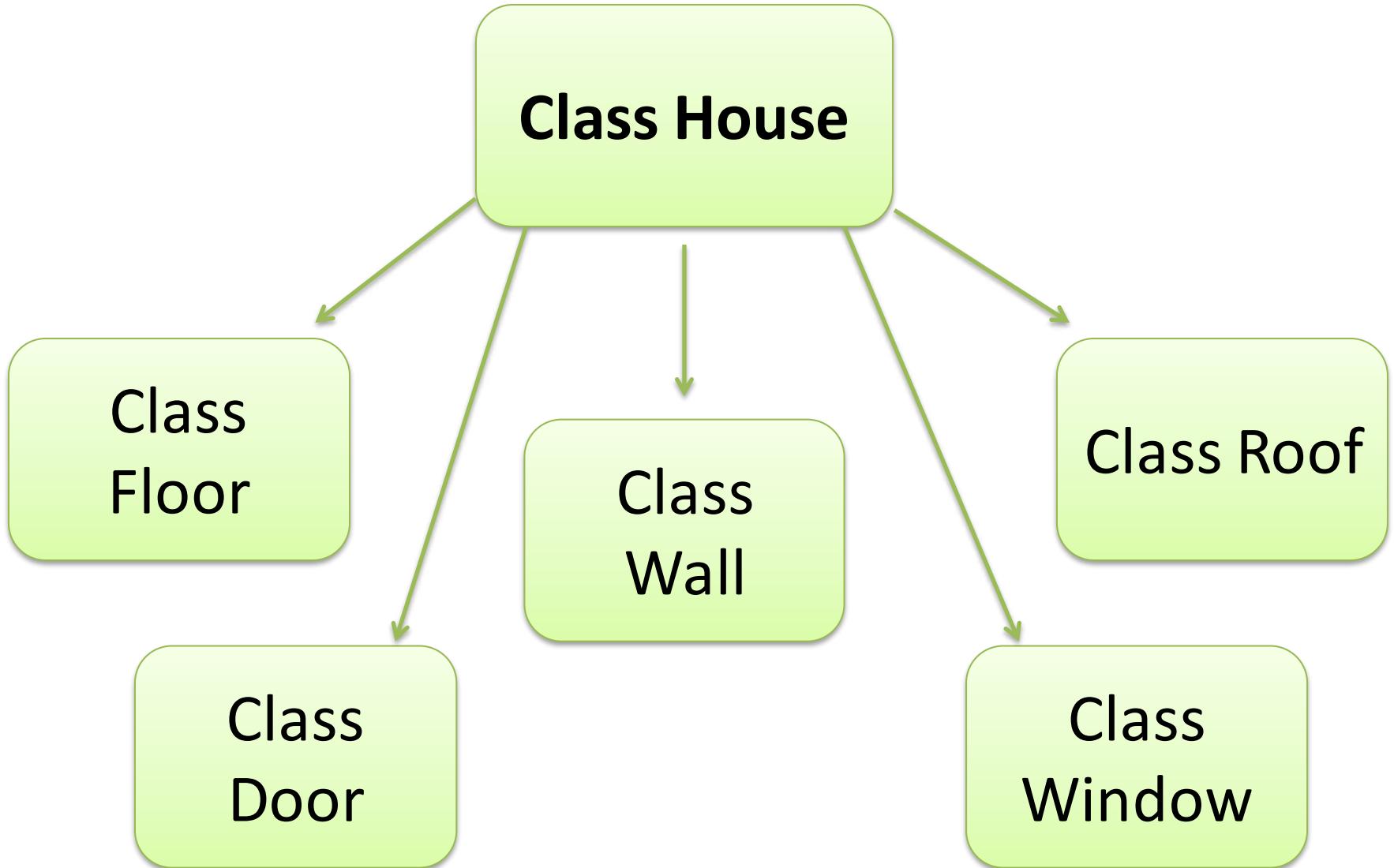
Id: 2  
Name: Mike  
Balance: 30\$

**His account:**

Id: 3  
Name: Ivan  
Balance: 10\$

**Объект  
(object)**

## Dividing into elements and Reusing



# Structure of Java class

The package statement

The import statement

Class declarations and definitions

Variables  
(State)

Methods  
(Behavior)

Constructors

Comments

# Object creation

BankAccount

bA =

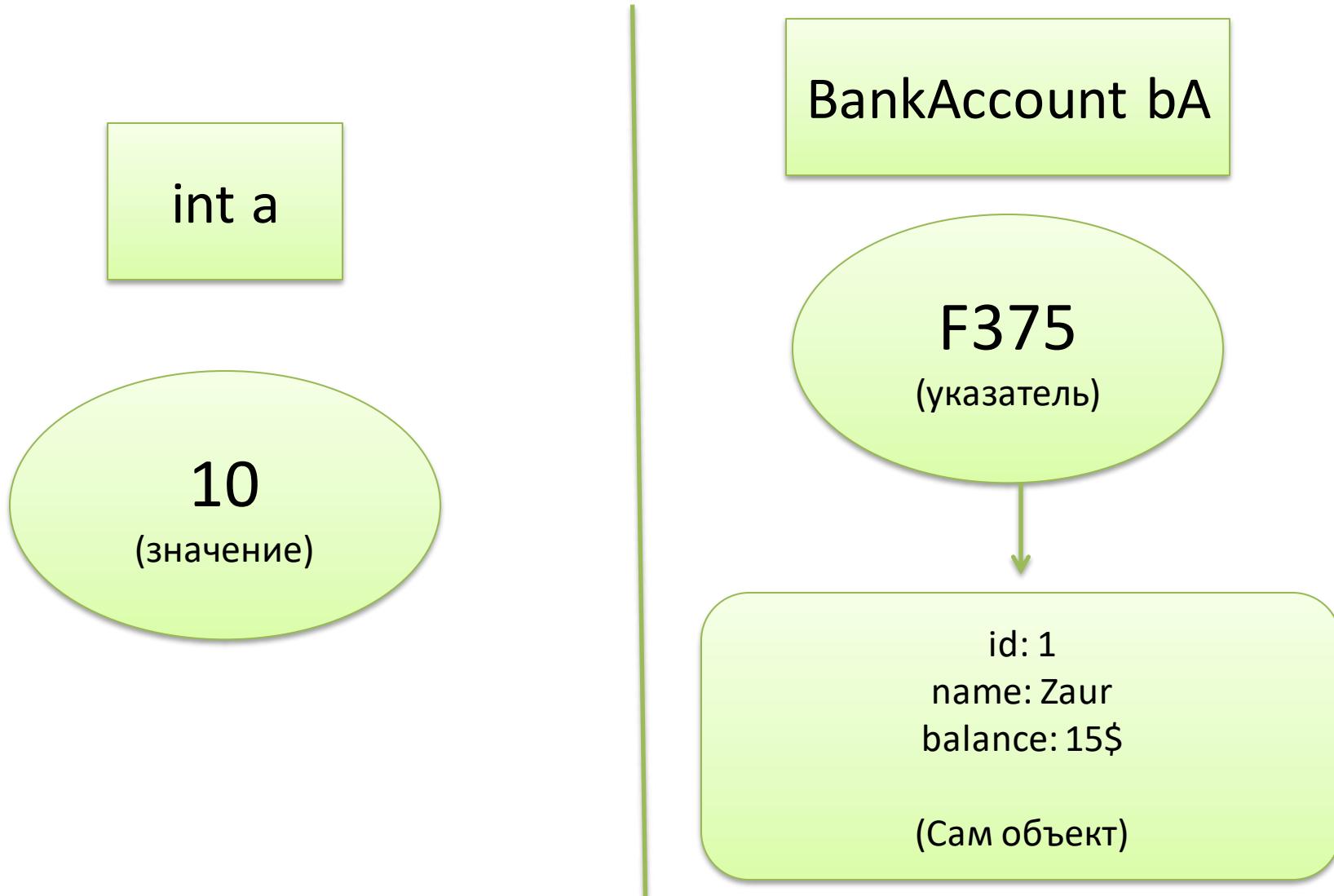
new BankAccount();

Тип данных  
переменной

Имя  
перемен-  
ной

Вызов  
конструктора –  
создание объекта

# Difference between primitive and reference data types



# Default значения типов данных

Primitive:

byte  
short } 0  
int  
long }  
float } 0.0  
double }  
char - '\u0000' или 0  
boolean - false

Reference:

String  
BankAccount  
Car  
House } null (ничто)

## 2 типа создания объекта String

1. `String name = "Zaur";`
2. `String name = new String("Zaur");`

## Подведение итогов

**Class/Object**

**Instance  
variables**

**Differences  
between data  
types**

**Constructor**

**Class structure**

**Reference data  
type**

**String**



## Домашнее задание:

1. Создайте 2 класса. 1-ый назовите `Student`. Он должен содержать в себе следующие атрибуты: номер студенческого билета, имя, фамилию, год обучения, средняя оценка по математике, средняя оценка по экономике, средняя оценка по иностранному языку. 2-ой класс назовите `StudentTest`. В нём Вы должны будете создать 3 разных объекта класса `Student`, вывести на экран в читабельном виде среднюю арифметическую оценку каждого студента (учитывая все 3 предмета).

P.S.: Страйтесь писать код, понятный не только для Вас, а для всех.

*The End*

**Путь к Java ОСА.**

**Эффективное пособие для начинающих.**

# **Урок 5**

# Краткий повтор пройденного материала

Понятия class, object. Class structure

Instance variables

Reference data types. Differences between reference and primitive data types

Constructor

class String

## Проверка д/з

1. Создайте 2 класса. 1-ый назовите Student. Он должен содержать в себе следующие атрибуты: номер студенческого билета, имя, фамилию, год обучения, средняя оценка по математике, средняя оценка по экономике, средняя оценка по иностранному языку. 2-ой класс назовите StudentTest. В нём Вы должны будете создать 3 разных объекта класса Student, вывести на экран в читабельном виде среднюю арифметическую оценку каждого студента (учитывая все 3 предмета).

P.S.: Страйтесь писать код, понятный не только для Вас, а для всех.

Ответ: программа Homework.Lesson4.Student

## Constructor

```
Car car1 = new Car();  
BankAccount bA = new BankAccount();  
String name = new String("Petr");
```

Конструктор всегда называется также как и класс:  
**name of constructor = name of class**

# Типы Констуктора



**Default**



**User defined**

Создаётся  
компилятором

Создаётся нами

Всегда без  
параметров

Может быть с  
параметрами или без

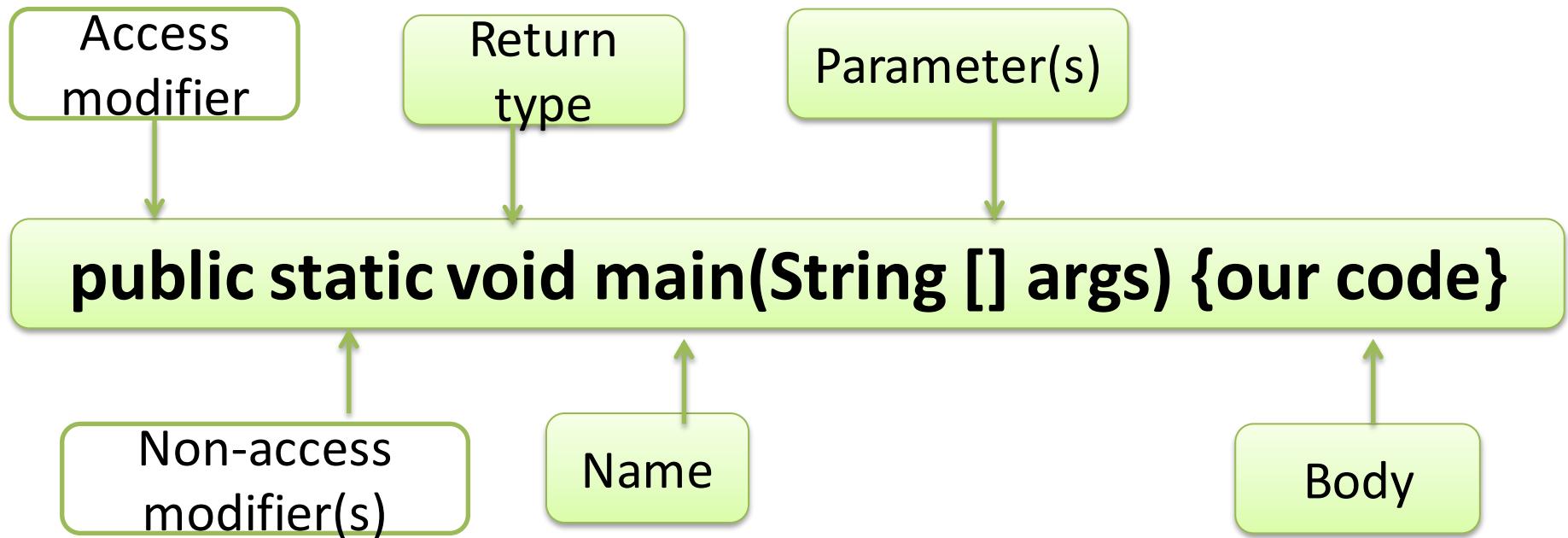
Тело всегда пустое

Тело может быть  
пустым или нет

Method



## Структура метода



## Создание метода (method creation)

```
int sum(int a, int b, int c) {  
    int result = a+b+c;  
    return result;  
}
```

## Вызов метода (method call)

```
int summaTrexChisel = sum(10, 5, 12);
```

## Difference between method and constructor

### Method

Всегда имеет return type

Можно придумать бесчисленное разнообразие имён

### Constructor

Никогда не имеет return type

Имя должно совпадать с именем класса

## Подведение итогов

**Default  
constructor**

**Return type**

**Method  
creation**



**User defined  
constructor**

**Parameter/  
argument**

**Method call**

**Differences  
between  
method and  
constructor**

## Домашнее задание:

1. В класс BankAccount добавьте 2 метода. Первый метод называется `popolnenieScheta` и увеличивает баланс на сумму, которая указана в параметре этого метода. Второй метод называется `snyatieSoScheta` и уменьшает баланс на сумму, которая указана в параметре этого метода.
2. Измените класс StudentTest так, чтобы среднюю арифметическую оценку студента выводил на экран метод. Т. е. создайте 1 метод, параметр которого – это объект класса Student, а в теле метода будет вычисляться средняя арифметическая оценка и выводиться на экран.
3. Создайте класс Employee с атрибутами `id`, `surname`, `age`, `salary`, `department`, которые должны задаваться в конструкторе. В данном классе также создайте метод увеличения зарплаты вдвое. Создайте второй класс EmployeeTest, в котором создайте 2 объекта класса Employee. Увеличьте зарплату каждому работнику вдвое с помощью метода и выведите на экран значение новой зарплаты.

*The End*

**Путь к Java ОСА.**

**Эффективное пособие для начинающих.**

# **Урок 6**

## Краткий повтор пройденного материала

Понятия constructor. Default & user defined constructors.

Понятия method

Создание method-а. Вызов method-а.

Return type

Parameters/Arguments

Difference between methods & constructors

## Проверка д/з

1. В класс BankAccount добавьте 2 метода. Первый метод называется `popolnenieScheta` и увеличивает баланс на сумму, которая указана в параметре этого метода. Второй метод называется `snyatieSoScheta` и уменьшает баланс на сумму, которая указана в параметре этого метода.
2. Измените класс StudentTest так, чтобы среднюю арифметическую оценку студента выводил на экран метод. Т. е. создайте 1 метод, параметр которого – это объект класса Student, а в теле метода будет вычисляться средняя арифметическая оценка и выводиться на экран.
3. Создайте класс Employee с атрибутами `id`, `surname`, `age`, `salary`, `department`, которые должны задаваться в конструкторе. В данном классе также создайте метод увеличения зарплаты вдвое. Создайте второй класс EmployeeTest, в котором создайте 2 объекта класса Employee. Увеличьте зарплату каждому работнику вдвое с помощью метода и выведите на экран значение новой зарплаты.

Ответ: программы пакета Homework.Lesson5

# Method Overloading

**Перезагруженные методы имеют одинаковые имена  
и разный список параметров**

Разный по типам данных:

```
void method (int a, String b)      { }  
void method (double a, boolean b) { }
```

Разный по количеству:

```
void method (int a, int b)      { }  
void method (int a, int b, int c) { }
```

Разный по порядку:

```
void method (int a, String b)      { }  
void method (String a, int b)      { }
```

## Rules of Overloading

Return type может быть одинаковый и различный:

```
void method (int a, String b) { }
```

```
void method (double a, boolean b) { }
```

```
int method (long a) { return 5; }
```

Access modifier может быть одинаковый и различный:

```
public void method (int a, int b) { }
```

```
private void method (int a, int b, int c) { }
```

```
public int method (long a) { return 5; }
```

Методы, отличающиеся только return type или access modifier не являются overloaded (**Compile ERROR**):

```
public void method (int a, String b) { }
```

```
private void method (int a, String b) { }
```

```
public String method (int a, String b) { return "a"; }
```

# Constructor Overloading

**Перезагруженные конструкторы имеют *разный список параметров***

Разный по типам данных:

Constructor (int a, String b) { }  
Constructor (double a, boolean b) { }

Разный по количеству:

Constructor (int a, int b) { }  
Constructor (int a, int b, int c) { }

Разный по порядку:

Constructor (int a, String b) { }  
Constructor (String a, int b) { }

## Rules of Overloading

Access modifier может быть одинаковый и различный:

```
public Constructor(int a, int b) { }
```

```
private Constructor(int a, int b, int c) { }
```

```
public Constructor(long a) { }
```

Конструкторы, отличающиеся только access modifier не являются overloaded (**Compile ERROR**):

```
public Constructor(int a, String b) { }
```

```
private Constructor(int a, String b) { }
```

this

Конструктор не может вызвать внутри себя overloaded конструктор по имени класса (**Compile ERROR**) :

```
Constructor(int a) { }
```

```
Constructor(int a, int b) { Constructor(5); }
```

Используйте “this” на первой строке в теле для вызова overloaded конструктора внутри конструктора :

```
public Constructor(int a) { }
```

```
private Constructor(int a, String b) { this(5);
                                         System.out.println("It is correct!"); }
```

## Подведение итогов

**Method  
overloading**

**Constructor  
overloading**

**“this”  
statement**

**Runtime &  
Compile time  
error**



## Домашнее задание:

1. Создайте класс, в котором будут 5 overloaded методов, которые вычисляют сумму нуля, одного, двух, трёх и четырёх целых чисел соответственно, передают эту сумму в output и выводят её на экран. В случае, когда слагаемые отсутствуют (т.е. когда параметров нет) сумма пусть равняется 0.
2. Измените класс Student так, чтобы он имел 3 конструктора. 1-ый принимает все параметры. 2-ой – только id, name, surname, course. 3-ий не принимает значений. Создайте в классе StudentTest 3 объекта с помощью разных конструкторов.

*The End*

**Путь к Java ОСА.**

**Эффективное пособие для начинающих.**

# **Урок 7**

Краткий повтор пройденного материала

Method overloading

Constructor overloading

“this” statement

Runtime & Compile time error

## Проверка д/з

1. Создайте класс, в котором будут 5 overloaded методов, которые вычисляют сумму нуля, одного, двух, трёх и четырёх целых чисел соответственно, передают эту сумму в output и выводят её на экран. В случае, когда слагаемые отсутствуют (т.е. когда параметров нет) сумма пусть равняется 0.
2. Измените класс Student так, чтобы он имел 3 конструктора. 1-ый принимает все параметры. 2-ой – только id, name, surname, course. 3-ий не принимает значений. Создайте в классе StudentTest 3 объекта с помощью разных констукторов.

Ответ: программы пакета Homework.Lesson6

# Пакет (Package)

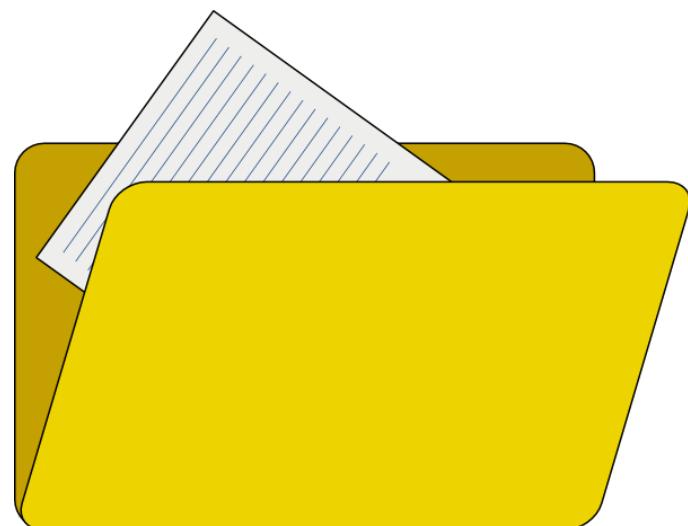
**package1**

class  
Test10

class  
Test20

class  
Test30

**АНАЛОГИЯ**



## Основные Цели Пакета

Защита доступа (Access protection)

Управление набором возможных имен  
(Namespace management)

Хранение связанных классов в одном месте  
(Keeping related classes in one place)

# Контроллер доступа (Access modifier)

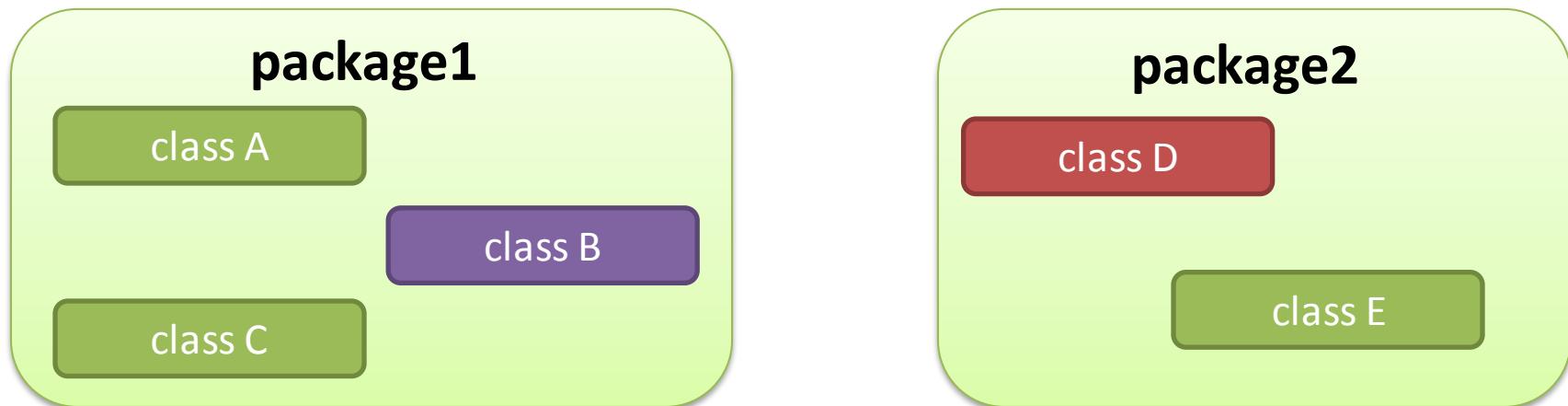
public

private

(default)

protected

# Описание access modifier-ов



**public** элемент класса А виден во всех классах всех пакетов

**private** элемент класса А виден только в нём

Элемент класса А с дефолтным access modifier виден во всех классах  
одного пакета – в котором расположен сам класс А

**protected** элемент класса А виден во всех классах одного пакета – в  
котором расположен сам класс А + во всех классах – его потомках (C, E)

# Описание access modifier-ов

	public	private	(default)	protected
class	+	-	+	-
constructor	+	+	+	+
variable	+	+	+	+
method	+	+	+	+

## Подведение итогов



**package**

**public**

**private**

**(default)**

**protected**

## Домашнее задание:

1. Пересоздайте класс Employee таким образом, чтобы переменная salary была недоступна вне класса, переменная surname была доступна отовсюду, а переменная id только внутри пакета. Также создайте 3 public метода, которые будут показывать на дисплее значения этих переменных. Создайте для данного класса 3 разных конструктора с public, default и private access modifier-ами. В констукторах присваивайте переменным класса значения из параметров. Создайте новые классы в том же и в другом пакете. Попытайтесь в них создать объекты класса Employee и вывести на экран значения переменных данного объекта с помощью метода println и методов самого класса.

*The End*

**Путь к Java ОСА.**

**Эффективное пособие для начинающих.**

# **урок 8**

## Краткий повтор пройденного материала

package statement

public access modifier

private access modifier

(default) access modifier

protected access modifier

## Проверка д/з

1. Пересоздайте класс Employee таким образом, чтобы переменная salary была недоступна вне класса, переменная surname была доступна отовсюду, а переменная id только внутри пакета. Также создайте 3 public метода, которые будут показывать на дисплее значения этих переменных. Создайте для данного класса 3 разных конструктора с public, default и private access modifierами. В констукторах присваивайте переменным класса значения из параметров. Создайте новые классы в том же и в другом пакете. Попытайтесь в них создать объекты класса Employee и вывести на экран значения переменных данного объекта с помощью метода println и методов самого класса.

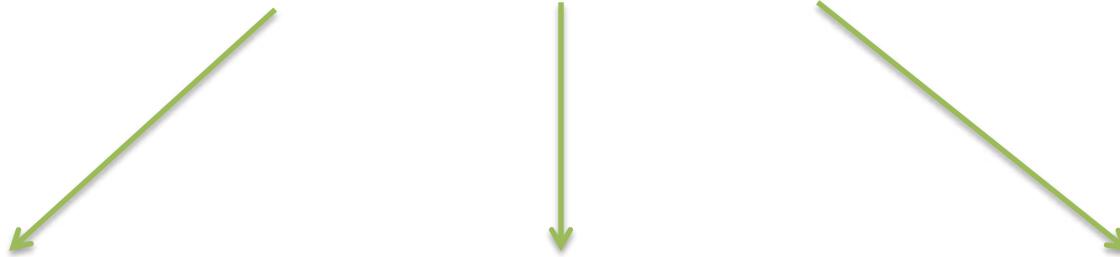
Ответ: программы пакета Homework.Lesson7

# Non-access Modifiers

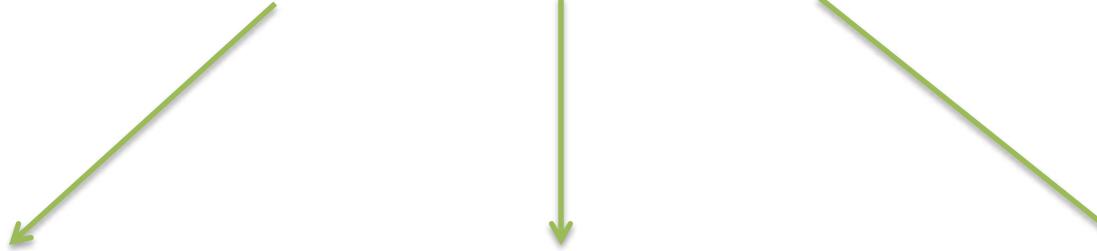
final

static

abstract



# final



variable  
(constant)

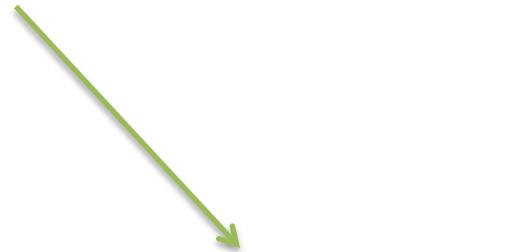
method

class

final variable должны быть инициализированы до их использования:

1. При определении переменной
2. В каждом конструкторе (если переменная не static)

# static



variable

method

static элементы принадлежат всему классу, а не  
отдельным его объектам. Существуют и могут быть  
использованы без создания объекта класса.

# static

```
class Student{  
String name = "Ivan";  
static count = 0;  
}
```

**count**

Student st1  
name = "Petya";

Student st2  
name = "Sasha";

Student st3  
name = "Kolya";

Student st4  
name = "Masha";

## static

static элементы: переменные и методы не могут вызывать, обращаться к instance переменным и методам.

К static элементам принято обращаться используя имя класса, а не ссылку на созданный объект.

## Подведение итогов



**constant**

**static variable**

**static methods**

## Домашнее задание:

1. В первом классе создайте 2 static метода. 1-ый пусть умножает 3 числа из параметра метода и возвращает их произведение; 2-ой – делит первое число из параметра на второе и ничего не возвращает, лишь выводит на дисплей в читабельном виде целое частное и остаток. Во втором классе по два раза используйте данные методы.
2. В первом классе создайте static final переменную Пи = 3,14. Используйте данную константу в non-static методе, который принимает в параметре значение радиуса и вычисляет площадь круга по формуле: Площадь = Пи \* радиус \* радиус. Также используйте данную константу в static методе, который принимает в параметре значение радиуса и вычисляет длину окружности по формуле: Площадь = 2 \* Пи \* радиус. Создайте ещё один non-static метод, который принимает в параметре значение радиуса и выводит на экран информацию о радиусе, площади круга и длине окружности. Используйте все 3 метода во 2-м классе.

*The End*

**Путь к Java ОСА.**

**Эффективное пособие для начинающих.**

# Урок 9

Краткий повтор пройденного материала

**constant**

**static variables**

**static methods**

## Проверка д/з

1. В первом классе создайте 2 static метода. 1-ый пусть умножает 3 числа из параметра метода и возвращает их произведение; 2-ой – делит первое число из параметра на второе и ничего не возвращает, лишь выводит на дисплей в читабельном виде целое частное и остаток. Во втором классе по два раза используйте данные методы.
2. В первом классе создайте static final переменную Пи = 3,14. Используйте данную константу в non-static методе, который принимает в параметре значение радиуса и вычисляет площадь круга по формуле: Площадь = Пи \* радиус \* радиус. Также используйте данную константу в static методе, который принимает в параметре значение радиуса и вычисляет длину окружности по формуле: Площадь = 2 \* Пи \* радиус. Создайте ещё один non-static метод, который принимает в параметре значение радиуса и выводит на экран информацию о радиусе, площади круга и длине окружности. Используйте все 3 метода во 2-м классе.

Ответ: программы пакета Homework.Lesson8

В зависимости от пределов видимости  
переменные делятся на 4 группы

Local variable

parameter

Instance variable

static variable

# this

this keyword при использовании с переменными и методами указывает на текущий экземпляр (объект) класса

this keyword нельзя использовать для обращения к переменным и методам внутри static метода или при присваивании значения static переменной

Не является ошибочным обращение с помощью this к static элементам

# Идентификаторы в Java

Длина не ограничена

Должен начинаться с прописной или заглавной буквы, с символов валют или \_

Символы валют и \_ могут быть в любой части идентификатора, а цифры везде, кроме начала

## Зарезервированные слова

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

## Garbage collector

- Мы можем определить какие объекты будут пригодны для того, чтобы garbage collector собрал их (удалил из памяти)
  - Мы не можем конкретно быть уверены в том, что в определённое время garbage collector удалит объект из памяти
- 
- Мы не можем контролировать garbage collector и управлять им
  - Управление garbage collector-ом лежит на JVM

## Подведение итогов



**Scope of  
variables**

**Correct  
identifiers**

**Garbage  
collector**

**this statement**

**Object's life  
cycle**

## Домашнее задание:

1. Напишите программу, в которой будет создано 8 объектов, но к последней строке метода main останется всего 2.
2. Вычислите устно output-ы данных программ, а затем проверьте в NetBeans:

```
class Test1{  
int a=1;  
static int a=2;  
void abc(int a){  
System.out.println(a);  
System.out.println(this.a);  
}  
public static void  
main(String [] args){  
Test1 t = new Test1();  
t.abc(3);  
}
```

```
class Test2{  
int a=1;  
static int b=2;  
static void abc(final int a){  
System.out.println(a);  
System.out.println(Test2.b);  
}  
public static void  
main(String [] args){  
abc(5);  
}
```

```
class Test3{  
int a=1;  
static int b=2;  
void abc(int a){  
System.out.println(b);  
System.out.println(a);  
System.out.println(this.a);  
System.out.println(Test3.b);  
}  
public static void  
main(String [] args){  
Test3 t = new Test3();  
t.abc(4);  
}
```

*The End*

**Путь к Java ОСА.**

**Эффективное пособие для начинающих.**

# **урок 10**

# Краткий повтор пройденного материала

Scope of variables

this statement

Correct identifiers

Object's life cycle

Garbage collector

# Проверка д/з

1. Напишите программу, в которой будет создано 8 объектов, но к последней строке метода main останется всего 2.
2. Вычислите устно output-ы данных программ, а затем проверьте в NetBeans:

```
class Test1{  
int a=1;  
static int a=2;  
void abc(int a){  
System.out.println(a);  
System.out.println(this.a);  
}  
public static void  
main(String [] args){  
Test1 t = new Test1();  
t.abc(3);  
}
```

```
class Test2{  
int a=1;  
static int b=2;  
static void abc(final int a){  
System.out.println(a);  
System.out.println(Test2.b);  
}  
public static void  
main(String [] args){  
abc(5);  
}
```

```
class Test3{  
int a=1;  
static int b=2;  
void abc(int a){  
System.out.println(b);  
System.out.println(a);  
System.out.println(this.a);  
System.out.println(Test3.b);  
}  
public static void  
main(String [] args){  
Test3 t = new Test3();  
t.abc(4);  
}
```

Ответ: программа Lesson9

# Для чого нужен import

package p1

class  
A

package p2

class  
B

```
package p1;
```

```
class A{  
p2.B b1 = new p2.B();  
}
```

```
package p1;  
import p2.B;
```

```
class A{  
B b1 = new B();  
}
```

# Subpackages and import

package p1

class  
A

package p2

class  
B

package p3

class  
C

# ERROR

```
package p1;  
import p2.*;
```

```
class A{  
    C c1 = new C();  
}
```

# Для чего нужен static import

package p1

class  
A

package p2

```
public class B{  
    public static int a;  
}
```

```
package p1;  
import static p2.B.a;
```

```
class A{  
    int x = a;  
}
```

# Comments

## end-of-line

```
// you can write comments here
```

## multiline

```
/* you can write comments here  
 * and here  
 * and here */
```

## JavaDoc

```
/** you can write comments here  
 * and here  
 * and here */
```

## Подведение итогов



**import  
statement**

**static import  
statement**

**Comments**

## Домашнее задание:

1. Создайте пакет p1, в нём класс A. В пакете p1 создайте подпакет p2, в нём класс B со static элементами. В подпакете p2 создайте подпакет p3, в нём класс C. Создайте новый пакет p4, в нём класс D. В пакете p4 создайте подпакет p5, в нём класс E. Внутри класса D напишите код, который будет задействовать любые элементы классов A, C, E, а также static элементы класса B. Используя выражения import и import static, работайте с не полными именами классов.

*The End*

**Путь к Java ОСА.**

**Эффективное пособие для начинающих.**

# **урок 11**

Краткий повтор пройденного материала

import statement

import static statement

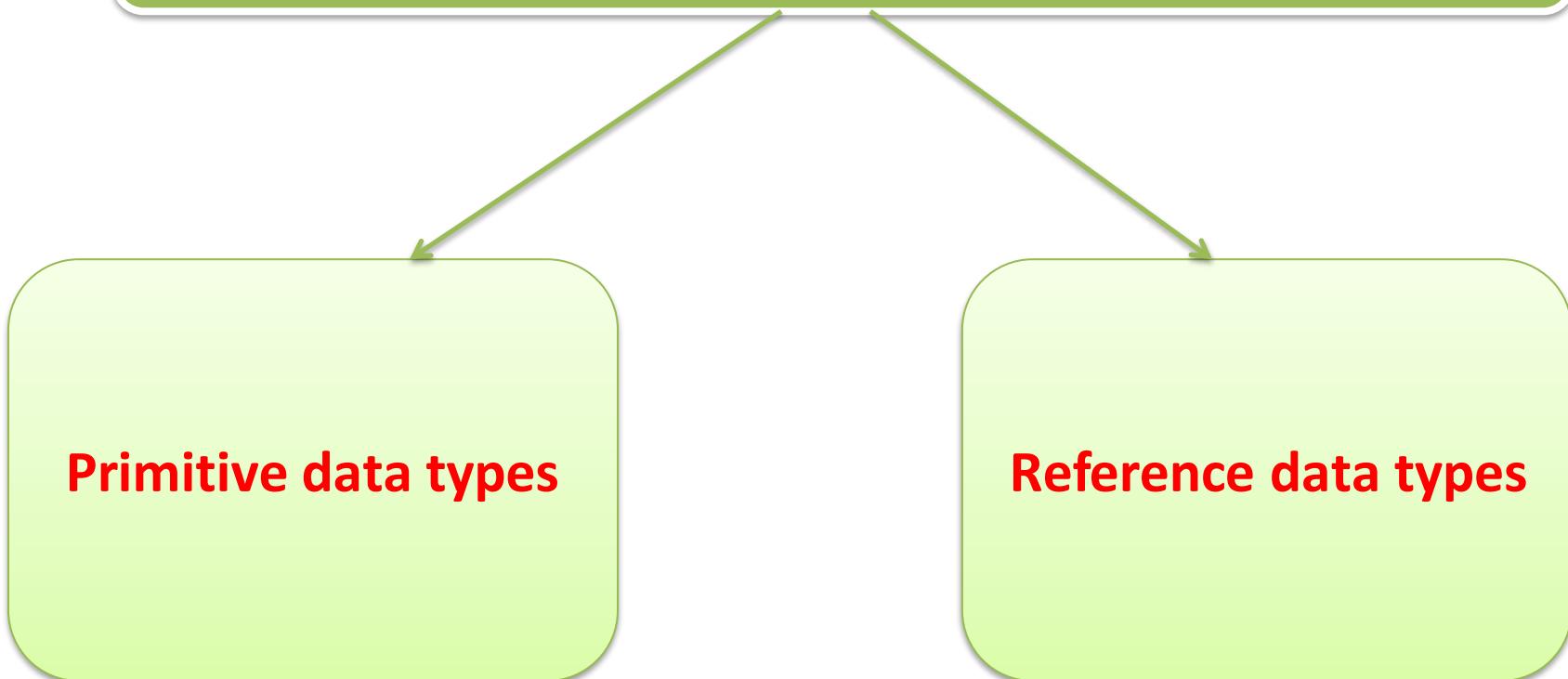
Comments

## Проверка д/з

1. Создайте пакет p1, в нём класс A. В пакете p1 создайте подпакет p2, в нём класс B со static элементами. В подпакете p2 создайте подпакет p3, в нём класс C. Создайте новый пакет p4, в нём класс D. В пакете p4 создайте подпакет p5, в нём класс E. Внутри класса D напишите код, который будет задействовать любые элементы классов A, C, E, а также static элементы класса B. Используя выражения import и import static, работайте с не полными именами классов.

Ответ: программы пакета Homework.Lesson10

## Аргументы методов



## Подведение итогов

**Passing  
primitives to  
method**



**Passing  
references to  
method**

## Домашнее задание:

1. Создайте класс Car с тремя переменными: цвет, мотор и количество дверей. Затем создайте класс CarTest, в котором должны быть 2 метода. 1-ый изменяет количество дверей объекта класса Car на количество, прописанное в параметре метода. 2-ой принимает в параметры 2 объекта класса Car и меняет их цвета местами. Примените оба метода в main и выведите на экран атрибуты всех объектов.

*The End*

**Путь к Java ОСА.**

**Эффективное пособие для начинающих.**

# **урок 12**

Краткий повтор пройденного материала

Passing primitives to method

Passing references to method

## Проверка д/з

1. Создайте класс Car с тремя переменными: цвет, мотор и количество дверей. Затем создайте класс CarTest, в котором должны быть 2 метода. 1-ый изменяет количество дверей объекта класса Car на количество, прописанное в параметре метода. 2-ой принимает в параметры 2 объекта класса Car и меняет их цвета местами. Примените оба метода в main и выведите на экран атрибуты всех объектов.

Ответ: программа Lesson11

## Синтаксис if statement

```
if (boolean expression) {  
    our code  
}  
  
else {  
    our code  
}
```

## Варианты использования if

```
if (boolean expression)
    { our code }
```

```
if (boolean expression)
    { our code }
else
    { our code }
```

```
if (boolean expression)
    { our code }
else if (boolean expression)
    { our code }
else
    { our code }
```

```
if (boolean expression)
if (boolean expression)
    { our code }
else
    { our code }
else
    { our code }
```

=

==

equals

**Works OK**

```
boolean b = false;  
if (b==true)  
    { our code }
```

**Works OK**

```
boolean b = false;  
if (b)  
    { our code }
```

**Logical error**

```
String st1 = new String("Hello");  
String st2 = new String("Hello");  
if (st1==st2)  
    { our code }
```

**Logical error**

```
boolean b = false;  
if (b=true)  
    { our code }
```

**Compile error**

```
int i = 5;  
if (i=10)  
    { our code }
```

**Works OK**

```
String st1 = new String("Hello");  
String st2 = new String("Hello");  
if (st1.equals(st2))  
    { our code }
```

## Ternary operator

**(boolean expression) ? (if true) : (if false)**

## Подведение итогов



**if else  
statement**

**Ternary operator**

**equals method**

## Домашнее задание:

1. В классе StudentTest написать 2 метода, которые принимают 2 input параметра – 2 объекта класса Student из Lesson11. Первый метод сравнивает 2-х студентов, используя 1 if statement и логические операторы внутри него и выводит на экран информацию о том, равны ли студенты. Второй метод использует nested if statement, сравнивает все атрибуты студента по отдельности, выводит на экран информацию о том, равны ли студенты, а если не равны, то в чём именно было обнаружено первое неравенство.

*The End*

**Путь к Java ОСА.**

**Эффективное пособие для начинающих.**

# **урок 13**

Краткий повтор пройденного материала

if else statement

equals method

Ternary operator

## Проверка д/з

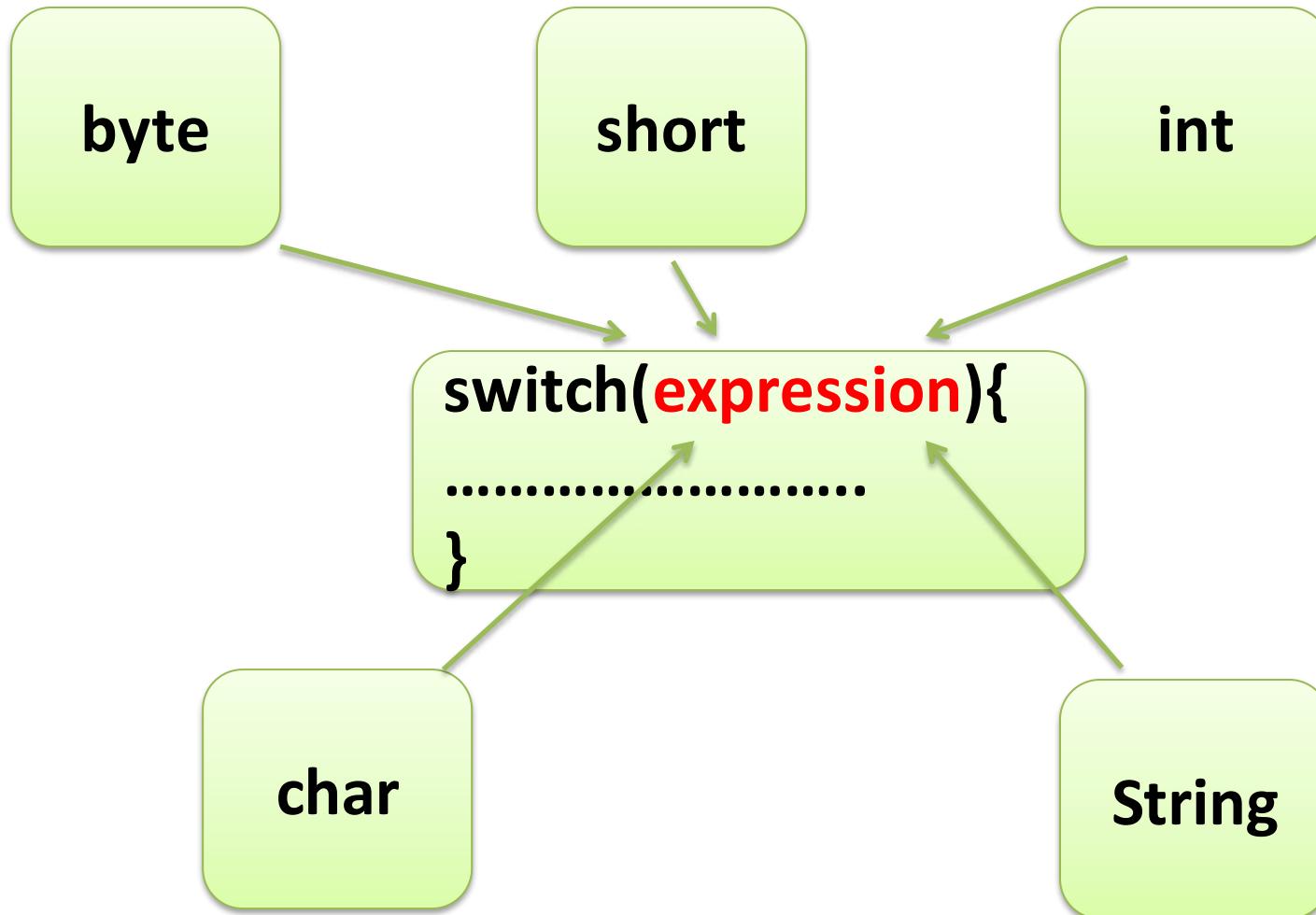
1. В классе StudentTest написать 2 метода, которые принимают 2 input параметра – 2 объекта класса Student из Lesson11. Первый метод сравнивает 2-х студентов, используя 1 if statement и логические операторы внутри него и выводит на экран информацию о том, равны ли студенты. Второй метод использует nested if statement, сравнивает все атрибуты студента по отдельности, выводит на экран информацию о том, равны ли студенты, а если не равны, то в чём именно было обнаружено первое неравенство.

Ответ: программа Lesson12

## Синтаксис switch statement

```
switch (expression) {  
    case value1:    our code;      break;  
    case value2:    our code;      break;  
    case value3:    our code;      break;  
    default:        our code;      break;  
}
```

## Тип данных switch expression



# Compile time constants

1, -3.14, “privet”, 5\*10

**YES**

a=5; b=10;  
a\*b

**NO**

final a=5; final b=10;  
a\*b

**YES**

final a; final b;  
a=5; b=10;  
a\*b

**NO**

## Подведение итогов



**switch**

## Домашнее задание:

1. В классе Month создайте метод, у которого 1 параметр типа данных int. Этот параметр будет указывать порядковый номер месяца. Используя функционал switch, выведите на экран количество дней этого месяца (для 2015 года). Проверьте работу данного метода в main.

*The End*

**Путь к Java ОСА.**

**Эффективное пособие для начинающих.**

# **урок 14**

Краткий повтор пройденного материала

`switch`

`switch expression, case expressions`

`break statement`

## Проверка д/з

1. В классе Month создайте метод, у которого 1 параметр типа данных int. Этот параметр будет указывать порядковый номер месяца. Используя функционал switch, выведите на экран количество дней этого месяца (для 2015 года). Проверьте работу данного метода в main.

Ответ: программа Lesson13

## Циклы в Java

**regular  
for**

**while**

**do while**

**foreach**

## Синтаксис for loop

**for(initialization ; condition ; update) {**

expressions;

**}**

# Unreachable statement

**Java does not allow to write code with unreachable statements**

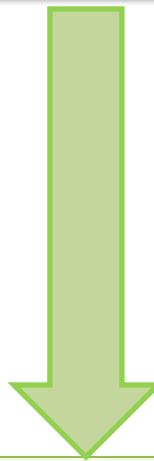
```
for(int i=0; false; i++){  
    some code  
}
```

**Compile time error!**

```
if(false)  
some code  
}
```

**Exception from the rule.**

# Loop statements



**break**



**continue**

# Nested loop

```
for(initialization_1 ; condition_1; update_1) {  
    expressions_1;  
    for(initialization_ 2; condition_ 2; update_2)  
        {expressions_2;}  
    expressions_3;  
}
```

You can name loops with labels. It will help you to write complex code.

## Подведение итогов

**loops**

**break**

**Nested  
loops**



**regular for**

**continue**

**Unreachable  
statements**

## Домашнее задание:

1. Создать класс. В классе создать статичный метод, который будет выводить на экран время в формате «час:минута:секунда» в интервале от 0 до 6 часов. Если час больше единицы и минута кратна 10-ти, то метод нужно закончить. Если же (секунда умноженная на час) больше минуты, то пора переходить на другую минуту. Продемонстрировать данный метод в действии.

*The End*

**Путь к Java ОСА.**

**Эффективное пособие для начинающих.**

# **урок 15**

# Краткий повтор пройденного материала

Loops. Regular for loop

break, continue statements

Nested loops

Unreachable statements

## Проверка д/з

1. Создать класс. В классе создать статичный метод, который будет выводить на экран время в формате «час:минута:секунда» в интервале от 0 до 6 часов. Если час больше единицы и минута кратна 10-ти, то метод нужно закончить. Если же (секунда умноженная на час) больше минуты, то пора переходить на другую минуту. Продемонстрировать данный метод в действии.

Ответ: программа Homework.Lesson14

## Синтаксис while loop

```
while (condition){  
Statement 1;  
Statement 2;  
.....  
Statement n;  
}
```

## Синтаксис do while loop

```
do {  
    Statement 1;  
    Statement 2;  
    .....  
    Statement n;  
}  
while (condition);
```

# Test

```
class Test {  
    public static void main(String[] args) {  
        do {  
            int a = 5;  
            System.out.print(a++ + " ");  
        } while (a <= 15);  
    }  
}
```

Compile time error at line

```
} while (a<=15);
```

## Подведение итогов

**while loop**

**break,  
continue**



**do while  
loop**

**Nested  
loops**

**Unreachable  
statements**

## Домашнее задание:

1. Переписать домашнее задание из Урока 14 так, чтобы outer и inner циклы представляли собой while loop, а middle цикл представлял собой do while loop.

*The End*

**Путь к Java ОСА.**

**Эффективное пособие для начинающих.**

# **урок 16**

# Краткий повтор пройденного материала

while loop

do while loop

break & continue statements

Nested loops

Unreachable statements

## Проверка д/з

1. Переписать домашнее задание из Урока 14 так, чтобы outer и inner циклы представляли собой while loop, а middle цикл представлял собой do while loop. Создать класс. В классе создать статичный метод, который будет выводить на экран время в формате «час:минута:секунда» в интервале от 0 до 6 часов. Если час больше единицы и минута кратна 10-ти, то метод нужно закончить. Если же (секунда умноженная на час) больше минуты, то пора переходить на другую минуту. Продемонстрировать данный метод в действии.

Ответ: программа Homework.Lesson15

## Объявление/создание объектов String

- **String s1 = new String("Good day!");**  
Всегда создаётся новый объект

### String pool

Новый объект создаётся  
только если подобного нет в  
String pool

- **String s2 = "Hello";**

- **System.out.print("Privet");**

## String objects counting

```
public class StringCounting {  
    public static void main(String[] args) {  
        String s1 = new String("Privet");  
        String s2 = "Privet";  
        System.out.println("Privet");  
        System.out.println("poka");  
        System.out.println("poka" == "privet");  
        String autumn = new String("Privet");  
    }  
}
```

## String is immutable

String хранит своё значение в **private** массиве, к тому же массив имеет фиксированную длину

String хранит своё значение в **final** массиве, следовательно его инициализация может быть только один раз

Ни один из методов класса String не изменяет индивидуальные элементы массива типа char

# Методы String

length() → int

indexOf(char c) → int  
indexOf(String s) → int  
indexOf(char c, int fromIndex) → int  
indexOf(String s , int fromIndex) → int

endsWith(String suffix) → boolean

trim() → String

concat(String str) → String

toLowerCase() → String

toUpperCase() → String

charAt(int index) → char

startsWith(String prefix) → boolean  
startsWith(String prefix, int from) → boolean

substring(int beginIndex) → String  
substring(int beginIndex, int endIndex) → String

replace(char oldChar, char newChar) → String  
replace(String oldString, String newString) → String

contains(String str) → boolean

При method chaining методы выполняются последовательно слева направо.

## Определение равенства String

`==`

**Method equals**

`!=`

**Method  
equalsIgnoreCase**

## Подведение итогов

**Создание  
String**

**Introduction  
to array**

**String  
methods**

**String pool**

**String is  
immutable**

**Equality of  
Strings**



## Домашнее задание:

1. Создайте класс, в котором создайте метод email. Данный метод должен принимать в input 1 String параметр. Данный параметр должен содержать в себе email-ы в следующем виде: [ya@yahoo.com](mailto:ya@yahoo.com); [on@mail.ru](mailto:on@mail.ru); [ona@gmail.com](mailto:ona@gmail.com); , т.е. После каждого e-mail должен стоять знак препинания ";". Ваш метод должен выводить на экран информацию о том, какой почтой пользуются, исходя из параметра, т.е. output должен быть следующего вида:

yahoo

mail

gmail

Продемонстрируйте данный метод.

*The End*

**Путь к Java ОСА.**

**Эффективное пособие для начинающих.**

# **урок 17**

Краткий повтор пройденного материала

Создание String

String pool

String is immutable

Introduction to array

String methods

Equality of Strings

## Проверка д/з

1. Создайте класс, в котором создайте метод email. Данный метод должен принимать в инпут 1 String параметр. Данный параметр должен содержать в себе email-ы в следующем виде: [ya@yahoo.com](mailto:ya@yahoo.com); [on@mail.ru](mailto:on@mail.ru); [ona@gmail.com](mailto:ona@gmail.com); , т.е. После каждого e-mail должен стоять знак препинания “;”. Ваш метод должен выводить на экран информацию о том, какой почтой пользуются, исходя из параметра, т.е. оутпут должен быть следующего вида:

**yahoo**

**mail**

**gmail**

Продемонстрируйте данный метод.

Ответ: программа Homework.Lesson16

## Создание объектов StringBuilder

```
StringBuilder sb1 = new StringBuilder();
```

```
StringBuilder sb2 = new StringBuilder("Good day!");
```

```
StringBuilder sb3 = new StringBuilder(50);
```

```
StringBuilder sb4 = new StringBuilder(sb2);
```

# методы StringBuilder

length() → int

charAt(int index) → char

indexOf(String s) → int

indexOf(String s , int fromIndex) → int

substring(int beginIndex) → String

substring(int beginIndex, int endIndex) → String

append(dataType dt) → StringBuilder

subsequence(int start, int end) → CharSequence

delete(int start, int end) → StringBuilder

insert(int toIndex, dataType dt) → StringBuilder

reverse() → StringBuilder

deleteCharAt(int index) → StringBuilder

capacity() → int

replace(int start, int end, String s) → StringBuilder

При method chaining методы выполняются последовательно слева направо.

# Test

```
public class Test {  
    public static void main(String[] args) {  
        StringBuilder sb1 = new StringBuilder("Hello, friend!");  
        String s = null;  
        s = sb1.append(" How are you?").substring(sb1.indexOf('f'),  
sb1.indexOf('!'));  
        System.out.println(s);  
    }  
}
```

Compile time error

## Ещё 2 конструктора класса String

```
StringBuilder sb1 = new StringBuilder("Hello");
String s1 = new String(sb1);
```

```
StringBuffer sb2 = new StringBuffer("Hello");
String s2 = new String(sb2);
```

## Подведение итогов

**Создание  
StringBuilder**

**StringBuilder  
is mutable**

**StringBuilder  
methods**

**StringBuffer  
class**

**Equality of  
StringBuilders**



## Домашнее задание:

1. Создайте класс, в котором создайте метод ravenstvo. Инпут параметрами данного метода будут 2 объекта класса StringBuilder. Метод должен иметь boolean return type, возвращать true, если значения объектов совпадают, false – если не совпадают.  
Продемонстрируйте данный метод.

*The End*

**Путь к Java ОСА.**

**Эффективное пособие для начинающих.**

# **урок 18**

Краткий повтор пройденного материала

Создание StringBuilder

StringBuilder is mutable

StringBuilder methods

Equality of Strings

## Проверка д/з

1. Создайте класс, в котором создайте метод ravenstvo. Инпут параметрами данного метода будут 2 объекта класса StringBuilder. Метод должен иметь boolean return type, возвращать true, если значения объектов совпадают, false – если не совпадают.  
Продемонстрируйте данный метод.

Ответ: программа Homework.Lesson17

## Виды массивов



Хранит коллекцию  
primitive data type



Хранит коллекцию  
reference data type

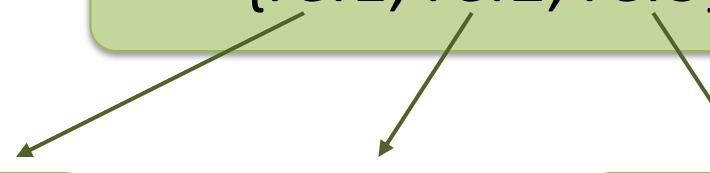
{5, 7, 9, -3, 0, 1}

{ref1, ref2, ref3}

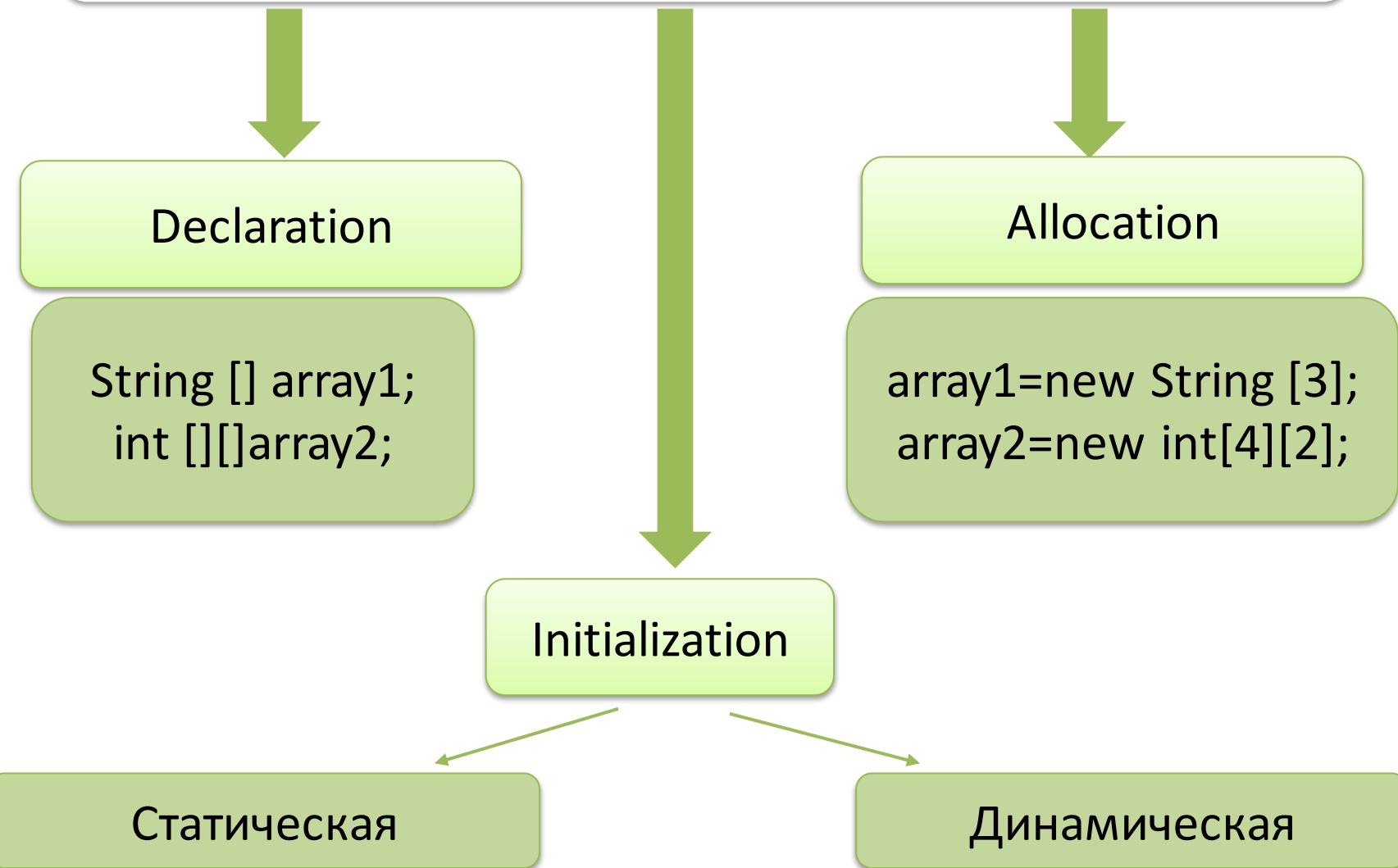
“hello”

“ok”

“bye”



## Этапы создания массива



## Смешанные варианты создания массива

```
int [] array3=new int[7];
```

```
int [] array2={1,2,3};
```

```
int [] array1;  
array1=new int []{1,2,3};
```

```
int [] array4=new int [] {1,2,3};
```

```
int [] array1;  
array1={1,2,3};
```

```
int [] array4=new int [3] {1,2,3};
```

# Introduction to Exceptions

ArrayIndexOutOfBoundsException

NullPointerException

```
int [] array1 = {1,2,3};  
array1 [4]=12;
```

```
int [][] array2 = {{5,6}, null};  
System.out.print(array2[1][0]);
```

## class Arrays

sort(array)

binarySearch(array, value)

```
int [] array1 = {1,2,3};  
Arrays.sort(array1);
```

```
int [] array1 = {1,2,3};  
Arrays.sort(array1);  
int index=Arrays.binarySearch(array1, 2);
```

# Test

Какие объявления массива не корректны?

- A. int[][] array1 = new int[10][];
- B. Car[][][] array2= new Car[1][0][7];
- C. String array3[] = new array3[9];
- D. java.lang.String[] array4[] = new java.lang.String[5][];
- E. int[][] array5 = new int[]{};
- F. int[][] array6 = new int[][]{};

# Test

Какие объявления массива корректны?

- A int array1[] = {3, 5, 6, 0};
- B int[] array2 = new int[1];
- C int[] array3 = new int[] {};
- D int[] array4 = new int[2] {};
- E int array5[] = new int[3] {0, 1, 2};



# Дополнения к материалу по классам String и StringBuilder

Ещё один конструктор класса String:

```
char[] array= new char[]{‘p’,‘r’,‘i’, ‘v’, ‘e’, ‘t’};  
String s= new String(array);
```

Метод append:

```
StringBuilder sb1 = new StringBuilder(“ok”);  
char[] array= {‘p’,‘r’,‘i’, ‘v’, ‘e’, ‘t’};  
sb1.append(array, 2, 3);
```

Метод insert:

```
StringBuilder sb1 = new StringBuilder(“privet”);  
char[] array= {‘p’,‘r’,‘i’, ‘v’, ‘e’, ‘t’};  
sb1.insert(1, array, 2, 3);
```

## Подведение итогов

array

Exception



class Arrays

## Домашнее задание:

1. Создайте класс, в котором создайте метод sortirovka. Инпут параметром данного метода будет одномерный массив типа int. Метод должен возвращать уже отсортированный по возрастанию массив .  
Продемонстрируйте данный метод.

1. Создайте класс, в котором создайте метод showArray. Инпут параметром данного метода будет двумерный массив типа String. Метод должен выводить на экран данный массив в следующем виде:  
{ {элемент00, элемент01}, {элемент10}, {элемент10, элемент11} }  
Содержимое,естественно, будет зависеть от Вашего массива.  
Продемонстрируйте данный метод.

*The End*

**Путь к Java ОСА.**

**Эффективное пособие для начинающих.**

# **урок 19**

Краткий повтор пройденного материала

Array

Introduction to Exceptions

class Arrays

## Проверка д/з

Создайте класс, в котором создайте метод sortirovka. Инпут параметром данного метода будет одномерный массив типа int. Метод должен возвращать уже отсортированный по возрастанию массив .

Продемонстрируйте данный метод.

Создайте класс, в котором создайте метод showArray. Инпут параметром данного метода будет двумерный массив типа String. Метод должен выводить на экран данный массив в следующем виде:

{ {элемент00, элемент01}, {элемент10}, {элемент10, элемент11} }

Содержимое, естественно, будет зависеть от Вашего массива.

Продемонстрируйте данный метод.

Ответ: пакет Homework.Lesson18

# Command line arguments

```
javac Test1.java
```

```
java Test1 Hello 14 true
```

=

```
String args [] = {"Hello", "14", "true"};
```

Data type is String

## Variable arguments = varargs

В листе параметров метода может быть только 1 varargs

В листе параметров метода varargs должен стоять самым последним

Следующий код вызывает Compile time error:

```
void abc(int ... a) {  
    //some code;  
}
```

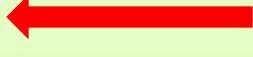
```
void abc(int array[]) {  
    // some code;  
}
```

# Test

Дан метод

```
private static void abc(int a, int ... b) {  
    System.out.print(b.length);    }
```

Какие методы из приведённых ниже выведут на экран число 2?

- A. abc(2, new int[2]); 
- B. abc(2, {0, 1});
- C. abc(1, {2});
- D. abc(0, 1, 2); 
- E. abc(0, 1);
- F. abc(3);
- G. abc();

# Test

Каков output данного кода?

```
static void abc(int a) {  
    System.out.println("Hello"); }  
static void abc(int ... a) {  
    System.out.println("Bye"); }  
  
public static void main(String [] args) {  
    abc(5); }  
}
```

Output: Hello

## Enhanced for = foreach

```
datatype [] array = {value1, value2,..., valueN};
```

```
for(datatype variable: array){  
    System.out.println(variable);  
}
```

С помощью foreach loop:

- невозможно изменить значения элементов массива примитивного типа данных;
- возможно изменить значения элементов (значения объектов) массива ссылочного типа данных, но заменить элемент невозможно;
- невозможно проводить динамическую инициализацию массива;
- невозможно в одном цикле одновременно работать более чем с одним массивом

## Подведение итогов

**Command  
line args**



**varargs**

**foreach loop**

## Домашнее задание:

1. Создайте метод abc, инпут параметр которого – N-ое количество массивов типа String. В методе создайте новый массив, который будет состоять из элементов массивов-параметров и будет оутпутом данного метода. В методе main вызовите метод abc и его элементам, которые равны значениям command line параметров, присвойте значение null. Выведите элементы обновлённого массива на экран. Везде, где возможно, используйте foreach loop.  
Запустите приложение с командной строки.

*The End*

**Путь к Java ОСА.**

**Эффективное пособие для начинающих.**

# **урок 20**

Краткий повтор пройденного материала

Command line arguments

Variable arguments

foreach

## Проверка д/з

1. Создайте метод abc, инпут параметр которого – N-ое количество массивов типа String. В методе создайте новый массив, который будет состоять из элементов массивов-параметров и будет оутпутом данного метода. В методе main вызовите метод abc и его элементам, которые равны значениям command line параметров, присвойте значение null. Выведите элементы обновлённого массива на экран. Везде, где возможно, используйте foreach loop.  
Запустите приложение с командной строки.

Ответ: программа D:\Test\Lesson19.java

# ArrayList

В основе ArrayList лежит массив

```
ArrayList <DataType> list1 = new ArrayList <DataType>();
```

```
ArrayList <DataType> list2 = new ArrayList <>();
```

```
ArrayList <DataType> list3 = new ArrayList <> (55);
```

```
ArrayList <DataType> list4 = new ArrayList <> (list3);
```

# методы ArrayList

add(DataType element) → boolean  
add(int index, DataType element) → boolean

get(int index) → DataType

set(int index, DataType element) → DataType

remove(Object element) → boolean  
remove(int index) → DataType

addAll(ArrayList aL) → boolean  
addAll(int index, ArrayList aL) → boolean

clear() → void

indexOf(Object element) → int

lastIndexOf(Object element) → int

size() → int

isEmpty() → boolean

contains(Object element) → boolean

toString() → String

# методы ArrayList и связанные с ArrayList

`clone() → Object`

`toArray() → Object []`  
`toArray(DataType [] array) → DataType []`

`Arrays.asList(DataType []) → List<DataType>`

`Collections.sort(ArrayList<DataType>) → void`

`equals(ArrayList<DataType>) → boolean`

# Iterator & ListIterator

some code

.....

```
Iterator<DataType> iter = aL.iterator();
while (iter.hasNext())
{
    System.out.println(iterator.next());
}
```

some code

.....

```
ListIterator <DataType> iter = aL.listIterator();
while (iter.hasNext())
{
    iter.next();
    iter.remove();
}
```

# Test

Каков оутпут приведённого ниже кода?

```
import java.util.ArrayList;
class Test {
    public static void main(String args[]) {
        ArrayList<String> students = new ArrayList<>();
        students.add(1, "Ivanov");
        students.add(2, "Petrov");
        students.add(3, "Sidorov");
        students.remove(2);
        for (String s : students) {
            System.out.print(s + " ");
        }
    }
}
```

IndexOutOfBoundsException будет выведен на экран.

## Подведение итогов

**Создание  
ArrayList**

**ArrayList  
methods**

**iterators**

**Object**



## Домашнее задание:

1. Создайте класс, в котором создайте метод abc. Испут параметром данного метода будет N-ое количество параметров типа String. Метод должен возвращать уже отсортированный объект ArrayList из неповторяющихся объектов типа String, взятых из параметра метода и выводить данный объект на экран.  
Продемонстрируйте данный метод.

*The End*

**Путь к Java ОСА.**

**Эффективное пособие для начинающих.**

# **урок 21**

Краткий повтор пройденного материала

Создание ArrayList

Методы ArrayList

iterators

Object class

## Проверка д/з

1. Создайте класс, в котором создайте метод abc. Инпут параметром данного метода будет N-ое количество параметров типа String. Метод должен возвращать уже отсортированный объект ArrayList из неповторяющихся объектов типа String, взятых из параметра метода и выводить данный объект на экран.  
Продемонстрируйте данный метод.

Ответ: Homework.Lesson20

# Повторение + Примеры

Ternary operator  
booleanExpression ? expression1 : expression2

import

package p1

class A

Конфликт имен

package p2

class A

```
package p3;  
import p1.A;  
import p2.A;  
class B {  
}
```

```
package p3;  
import p1.*;  
import p2.*;  
class B {  
A a = new A();  
}
```

```
package p3;  
import p1.*;  
import p2.*;  
class B {  
    A a = new A();  
}
```

```
package p3;  
import p1.*;  
import p2.*;  
class B {  
    p1.A a = new p1.A();  
    p2.A a = new p2.A();  
}
```

## Повторение + Примеры

Garbage Collection. Methods System.gc(), finalize().

public элементы класса, access modifier которого default, не видны в классах другого пакета

Объект public класса не может быть создан в классе другого пакета, с помощью конструктора, access modifier которого default

access modifier default-ного конструктора такой же, что и у его класса

Компилятор не разрешает написание рекурсивного конструктора

## Повторение + Примеры

Превращение конструктора в метод путём добавления к нему return type

Названия переменной и метода могут совпадать

Параметры в overloaded методах могут быть как примитивного, так и ссылочного типа данных

При вызове метода, требующего в параметр тип данных int мы можем использовать тип данных char

Некоторые моменты, связанные с ключевым словом return

# Test

Каков вывод приведённого ниже кода?

```
class Test {  
    public static void main(String[] args) {  
        int a = 7;  
        System.out.println(a > 2 ? a < 5 ? 3 : 6 : 9);  
    }  
}
```

- A 2
- B 5
- C 3
- D 6 ←
- E 9
- F. The code will not compile

# Test

На каких строчках объекты car1, car2 , car3 и car4 становятся пригодными для удаления их Garbage collector-ом?

```
1 public class Car{  
2     public static void main(String[] args) {  
3         Car car1 = new Car();  
4         Car car2 = new Car();  
5         Car car3 = car1;  
6         car1 = null;  
7         Car car4 = car1;  
8         car3 = null;  
9         car2 = null;  
10        car2 = new Car();  
11        System. gc();  
12    }  
13 }
```

К концу строки №8

К концу строки №9

К концу строки №12

# Test

Какое утверждение верно для данного кода?

```
1 class Employee {  
2     Employee manager = new Employee();  
3 }  
4  
5 class TestEmployee {  
6     public static void main(String args[]) {  
7         Employee e = null;  
8         e = new Employee();  
9         e = null;  
10    }  
11 }
```

- А Объект, созданный на строке №8, пригоден для удаления Garbage collector-ом в конце строки №9
- В Объект, созданный на строке №8, пригоден для удаления Garbage collector-ом в конце строки №10
- С Выполняющийся код никогда не достигнет строк №9, 10, 11

*The End*

**Путь к Java ОСА.**

**Эффективное пособие для начинающих.**

# **урок 22**

## 3 принципа ООП

Encapsulation (сокрытие данных)

Inheritance (наследование)

Polymorphism (полиморфизм)

# Encapsulation

Сокрытие данных, защита их от внешнего нежелательного вмешательства, помещение их в «капсулу».

Характеризуется private переменными и public методами get и set, которые нередко проверяют какие-либо условия.

Если return type метода get – это mutable тип данных, то лучше возвращать его копию.

# Inheritance

```
class Parent { some code }  
class Child extends Parent { some code }
```

Derived class  
Child class  
Subclass  
Extended class

Super class  
Parent class  
Base class



# Зачем нужно наследование?

Более короткое написание классов

Лёгкость в изменении/добавлении общих элементов

Extensibility

Более лёгкое тестирование классов

Группировка классов под общим типом

# Отношения между классами. Object класс.

“Is-a” relationship



```
class Animal { }  
class Mouse extends Animal { }  
Mouse is Animal
```

“Has-a”  
relationship



```
class Window { }  
class House {  
    Window w = new Window(); }  
House has Window
```

Класс Object является прародителем всех классов в Java. Это единственный класс, у которого нет родителя.

# Что наследуется классом?

Элементы с access modifier public

Элементы с access modifier default, только если child класс находится в том же пакете, что и parent класс

Элементы с access modifier private не наследуются, но наследуются public методы, которые могут работать с ними

Элементы с access modifier protected видны там же, где и элементы с access modifier default + в subclassах parent класса.

Таким образом, элементы с access modifier protected наследуются вне зависимости от того, где находится child класс.

Конструкторы не способны быть наследованы.

# Keyword “super” в конструкторе

```
class X {  
    public X(String s) {  
        System.out.println(s);  
    }  
}
```



```
class Y extends X {  
    public Y() {  
        super("ok");  
        System.out.println("privet");  
    }  
}
```

Выражение `super` вызывает конструктор `super` класса, который заканчивает свою работу всегда раньше конструктора `child` класса

Выражение `super` если есть, то должно стоять на 1-ой строке конструктора

Если мы сами не пишем выражение `super`, то компилятор дописывает его сам, обращаясь к конструктору без параметров `super` класса

Выражения `super` и `this` не могут одновременно находиться в теле конструктора

## Подведение итогов

**encapsulation**

**inheritance**

**Is-a/Has-a**

**“super” in  
constructor**



## Домашнее задание:

1. Создайте класс Student со следующими переменными: name (используйте StringBuilder), course, grade. Примените инкапсуляцию к данному классу. Длина имени объектов не должна быть менее 3-х символов, оценки должны быть числами в интервале от 1 до 10, курс должен быть числом от 1 до 4 включительно. Создайте метод showInfo, который будет выводить всю информацию о студенте, не используя переменные класса напрямую. Создайте класс TestStudent, в котором создайте объект класса Student, придайте его переменным значения. Произведите вызов метода showInfo.

## Домашнее задание:

2. Создайте класс Animal. При вызове его конструктора пусть на экран выводится "I am animal". В классе пусть будут переменная eyes, характеризующая количество глаз; методы eat (выводящий на экран "Animal eats") и drink (выводящий на экран "Animal drinks").

Создайте класс Pet, который является child классом класса Animal. При вызове его конструктора пусть на экран выводится "I am pet" и переменной eyes придаётся значение 2. В классе пусть будут переменные name; tail, характеризующая количество хвостов и равная 1; paw, характеризующая количество лап и равная 4; методы run (выводящий на экран "Pet runs") и jump (выводящий на экран "Pet jumps").

Создайте класс Dog, который является child классом класса Pet. При вызове его конструктора с параметром, который будет передавать имя, пусть на экран выводится "I am dog and my name is: " + имя питомца. В класс добавьте метод play (выводящий на экран "Dog plays").

Создайте класс Cat, который является child классом класса Pet. При вызове его конструктора с параметром, который будет передавать имя, пусть на экран выводится "I am cat and my name is: " + имя питомца. В класс добавьте метод sleep (выводящий на экран "Cat sleeps").

Создайте класс Test, в методе main которого выведите на экран количество лап объекта класса Dog и вызовите метод sleep объекта класса Cat.

*The End*

**Путь к Java ОСА.**

**Эффективное пособие для начинающих.**

# **урок 23**

# Краткий повтор пройденного материала

Encapsulation

Inheritance

Is-a/Has-a relationships

Keyword “super” in constructor

## Проверка д/з

1. Создайте класс Student со следующими переменными: name (используйте StringBuilder), course, grade. Примените энкапсулацию к данному классу. Длина имени объектов не должна быть менее 3-х символов, оценки должны быть числами в интервале от 1 до 10, курс должен быть числом от 1 до 4 включительно. Создайте метод showInfo, который будет выводить всю информацию о студенте, не используя переменные класса напрямую. Создайте класс TestStudent в котором создайте объект класса Student, придайте его переменным значения. Произведите вызов метода showInfo.

Ответ: пакет Homework.Lesson22

# Проверка д/з

2. Создайте класс Animal. При вызове его конструктора пусть на экран выводится "I am animal". В классе пусть будут переменная eyes, характеризующая количество глаз; методы eat (выводящий на экран "Animal eats") и drink (выводящий на экран "Animal drinks").

Создайте класс Pet, который является child классом класса Animal. При вызове его конструктора пусть на экран выводится "I am pet" и переменной eyes придаётся значение 2. В классе пусть будут переменные name; tail, характеризующая количество хвостов и равная 1; paw, характеризующая количество лап и равная 4; методы run (выводящий на экран "Pet runs") и jump (выводящий на экран "Pet jumps").

Создайте класс Dog, который является child классом класса Pet. При вызове его конструктора с параметром, который будет передавать имя, пусть на экран выводится "I am dog and my name is: " + имя питомца. В класс добавьте метод play (выводящий на экран "Dog plays").

Создайте класс Cat, который является child классом класса Pet. При вызове его конструктора с параметром, который будет передавать имя, пусть на экран выводится "I am cat and my name is: " + имя питомца. В класс добавьте метод sleep (выводящий на экран "Cat sleeps").

Создайте класс Test, в методе main которого выведите на экран количество лап объекта класса Dog и вызовите метод sleep объекта класса Cat.

Ответ: пакет Homework.Lesson22

# Типы reference переменных и объектов

```
class Animal { }  
class Mouse extends Animal { }
```

```
class Test {  
    public static void main(String [] args){  
        Animal a = new Animal();  
        Mouse m = new Mouse();  
        Animal am = new Mouse();  
    } // because Mouse IS Animal  
}
```

Если переменная типа данных super класса ссылается на объект sub класса, то с помощью этой переменной можно вызывать только унаследованные от super класса элементы.

```
class Test2 {  
    public Animal abc(){  
        return new Mouse();  
    } // because Mouse IS Animal  
}
```

```
class Test3 {  
    public Object def(){  
        return new int[] {2,6,9,3};  
    } // because array IS Object  
}
```

# Method overriding

Это изменение non-static, non-final метода в sub классе, который он унаследовал от parent класса.

Методы считаются overridden, если:

Имя в sub классе такое же, что и в parent классе;

Список аргументов в sub классе такой же, что и в parent классе;

Return type в sub классе такой же, что и в parent классе или же return type в sub классе - это sub класс return type из parent класса (covariant return types);

Access modifier в sub классе такой же или менее строгий, чем в parent классе;

Метод в sub классе тоже должен быть non-static

Понятия variable overriding не существует

Binding – определение вызываемого метода,  
основываясь на объекте, который производит вызов  
или типе данных reference variable



Compile time binding

Run time binding

private methods

All other methods

static methods

final methods

Все переменные имеют compile time binding

# Method hiding

Это перекрытие static методов из parent класса в sub классе.

Методы считаются hidden, если:

Имя в sub классе такое же, что и в parent классе;

Список аргументов в sub классе такой же, что и в parent классе;

Return type в sub классе такой же, что и в parent классе или же return type в sub классе - это sub класс return type из parent класса (covariant return types);

Access modifier в sub классе такой же или менее строгий, чем в parent классе;

Если в parent классе метод является static, то и в sub классе он должен быть static.

Variable hiding – объявление в sub классе переменной с таким же именем (не обязательно типом), что и в parent классе.

## Что можно делать override:

Все методы, которые не являются static, final или private.

## Что можно делать hide:

static методы

Non-private variables

private методы и переменные нельзя делать ни override, ни hide.

final методы нельзя делать ни override, ни hide.

Вы можете использовать annotation @Override, когда перезаписываете метод.

Почему static методы нельзя сделать override?

Overriding и polymorphism напрямую связаны с созданием объектов – в зависимости от типа объектов вызываются те или иные методы. А само понятие static не подразумевает под собой процесс создания объектов.

final метод – это метод который не может быть overridden или hidden.

final класс – это класс, который не может иметь потомков.

# Обращение к элементам super класса с помощью keyword “super”

Keyword “super” означает обращение к объекту родительского класса.

С помощью keyword “super” можно обращаться как к методам, так и к переменным родительского класса.

Невозможно использование keyword “super” в static методах и переменных.

Для того чтобы обратиться к элементам родительского класса с помощью keyword “super”, эти элементы должны быть видны в дочернем классе.

При обращение к методу родительского класса, выражение “super” не обязательно должно быть первой строкой тела метода.

## Подведение итогов

**Data types of  
reference variable  
and object**

**Method  
overriding**

**keyword  
“super”**

**Binding**

**Method hiding**



# Домашнее задание:

## Test 1

Каков будет результат компиляции и запуска класса Y?

```
package p1;
public class X{
    X()    { }
    public void abc()    { System.out.println('X');    }
}
```

---

```
-----  
package p2;
import p1.*;
public class Y extends X{
    Y()    { }
    public void abc()    { System.out.println('Y');    }
    public static void main(String[] args){
        Y y = new Y();
        y.abc();
    }
}
```

# Домашнее задание:

Test 2

Каков будет результат компиляции и запуска класса Y?

```
package p1;
public class X{
protected void abc() { System.out.println('X'); }
}
```

---

```
package p2;
import p1.*;
public class Y extends X {
public void abc() { System.out.println('Y'); }
public void def() { Y y = new Y(); y.abc(); }
public void ghi() { X x = new Y(); x.abc(); }
public static void main(String[] args)
{ Y a = new Y(); a.abc(); a.def(); a.ghi(); }
}
```

# Домашнее задание:

## Test 3

Каков будет результат компиляции и запуска классов X и Y?

```
public class X{  
    public X() { System.out.println("X"); }  
    public X(int i) { System.out.println("X" + i); }  
    private boolean abc() { return false; }  
    public static void main(String[] args) {  
        X x = new Y(18);  
        System.out.println( x.abc() );  
    }  
}
```

```
class Y extends X{  
    public Y(int i) { System.out.println("Y"); }  
    public boolean abc () { return true; }  
}
```

# Домашнее задание:

## Test 4

Каков будет результат компиляции и запуска класса Test?

```
class X {}  
class Y extends X {}  
  
public class Test {  
    public static void abc(X x, Y y)  
    {  
        System.out.println("privet");  
    }  
    public static void abc(Y y, X x)  
    {  
        System.out.println("poka");  
    }  
    public static void main(String[] args) {  
        Y a = new Y();  
        abc(a, a);  
    }  
}
```

## Домашнее задание:

Test 5

Каков будет результат компиляции и запуска класса Test?

```
class X{  
String s1 = "privet";  
}
```

```
class Y extends X {  
boolean bool = false;  
}
```

```
class Test {  
public static void main(String args[]) {  
X x = new Y ();  
System.out.println(x.s1 + " " + x.bool);  
}  
}
```

*The End*

**Путь к Java ОСА.**

**Эффективное пособие для начинающих.**

# **урок 24**

Краткий повтор пройденного материала

Data types of reference variable and object

Binding

Method overriding

Method hiding

keyword “super”

# Проверка д/з

Test 1

Каков будет результат компиляции и запуска класса Y?

```
package p1;
public class X{
    X()    { }
    public void abc()    { System.out.println('X');    }
}
```

---

```
-----  
package p2;
import p1.*;
public class Y extends X{
    Y()    { }
    public void abc()    { System.out.println('Y');    }
    public static void main(String[] args){
        Y y = new Y();
        y.abc();
    }
}
```

# Проверка д/з

Test 2

Каков будет результат компиляции и запуска класса Y?

```
package p1;
public class X{
protected void abc() { System.out.println('X'); }
}
```

---

```
package p2;
import p1.*;
public class Y extends X {
public void abc() { System.out.println('Y'); }
public void def() { Y y = new Y(); y.abc(); }
public void ghi() { X x = new Y(); x.abc(); }
public static void main(String[] args)
{ Y a = new Y(); a.abc(); a.def(); a.ghi(); }
}
```

# Проверка д/з

Test 3

Каков будет результат компиляции и запуска классов X и Y?

```
public class X{  
    public X() { System.out.println("X"); }  
    public X(int i) { System.out.println("X" + i); }  
    private boolean abc() { return false; }  
    public static void main(String[] args) {  
        X x = new Y(18);  
        System.out.println( x.abc() );  
    }  
}
```

```
class Y extends X{  
    public Y(int i) { System.out.println("Y"); }  
    public boolean abc () { return true; }  
}
```

# Проверка д/з

Test 4

Каков будет результат компиляции и запуска класса Test?

```
class X {}  
class Y extends X {}  
  
public class Test {  
    public static void abc(X x, Y y)  
    {  
        System.out.println("privet");  
    }  
    public static void abc(Y y, X x)  
    {  
        System.out.println("poka");  
    }  
    public static void main(String[] args) {  
        Y a = new Y();  
        abc(a, a);  
    }  
}
```

# Проверка д/з

Test 5

Каков будет результат компиляции и запуска класса Test?

```
class X{  
String s1 = "privet";  
}
```

```
class Y extends X {  
boolean bool = false;  
}
```

```
class Test {  
public static void main(String args[]) {  
X x = new Y ();  
System.out.println(x.s1 + " " + x.bool);  
}  
}
```

# abstract классы и abstract методы

```
abstract class Figura{  
    public void abc() { some code }  
    abstract int ploshad();  
}
```

У abstract методов не бывает тела.

Невозможно создать объект abstract класса.

Если в классе есть abstract метод, то этот класс тоже должен быть abstract.

abstract класс может содержать, а может и не содержать abstract методы.

Дочерний класс должен перезаписать все abstract методы родительского класса или тоже быть abstract.

## abstract классы и abstract методы

Можно использовать reference variable типа abstract класса чтобы ссылаться на объект дочернего класса, который не является abstract .

abstract класс не может быть final.

Переменные не могут быть abstract.

У abstract классов есть конструктор.

Любой перезаписанный метод может быть как abstract, так и не abstract.

Для методов недопустимо сочетание: final abstract, private abstract, static abstract

# interface

```
interface Jumpable {  
void jump(int santimetr);  
int CONSTANT = 18;  
}
```

```
class Animal implements Jumpable {  
public void jump(int santimetr) {  
System.out.println("Animal can jump");  
System.out.println("Number"+CONSTANT);} }
```

Интерфейс – это конструкция языка программирования, которую часто сравнивают с контрактом. В этом контракте указано, что класс сможет делать, т.е. какие методы в нём будут присутствовать, если он имплементирует данный интерфейс. Когда класс имплементирует какой либо интерфейс, он обязуется снабдить методы этого интерфейса телами (перезаписать абстрактные методы); в противном случае класс должен стать абстрактным. Т.о. если известно, что класс имплементировал какой либо интерфейс, то в этом классе гарантированно будут методы из интерфейса.

# interface

Невозможно создать объект интерфейса, потому что это не класс.

У интерфейса нет конструкторов.

Access modifier у всех топ-левел интерфейсов или public, или default.

Если не указать самостоятельно, то компилятор добавит в определение интерфейса слово abstract.

Интерфейс не может быть final.

# interface

Если не указать самостоятельно, то компилятор добавит в определение всех non-default (не access modifier) и non-static методов слова abstract и public.

Методы интерфейса не могут быть final.

Из переменных в интерфейсе могут быть только константы, которые должны быть в нём инициализированы.

Если не указать самостоятельно, то компилятор добавит в определение всех переменных слова public, final и static.

Если класс, который implementsовал интерфейс не перезаписал все его методы, то этот класс должен объявляться abstract.

# interface

## It is OK:

```
interface I1{ void abc(int a); }
interface I2{ void abc(String s); }
class Test implements I1, I2{
    public void abc(int a){ some code }
    public void abc(String s){ some code }
}
```

## It is NOT OK:

```
interface I1{ void abc(int a); }
interface I2{ int abc(int a); }
class Test implements I1, I2{
    public void abc(int a){ some code }
    public int abc(int a){ return 5; }
}
```

```
interface Jumpable{ void jump(); }
class Human implements Jumpable {
    public void jump(){ some code }
}
class Animal implements Jumpable {
    public void jump(){ some code }
}
```



```
class Test{
    public static void main(String [] args){
        Jumpable j1 = new Human();
        Jumpable j2 = new Animal();
        j1.jump();           j2.jump();
    }
}
```

# class & interface inheritance

Класс может наследовать 0 или 1 класс. Для наследования класс использует keyword “extends”.

Класс может implementировать 0 или более интерфейсов. Для implementationa класс использует keyword “implements”.

Класс не может наследовать интерфейс.

Интерфейс не может наследовать или implementировать класс.

Интерфейс может наследовать 0 или более интерфейсов. Для наследования интерфейс использует keyword “extends”.

Конкретный класс может наследовать конкретный или абстрактный класс.

abstract класс может наследовать конкретный или абстрактный класс и implementировать интерфейс.

Первый конкретный класс в иерархии должен снабдить все abstract методы телами.

## default методы в interface

```
interface Jumpable{  
    default void jump() {  
        System.out.println("You can jump!!!"); }  
}
```

Данное слово **default** никаким образом не связанно с access modifier. Access modifier **default**-ных методов в интерфейсе – **public**, который будет добавлен компилятором, если его не указать самостоятельно.

**default** методы предоставляют дефолтную реализацию метода и могут быть созданы только в интерфейсе.

Если в интерфейсе создан **default** метод, то он должен иметь тело.

**default** методы не должны быть **static**, **final** или **abstract**.

## default методы в interface

```
interface I1 { default void abc (){  
System.out.println("I1"); } }  
interface I2 { default void abc (){  
System.out.println("I2"); } }  
interface I3 { void abc (); }
```

### NOTOK:

```
class Test1 implements I1, I2{ }  
abstract class Test2 implements I1, I3{ }
```

### OK:

```
class Test3 implements I1, I2{  
public void abc (){ System.out.println("Test3"); } }  
class Test4 implements I1, I3{  
public void abc (){ System.out.println("Test4"); } }
```

## static методы в interface

```
interface I4{  
static void abc() {  
System.out.println("Static method"); } }
```

Если не указать самостоятельно, то компилятор добавит в определение всех static методов слово public.

static методы не наследуются ни одним классом, который имплементирует интерфейс.

Для вызова static метода необходимо использовать имя интерфейса.

## Подведение итогов

**abstract**  
методы



**abstract**  
классы

**interface**

Что было новым в  
java8 для  
интерфейсов

## Домашнее задание:

Создайте абстрактный класс Animal, его конструктор пусть имеет параметр, значение которого назначается переменной String name. В классе Animal создайте абстрактные методы eat и sleep. Создайте абстрактный класс Fish, который является дочерним классом класса Animal, его конструктор пусть имеет параметр, значение которого назначается переменным name данного и родительского класса. В классе Fish перезапишите метод sleep так, чтобы он выводил на экран "Vsegda interesno nablyudat, kak spyat ribi". Также здесь создайте абстрактный метод swim.

Создайте абстрактный класс Bird, который является дочерним классом класса Animal, его конструктор пусть имеет параметр, значение которого назначается переменным name данного и родительского класса. Также здесь создайте абстрактный метод fly.

Создайте абстрактный класс Mammal, который является дочерним классом класса Animal, его конструктор пусть имеет параметр, значение которого назначается переменным name данного и родительского класса. Также здесь создайте абстрактный метод run.

Создайте интерфейс Speakable, в котором пусть будет дефолтный метод speak, который выводит на экран "Somebody speaks". Пусть класс Mammal implements этот интерфейс. Также пусть класс Bird implements этот интерфейс и перезаписывает его метод так, чтобы он выводил на экран имя + " sings".

Создайте класс Mechenosec, который является дочерним классом класса Fish, его конструктор пусть имеет параметр, значение которого назначается переменным name данного и родительского класса. В классе Mechenosec перезапишите метод swim так, чтобы он выводил на экран "Mechenosec krasivaya riba, kotoraya bistro plavaet!". Также перезапишите метод eat так, чтобы он выводил на экран "Mechenosec ne xishnaya riba, i ona est obichniy ribiy korm!".

## Домашнее задание:

Создайте класс Pingvin, который является дочерним классом класса Bird, его конструктор пусть имеет параметр, значение которого назначается переменным паме данного и родительского класса. В классе Pingvin перезапишите метод eat так, чтобы он выводил на экран "Pingvin lyubit est ribu!". Также перезапишите метод sleep так, чтобы он выводил на экран "Pingvini spyat prijavshis drug k drugu!". Также перезапишите метод fly так, чтобы он выводил на экран "Pingvini ne umeyut letat!". Также перезапишите метод speak так, чтобы он выводил на экран "Pingvini ne umeyut pet kak solovyi!".

Создайте класс Lev, который является дочерним классом класса Mammal, его конструктор пусть имеет параметр, значение которого назначается переменным паме данного и родительского класса. В классе Lev перезапишите метод eat так, чтобы он выводил на экран "Lev, kak lyuboy xishnik, lyubit myaso!". Также перезапишите метод sleep так, чтобы он выводил на экран "Bolshuyu chast dnya lev spit!". Также перезапишите метод run так, чтобы он выводил на экран "Lev-eto ne samaya bistraya koshka!".

В классе Lesson24 создайте main метод, в котором:

1. Создайте объект класса Mechenosec, на который ссылается переменная типа Mechenosec, выведите переменную паме данного объекта и вызовите все методы, которые сможете вызвать с помощью данной переменной;
2. Создайте объект класса Pingvin, на который ссылается переменная типа Speakable, вызовите все методы, которые сможете вызвать с помощью данной переменной;
3. Создайте объект класса Lev, на который ссылается переменная типа Animal, выведите переменную паме данного объекта и вызовите все методы, которые сможете вызвать с помощью данной переменной;
4. Создайте ещё один объект класса Lev, на который ссылается переменная типа Mammal, выведите переменную паме данного объекта и вызовите все методы, которые сможете вызвать с помощью данной переменной.

*The End*

**Путь к Java ОСА.**

**Эффективное пособие для начинающих.**

# **урок 25**

# Краткий повтор пройденного материала

abstract methods

abstract class

interface

Что было новым в java8 для интерфейсов

# Проверка д/з

Создайте абстрактный класс Animal, его конструктор пусть имеет параметр, значение которого назначается переменной String name. В классе Animal создайте абстрактные методы eat и sleep. Создайте абстрактный класс Fish, который является дочерним классом класса Animal, его конструктор пусть имеет параметр, значение которого назначается переменным name данного и родительского класса. В классе Fish перезапишите метод sleep так, чтобы он выводил на экран "Vsegda interesno nablyudat, kak spyat ribi". Также здесь создайте абстрактный метод swim. Создайте абстрактный класс Bird, который является дочерним классом класса Animal, его конструктор пусть имеет параметр, значение которого назначается переменным name данного и родительского класса. Также здесь создайте абстрактный метод fly.

Создайте абстрактный класс Mammal, который является дочерним классом класса Animal, его конструктор пусть имеет параметр, значение которого назначается переменным name данного и родительского класса. Также здесь создайте абстрактный метод run.

Создайте интерфейс Speakable, в котором пусть будет дефолтный метод speak, который выводит на экран "Somebody speaks". Пусть класс Mammal implements этот интерфейс. Также пусть класс Bird implements этот интерфейс и перезаписывает его метод так, чтобы он выводил на экран имя + " sings".

Создайте класс Mechenosec, который является дочерним классом класса Fish, его конструктор пусть имеет параметр, значение которого назначается переменным name данного и родительского класса. В классе Mechenosec перезапишите метод swim так, чтобы он выводил на экран "Mechenosec krasivaya riba, kotoraya bistro plavaet!". Также перезапишите метод eat так, чтобы он выводил на экран "Mechenosec ne xishnaya riba, i ona est obichniy ribiy korm!".

# Проверка д/з

Создайте класс Pingvin, который является дочерним классом класса Bird, его конструктор пусть имеет параметр, значение которого назначается переменным name данного и родительского класса. В классе Pingvin перезапишите метод eat так, чтобы он выводил на экран "Pingvin lyubit est ribu!". Также перезапишите метод sleep так, чтобы он выводил на экран "Pingvini spyat prijavshis drug k drugu!". Также перезапишите метод fly так, чтобы он выводил на экран "Pingvini ne umeyut letat!". Также перезапишите метод speak так, чтобы он выводил на экран "Pingvini ne umeyut pet kak solovyi".

Создайте класс Lev, который является дочерним классом класса Mammal, его конструктор пусть имеет параметр, значение которого назначается переменным name данного и родительского класса. В классе Lev перезапишите метод eat так, чтобы он выводил на экран "Lev, kak lyuboy xishnik, lyubit myaso!". Также перезапишите метод sleep так, чтобы он выводил на экран "Bolshuyu chast dnya lev spit!". Также перезапишите метод run так, чтобы он выводил на экран "Lev-eto ne samaya bistraya koshka!".

В классе Lesson24 создайте main метод, в котором:

1. Создайте объект класса Mechenosec, на который ссылается переменная типа Mechenosec, выведите переменную name данного объекта и вызовите все методы, которые сможете вызвать с помощью данной переменной;
2. Создайте объект класса Pingvin, на который ссылается переменная типа Speakable, вызовите все методы, которые сможете вызвать с помощью данной переменной;
3. Создайте объект класса Lev, на который ссылается переменная типа Animal, выведите переменную name данного объекта и вызовите все методы, которые сможете вызвать с помощью данной переменной;
4. Создайте ещё один объект класса Lev, на который ссылается переменная типа Mammal, выведите переменную name данного объекта и вызовите все методы, которые сможете вызвать с помощью данной переменной.

Ответ: Homework.Lesson24

# Polymorphism

Дословный перевод слова «полиморфизм» - множество форм.

Полиморфизм – это способность объекта принимать несколько форм.

Объект в Java считается полиморфным, если он имеет более 1 связи IS-A.

Полиморфизм – это способность метода вести себя по разному в зависимости от того, объект какого класса вызывает этот метод.

Перезаписанные методы также часто называют полиморфными.

## Оператор instanceof

```
Car c = new Car();
System.out.print(c instanceof Object);
```

Оператор instanceof проверяет, есть ли между объектом и классом/интерфейсом связь IS-A. Если связь IS-A невозможна, то компилятор выдаёт ошибку.

## Reference data types casting

Casting – это процесс когда вы заставляете переменную одного типа данных вести себя как переменная другого типа данных.

Casting возможен только тогда, когда между классами/интерфейсами существует IS-A взаимоотношение.

Делая casting, вы не меняете тип данных объекта, а заставляете его чувствовать себя как объект другого типа.

Casting из sub класса в super класс происходит автоматически - Upcasting

Casting из super класса в sub класс НЕ происходит автоматически - Downcasting

Если между классами/интерфейсами нет IS-A взаимоотношения, компилятор не допустит casting.

Даже если компилятор допустил casting, выскочит runtime exception, если объект, который мы делаем cast на самом деле не принадлежит классу, на который мы его делаем cast.

# Primitive data types casting

## 19 forms of widening:

byte to short, int, long, float, or double  
short to int, long, float, or double  
char to int, long, float, or double  
int to long, float, or double  
long to float or double  
float to double

Narrowing without casting происходит,  
если выполняются 3 условия:

1. Если int cast-ится в byte, short или char;
2. Если значение int – это константа;
3. Если значение int помещается в  
соответствующий тип данных.

## 22 forms of narrowing:

short to byte or char  
char to byte or short  
int to byte, short, or char  
long to byte, short, char, or int  
float to byte, short, char, int, or long  
double to byte, short, char, int, long, or float

# Numeric promotion

Это конвертация меньшего численного типа данных в больший.

Numeric promotion происходит в следующих случаях:

Если имеются 2 значения разных типов данных, Java автоматически конвертирует меньший числовой тип данных в больший.

Если одно из значений - это целое число, а другое дробное, Java автоматически конвертирует целочисленный тип данных в дробный.

Если значения типов данных byte, short, и char участвуют в арифметических операциях, то они перед этим конвертируются в тип данных int, даже если в данных арифметических операциях значение типа данных int не участвует.  
Иключение: Унарные операторы. Использование ++ к типу данных byte не конвертирует результат оператора в int.

После применения вышеуказанных 3-х правил, когда все операнды стали одного типа данных, результирующее значение, которое получается после срабатывания всех операторов будет такого же типа данных, что и тип данных operand.

# Test

Каков будет результат компиляции и запуска данного кода?

```
class Test1 implements interface1, interface2{
    public void abc() { System.out.println("OK"); }
    public static void main(String[] args){
        Test1 t = new Test1();
        ( (interface1) t).abc();
    }
}
interface interface1{
    int a = 5;
    void abc();
}
interface interface2{
    int a = 10;
    void abc();
}
```

# Test

Рассмотрите данный код:

```
class Employee {  
void sleep() { System.out.println("Employee sleeps"); } }
```

```
class Doctor extends Employee {  
void sleep() { System.out.println("Doctor sleeps"); } }
```

```
class Test2 {  
public static void main(String [] args){  
Employee e = new Employee();  
// some code  
e.print();  
x.print(); } }
```

Какие выражения (по отдельности) могут быть написаны на строке //some code чтобы на экран выводилось:

Employee sleeps  
Doctor sleeps

- A) Employee x = new Employee();
- B) Employee x = new Doctor();
- C) Doctor x = new Doctor();
- D) Doctor x = new Employee();
- E) Employee x = (Doctor)new Employee();
- F) Doctor x = (Doctor)new Employee();

# Test

Каков будет результат компиляции и запуска данного кода?

```
class Animal {  
}
```

```
class Mouse extends Animal {  
    void abc() {  
        def(new Animal(), new Mouse());  
        def((Mouse) new Animal(), new Mouse());  
    }  
    void def(Animal a1, Mouse m1) {  
        Mouse m2 = (Mouse) a1;  
        Animal a2 = (Animal) m1;  
    }  
    public static void main(String[] args) {  
        new Mouse().abc();  
    }  
}
```

## Подведение итогов

**Polymorphism**

**instanceof**

**casting**

**Numeric  
promotion**



## Домашнее задание:

Переименуйте класс Lesson24 из последнего домашнего задания в Lesson25 и измените его метод main следующим образом. Создайте в нём 2 массива типа Speakable и типа Animal, которые будут содержать все возможные ссылочные переменные, ссылающиеся на все возможные объекты. Используя if и instanceof проверяйте на какой объект ссылается переменная и выводите на экран соответствующие переменные данного объекта и вызывайте его методы.

*The End*

**Путь к Java ОСА.**

**Эффективное пособие для начинающих.**

# **урок 26**

Краткий повтор пройденного материала

Polymorphism

instanceof

Casting

Numeric promotion

## Проверка д/з

Переименуйте класс Lesson24 из последнего домашнего задания в Lesson25 и измените его метод main следующим образом. Создайте в нём 2 массива типа Speakable и типа Animal, которые будут содержать все возможные ссылочные переменные, ссылающиеся на все возможные объекты. Используя if и instanceof проверяйте на какой объект ссылается переменная и выводите на экран соответствующие переменные данного объекта и вызывайте его методы.

Ответ: Homework.Lesson25

## Методы equals и toString.

Если вы перезаписываете метод equals, всегда используйте в его параметре тип данных Object.

Правильно и логично перезаписанный метод equals должен обладать следующими свойствами:

1. Симметричность – для non-null ссылочных переменных a и b, a.equals(b) возвращает true тогда и только тогда, когда b.equals(a) возвращает true;
2. Рефлексивность – для non-null ссылочной переменной a, a.equals(a) всегда должно возвращать true;
3. Транзитивность – для non-null ссылочных переменных a, b и c, если a.equals(b) и b.equals(c) возвращает true, то a.equals(c) тоже должно возвращать true;
4. Постоянство – для non-null ссылочных переменных a и b, неоднократный вызов a.equals(b) должен возвращать или только true, или только false;
5. Для non-null ссылочной переменной a, a.equals(null) всегда должно возвращать false;

Метод toString принадлежит классу Object, возвращает строковое представление объекта. Дефолтная реализация данного метода возвращает имя класса, @, число (результат метода hashCode данного объекта).

# Wrapper classes

byte – Byte  
short - Short  
int - Integer  
long - Long

float – Float  
double - Double  
char - Char  
boolean - Boolean

Autoboxing – это конвертирование примитивных типов данных в соответствующий wrapper класс.

Unboxing – это конвертирование объекта типа wrapper класс в соответствующий примитивный тип данных.

Метод parse позволяет нам конвертировать подходящее значение типа данных String в соответствующий примитивный тип данных.

Метод valueOf позволяет нам создавать новый объект wrapper класса того типа, на котором данный метод был вызван.

## Method overloading, приоритетность методов

Если при вызове метода его параметр лист соответствует нескольким параметрам listам overloaded методов, то приоритет их вызова выглядит следующим образом:

1. Точное совпадение типов данных;
2. Поглощающие типы данных (большие типы данных для primitive и parent классы для reference типов);
3. Autoboxed типы данных;
4. Varargs.

Конвертация типов данных для соответствия параметру листу метода не может происходить в 2 этапа.

# Non-static and static initializers

Initializer block срабатывает каждый раз, когда создаётся новый объект соответствующего класса.

Static initializer block срабатывает один раз, когда класс загружается в память.

Равнозначные initializer блоки выполняются в той последовательности, в которой они описаны в классе.

Последовательность инициализации initializer блоков и переменных:

1. Статические блоки и переменные родительского класса;
2. Статические блоки и переменные дочернего класса;
3. Не статические блоки и переменные родительского класса;
4. Конструктор родительского класса;
5. Не статические блоки и переменные дочернего класса;
6. Конструктор дочернего класса.

Инициализация пунктов 3-6 происходит только и при каждом создании объекта.

## Non-access modifiers

**transient.** Переменные класса с ключевым словом transient не сериализуются.

**native.** Методы с ключевым словом native реализованы не на Java. В своём описании они не имеют тела и заканчиваются как абстрактные методы символом «;»

**synchronized.** Методы с ключевым словом synchronized могут быть использованы в одно и то же время только одним потоком.

**volatile.** Переменные с ключевым словом volatile могут быть изменены разными потоками и данные изменения будут видны во всех потоках.

**strictfp.** Ключевое слово strictfp в методах и классах ограничивает точность вычислений с float и double по стандарту IEEE.

# Test

Каков будет результат компиляции и запуска данного кода?

```
public class Test {  
    void abc(short s)    { System.out.println("byte"); }  
    void abc(int i)      { System.out.println("int"); }  
    void abc(float f)    { System.out.println("float"); }  
    void abc(Object o)   { System.out.println("Object"); }  
}
```

```
public static void main(String[] args) {  
    Test t = new Test();  
    char c = 55;  
    t.abc(c);  
    t.abc(false);  
    t.abc(3.14);  
}  
}
```

# Test

Каков будет результат компиляции и запуска данного кода?

```
class X {  
    static String s = "";  
    { s += "A"; }  
    static  
    { s += "B"; }  
    { s += "C"; }  
}  
class Z {  
    public static void main(String[] args) {  
        System.out.println(X.s);  
        System.out.println(X.s);  
        new X();  
        new X();  
        System.out.println(X.s);  
    }  
}
```

## Подведение итогов

**equals**

**Wrapper  
classes**

**initializers**

**toString**

**Method  
overloading**

**Non-access  
modifiers**



*The End*

**Путь к Java ОСА.**

**Эффективное пособие для начинающих.**

# **урок 27**

# Краткий повтор пройденного материала

Method equals

Method toString

Wrapper classes

Method overloading

Initializers

Non-access modifiers

# Введение в java.io

java.io – это пакет, в котором собраны классы и интерфейсы, которые предназначены, если обобщить, для чтения и записи информации из/в какой-либо источник, например файл.

Класс File – абстрактная репрезентация пути к файлу или папке.

Класс FileInputStream предназначен для создания потока, с помощью которого можно читать информацию из источника.

Класс FileOutputStream предназначен для создания потока, с помощью которого можно писать информацию в источник.

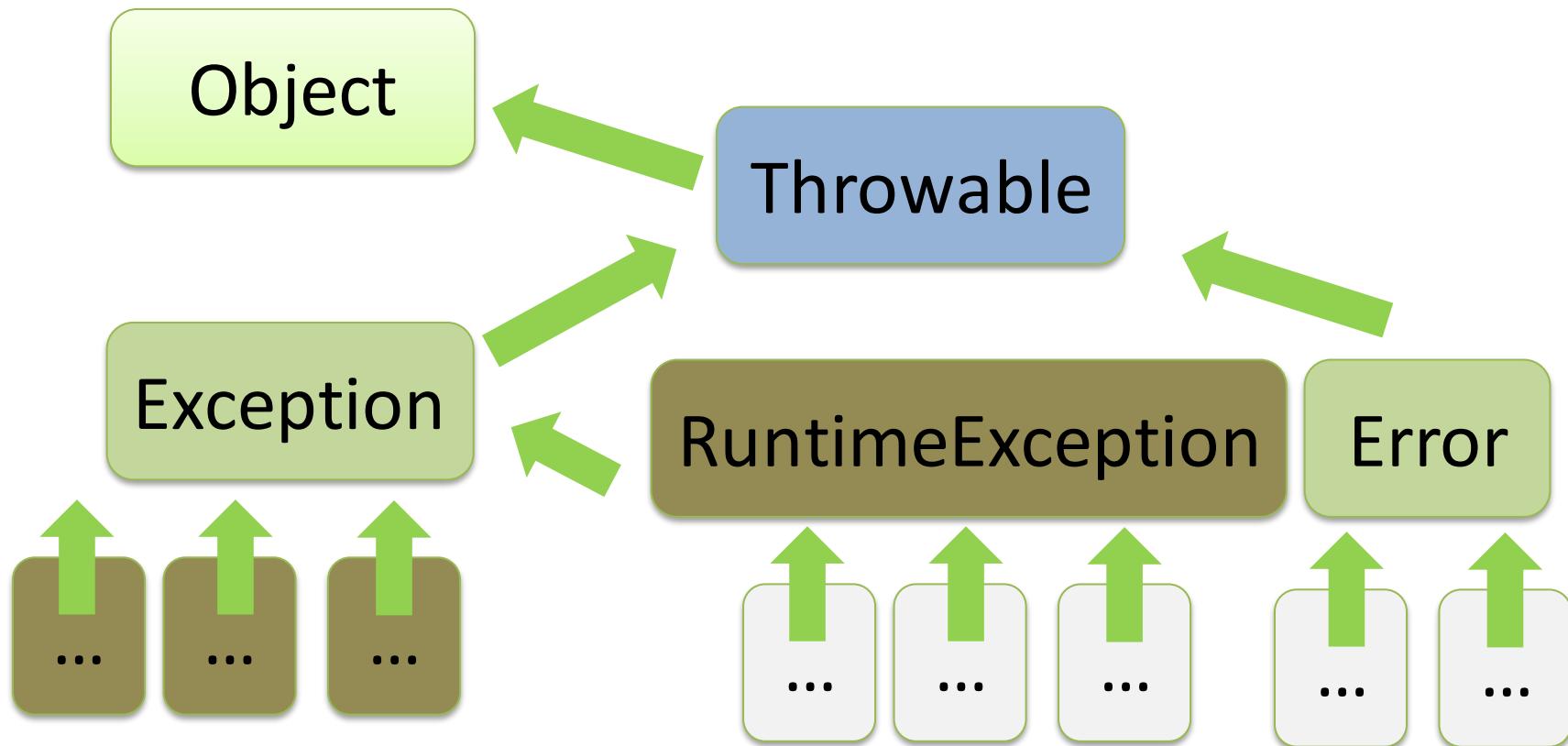
Конструкторы классов FileInputStream и FileOutputStream могут выбрасывать исключение FileNotFoundException.

Методы read и write классов FileInputStream и FileOutputStream могут выбрасывать исключение IOException.

# Исключения

Обработка исключений позволяет нам разграничивать код на код, который должен выполняться при обычном протекании программы и код, который должен выполняться при выбросе исключений.

## Иерархия классов.



# try/catch/finally

```
try{some code}  
catch(Exception_name_1 e){some code}  
catch(Exception_name_2 e){some code}  
.....  
catch(Exception_name_n e){some code}  
finally{some code}
```

Несколько catch блоков могут следовать за try блоком, но лишь 1 finally блок может следовать за catch блоками.

Одновременно вы можете использовать все 3 типа блоков или по 2: try блок с catch блоком/ами; try блок с finally блоком. По отдельности блоки использовать невозможно.

Последовательность блоков всегда должна соответствовать следующему порядку: try → catch → finally

finally блок выполняется вне зависимости от того выбросилось исключение или нет.

## Сабклассы RuntimeException = unchecked exceptions

Runtime исключения бывают в коде, в котором присутствуют ошибочные выражения. Т.е. в выбросе данных исключений виноват программист. Компилятор НЕ в состоянии проверить возможность выброса runtime исключений.

Runtime исключения можно не объявлять и не обрабатывать, но при желании можно сделать и то, и другое.

## Сабклассы Exception = checked exceptions

Checked исключения указывают на часть кода, который находится за пределами непосредственного контроля программой и который может являться причиной выброса исключений. Они как правило возникают при взаимодействии вашей программы с внешними источниками (работа с файлами, с БД, с сетью), из за которых и могут возникать проблемы. Компилятор всегда проверяет возможность выброса checked исключений.

Checked исключения всегда должны быть или объявленны и/или обработаны.

## Checked exceptions

Если метод а использует внутри себя метод b, который может выбросить checked исключение, то метод а должен:

1. Или заключить вызов метода b в try/catch блоки;
2. Или/и объявить, что он тоже может выбросить это checked исключение или его super класс.

## Error

Error – это очень серьёзные ошибки, которые не могут быть напрямую контролированы вашей программой.

Error-ы могут быть объявленны, но объявлять error-ы считается глупой практикой. Некоторые error-ы могут быть обработаны, но это тоже будет очень глупой затеей. Error-ы, как и runtime исключения считаются unchecked.

## Важные моменты в исключениях

Очерёдность catch блоков очень важна. Компилятор не пропустит код, если исключение «super class» будет стоять перед исключением «sub class».

Если в части кода, которая не находится в блоке try или в блоке try выбрасывается исключение, то соответствующая оставшаяся часть кода уже не обрабатывается.

После выброса исключения мы можем увидеть стэк трейс для всех методов, задействованных в выбросе этого исключения.

При создании объекта исключения вы можете воспользоваться его конструктором, который принимает String параметр и вписать в него необходимую информацию. Вы также можете воспользоваться конструктором, который не принимает параметры.

```
catch (Exception e) {  
    System.out.println(e); //выводит на экран тип исключения и сообщение(мессаж)  
    System.out.println(e.getMessage()); //выводит на экран сообщение(мессаж)  
    e.printStackTrace(); } //выводит на экран стэк трейс
```

## Важные моменты в исключениях

finally блок выполняется даже в том случае, если в try блоке или в catch блоке имеется return statement.

finally блок не выполнится только в том случае, если вы прекращаете работу программы с помощью System.exit в try блоке или в catch блоке или же, если происходит крушение JVM или, например, операционной системы.

Если return statement имеется и в catch блоке, и в finally блоке, то оутпутом метода будет возвращаемое значение из finally блока.

Если return statement в catch блоке возвращает primitive data type, то в finally блоке вы его изменить не сможете. Если return statement в catch блоке возвращает reference data type, то в finally блоке вы его сможете изменить (естественно, если тип mutable).

Исключение может быть перевыброшено. Это обычно делают тогда, когда код вашего текущего метода обработал данное исключение не полностью и выбрасывает его вновь, чтобы метод, который будет вызывать текущий метод завершил обработку данного исключения.

## Важные моменты в исключениях

При написании кода, вы можете использовать вложенные блоки try, catch и finally.

Если исключение выбрасывается из catch блока, то оно не может быть обработано одноуровневым catch блоком.

При написании кода, вы можете создавать свои собственные исключения. В зависимости от необходимости, создавайте исключения, которые наследуются от классов Exception или RuntimeException.

# Распространённые сабклассы RuntimeException

ArrayIndexOutOfBoundsException

IndexOutOfBoundsException

ArithmaticException

ClassCastException

IllegalArgumentException

IllegalStateException

NullPointerException

NumberFormatException

# Распространённые сабклассы Error

ExceptionInInitializerError

StackOverflowError

OutOfMemoryError

NoClassDefFoundError

## Исключения, method overriding and overloading, constructors

Если класс перезаписывает метод из супер класса или implements метод из интерфейса непозволительно добавлять в его сигнатуру новые checked исключения. Можно в сигнатуре метода использовать только исключения из перезаписанного метода супер класса или дочерние классы данных исключений.

Вышеннаписанное правило никаким образом не относится к перегруженным методам.

Конструктор может выбрасывать исключения. Конструктор в своей сигнатуре должен описывать все исключения конструктора супер класса, который он вызывает, может описывать супер классы данных исключений, а также добавлять новые исключения.

# Важные моменты в исключениях

Указание в сигнатуре метода исключения, которое не будет выбрасываться в данном методе не является ошибкой.

## NOT OK:

```
static void abc() { }
public static void main(String[] args) {
    try{ abc(); }
    catch(IOException e){}
}
```

## OK:

```
static void abc() throws IOException { }
public static void main(String[] args) {
    try{ abc(); }
    catch(IOException e){}
}
```

## Unreachable statements

```
public int abc() throws Exception{
    throw new Exception();
    return 5;
}
```

```
public int abc() throws Exception{
    return 5;
    throw new Exception();
}
```

# Test

Каков будет результат компиляции и запуска данного кода?

```
class Test{
    static String str = "";
    void ghi()throws Exception{
        throw new Exception();
    }
    void def()throws Exception{
        ghi();
        str+="2";
        ghi();
        str+="3";
    }
    void abc(){
        try{def();}
        catch(Exception e) {str+="1";}
    }
    public static void main(String[] args) {
        Test t = new Test();
        t.abc();
        System.out.println(str);
    }
}
```

Output: 1

# Test

Какие из этих классов представляют собой unchecked исключения?

- A) CompilationException
- B) NumberException
- C) NullPointerException ←
- D) Throwable
- E) StackOverflowException
- F) RuntimeException ←
- G) ArrayIndexOutOfBoundsException ←
- H) IllegalArgumentException ←
- I) MemoryOutOfBoundsException
- J) CheckedException

## Подведение итогов

**Checked  
exceptions**

**Unchecked  
exceptions**

**try/catch/  
finally**

**Errors**



## Домашнее задание:

Создайте 2 исключения. 1-ый пусть является дочерним классом класса RuntimeException и называется NeMyasoException; в нём создайте конструктор, который принимает 1 String параметр и передаёт его конструктору super класса. 2-ой пусть является дочерним классом класса Exception и называется NeVodaException; в нём создайте конструктор, который принимает 1 String параметр и передаёт его конструктору super класса.

Создайте класс Tiger. Первый метод класса - метод eat будет иметь return type void и принимать 1 String параметр. Если данный параметр не равен String-у "myaso", то пусть метод выбрасывает объект NeMyasoException с параметром "Tigr ne est " + параметр метода. Если данный параметр равен String-у "myaso", то пусть на экран выводится "Tigr est myaso". Второй метод класса - метод drink будет иметь return type void и принимать 1 String параметр. Если данный параметр не равен String-у "voda", то пусть метод выбрасывает объект NeVodaException с параметром "Tigr ne pyet " + параметр метода. Если данный параметр равен String-у "voda", то пусть на экран выводится "Tigr pyet vodu".

В классе Lesson27 внутри метода main создайте объект класса Tiger. Вызовите метод eat с параметром "myaso". Затем в блоке try вызовите метод drink с параметром voda. В данном блоке try создайте ещё один блок try, где вызовите метод drink с параметром pivo. Во внешнем блоке try пусть также размещаются блок catch, который ловит исключения типа Exception и в своем теле выводит на экран мессаж данного исключения; блок catch, который ловит исключения типа NeVodaException и в своем теле выводит на экран мессаж данного исключения; блок finally, который в своем теле выводит на экран "Eto inner finally block". К внешнему try блоку пусть относятся блок catch, который ловит исключения типа Exception и в своем теле выводит на экран мессаж данного исключения; блок catch, который ловит исключения типа RuntimeException и в своем теле выводит на экран мессаж данного исключения.

*The End*

**Путь к Java ОСА.**

**Эффективное пособие для начинающих.**

# **урок 28**

Краткий повтор пройденного материала

Checked exceptions

Unchecked exceptions

Errors

try/catch/finally blocks

# Проверка д/з

Создайте 2 исключения. 1-ый пусть является дочерним классом класса RuntimeException и называется NeMyasoException; в нём создайте конструктор, который принимает 1 String параметр и передаёт его конструктору super класса. 1-ой пусть является дочерним классом класса Exception и называется NeVodaException; в нём создайте конструктор, который принимает 1 String параметр и передаёт его конструктору super класса.

Создайте класс Tiger. Первый метод класса - метод eat будет иметь return type void и принимать 1 String параметр. Если данный параметр не равен String-у "myaso", то пусть метод выбрасывает объект NeMyasoException с параметром "Tigr ne est " + параметр метода. Если данный параметр равен String-у "myaso", то пусть на экран выводится "Tigr est myaso". Второй метод класса - метод drink будет иметь return type void и принимать 1 String параметр. Если данный параметр не равен String-у "voda", то пусть метод выбрасывает объект NeVodaException с параметром "Tigr ne pyet " + параметр метода. Если данный параметр равен String-у "voda", то пусть на экран выводится "Tigr pyet vodu".

В классе Lesson27 внутри метода main создайте объект класса Tiger. Вызовите метод eat с параметром "myaso". Затем в блоке try вызовите метод drink с параметром voda. В данном блоке try создайте ещё один блок try, где вызовите метод drink с параметром pivo. Во внешнем блоке try пусть также размещаются блок catch, который ловит исключения типа Exception и в своем теле выводит на экран мессаж данного исключения; блок catch, который ловит исключения типа NeVodaException и в своем теле выводит на экран мессаж данного исключения; блок finally, который в своем теле выводит на экран "Eto inner finally block". К внешнему try блоку пусть относятся блок catch, который ловит исключения типа Exception и в своем теле выводит на экран мессаж данного исключения; блок catch, который ловит исключения типа RuntimeException и в своем теле выводит на экран мессаж данного исключения.

Ответ: Homework.Lesson27

## Дни и время

LocalDate класс содержит информацию о дне: год, месяц, день.

LocalTime класс содержит информацию о времени: час, минуты, секунда, наносекунда.

LocalDateTime класс содержит информацию о дне и времени: год, месяц, день, час, минуты, секунда, наносекунда.

Данные классы содержатся в пакете java.time

Данные 3 класса имеют статический метод now(), который возвращает соответствующие объекты с текущими значениями.

# Создание объектов классов LocalDate, LocalTime и LocalDateTime

У этих 3-х классов конструктор имеет access modifier private. Поэтому, мы не можем создавать объекты этих классов используя конструкторы. Методы of возвращают объект соответствующего типа.

## Методы of

public static LocalDate of(int год, int месяц, int день\_месяца)

public static LocalDate of(int год, Month месяц, int день\_месяца)

public static LocalTime of(int час, int минута)

public static LocalTime of(int час, int минута, int секунда)

public static LocalTime of(int час, int минута, int секунда, int наносекунда)

## Методы of

public static LocalDateTime of(int год, int месяц, int день\_месяца,  
int час, int минута)

public static LocalDateTime of(int год, int месяц, int день\_месяца,  
int час, int минута, int секунда)

public static LocalDateTime of(int год, int месяц, int день\_месяца,  
int час, int минута, int секунда, int наносекунда)

public static LocalDateTime of(int год, Month месяц, int день\_месяца,  
int час, int минута)

public static LocalDateTime of(int год, Month месяц, int день\_месяца,  
int час, int минута, int секунда)

public static LocalDateTime of(int год, Month месяц, int день\_месяца,  
int час, int минута, int секунда, int наносекунда)

public static LocalDateTime of(LocalDate день, LocalTime время)

При некорректном указании параметров метода of выбрасывается  
соответствующий exception.

# Изменение объектов классов LocalDate, LocalTime и LocalDateTime

**Объекты данных 3-х классов являются immutable.**

## Методы LocalDate

plusDays(long количество\_дней) → LocalDate

minusDays(long количество\_дней) → LocalDate

plusWeeks(long количество\_недель) → LocalDate

minusWeeks(long количество\_недель) → LocalDate

plusMonths(long количество\_месяцев) → LocalDate

minusMonths(long количество\_месяцев) → LocalDate

plusYears(long количество\_лет) → LocalDate

minusYears(long количество\_лет) → LocalDate

## Методы LocalTime

plusHours(long количество\_часов) → LocalTime

minusHours(long количество\_часов) → LocalTime

plusMinutes(long количество\_минут) → LocalTime

minusMinutes(long количество\_минут) → LocalTime

plusSeconds(long количество\_секунд) → LocalTime

minusSeconds(long количество\_секунд) → LocalTime

plusNanos(long количество\_наносекунд) → LocalTime

minusNanos(long количество\_наносекунд) → LocalTime

## Методы LocalDateTime

Методы LocalDateTime охватывают методы LocalDate и LocalTime и в оутпуть возвращают LocalDateTime.

# Методы isAfter и isBefore

Методы isAfter и isBefore могут быть использованы для сравнения объектов классов LocalDate, LocalTime и LocalDateTime. Данные методы возвращают boolean. Компилятор разрешает сравнивать только объекты соответствующих классов.

## Класс Period

Класс Period имеет конструктор с access modifier private. Методы of возвращают объект типа Period.

public static Period ofDays(int колич\_дней)

public static Period ofWeeks (int колич\_недель)

public static Period ofMonths(int колич\_месяцев)

public static Period ofYears(int колич\_лет)

public static Period of(int колич\_лет, int колич\_месяцев, int колич\_дней)

Методы plus и minus могут быть использованы для прибавления и отнимания периодов к/от объектов класса LocalDate и LocalDateTime. При попытке использования методов plus или minus для сложения или отнимания периода к/от объекта класса LocalTime будет выброшен exception.

При создании объекта класса Period не работает method chaining метода of. При попытке method chaining только последний метод of имеет значение.

## Класс Duration

Класс Duration имеет конструктор с access modifier private. Методы of возвращают объект типа Duration.

public static Duration ofDays(long колич\_дней)

public static Duration ofHours (long колич\_часов)

public static Duration ofMinutes(long колич\_минут)

public static Duration ofSeconds(long колич\_секунд)

public static Duration ofMillis(long колич\_миллисекунд)

public static Duration ofNanos(long колич\_наносекунд)

Методы plus и minus могут быть использованы для прибавления и отнимания объекта класса Duration к/от объектов класса LocalTime и LocalDateTime. При попытке использования методов plus или minus для сложения или отнимания объекта класса Duration к/от объекта класса LocalDate будет выброшен exception.

При создании объекта класса Duration не работает method chaining метода of. При попытке method chaining только последний метод of имеет значение.

## Получение информации из класса LocalDate

getDayOfWeek() → DayOfWeek

getDayOfMonth() → int

getDayOfYear() → int

getMonth() → Month

getMonthValue() → int

getYear() → int

## Получение информации из класса LocalTime

getNano() → int

getSecond() → int

getMinute() → int

getHour() → int

## Получение информации из класса LocalDateTime

Данный класс включает в себя все методы из классов LocalDate и LocalTime.

# Класс DateTimeFormatter

Данный класс находится в пакете java.time.format

С помощью метода format вы можете изменять вывод вашей даты или вашего времени на экран.

```
LocalDateTime ldt = LocalDateTime.of(2014, Month.MARCH, 10, 15, 20, 30);
DateTimeFormatter f = DateTimeFormatter.ISO_WEEK_DATE;
System.out.println(ldt.format(f));
```

```
LocalDateTime ldt = LocalDateTime.of(2014, Month.MARCH, 10, 15, 20, 30);
DateTimeFormatter f1 = DateTimeFormatter.ofLocalizedDate(FormatStyle.SHORT);
System.out.println(ldt.format(f1));
DateTimeFormatter f2 = DateTimeFormatter.ofLocalizedDate(FormatStyle.MEDIUM);
System.out.println(ldt.format(f2));
```

Методы класса DateTimeFormatter.ofLocalizedDate, ofLocalizedTime и ofLocalizedDateTime должны создавать формат, который будет применяться для соответствующих классов.

Метод format имеется не только у объектов классов LocalDate, LocalTime, LocalDateTime, но и у класса DateTimeFormatter, что делает возможным написание последнего выражения в следующем виде:

```
System.out.println(f2.format(ldt));
```

И эти 2 выражения имеют один и тот же результат.

# Класс DateTimeFormatter

С помощью метода ofPattern вы можете создавать свой формат.

```
LocalDateTime ldt = LocalDateTime.of(2014, Month.MARCH, 10, 15, 20, 30, 5555);
```

"Y" = 2014

"YY" = 14

"YYYY" = 2014

"M" = 3

"MM" = 03

"MMM" = мар

"MMMM" = марта

"w" = 11

"ww" = 11

"d" = 10

"dd" = 10

"h" = 3

"hh" = 03

"m" = 20

"mm" = 20

"s" = 30

"ss" = 30

"n" = 5555

"nnnnn" = 05555

У объектов классов LocalDate, LocalTime, LocalDateTime нужно пытаться брать ту информацию, которую они могут содержать. В противном случае будет выбрасываться исключение.

## Класс DateTimeFormatter

Метод `parse` переводит `String` в объект классов `LocalDate`, `LocalTime` и `LocalDateTime`.

Если ваш `String` объект соответствует формату даты или времени по умолчанию, то вы можете не использовать 2-ой параметр метода `parse` – шаблон. Если не соответствует или если вы не знаете форматов по умолчанию, то используйте шаблон.

## Подведение итогов

**LocalDate  
LocalTime  
LocalDateTime**

**Period  
Duration**

**DateTimeFormatter**

**Методы этих  
классов**



## Домашнее задание:

Создайте класс. Внутри класса создайте 2 шаблона с помощью класса `DateTimeFormatter`. 1-ый шаблон подгоните под вид «2016, января-01 !! 09:00», 2-ой - под вид «09:00, 03/фев/16». Создайте метод `smena`, который принимает в параметры объекты следующих классов: 2 объекта `LocalDateTime`, 1 объект `Period`, 1 объект `Duration`. До тех пор, пока 1-ый объект `LocalDateTime` меньше (раньше) 2-го проделывайте все следующие действия (даже если во время данных действий 1-ый объект `LocalDateTime` уже не меньше 2-го) раз за разом:

1. Выводите на экран «Работаем с: » + дата и время 1-го объекта с использованием 1-го шаблона;
2. Увеличивайте данный (1-ый) объект на период и выводите на экран « До: » + дата и время изменённого 1-го объекта с использованием 1-го шаблона;
3. Выводите на экран «Отдыхаем с: » + дата и время изменённого 1-го объекта с использованием 2-го шаблона;
4. Увеличивайте данный (1-ый) объект на продолжительность и выводите на экран « До: » + дата и время изменённого 1-го объекта с использованием 2-го шаблона.

Старайтесь, чтобы вывод был читабельным.

В методе `main` создайте все необходимые объекты и запустите с их помощью метод `smena`.

*The End*

**Путь к Java ОСА.**

**Эффективное пособие для начинающих.**

# **урок 29**

Краткий повтор пройденного материала

LocalDate, LocalTime, LocalDateTime

Period, Duration

DateTimeFormatter

Методы этих классов

# Проверка д/з

Создайте класс. Внутри класса создайте 2 шаблона с помощью класса DateTimeFormatter. 1-ый шаблон подгоните под вид «2016, января-01 !! 09:00», 2-ой - под вид «09:00, 03/фев/16». Создайте метод smena, который принимает в параметры объекты следующих классов: 2 объекта LocalDateTime, 1 объект Period, 1 объект Duration. До тех пор, пока 1-ый объект LocalDateTime меньше (раньше) 2-го проделывайте все следующие действия (даже если во время данных действий 1-ый объект LocalDateTime уже не меньше 2-го) раз за разом:

1. Выводите на экран «Работаем с: » + дата и время 1-го объекта с использованием 1-го шаблона;
2. Увеличивайте данный (1-ый) объект на период и выводите на экран « До: » + дата и время изменённого 1-го объекта с использованием 1-го шаблона;
3. Выводите на экран «Отдыхаем с: » + дата и время изменённого 1-го объекта с использованием 2-го шаблона;
4. Увеличивайте данный (1-ый) объект на продолжительность и выводите на экран « До: » + дата и время изменённого 1-го объекта с использованием 2-го шаблона.

Старайтесь, чтобы вывод был читабельным.

В методе main создайте все необходимые объекты и запустите с их помощью метод smena.

Ответ: Homework.Lesson28

# Lambda expressions

Самый короткий вариант написания лямбда выражения:

stud -> stud.avgGrade > 8.5

Более полный вариант написания лямбда выражения:

(Student stud) -> {return stud.avgGrade > 8.5;}

В лямбда выражении оператор стрелка разделяет параметры метода и тело метода.

В лямбда выражении справа от оператора стрелка находится тело метода, которое было бы у метода соответствующего класса, имплементировавшего наш интерфейс с единственным методом.

# Lambda expressions

Вы можете использовать смешанный вариант написания лямбда выражения: слева от оператора стрелка писать короткий вариант, справа – полный. Или наоборот.

Если вы используете полный вариант написания для части лямбда выражения справа от стрелки, то вы должны использовать слово return и знак «;»

Левая часть лямбда выражения может быть написана в краткой форме, если метод интерфейса принимает только 1 параметр. Даже если метод интерфейса принимает 1 параметр, но в лямбда выражении вы хотите писать данный параметр используя его тип данных, тогда уже вы должны писать левую часть лямбда выражения в скобках.

Если в правой части лямбда выражения вы пишите более одного statement-а, то вы должны использовать его полный вариант написания.

# Lambda expressions

```
def( () -> 5 );
```

Compile time errors:

```
def( (x) -> x.length() );
```

```
def( x -> {x.length();} );
```

```
def( (String x) -> x.length() );
```

```
def( x -> {return x.length();} );
```

```
def( (x, y) -> x.length() );
```

```
def( x, y -> x.length() );
```

```
def( (String x, String y) -> x.length() );
```

# Lambda expressions

```
method( (int x, int y) -> {int x=5; return10;} );
```

NOT OK

```
method( (int x, int y) -> {x=5; return10;} );
```

OK

```
method( (int x, int y) -> {int x2=5; return10;} );
```

OK

## interface Predicate<T>

Лямбда выражения работают с интерфейсом, в котором есть только 1 метод. Такие интерфейсы называются функциональными интерфейсами, т.е. интерфейсами, пригодными для функционального программирования.

```
public interface Predicate<T>{  
    boolean test(T t);  
}
```

Интерфейс Predicate<T> находится в java.util.function

Метод класса ArrayList removeIf использует в параметре Predicate<T>.

## Подведение итогов

Lambda  
expressions

Интерфейс  
Predicate



Метод  
removeIf

## Домашнее задание:

Создайте класс Employee, у которого будут переменные name, department и salary. Задавайте значения этим переменным при создании объекта.

В классе TestEmployee создайте метод printEmployee, который принимает в параметр объект класса Employee и выводит на экран всю информацию о данном работнике. Используя интерфейс Predicate, создайте в классе TestEmployee метод filtraciyaRabotnikov, который помимо объекта Predicate принимает в параметр ArrayList работников и выводит на экран информацию о всех работниках из ArrayList , которые подходят под определённые условия.

В методе main класса TestEmployee создайте ArrayList работников и заполните его объектами класса Employee. Затем, используя данный ArrayList и лямбда выражения, трижды вызовете метод filtraciyaRabotnikov таким образом, чтобы выведенные на экран работники подходили под следующие условия:

В первый раз: департамент работника должен быть "IT", а з/п больше 200;

Во второй раз: имя работника должно начинаться с "Е", а з/п не должна быть 450;

В третий раз: имя работника должно быть такое же, что и у его департамента.

*The End*

**Путь к Java ОСА.**

**Эффективное пособие для начинающих.**

# **урок 30**

Краткий повтор пройденного материала

Lambda expressions

Интерфейс Predicate

Метод removeIf

# Проверка д/з

Создайте класс Employee, у которого будут переменные name, department и salary.

Задавайте значения этим переменным при создании объекта.

В классе TestEmployee создайте метод printEmployee, который принимает в параметр объект класса Employee и выводит на экран всю информацию о данном работнике.

Используя интерфейс Predicate, создайте в классе TestEmployee метод filtraciyaRabotnikov, который помимо объекта Predicate принимает в параметр ArrayList работников и выводит на экран информацию о всех работниках из ArrayList , которые подходят под определённые условия.

В методе main класса TestEmployee создайте ArrayList работников и заполните его объектами класса Employee. Затем, используя данный ArrayList и лямбда выражения, трижды вызовете метод filtraciyaRabotnikov таким образом, чтобы выведенные на экран работники подходили под следующие условия:

В первый раз: департамент работника должен быть "IT", а з/п больше 200;

Во второй раз: имя работника должно начинаться с "Е", а з/п не должна быть 450;

В третий раз: имя работника должно быть такое же, что и у его департамента.

Ответ: Homework.Lesson29

# Test 1

Что будет результатом компиляции и запуска данного кода?

```
class B extends A {  
    public int a = 20;  
    public void abc() { System.out.println("child non-static"); }  
    public static void abc2() { System.out.println("child static"); }  
    public static void main(String[] args) {  
        B b = new B();  
        System.out.println(b.a);  
        System.out.println(((A) b).a);  
        b.abc();  
        ((A) b).abc();  
        b.abc2();  
        ((A) b).abc2();    }    }
```

Output:  
20  
10  
child non-static  
child non-static  
child static  
base static

```
class A {  
    public int a = 10;  
    public void abc() { System.out.println("base non-static"); }  
    public static void abc2() { System.out.println("base static"); }  
}
```

## Test 2

Что будет результатом компиляции и запуска данного кода?

```
class A {  
    public A() { str1 = abc("String1"); }  
    static String str1 = abc("String2");  
    String str3 = abc("String3");  
    { str1 = abc("String4"); }  
    static { str1 = abc("String5"); }  
    static String str2 = abc("String6");  
    String str4 = abc("String7");  
  
    public static void main(String args[]) { A a = new A(); }  
  
    static String abc(String str) {  
        System.out.println(str);  
        return str;  
    }  
}
```

Output:  
String2  
String5  
String6  
String3  
String4  
String7  
String1

## Test 3

Какие из следующих фрагментов кода являются корректными?

- A) new Object[]{new Object(), "privet", 18, new ArrayList()}; ←
- B) new Object[]{ "poka", new Object(), {} , new ArrayList()};
- C) new Object[]{new String[]{"\""}, new Object(), "ok", new ArrayList()}; ←
- D) new Object[1]{ new Object() };

## Test 4

Что будет результатом компиляции и запуска данного кода?

```
import java.util.*;  
  
class Test {  
    public static void main(String[] args) {  
        ArrayList<String> al;  
        if (al != null){  al.add("1");  }  
    }  
}
```

Compile time error

# Test 5

Что будет результатом компиляции и запуска данного кода?

```
class Test {  
    static int j = 5;  
  
    static void abc(int i) {  
        System.out.println(i);  
    }  
  
    public static void main(String[] args) {  
        abc(j++);  
        System.out.println(j);  
    }  
}
```

Output:

5  
6

# Test 6

Что будет результатом компиляции и запуска данного кода?

```
class Test {  
  
    int x =10;  
    int a=5;  
    int b=10;  
    int c = 4;  
  
    public double sredArifm() {  
        if (x > 0) {  
            double avg = 0;  
            avg = (a + b + c) / 3;  
            return avg;  
        } else {  
            avg = 0;  
            return avg;  
        }  
    }  
}
```

Compile time error

## Test 7

Что будет результатом компиляции и запуска данного кода?

```
class Test {  
  
    public static void main(String[] args) {  
        int a = 8;  
        do {  
            while (a++ < 12) {  
                a += 4;  
            }  
        } while (a < 4);  
        System.out.println(a);  
    }  
}
```

Output:  
14

# Test 8

Что будет результатом компиляции и запуска данного кода?

```
import java.io.*;
class Test {

    public static void main(String args[]) {
        int a = abc();
        System.out.println(a);
    }

    static int abc() {
        try {
            FileInputStream fis = new FileInputStream("file_kotorogo_net.txt");
        } catch (FileNotFoundException e) {
            System.out.println("file ne nayden");
        } finally {
            System.out.println("eto finally block");
        }
        System.out.println("Programma prodoljaetsya");
        return 18;
    }
}
```

Output:  
file ne nayden  
eto finally block  
Programma prodoljaetsya  
18

# Test 9

Что будет результатом компиляции и запуска данного кода?

```
import java.io.*;
class Test {

    public static void main(String args[]) {
        int a = abc();
        System.out.println(a);
    }

    static int abc() {
        try {
            FileInputStream fis = new FileInputStream("file_kotorogo_net.txt");

        } catch (FileNotFoundException e) {
            System.out.println("file ne nayden");
            return 17;
        } finally {
            System.out.println("eto finally block");
        }
        System.out.println("Programma prodoljaetsya");
        return 18;
    }
}
```

Output:  
file ne nayden  
eto finally block  
17

# Test 10

Что будет результатом компиляции и запуска данного кода?

```
import java.io.*;
class Test {
    public static void main(String args[]) {
        int a = abc();
        System.out.println(a);
    }

    static int abc() {
        try {
            FileInputStream fis = new FileInputStream("file_kotorogo_net.txt");

        } catch (FileNotFoundException e) {
            System.out.println("file ne nayden");
            return 17;
        } finally {
            System.out.println("eto finally block");
            return 16;
        }
        System.out.println("Programma prodoljaetsya");
        return 18;
    }
}
```

Compile time error

# Test 11

Что будет результатом компиляции и запуска данного кода?

```
class Book {  
  
    String bookName;  
  
    Book() {  
        Book b1 = new Book();  
        b1.bookName = "Java OCA";  
    }  
}
```

```
class TestBook {  
  
    public static void main(String args[]) {  
        Book b2 = new Book();  
        b2.bookName = "Java OCP";  
        System.out.println(b2.bookName);  
    }  
}
```

Output:  
StackOverflowError

# Test 12

Что будет результатом компиляции и запуска данного кода?

```
class Test {  
    void abc() {  
        try {  
            def();  
            return;  
        } finally {  
            System.out.println("finally");  
        }  
    }  
    void def() {  
        System.out.println("def");  
        throw new StackOverflowError();  
    }  
  
    public static void main(String args[]) {  
        Test t = new Test();  
        t.abc();  
    }  
}
```

Output:  
def  
finally  
StackOverflowError

# Test 13

Что будет результатом компиляции и запуска данного кода?

```
class Book {  
}  
interface Readable {  
}  
  
class PaperBook extends Book implements Readable {  
}  
  
class ElectronicBook extends Book {  
}  
  
class TestBook {  
  
    public static void main(String args[]) {  
        Readable r = null;  
        ElectronicBook eBook = new ElectronicBook();  
        r = (Readable) eBook;  
    }  
}
```

Output:  
ClassCastException

# Test 14

Что будет результатом компиляции и запуска данного кода?

```
class Test {  
  
    public static void main(String[] args) {  
        String[][] array  
        = {{"A", "B", "C"}, null,  
            {"D"}, new String[8]};  
        System.out.println(array);  
        System.out.println(array[1]);  
        System.out.println(array[0][2]);  
        System.out.println(array[3][0]);  
        System.out.println(array[3][5].length());  
    }  
}
```

Output:  
[[Ljava.lang.String;@2a139a55  
null  
C  
null  
NullPointerException

# Test 15

Что будет результатом компиляции и запуска данного кода?

```
class Test {  
  
    public static void main(String[] args) {  
        boolean b = false;  
        if (b) {  
            System.out.println("privet1");  
        }  
        while (b) {  
            System.out.println("poka1");  
        }  
  
        if (1 != 1) {  
            System.out.println("privet2");  
        }  
        while (1 != 1) {  
            System.out.println("poka2");  
        }  
    }  
}
```

Compile time error

# Test 16

Что будет результатом компиляции и запуска данного кода?

```
interface Jumpable{}  
class Cats implements Jumpable {}  
class Tiger extends Cats {}  
class House {}  
  
class Test {  
  
    public static void main(String[] args) {  
        Jumpable j = new Tiger();  
        Tiger t = new Tiger();  
        House h = new House();  
  
        if(j instanceof Jumpable) {System.out.println("j is Jumpable");}  
        if(t instanceof Cats) {System.out.println("t is Cat");}  
        if(h instanceof Cats) {System.out.println("h is Cat");}  
    }  
}
```

Compile time error

# Test 17

Что будет результатом компиляции и запуска данного кода?

```
class Test {  
  
    public static void main(String[] args) {  
        Integer i = new Integer(5);  
        Long l = new Long(10);  
        Double d = new Double(5);  
  
        System.out.println(i == l);  
        System.out.println(i.equals(l));  
  
        System.out.println(i == d);  
        System.out.println(i.equals(d));  
    }  
}
```

Compile time error

# Test 18

Что будет результатом компиляции и запуска данного кода?

```
class Employee {  
  
    public static void main(String... args) {  
        int id;  
        double salary;  
  
        System.out.println(id + salary);  
    }  
}
```

Compile time error

# Test 19

Что будет результатом компиляции и запуска данного кода?

```
class Employee {  
    String name = "Ivan";  
    int id = 5;  
  
    static void printInfo() {  
        System.out.println("Imya studena: " + name + "id: " + id);  
    }  
}  
  
class TestEmployee {  
    public static void main(String args[]) {  
        Employee emp = new Employee();  
        emp.printInfo();  
    }  
}
```

Compile time error

# Test 20

Что будет результатом компиляции и запуска данного кода?

```
class Tiger {  
    static int counter = 0;  
}
```

```
class TestTiger {  
  
    public static void main(String args[]) {  
        boolean flag = false;  
        while (flag = true) {  
            Tiger.counter++;  
        }  
        System.out.println(Tiger.counter);  
    }  
}
```

Infinite loop

# Test 21

Что будет результатом компиляции и запуска данного кода?

```
class Walker {  
  
    public static void main(String args[]) {  
        boolean flag = true;  
        do  
            System.out.println("walk and enjoy");  
            flag = false;  
        while (!flag);  
    }  
}
```

Compile time error

# Test 22

Что будет результатом компиляции и запуска данного кода?

```
class Test {  
  
    public static void main(String args[]) {  
        String s = "";  
        while (false) {  
            s += "hello";  
        }  
        System.out.println(s);  
    }  
}
```

Compile time error

# Test 23

Что будет результатом компиляции и запуска данного кода?

```
class Test {  
  
    public static void main(String args[]) {  
        String array1[] = new String[][]{{new String[]{"privet", "poka", "ok"}, {new String()},  
                                         {null}}[2];  
        String array2[] = {null};  
        String array3[] = null;  
  
        System.out.println(array1[0]);  
        System.out.println(array2[0]);  
        System.out.println(array3[0]);  
    }  
}
```

Output:  
null  
null  
NullPointerException

# Test 24

Какие из следующих фрагментов кода позволяют вызвать метод abc внутри метода def?

```
class MyException extends Exception { }
```

```
class Test {  
    void abc(double d) throws MyException  
    { System.out.println(d); }
```

```
    void def() {  
//ВСТАВЬТЕ КОД  
    }  
}
```

- A) try { abc(3.14); }  
 catch (Exception e) { }
- B) try { new Test().abc(3.14); }  
 catch (RuntimeException e) { }
- C) try { new Test().abc(3.14); }  
 catch (MyException e) { }
- D) try { new Test().abc(3.14); }  
 finally { }



# Test 25

Что будет результатом компиляции и запуска данного кода?

```
class Test {  
  
    public static void main(String[] args) {  
        int i = 5;  
        if (i++ <= 5) {  
            System.out.println(i);  
        }  
    }  
}
```

Output:

6

# Test 26

Что будет результатом компиляции и запуска данного кода?

```
class Test {  
  
    public static void main(String[] args) {  
        int[][] array = {{1, 2}, {3, 4}};  
        int i = 5;  
        try {  
            array[abc()][i = 0]++;  
        } catch (Exception e) {  
            System.out.println(i + " " + array[1][1]);  
        }  
    }  
  
    static int abc() throws Exception {  
        throw new Exception();  
    }  
}
```

Output:

5 4

# Test 27

Что будет результатом компиляции и запуска данного кода?

```
class A {  
    int a = 3;  
    int returnA() {  
        System.out.println("Klass A " + a);  
        return a;  
    } }
```

```
class B extends A {  
    int a = 5;  
    public int returnA() {  
        System.out.println("Klass B " + a);  
        return a;  
    }  
    public static void main(String[] args) {  
        A test1 = new B();  
        System.out.println(test1.a + " " + test1.returnA());  
        B test2 = (B) test1;  
        System.out.println(test2.a + " " + test2.returnA());  
    } }
```

Output:  
Klass B 5  
3 5  
Klass B 5  
5 5

# Test 28

Что будет результатом компиляции и запуска данного кода?

```
class Test {  
  
    static int a = 3;  
  
    public static void main(String[] args) {  
        int b = 1 + (a = 5);  
        System.out.println(b);  
    }  
}
```

Output:

6

# Test 29

Что будет результатом компиляции и запуска данного кода?

```
class A {  
    private int a = 3;  
    public void abc() { System.out.println("method abc"); }  
    public void def() { System.out.println("method def"); }  
}
```

```
class B extends A {  
    public int a = 5;  
    public void def() { System.out.println("method def"); }  
}
```

```
class C {  
    public static void main(String[] args) {  
        A test1 = new A();  
        A test2 = new B();  
        System.out.println(test1.a);  
        System.out.println(test2.a);  
    }  
}
```

Compile time error

## Test 30

Какие из конструкторов написаны корректно?

class Student{

Student(Student st) { System.out.println("constructor 1");}



Student Student( ) { System.out.println("constructor 1");}

protected final Student( ) { System.out.println("constructor 1");}

void Student( ) { System.out.println("constructor 1");}

public static void Student(String args[ ] ) { System.out.println("constructor 1");}  
}

# Test 31

Что будет результатом компиляции и запуска данного кода?

```
interface Jumpable{}  
class Cats implements Jumpable {}  
class Tiger extends Cats {}  
class House {}  
  
class Test {  
  
    public static void main(String[] args) {  
        Jumpable j = new Tiger();  
        Tiger t = new Tiger();  
        House h = new House();  
  
        if(j instanceof Cats) {System.out.println("j is Cat");}  
        if(t instanceof Jumpable) {System.out.println("t is Jumpable");}  
        if(h instanceof Jumpable) {System.out.println("h is Jumpable");}  
    }  
}
```

Output:  
j is Cat  
t is Jumpable

# Test 32

Что будет результатом компиляции и запуска данного кода?

```
class Test{
    static int x = abc(1);
    x = abc(2);
}
static{
    x = abc(3);
}
public static void main(String args[]){
    Test t = new Test();
}
static int abc(int i){
    System.out.println(i); return i;
}
```

Output:  
1  
3  
2

# Test 33

Что будет результатом компиляции и запуска данного кода?

```
class Test{  
    public static void main(String[ ] args){  
        int[] array1 = { -3, 2, 0, 8, 1};  
        int[] array2 = { 9, 0, 4, -10 };  
        System.out.println(array1 [ (array1 = array2)[1] ] );  
    }  
}
```

Output:  
-3

# Test 34

Что будет результатом компиляции и запуска данного кода?

```
class Test {  
  
    public static void main(String[] args) {  
        do {  
            int a = 1;  
            System.out.println(a++);  
        } while (a < 7);  
  
    }  
}
```

Compile time error

# Test 35

Что будет результатом компиляции и запуска данного кода?

```
class Test {  
  
    public static void abc() {  
        System.out.print("method abc");  
    }  
  
    public void def() {  
        System.out.println("method def");  
    }  
  
    public static void hig() {  
        abc();  
        def();  
    }  
  
    public static void main(String[] args) {  
        Test t = null;  
        t.hig();  
    } }
```

Compile time error

## Test 36

Даны 2 пакета со следующими классами. Что будет результатом компиляции и запуска данного кода?

```
package p1;  
public class A {  
    public static int x = 3; }
```

```
package p2;  
import p1.*;  
import static p1.A.*;  
public class B {  
    static A a1 = new A();  
    static A a2 = new A();  
    {  
        System.out.println("Vsem xoroshego dnya!");  
    }  
    public static void main(String[] args) {  
        a1.x = 5;  
        a2.x = 10;  
        System.out.println(a1.x);  
    } }
```

Output:  
10

# Test 37

Сколько объектов будут пригодными для удаления их GC на строке с комментарием данного кода?

```
class Test {  
    Byte b = 10;  
    Test abc(Test t) {  
        t = null;  
        return t;  
    }  
  
    public static void main(String[] args) {  
        Test t1 = new Test();  
        Test t2 = new Test();  
        Test t3 = t1.abc(t2);  
        t1 = null;  
        // kommentariy  
    }  
}
```

2

# Test 38

Что будет результатом компиляции и запуска данного кода?

```
class Test {  
    int a = 3;  
  
    public static Test abc(Test test1, Test test2) {  
        final Test b = test1;  
        b.a = 5;  
        return b;  
    }  
  
    public static void main(String[] args) {  
        final Test t1 = new Test();  
        Test t2 = new Test();  
        Test t3 = abc(t1, t2);  
        System.out.println(t1 == t3);  
        System.out.println(t1.equals(t3));  
        System.out.println(t1.a == t3.a);  
    }  
}
```

Output:  
true  
true  
true

# Test 39

Что будет результатом компиляции и запуска данного кода?

```
class Test {  
    static int a = 3;  
    static void abc() {  
        int a = 5;  
        def();  
    }  
  
    static void def() {  
        a += 2;  
        ghi(a);  
        System.out.println(a);  
    }  
  
    static void ghi(int a) {  
        a -= 1;  
    }  
    public static void main(String[] args) {  
        abc();  
    } }
```

Output:  
5

# Test 40

Что будет результатом компиляции и запуска данного кода?

```
class Test {  
  
    public static void main(String[] args) {  
        int a = 5;  
        String s = (a > 3) ? "1" : (a > 0) ? "2" : "3";  
        System.out.println(s);  
    }  
}
```

Output:  
1

# Test 41

Что будет результатом компиляции и запуска данного кода, если программу запускать с помощью следующей строки: java Test ok privet poka

```
class Test {  
  
    public static void main(String[] args) {  
        String[] array = new String[7];  
        for (int i = 0; i < array.length; i++) {  
            if (i < args.length)  
                array[i] = args[i];  
            System.out.println(array[i].toUpperCase());  
        }  
    }  
}
```

Output:  
ok  
privet  
poka  
NullPointerException

## Test 42

Сколько объектов были созданы и сколько объектов будут пригодными для удаления их GC на строке с комментарием данного кода?

```
class Test1 {  
    int[] array1 = {-3, 0, 3};  
}
```

```
class Test2 {  
  
    public static void main(String[] args) {  
        Test1[] array2 = new Test1[5];  
        array2[0] = new Test1();  
        Test1 t = new Test1();  
        array2[1] = t;  
        t = null;  
        array2[1] = null;  
        // kommentariy  
    }  
}
```

Созданы: 5  
Пригодны для удаления: 2

# Test 43

Что будет результатом компиляции и запуска данного кода?

```
class Test {  
  
    public static void main(String[] args) {  
        String s1 = "ok";  
        String s2 = s1;  
        s1 += "!!!";  
        System.out.println(s1 + ", " + s2 + ", " + (s1 == s2) + ", " + s1.equals(s2));  
        StringBuilder sb1 = new StringBuilder("ok");  
        StringBuilder sb2 = sb1;  
        sb1.append("!!!");  
        System.out.println(sb1 + ", " + sb2 + ", " + (sb1 == sb2) + ", " + sb1.equals(sb2));  
    }  
}
```

Output:  
ok!!!, ok, false, false  
ok!!!, ok!!!, true, true

# Test 44

Что будет результатом компиляции и запуска данного кода?

```
class Test {  
  
    public static void main(String[] args) {  
        while (true) {  
            System.out.println("privet");  
        }  
        System.out.println("poka");  
    }  
}
```

Compile time error

# Test 45

Что будет результатом компиляции и запуска данного кода?

```
class Test {  
    static String s = "";  
    public static void main(String[] args) {  
        try {  
            throw new RuntimeException();  
        } catch (Exception e1) {  
            try {  
                try {  
                    throw new Exception();  
                } catch (Exception e2) {  
                    s += "1";  
                }  
                throw new Exception();  
            } catch (Exception e3) {  
                s += "2";  
            } finally {  
                s += "3";  
            }  
        } finally {  
            s += "4";  
        }  
        System.out.println(s);    } }
```

Output:  
1234

# Test 46

Что будет результатом компиляции и запуска данного кода?

```
class Test {  
  
    static String s = "";  
  
    public static void main(String[] args) {  
        int[] array = {1, 2, 5, 8};  
        for (int a : array){  
            for (int j = 0; j < 4; j++) {  
                if (a > 1.5 && a < 6) {  
                    continue;  
                }  
                System.out.println(a);  
                if (j == 1)  
                    break;  
                continue;  
            }  
            continue;  
        }  
    }  
}
```

Output:  
1  
1  
8  
8

ссылка:

<http://enthuware.com/>

*The End*