Java for WEB

Lesson 1

Организация занятий

- Коммуникация (рассылка, Skype, форум)
- Репозиторий для домашних заданий (GitHub)
- IDE
- JDK

Цель курса

Научить вас создавать Java-приложения

Цель первого занятия

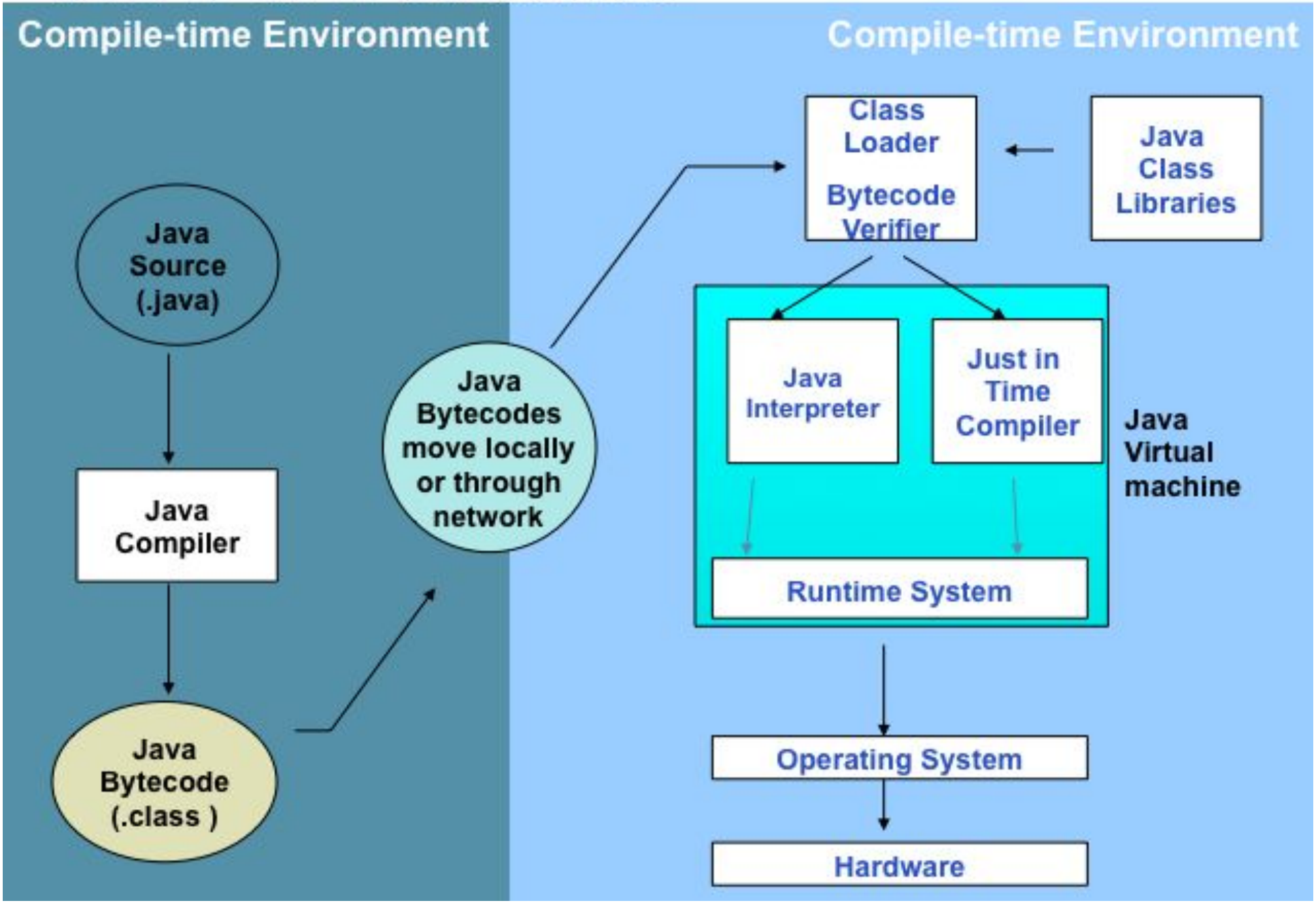
Дать представление что такое Java

Дать самые базовые знания для написания приложений на Java

Особенности языка Java

- Простой
- Объектно-оринетированный
- Кроссплатформенный
- Интерпретируемый
- Распределенный
- Надежный
- Безопасный
- Многопоточный
- Динамичный
- Высокопроизводительный

How it works…!


Compile-time Environment
Compile-time Environment
Java Source (.java)
Java Compiler
Java Bytecode (.class)
Java Bytecodes move locally or through network
Class Loader
Bytecode Verifier
Java Class Libraries
Java Interpreter
Just in Time Compiler
Java Virtual machine
Runtime System
Operating System
Hardware

Hello World

```
public class HelloWorld {
    public static void main(String argv[] ) {
      System.out.println("Hello World!!!");
    }
  }
```

Анализ программы компилятором

- пробелы (white spaces);
- комментарии (comments);
- основные лексемы (tokens).

```
double a = 1, b = 1, c = 6;
double D = b * b - 4 * a * c;

if (D >= 0) {
   double x1 = (-b + Math.sqrt (D)) / (2 * a);
   double x2 = (-b - Math.sqrt (D)) / (2 * a);
}
```

Можно записать и в таком виде:

```
double a=1,b=1,c=6;double D=b*b-4*a*c;if(D>=0)
{double x1=(-b+Math.sqrt(D))/(2*a);double
x2=(-b-Math.sqrt(D))/(2*a);}
```

Java Syntax

- Синтаксис унаследован от C++. Упрощен. Автоматизирована уборка мусора.
- Сходство с C++: терминология, типы, синтаксис.
- Отличия от C++: управление памятью, error & exception handling, full OO.

Важные моменты:

- Case sensitive.
- Каждый statement заканчивается точкой с запятой - ‘;’.
- Блоки помещаются в фигурные скобки - ‘{‘ и ‘}’ .
- Пробел, табуляция, и перевод строки используются

для форматирования кода, удобства чтения и понимания.

- Комментарии (// one-line, /* Multiple line */, /** JavaDoc */)

Идентификаторы

Это имена, которые даются различным элементам языка для упрощения доступа к ним.

Имена имеют пакеты, классы, интерфейсы, поля, методы, аргументы и локальные переменные.

Правило: любая комбинация латинских букв (uppercase, lowercase), чисел (но не начинается с числа) и `_', `$'.

Character, c, D, x1, x2, Math, sqrt, x,

PI, condition, getWidth, getHeight,

lang, stack, Stack, STACK_SIZE, wav2snd, _snd, $snd

Ключевые слова

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

Литералы

Позволяют задать в программе значения для числовых, символьных и строковых выражений, а также null- литералов.

- целочисленный (integer);
- дробный (floating-point);
- булев (boolean);
- символьный (character);
- строковый (string);
- null- литерал (null-literal).

Переменные

Переменные используются в программе для хранения данных.
Любая переменная имеет три базовые характеристики:

- имя;
- тип;
- значение.

```
int a;
int b = 0, c = 3+2;
int d = b+c;
int e = a = 5;
```

Примитивные типы

Целочисленные типы – это byte, short, int, long, также к ним относят и char.

Дробные типы – это float и double .

Булев тип представлен всего одним типом boolean

Примитивные и ссылочные типы данных

Что произойдет со второй переменной?

```
int a=5;   // объявляем первую переменную и
// инициализируем ее
int b=a;   // объявляем вторую переменную и
// приравниваем ее к первой
a=3;       // меняем значение первой
print(b);  // проверяем значение второй
```

```
Point p1 = new Point(3,5);
Point p2=p1;
p1.x=7;
print(p2.x);
```

```
Point p1 = new Point(3,5);
Point p2=p1;
p1 = new Point(7,9);
print(p2.x);
```

Операторы

Java операторы (по приоритетам):

- () [] . ; ,
- ++ -- ~ !
- * / %
- + -
- >> >>> << <<<
- > >= < <=
- == !=
- &
- ^
- |
- ||
- ?: (тернарный)
- =
- No operator overloading like in C++!

Управляющие операторы

if else

```
 if (username == null) // Если имя пользователя равно null, то
     username = "John Doe"; // определить его

if (i == j) {
    if (j == k)
        System.out.println("i равно k");
} else {
    System.out.println("i не равно j");
}

 if (n == 1) {
  // Выполнить блок кода No1
} else if (n == 2) {
  // Выполнить блок кода No2
} else if (n == 3) {
  // Выполнить блок кода No3
} else {
  // Если ни одно условие не выполнилось, выполнить блок No4
}
```

switch

```
switch(n) {
    case 1:             // Начать здесь, если n == 1
       // Выполнить блок кода No1
       break;          // Остановиться здесь
    case 2:             // Начать здесь, если  n == 2
       // Выполнить блок кода No2
       break;           // Остановиться здесь
    default:            // Если ни одно условие не выполнилось,
       // Выполнить блок кода No4
       break;          // Остановиться здесь
}
```

```
boolean parseYesOrNoResponse(char response) {
switch(response) {
    case 'y':
    case 'Y': return true;
    case 'n':
    case 'N': return false;
    default:
       throw new IllegalArgumentException("Ответ должен быть Y или N");
}
```

Циклы

while

```
int count = 0;
while (count < 10) {
    System.out.println(count);
    count++;
}
```

do while

```
int count = 0;
do {
    System.out.println(count);
    count++;
} while(count < 10);
```

for

```
int count;
for(count = 0 ; count < 10 ; count++)
    System.out.println(count);
```

```
for(int count = 0 ; count < 10 ; count++)
   System.out.println(count);
```

break

```
for(int i = 0; i < data.length; i++) {
    if(data[i] == target) {
        index = i;
        break;
    }
}
```

continue

```
for(int i = 0; i < data.length; i++) {
    if(data[i] == -1)
        continue;
    square(data[i]);
}
```

return

```
double square(double x) {
    return x * x;
}
```

Массивы

В отличие от обычных переменных, которые хранят только одно значение, массивы (arrays) используются для хранения целого набора значений.

```
int array[]=new int[5];
for (int i=0; i<5; i++) {
  array[i]=i*i;
 }
 for (int j=0; j<5; j++) {
    System.out.println(j+"*"+j+"="+array[j]);
 }
```

```
Object arr[] = new Object[3];
 arr[0]=new Object();
 arr[1]=null;
 arr[2]=arr;   // Элемент ссылается
               // на весь массив!
```

```
int i[]={1, 3, 5};
 int j[]={};   // эквивалентно new int[0]
```

```
Point p=new Point(1,3);
 Point arr[]={p, new Point(2,2), null, p};
 // или
 String sarr[]={"aaa", "bbb", "cde"+"xyz"};
```

```
int pithagor_table[][]=new int[5][5];
for (int i=0; i<5; i++) {
   for (int j=0; j<5; j++) {
      pithagor_table[i][j]=i*j;
      System.out.print(pithagor_table[i][j] + "\t");
   }
   System.out.println();
}
```

Методы

Сигнатура

- Имя метода
- Количество, порядок, тип и имена параметров
- Тип возвращаемого значения
- Проверяемые исключения, которые может генерировать метод
- Различные модификаторы метода, предоставляющие дополнительную информацию

modifiers type name (paramlist) [**throws** exceptions]

```
public static void main(String[] args) { ... }
public final synchronized int indexOf(Object element, int startIndex) { ... }
    double distanceFromOrigin() { ... }
static double squareRoot(double x) throws IllegalArgumentException { ... }
protected abstract String readText(File f, String encoding) throws                FileNotFoundException,
UnsupportedEncodingException;
```

Наследование, Полиморфизм, Инкапсуляция

Наследование означает возможность заимствования одним классом у другого класса его полей и методов. В дальнейшем будут использоваться два термина: *суперкласс* и *суб- класс*.

С концепцией наследования тесно связана концепция *полиморфизма* типа. Поли-морфизм типа, вообще говоря, означает, что объект субкласса может быть применён в любом контексте, где предусмотрено использование объектов суперкласса.

Инкапсуляция это сокрытие данных и реализации.

Классы и объекты

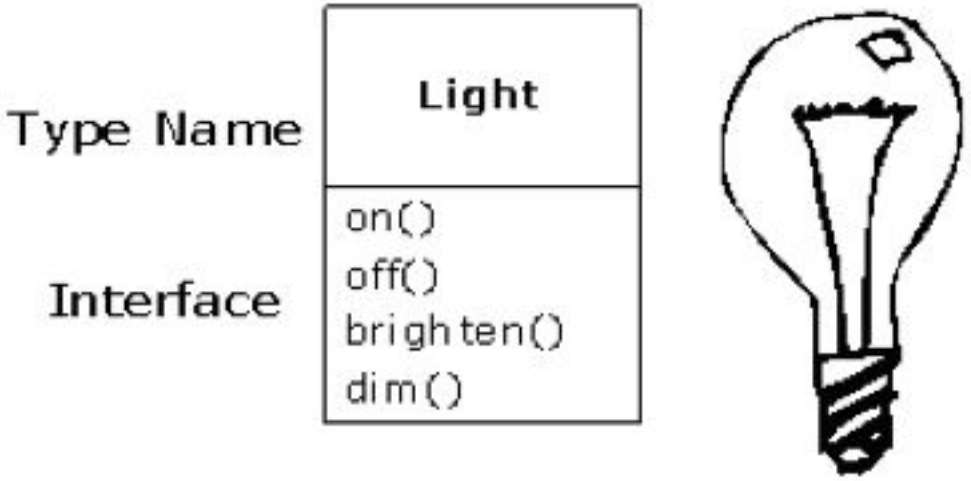
Любую сущность можно рассматривать как объект.

Программа представляет собой набор объектов, обменивающихся сообщениями, при помощи которых передаются запросы на выполнение методов.

Каждый объект обладает памятью, скрытой от остальных объектов.

Каждый объект имеет тип.

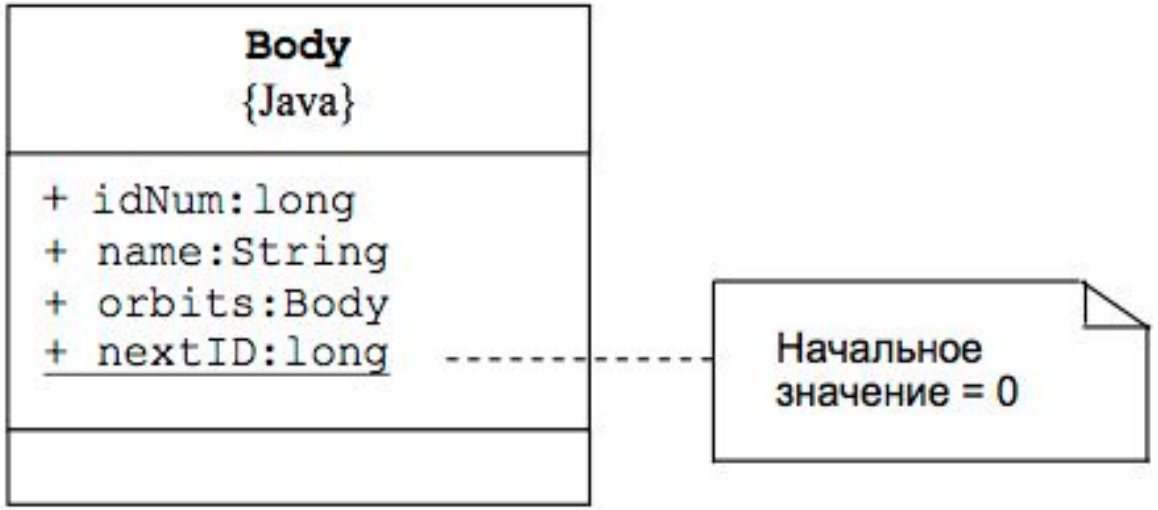
Все объекты одного типа могут получать одинаковые сообщения.




```
Light lt = new Light();
lt.on();
```

Простой класс

Объявление класса `Body`, предназначенного для хранения данных о небесных телах.




```
class Body {
   public long idNum;
   public String name;
   public Body orbits;

   public static long nextID = 0;
}
```

Модификаторы объявления класса

public. Модификатор *public* объявляет класс *общедоступным*.

Это означает, что в любом коде разрешается объявлять ссылки на объекты класса и обращаться к его членам.

abstract. Модификатор *abstract* объявляет класс *абстрактным*. Главное отличительное свойство абстрактного класса, – это невозможность его использования для генерации объектов.

final. Класс,определённый как *final* не допускает *наследования*.

strict floating point . Присутствие в объявлении класса модификатора *strictfp* означает, что операции с плавающей точкой, предусмотренные методами класса, должны выполняться *единообразно* всеми виртуальными машинами Java.

Модификаторы объявления поля

`private, package, protected, public` (модификаторы доступа);

`static` (модификатор статическог ополя);

`final`(модификатор изменения поля);

`transient` (модификатор *сериализации* или преобразования объекта в байтовый поток, передаваемый по сети);

`volatile` (значение поля с этим модификатором может быть изменено параллельно выполняющимися потоками).

Статические елементы

```
class Cat {
    public static int totalCount;
    public String name;
}
```

```
Cat.totalCount++;
// рождение еще одного кота


Cat markiz = new Cat();
markiz.totalCount = 100;


    Cat.totalCount=100;


Cat markiz = null;
markiz.totalCount+=10;
```

```
class Cat {
    private static int totalCount;

    public static int getTotalCount() {
      return totalCount;
    }
}
```

```
Cat markiz = null;
markiz.getTotalCount();// два эквивалентных
Cat.getTotalCount();
```

Конструкторы

Инициализация при помощи конструкторов завершается до того момента, когда оператор `new` формирует ссылку на вновь созданный объект.

```
class Body {
   public long idNum;
   public String name = "Астероид";
   public Body orbits = null;
   private static long nextID = 0;

   public Body() {
      idNum = nextID++;
   }

   Body(String bodyName, Body orbitsAround) {
      this();
      name = bodyName;
      orbits = orbitsAround;
    }
}
```

```
Body sun = new Body();     // idNum = 0
sun.name = "Солнце";
Body earth = new Body();   // idNum = 1
earth.name = "Земля";
earth.orbits = sun;

Body mars = new Body("Марс", null);
Body fobos = new Body("Фобос", mars);
```

Ключевые слова this и super

```
class Test {
   public Object getThis() {
     return this;
     // Проверим, куда указывает эта ссылка
   }
   public static void main(String s[]) {
     Test t = new Test();
     System.out.println(t.getThis()==t);
     // Сравнение
   }
 }
```

```
class Human {
   private String name;

   public void setName(String name) {
     this.name=name;
   }
}
```

```
class Human {
   public static void register(Human h) {
     System.out.println(h.name+
       " is registered.");
   }

   private String name;
   public Human (String s) {
     name = s;
     register(this); // саморегистрация
   }

   public static void main(String s[]) {
     new Human("John");
   }
}
```

Ключевые слова this и super

```
class Parent {
    public int getValue() {
      return 5;
    }
}

class Child extends Parent {

    // переопределение метода
    public int getValue() {
      // обращение к методу родителя
      return super.getValue()+1;
    }

    public static void main(String s[]) {
      Child c = new Child();
      System.out.println(c.getValue());
    }
}
```

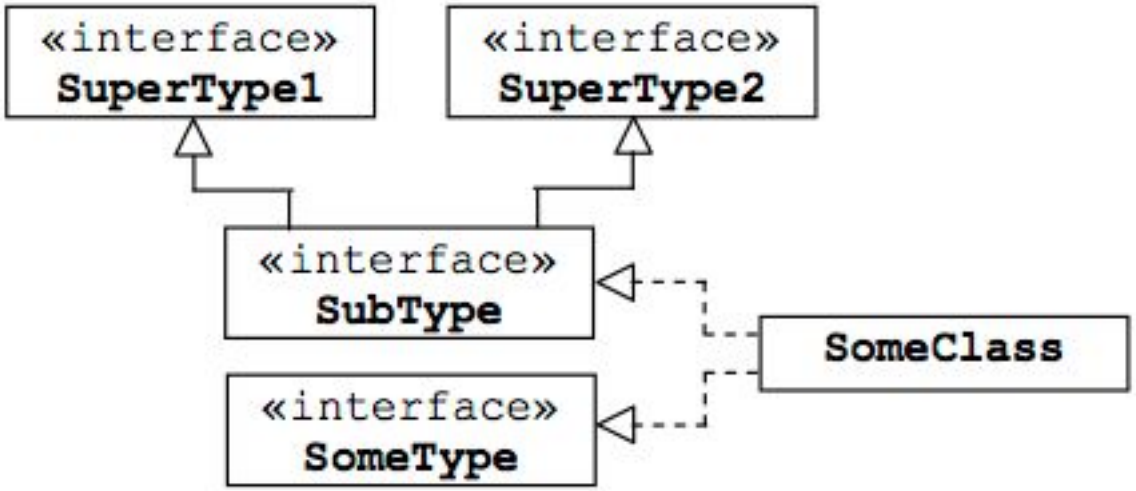
Ключевое слово abstract

```
// Базовая арифметическая операция
  abstract class Operation {
    public abstract int calculate(int a, int b);
  }
  // Сложение
  class Addition extends Operation {
    public int calculate(int a, int b) {
      return a+b;
    }
  }

  // Вычитание
  class Subtraction extends Operation {
     public int calculate(int a, int b) {
       return a-b;
     }
  }

  class Test {
     public static void main(String s[]) {
       Operation o1 = new Addition();
       Operation o2 = new Subtraction();

       o1.calculate(2, 3);
       o2.calculate(3, 5);
     }
  }
```

```
abstract class Test {
    public abstract int getX();
    public abstract int getY();
    public double getLength() {
      return Math.sqrt(getX()*getX()+
                       getY()*getY());
    }
 }
```

Интерфейсы

Интерфейс позволяет описать тип в абстрактной форме – в виде набора заголовков методов и объявлений именованных констант. Интерфейс не содержит блоков реализации и, поэтому, невозможно создавать экземпляры интерфейса.



Интерфейсы

`Cloneable`.

Объекты этого типа поддерживают операцию клонирования.

`Comparable`.

Объекты этого типа допускают упорядочение и поэтому могут сравниваться и, следовательно, сортироваться.

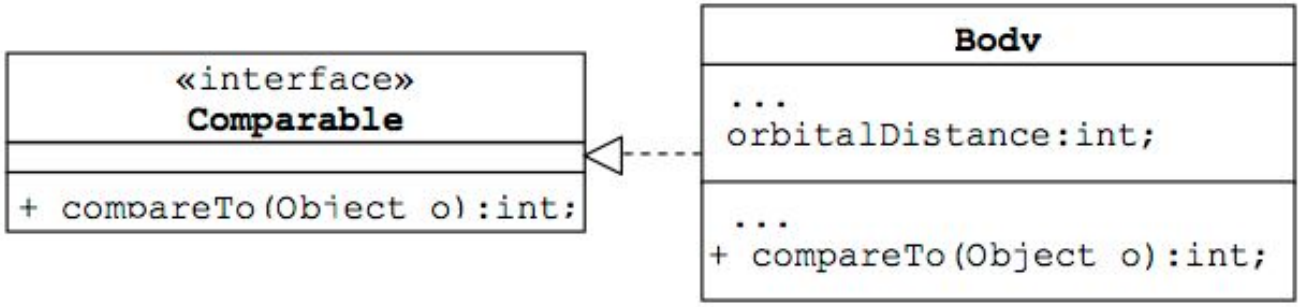
`Runnable`.

Объекты этого типа содержат код, способный выполняться в виде независимого потока вычислений.

`Serializable`.

Объекты этого типа могут быть преобразованы в последовательность байтов с целью сохранения на носителях или передачи в среду другой виртуальной машины, а затем, при необходимости, восстановлены в исходном виде.

Интерфейсы

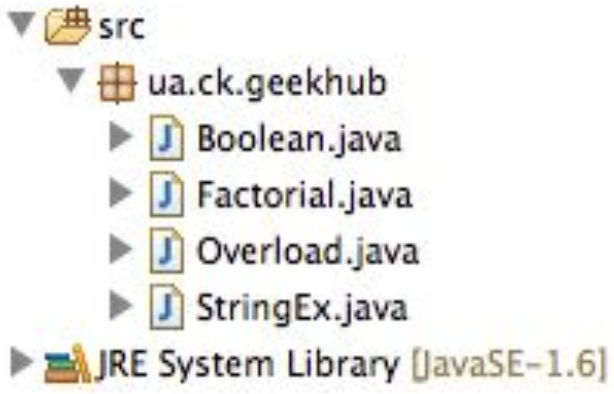



```
class Body implements Comparable {
   // Объявления полей и конструкторов опущены
   int orbitalDistance        // Инициализируется
                              // конструктором
   public int compareTo(Object o){
      Body other = (Body)o;
      if (o.orbits == other.orbits) // Если тела вращаются
                                    // вокруг одного центра
         return o.orbitalDistance - other.orbitalDistance;
      else
         throw new IllegalArgumentExeption("Неверное значение orbits");
   }
}
```

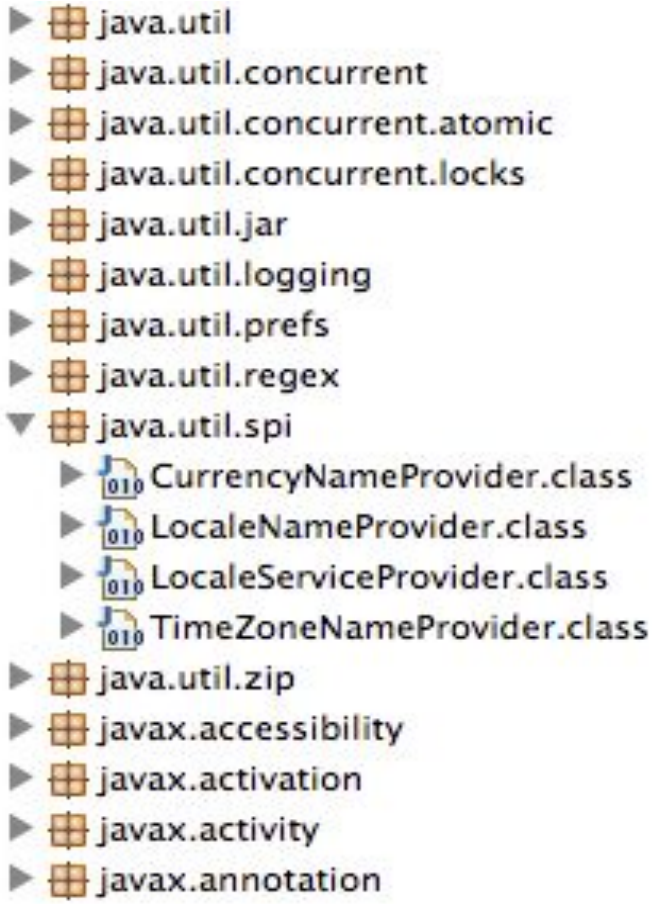
Пакеты и пространство имен

Пакет (package) представляет собой именованную совокупность классов (и, возможно, подпакетов).

Java содержит пакеты, имена которых начинаются на java,javax и org.omg.



Одной из важных функций пакетов является разделение пространства имен Java и предотвращение конфликта имен между классами.



Импорт классов и пакетов

- импорт одного типа

```
import java.net.URL;
```

- импотр пакета

```
import java.awt.*;
```

// пример вызовет ошибку компиляции

```
import java.awt.image;
```

Правильно:

```
import java.awt.*;
```

Допускается одновременно импортировать пакет и какой-нибудь тип из него:

```
import java.awt.*;
import java.awt.Point;
```

// пример вызовет ошибку компиляции

```
package ua.ck.geekhub;
import java.awt.Point;
class Point {
}
```

Прочтите код

```
package ua.ck.geekhub;

/**
 * Эта программа подсчитывает факториал числа.
 */
public class Factorial { // Определить класс
    // Программа начинается здесь
    public static void main(String[] args) {
        // Получить входные данные от пользователя
        int input = Integer.parseInt(args[0]);
        double result = factorial(input);   // Посчитать факториал
        System.out.println(result);         // Распечатать результат
    }                                   // метод main() заканчивается здесь
    // Этот метод подсчитывает x!
    public static double factorial(int x) {
        assert x == 2 : "Error";
        if (x < 0)              // Проверить правильность входных данных
            return 0.0;         // Если данные неправильные, отобразить 0
        double fact = 1.0;      // Начинать с исходной величины
        while (x > 1) {         // цикл выполняется, пока x больше 1
            fact = fact * x;    // умножать на x каждый раз
            x = x - 1;          // а затем уменьшить x
        }                       // Вернуться к началу цикла
        return fact;            // возвратить результат
    }                           // метод factorial() заканчивается здесь
}                               // класс заканчивается здесь
```

Задание

- Завести всем GitHub-аккаунты и дать доступ преподавателям. Выкладывать туда все домашние задания.
- - Написание приложения FibonacciSequence - вывести первые n чисел последовательности Фибоначи (n передается как параметр).
- - Написание приложения FractionSequence - вывести первые n чисел по формуле (1/n) (n передается как параметр).
- - Написание приложения CharSequence - вывести символы с кодами от m до n (m и n передаются как параметры).
- - Написание приложения Sum - вывести сумму чисел m и n (m и n передаются как параметры; могут быть дробными или целыми заданными в 8-, 10- или 16-ричном формате).