Java
Standart Edition

Trainer: Maksym Konovaliuk

GEORGIA'S INNOVATION AND TECHNOLOGY AGENCY

The expert of IT education

Skillup.com.ua

History

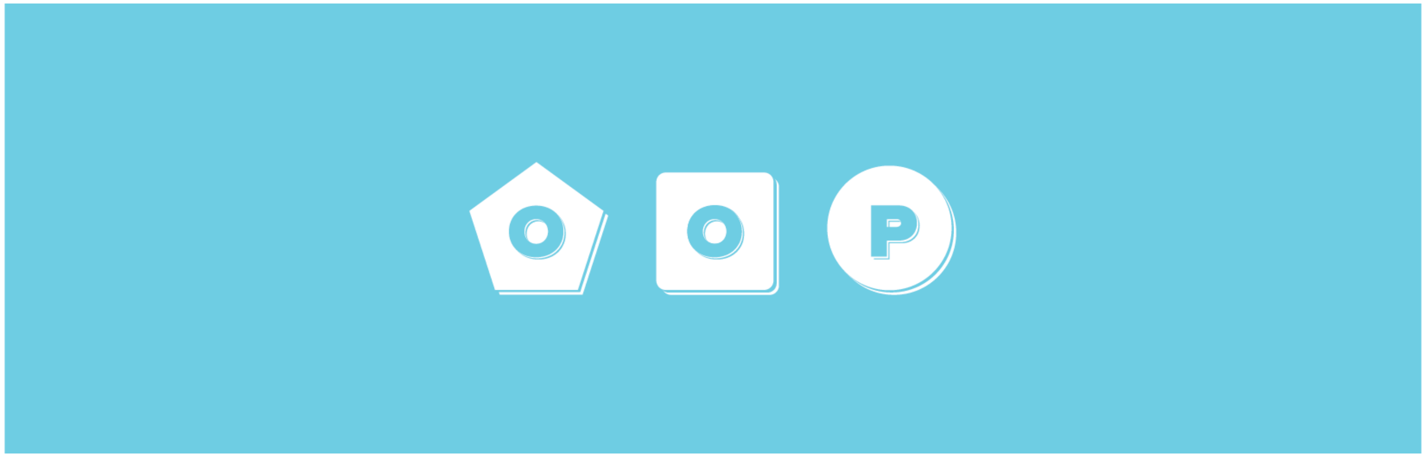
Sun Microsystems released the first public implementation as Java 1.0 in 1995.

In 1997, Sun Microsystems approached the ISO/IEC JTC 1 standards body and later the Ecma International to formalize Java, but it soon withdrew from the process.

On November 13, 2006, Sun released much of Java as free and open-source software, (FOSS), under the terms of the GNU General Public License (GPL).

Following Oracle Corporation's acquisition of Sun Microsystems in 2009–2010, Oracle has described itself as the "steward of Java technology with a relentless commitment to fostering a community of participation and transparency".[24]

Object-oriented programming



Java is an object oriented language, so before considering Java let's look to basic elements of object oriented programming. Nowadays, documents of various spheres are represented as electronic documents and this stimulates not only the development of hardware, but also the evolution of the tools for developing software. The transition from procedural programming to object oriented programming (OOP) was a significant step in the programming theory that reduced the time and resources of software development. Idea is to create their own data types which are named classes.

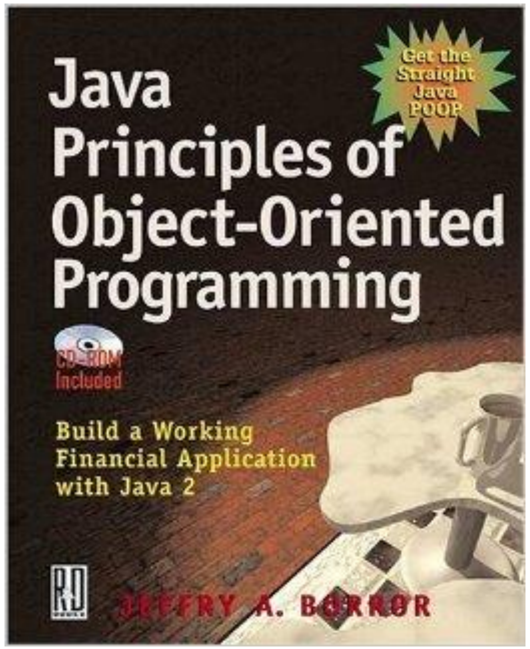
Object-oriented programming

OOP – is a programming methodology which based upon the representation of the software in the form of a set of objects, each of which is an instance of a particular class.

Object – is real named entities with the properties and behavior. Object is the instance of class. Any object is always apply to a particular class. Class consists of describing data and operations with them.

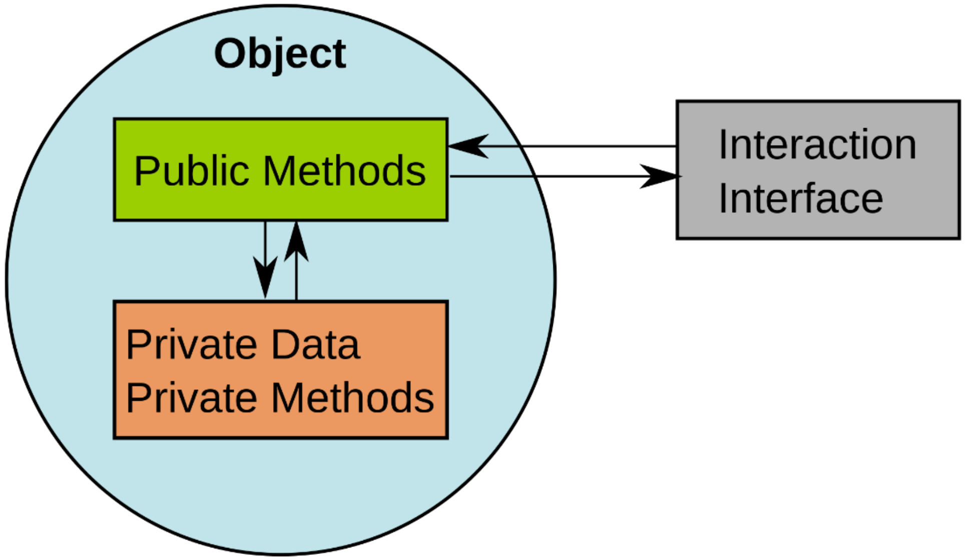
Object-oriented programming

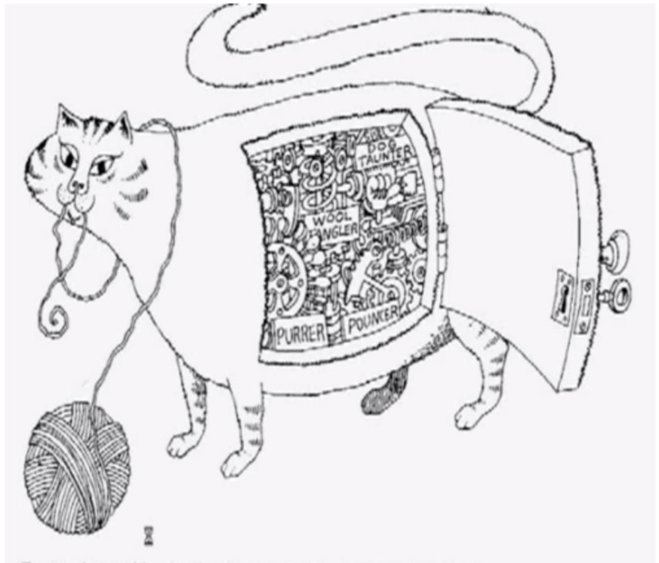
Object oriented programming based on next principles:
- encapsulation;
- inheritance;
- polymorphism (in particular “late binding”).


Java
Principles of Object-Oriented Programming
Get the Straight Java POOP
CD-ROM Included
Build a Working Financial Application with Java 2
JEFFRY A. BORROR

Object-oriented programming

Encapsulation – is a principle that unites data and code which operates with this data and also defends data from direct external access. In practical programming it means that there is no access to fields from other classes (fields has private access from other classes) besides the methods of this class (methods has public access from other classes).





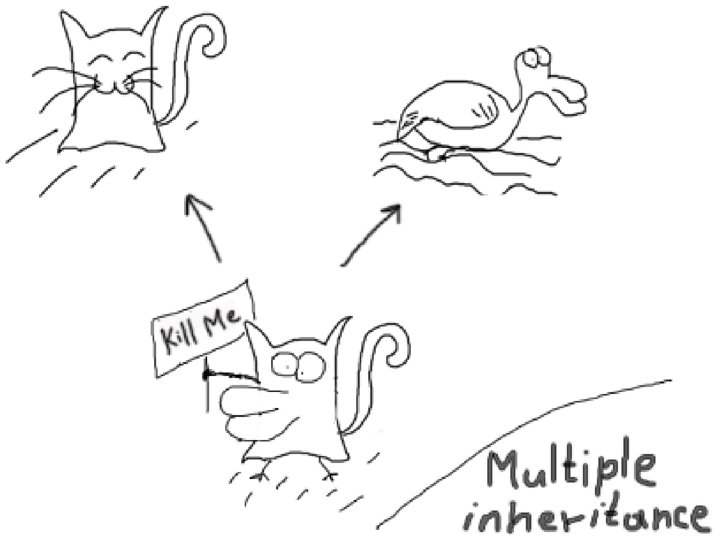
Encapsulation hides the details of the implementation of an object.

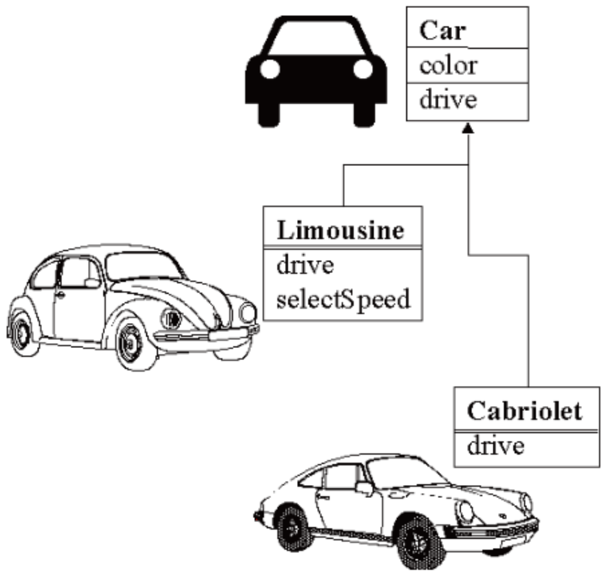
Object-oriented programming

Inheritance – is the process by which one class adopts the behavior of another class and adds a specific behavior for this class. It means that derived class gets properties and methods from inherited class and adds their own fields and/or methods.

In general there is two types of inheritance:
- single inheritance;
- multiple inheritance (denied in Java).

Single inheritance means that class can have only one superclass. Multiple inheritance means that class have multiple superclasses.





Object-oriented programming

Polymorphism – is the process that based on the inheritance of classes that have override methods and ability of link to invoke dynamically the version of the overridden method depending on the type of the object. The idea is: one interface and several methods.

Meow!
Woof!
Quack!
Animal
Animal
Animal
Polymorphism
speak();
by Sinipull for codecall.net

Object-oriented programming

Instance of class can be used anywhere, where using instance of superclass.

Base Java

Getting started

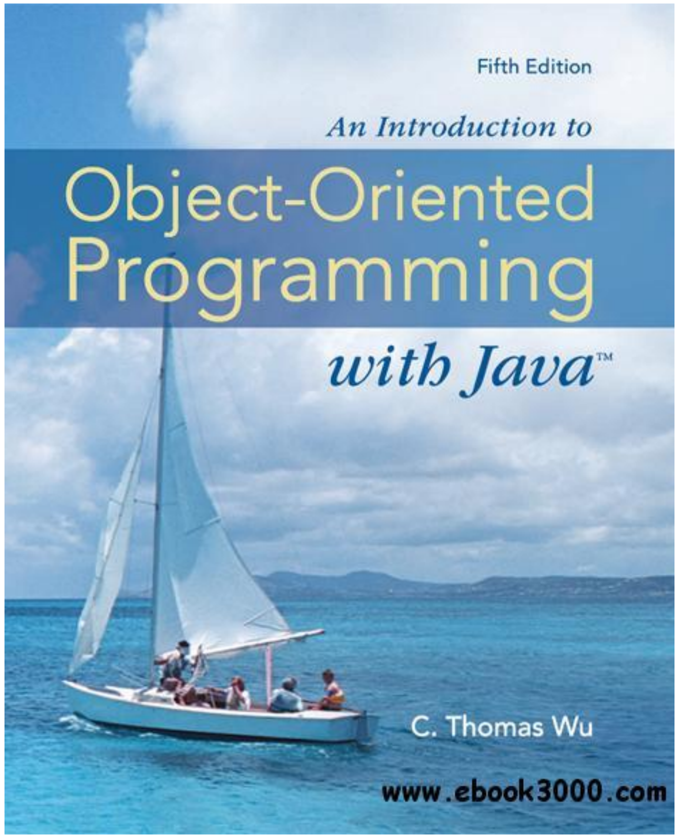
Java overview

Java programming language characterize as:

- object oriented;
- simple;
- secure;
- architecture independent and portable;
- multithreaded;
- dynamic;
- distributed.

Java overview

Object oriented – Java creates as object oriented programming language.



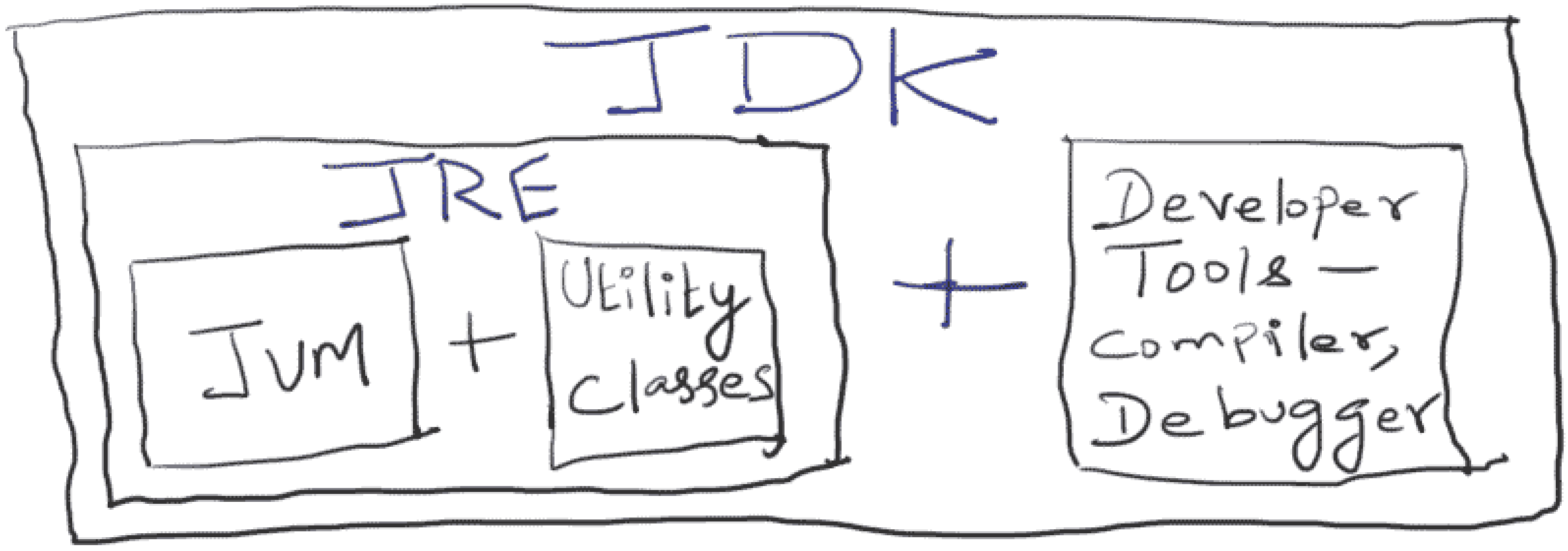
Java overview

Simple - syntax of Java is simpler than C++. From Java were excluded all complex means of C++ programming language, which makes use of Java simple. Java has no pointers, unions, header files, multiple inheritance and so on.

Java overview

Secure – Java Virtual Machine (JVM) controls the execution of Java application, which makes execution secure and prevents situation as intentional stack overflow.

Java overview



Architecture independent and portable – Java application executes in any environment and any operation system that have JVM, which makes Java portable.

Java overview

Multithreaded - it is possible in Java to create application that execute multiple operations at the same time.

Java overview

Dynamic - Java easy adopts to unsustainable environment by adding new functionality to libraries.

Distributed – Java has tools for network execution. Java components have possibility of running in different places in the world and execute as single application.

Base characteristics

All instances of classes are located in heap memory and available by links that located in stack.

In Java the header of class and his implementation are located in one file. Java doesn't support structure, overloading operators, multiple inheritance and typedef unlike C++ .

Mechanism of multiple inheritance in C++ are replaced in Java by using interfaces.

Constructors exist in Java, but there are no destructors. Java doesn't support operator "goto" and “const”, however they exist in syntax.

Base characteristics

Here is the list of register words and keywords in Java:

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

Apart from keywords represented in table Java has three literals: **null**, **true**, **false**:

JDK

Developers must have Java Development Kit (JDK) to create Java application. JDK consists from: compiler, standard library, samples, documentation, different utilities and Java Runtime Environment (JRE).

Here is the structure of JDK directories:
bin – compiler and other execution tools;
demo – directory with samples;
docs – documentation in html format;
include – files for developing platform-oriented methods;
jre\lib – Java libraries for end user and configuration files;
lib – Java libraries;

Simple application

Java class must be defined in a text file with a “.java” extension. In addition, if the class is declared **public**, then the name of the file must match the name of the class. A “.java” file can only contain at most one top-level **public** class.

```
package first;

public class First {
        public static void main(String[] args) {
            System.out.println("Hello from Kyiv!");
        }
  }
```

Compiling the First.java source file creates a bytecode file named First.class.

Simple application

Below, the simple “Hello world” application will be done with the use of a class method, implemented in the simple object-oriented application.

```
package first;

public class First {
	public static void main(String[] args) {
   //instantiating new instance objectName
		ClassName objectName = new ClassName();
   //method call
		objectName.methodName();
	}
}
// simple class: ClassName
class ClassName {
   public void methodName() {// method implementation
   //print string “Hello”
		System.out.println("Hello from Kyiv!");
	}
}
```

Base code convention

Name of the class, the field or method must use whole words, completely exclude contraction. Abbreviations used only when they are obvious. Class name is always spelled with a capital letter: **Class**.

If the class name consists of two or more words, then the second and the following words are written together with the previous and begin with a capital letter: **ClassName**. The method name is always spelled with a lowercase letter: **method**().

If the name of the method consists of two or more words, then the second and following word are written together with the previous and begin with a capital letter: **methodName**().

Base code convention

Name of class field, local variables and parameters of the method always are spelled with a small letter: **field**. If the name of the class fields, local variables and parameters of the method consist of two or more words, then the second and the following words are written together with the previous and begin with a capital letter: **fieldName**. Constants and enumerations are written in upper case: **MAX_VALUE**.

All package names are written in small letters. Abbreviations are allowed only if the package name is too long: 10 or more characters. Use of numbers and other symbols are undesirable.

Running first application

To run the application you need to download the latest version of JDK for the platform you will use (www.oracle.com/technetwork/java/javase/downloads/index.html). After installation, check that your system **PATH** added the directory that contains the **java.exe** and **javac.exe**. You can do it like this (in Windows):
Start -> Settings -> Control Panel -> System -> Advanced -> Environment Variables
Find a list of the **PATH** variable and add to it the path to the directory containing the files **java.exe** and **javac.exe**. For example - C:\Program Files\Java\jdk_1.8\bin. If you do not already have the **PATH**, create it. Also create a variable named **JAVA_HOME**, its value will be the path to the directory where you installed JDK (for example, C:\Program Files\Java\jdk_1.8). Check if everything is OK, you can: Start a command prompt and enter the command java. You should get the following result:

```
Usage: java [-options] class [args...]
          (to execute a class)
or  java [-options] -jar jarfile [args...]
          (to execute a jar file)
where options include:
   -client       to select the "client" VM
   -server       to select the "server" VM
   -hotspot      is a synonym for the "client" VM  [deprecated]
                    The default VM is client.
```

Running first application

This means that the virtual machine was found, but you call it with incorrect arguments. If you receive a message «**'java' is not recognized as an internal or an external command, operable program or batch file**», it means that you did something wrong.

Compile the file with the command **javac first/First.java**, in same directory. Once the compilation is successful you can find in the directory next files **First.class** and **ClassName.class**. Now you can run your application by typing a string java **first.First** being out of the first directory.

If you have received as a result the error message «**ClassNotFoundException**», than you need to install another system variable - **CLASSPATH**. This can be done in the same way as described above. Value of the variable should be a list of ways in which Java classes will look for at startup. For example: **.;C:\Java\MyProject\classes**. Point is needed to include in your search the current directory.

You can do without the installation of the system variables. Suppose we installed the JDK in the directory **D:\JDK** and created a file **D:\First.java**. Then compilation and running can be as follows:

```
D:\JDK\bin\javac.exe D:\projects\First.java
D:\JDK\bin\java.exe -classpath D:\First
```

Question?