Java language

Version 1.6

References:

Java in a Nutshell - 6th Edition; David Flanagan; O'REILLY.

Presentation of the Java language

- similar to C and C++, but no pointers

- strongly object oriented
  → well suited for the manipulation of complex sets of data

  class = data + functions applied over these data ("methods")
  object = instance of a class

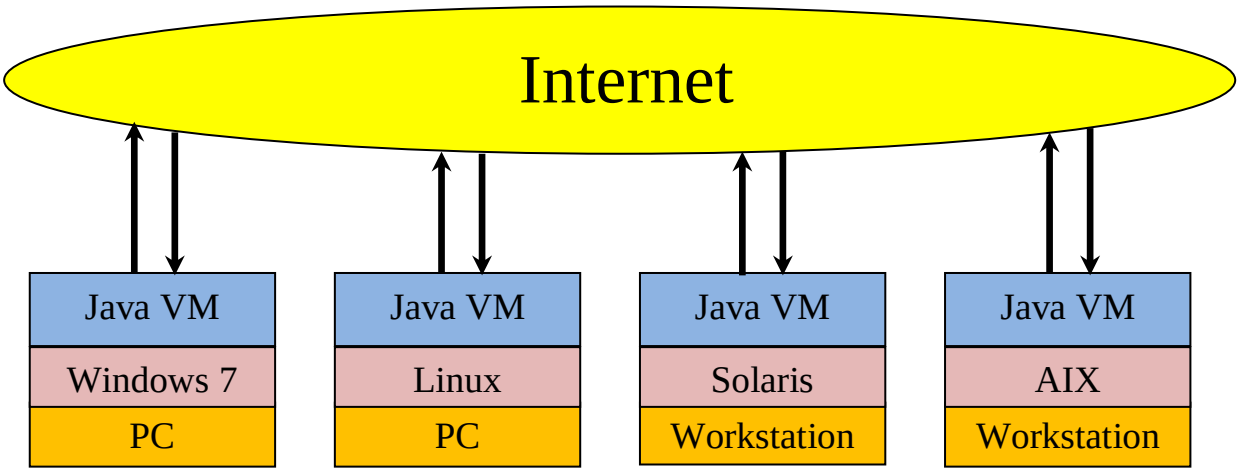
- robust: strongly typed → extensive checking during compilation
  no hazardous operations on memory

- the executable code obtained by compilation of the source code is interpreted by the Java Virtual Machine (JVM) and not directly by such or such processor
  → execution strictly independent from the platform (computer and OS)
  → "universal", standard language

  but interpretation slows down execution → "Just In Time" compiler

- dynamic linking: definitions of classes are loaded by the JVM during runtime, only if and when execution needs it

- platform independent + dynamic linking → distributed execution on several different computers over the Internet:



- The language itself is relatively simple but it comes with many libraries (*packages*) of predefined classes for several application fields (create GUIs, ...)

Plan of the course

Data structures and programming structures

Primitive data types
Reference data types
Operators and expressions
Statements
Methods and passing of arguments
Arrays
Strings
Main method

Classes of objects and object oriented programming

Principle of object oriented programming
Using objects
Definition of classes
Subclasses and inheritance
Nested top-level classes and interfaces; inner classes

Exception handling

Catching and throwing exception objects
predefined exception classes

Threads

Running several threads concurrently
Coherence of shared data
thread termination

Packages of classes; programming environment (JDK)

Packages
Predefined packages
File organization

Various useful classes

String manipulation
Mathematical functions
Standard input / output
File input / output
Various system features
Hashtable

example:

→ *install JDK 1.6 (Java Development Kit) on your PC*
→ *In Computer Properties, Advanced system settings Environment Variables, edit Path variable and add at the end of it: ; path to bin folder of JDK*

→ *write in file MyProg.java :*

```
public class MyProg
{
   public static void main(String[] arg)
   {
      int[] a = new int[3];

      a[0] = 3;    a[1] = 23;    a[2] = 35;

      for (int i = 0 ; i < a.length ; i++)
         System.out.println("a[" + i + "] = " + a[i]);
   }
}
```

→ *open a shell in the same folder:*

compilation creates MyProg.class

```
> javac MyProg.java
```

```
> java MyProg

a[0] = 3
a[1] = 23
a[2] = 35
```

execution prints results in shell

example:

→ *write in file MyProg.java :*

```
public class MyProg
{
   public static void main(String[] arg)
   {
      Man mehmet = new Man();

      mehmet.weight = 70;

      mehmet.add_weight(10);

      System.out.println("mehmet's new weight is " + mehmet.weight);
   }
}

class Man
{
   int weight;

   void add_weight(int dw)
   {
      this.weight += dw;
   }
}
```

→ *open a shell in the same folder:*

> javac MyProg.java

compilation creates MyProg.class *and* Man.class

> java MyProg

mehmet's new weight is 80