PROGRAMMING III
OOP. JAVA LANGUAGE

COURSE 2

COURSE CONTENT

- **Classes**
  - Class modifiers
  - Fields modifiers
  - Method modifiers
- **Objects**
- **Display objects**
  - toString() method
  - StringBuffer
  - StringBuilder

CLASSES

- **Classes**
  - Groups objects with similar characteristics
- **Syntax**

  [classModifier] **class** ClassName [extends BaseClassName]
  [implements Interface1 [, Interface2] …[, InterfaceN]]{
  member fields and methods
  }

  Where
  - ClassName
    - Variable name that starts with upper case (does not contain spaces)
  - classModifiers
    - public, abstract, final

CLASSES

- **Classes**
  - Groups objects with similar characteristics
- **Syntax**

```
[classModifier] class ClassName [extends BaseClassName]
    [implements Interface1 [, Interface2] …[, InterfaceN]]{
     member fields and methods
}
```

CLASS MODIFIERS

- **Can appear only once in class declaration**
- **public modifier**
  - Class is visible in all packages
  - the name of the class has to be the same like the name of the file
- **abstract modifier**
  - used for classes that contain abstract methods
  - Used for classes that inherits abstract methods from a base class
  - If the class does not implement all the methods exposed by an interface

What is a package?

What is an abstract method?

CLASS MODIFIERS

- **final modifier**
  - The class definition is complete
  - The class cannot base class for other classes
    - A class cannot be in the same time final and abstract

OBJECT CLASS

- **All Java classes are inherited from Object class**
- **Some of most common used Object class methods**
  - protect void equals(Object obj)
    - Tests if the current object is equal with the one passed like parameter
  - protected void finalize()
    - Method called by *garbage collection* when the is no reference to the current object
  - public class getClass()
    - Method that returns the current class of the object
  - public int hashCode()
    - Homework: which is the role of hashCodeMethod()
  - public String toString()
    - Returns the string representation of the object

CLASSES

- **Classes**
  - Groups objects with similar characteristics
- **Syntax**

[classModifier] **class** ClassName [extends BaseClassName]
[implements Interface1 [, Interface2] …[, InterfaceN]]{
member fields and methods
}

CLASS MEMBER FIELDS

- **Class member fields/variables**
  - Describe the properties of a class
- **Syntax**
  - [fieldsModifier] variableType variableName [, variableName1 …[, variableNameN]];
  - [fieldsModifier] variableType variableName [=variable initialization]
  - [fieldsModifier] variableType variableName[] [=variable initialization]
- **Fields modifiers**
  - Access modifiers
    - public, protected, implicit/default, private
  - Others
    - final, static, transient, volatile

MEMBER FIELDS MODIFIERS

- **Access modifiers**
  - public
    - Visible all classes and packages
  - protected
    - Visible in derived classes
  - implicit/default
    - Visible in all classes in same package
  - private
    - Visible in current class

MEMBER FIELDS MODIFIERS

- **Others**
  - final
    - Constants
    - The value of the attribute is the same during the hole program execution
    - In many cases is used with static modifier
    - Constants in Java are written with upper cases
    - Must be initialized when they are declared
  - static
    - Allocates a single memory location that is shared by all class objects
    - Accessible by class name
  - transient
    - Variables that does not persist (are not serializable)
  - volatile
    - The value of this variable will never be cached thread-locally: all reads and writes will go straight to "main memory";
    - Access to the variable acts as though it is enclosed in a synchronized block, synchronized on itself.

CLASS MEMBER FIELDS

- **Example**

```
public class ExVariables {
   public static final int MAXIMUM_CAPACITY = 100;
   int age;
   private String name;
   transient double mean;
   protected double marks[];
   private int I, j, k=9;
   private double b[] = new double [10];
}
```

CLASS MEMBERS

- ***this* keyword**
  - A reference to the current object
- ***super* keyword**
  - A reference to base class

CLASS MEMBERS TYPES

- **Local variables**
  - Variables defined inside methods, constructors or blocks are called local variables.
  - The variable will be declared and initialized within the method and the variable will be destroyed when the method has completed.

- **Instance variables**
  - Instance variables are variables within a class but outside any method.
  - These variables are initialized when the class is instantiated.
  - Instance variables can be accessed from inside any method, constructor or blocks of that particular class.

- **Class variables**
  - Class variables are variables declared within a class, outside any method, with the static keyword.

CLASSES

- **Classes**
  - Groups objects with similar characteristics
- **Syntax**

```
[classModifier] class ClassName [extends BaseClassName]
    [implements Interface1 [, Interface2] …[, InterfaceN]]{
     member fields and methods
}
```

CLASS METHODS

- **Class member methods/function**
  - Describe the behavior of a object of the class
- **Syntax**

  [methodModifiers] returnType methosName ([parameter list]) [throws Exception1[, …, ExceptionN]] { …}

  Where
  - methodModifier: public, protected, private, default, abstract, final, static, synchronized, native
  - returnType: void, primitive or reference type
  - parameterList: formal parameters list

CLASS METHODS MODIFIERS

- **Access modifiers**
  - public
    - Visible all classes and packages
  - protected
    - Visible in derived classes
  - implicit/default
    - Visible in all classes in same package
  - private
    - Visible in current class

CLASS METHODS MODIFIERS

- **Others**
  - abstract
    - Offers only the signature of the method
    - The method does not provide an implementation
    - Cannot be: private, static, final, native or synchronized
  - static
    - Class method
    - Does not have access to *this* referese
  - final
    - Cannot be overwritten
  - synchronized
    - Only one thread can access the method when is executed
  - native
    - A native method in other programming language (like C, C++)

JAVA METHODS WITH VARIABLE ARGUMENTS LENGTH

- **Example**

```
class X {
   void method1 (int a, String … words) {
      for (String s: words) {
         System.out.println("argument: " + s);
      }
   }
   void method2 (double … numbers) { }
}
```

- **Properties**
  - It must be the last argument of the method
  - The argument is an array of objects of the type of the argument

- **Call**
  - method1(10)
  - method1(10, "s1","s2")
  - method1(10, "s1", "s2","s3")
  - …
  - method2()
  - method2(4.5)
  - method2(5.7, 7.8)
  - …

CONSTRUCTORS

- **Properties**
  - Function that an object calls when an object is instantiated
  - Has the same name like the class
  - It does not have a return value
  - If no constructor is defined a default constructor is provided by the compiler
- **Example**

```
public class X {
   int y;
//default constructor
   public X() { }
   public X(int v) {
       this.v = x;
   }
}
```

Constructor call:

```
X obj1 = new X();
X obj2 = new X(3);
```

CLONABLE AND COPY CONSTRUCTOR

- **Copy Constructor**
  - X(X obj)

- **Clonable interface**
  - clone(Object x)

TRANSFORMING OBJECTS TO STRING

- **Implemenatation of toString() methods**

- **Variants**
  - String class
  - StringBuilder
  - StringBuffer

OBJECTS

- Objects have states and behaviors
- Objects creation steps
  - Declaration
    - A variable declaration with a variable name with an object type.
  - Instantiation
    - The 'new' keyword is used to create the object.
  - Initialization
    - The 'new' keyword is followed by a call to a constructor.
    - This call initializes the new object.