UNIVERSITÀ DEGLI STUDI DE L'AQUILA

disim

Mobile Programming Laboratory

Overview of Object-Oriented Programming

Teachers

Ing. Tarquini Francesco, Ph.D

Ph.D in Computer Science Engineering

francesco.tarquini@univaq.it

Ing. D'Errico Leonardo

Ph.D Student in Computer Science Engineering

leonardo.derrico@graduate.univaq.it

Teaching Materials

Available on MOODLE platform

http://www.didattica.univaq.it

Google Drive Repository

https://drive.google.com/drive/folders/1ISqZfn0i9Ub3eWNXbvW00rd0hD9ya8OL?usp=sharing

Topics

- JAVA 7
  - Package
  - Classes and Objects
  - Variables and Methods
  - Visibility
  - Heritage
  - Abstract Class
  - Interfaces

JAVA

- Java is a programming language originally created by James Gosling in 1995 at Sun Microsystems (now owned by Oracle Corporation).
- Now the last version is Java SE 10, released on March 2018.
- The most common version is Java SE 8 (1.8.0), released on March 2014.
- In this course we will use Java SE 7 (1.7.x) because is fully supported by IDE.

- You can download it in the Oracle web site:
  - http://www.oracle.com/technetwork/java/javase/downloads/index.html
- An Open Source version, named OpenJDK, is available on the follow link:
  - http://openjdk.java.net/install/

JAVA

- In your computer you must install at least the Java JDK 1.7
  - In Oracle is available only the last JDK, version

Packages

- A Package is a collection of linked classes
- All source code in Java applications is defined in a package structure
- The package is **main** identifier of your application
- There are several packages that include some of the most used Classes
  - java.lang: language support
  - java.util: utility class
  - java.io: input/output

Packages

- Every Classes must be included in the application package.
- How can you add a Class in a package?
  - Using the “package” keyword in the top of the file as the first row:

```
package it.univaq.mobileprogramming;
```

- How can you use the class defined in other package?
  - Using the “import” keyword between package row and the class definition:

```
import java.io.*;
```

Classes and Object

- A Class is an extensible program-code-template for creating objects, providing initial values for state (member variables) and implementations of behavior (member functions or methods).

- When an object is created by a constructor of the class, the resulting object is called an instance of the class, and the member variables specific to the object are called instance variables, to contrast with the class variables shared across the class.

Classes and Object

- To create a class we must use the “class” keyword in the following way:

```
class ClassName {

    ClassName() {
        // Constructor implementation
    }

    // Here you can define the methods and variables of the class
}
```

- The Class constructor is used to instantiate a Class Object

```
ClassName instance = new ClassName();
```

Variables and Methods

- The variables are the attributes of the class and they store a value.

```
int state = 0;

char[] name = { 'H', 'E', 'L', 'L', 'O' };
```

- The primitive types of a variable are:
  - **short**: small numeric integer
  - **int**: numeric integer
  - **float**: numeric decimal
  - **long**: long numeric integer
  - **double**: long numeric decimal
  - **boolean**: true or false
  - **char**: character
  - **byte**: byte


Variables and Methods

- Methods are defined in the following way:

```
void setState(int state) {
        // Implementation Here
}

int getState() {
        // Implementation Here
}
```

Visibility

- Classes, Variables and Methods have several visibility options.
- Visibility can be:
  - **public**: every class can use the item
  - **private**: only the class that defines this item can use it
  - **protected**: as private visibility but extended to child class
  - **none**: as public but only for classes in the same package

Heritage

- Child Classes inherit the methods and attributes of the parent Class.
- Child Classes are declared as follows:

```
class ChildClassName extends ParentClassName {

        ChildClassName(){
                super();
                // Constructor implementation
        }

        // Here you can define the methods and variables of the class
}
```

Abstract Class

- Sometime you can need a Class some methods undefined.
  These methods, and the Class, are defined “abstracted”, using the “abstract” keyword.

```
abstract class ClassName {

    // Abstract methods
    abstract void methodName();
}
```

- Example:

```
abstract class Polygon {

    double perimeter;
    double area;

    abstract double getArea();

    abstract double getPerimeter();
}
```

```
class Square extends Polygon {

    double side;

    @Override
    double getArea(){
        return side *side;
    }
}
```

Interfaces

- A particular Class without attributes and with only abstract methods

```
interface InterfaceName {

    // Abstract methods
    void methodName();
}
```

- Classes can implement Interfaces using implements keyword

```
class ClassName implements InterfaceName {

}
```

- Java supports multiple interfaces