Unit 1
Introduction to Object Oriented Programming

Dr. Magdi AMER

Course Overview

References

Code: https://github.com/drmagdiamer/oop
Slides: https://github.com/drmagdiamer/oop/tree/main/src/com/javainbabysteps/oop/slides
Lectures: https://youtube.com/playlist?list=PLXtd__vEzNb7i9lWljXSzf-gGPhjNm8bS&si=sHaMhrxoZj98R80l

OOP History

- First OO language appeared in 1967: Simula ➔ too slow
- In 1972, SmallTalk was invented.
- Used mainly in research.
- Limited commercial applications.
- 1979, Bjarne Stroustrup extended C with classes during his Ph.D.
- 1983, Stroustrup, working at AT&T, further developed it to become C++.
- C++ was a big success, with a huge interest from the industry.
- The first commercial application was a C++ to C compiler.
  - This means that all the commands of C++ can be expressed in C.
  - C++ does not bring new commands, but a new modeling framework
  - OOP main drive come from software engineering.
- The world needed a commercial OO Language: the story of the Y2K bug
  - Y2K correction cost: 300 Billions.
- OOP main concepts:
  - Encapsulation
  - Extension
  - Classes ➔ Factory, Object➔ goods constructed by the factory.

Java History

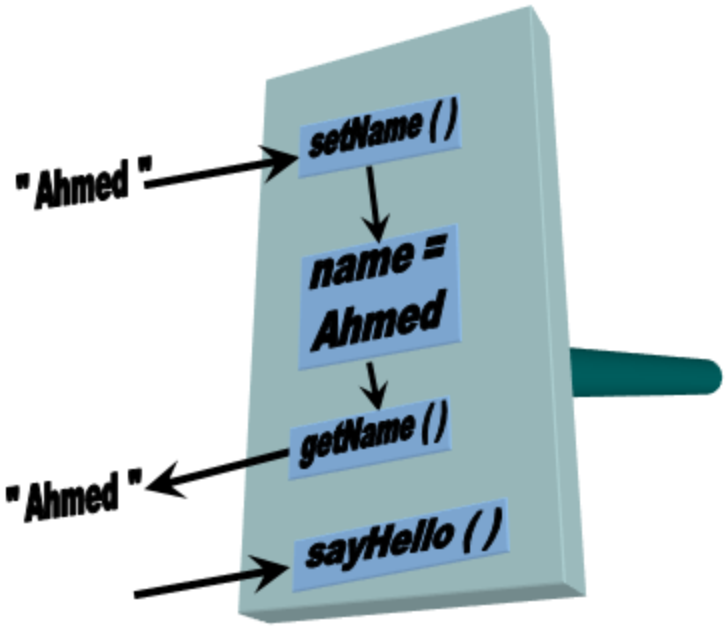
- **1991➔ Oak: a language for consumer electronics**
- **1994➔ Team is dissolved and Gosling resign.**
- **Gosling is given cart blanche, Java is born**
  - **A language that runs in any platform**
  - **A language that has security built in.**
- **1995➔ Java is made available for the industry.**
- **Since 1996, Java became the main language for server programming**
  - **CGI: old technology**
  - **ASP: Microsoft solution for small business**
  - **ColdFusion: another solution for small business**
- **In 2002, Microsoft launched dot net, a solution for small and medium business.**
- **Since them, Microsoft is enhancing dot net to make it a valid solution for large enterprise.**

Using Objects

- To build a car, we do not start by building factories that produce every component (tires, class, etc). Instead, we choose from the products that are available on the market. Other specialized products, such as car body and motor, will have to be specially manufactured. Building a car will become the process of manufacturing components and assembling them to build a car.
- Similarly, the process of building a software product should follow the same concept. The programmer should be able to use a set of pre-manufactured elements in building the application.
- This is done through the use of the concept of classes. A class is a factory that manufactures a specific type of product.
- When designing a program, we need first to identify the set of objects that are needed to build the application. These object need factories to build them, which are the classes. Some of these classes may be available as part of the standard Java libraries (packages). Other objects will be specific to the program that we are building, and we need to build the factory (or classes) that can build these objects.
- Building a program means creating objects and connecting them, exactly like we need to assemble components to build a car.

Using Objects

```
public class Student  {
  protected String name;
public Student()
{name = "" ;}
  public void setName(String aName)
 {  name = aName;   }
  public String getName( )
 {   return name;   }
public void sayHello()
{ System.out.println("Hello");}
}
```



Dissection Student

```
public class Student
{
.....
}
```

- Class declaration
- public➔ can be used from any other class
- class➔ keyword ➔ start of a class declaration
- Class Name:
  - Start with a letter
  - Followed by letters, numbers or underscore
  - No special character is allowed
  - As a convention: First letter is capital, Start of every word is capital.

Class Name

- The name of a class must begin with either:
  - a letter (a, b, c,…A, B, C….)
- The name of the class can contain (after the first character)
  - a letter (a, b, c,…A, B, C….)
  - a digit (1, 2, 3, …)
  - _
- Upper and lower case characters are distinct.
- The name of a class CAN NOT contain
  - a space
  - any other special character (*, &, %, $)
  - any non English letter (ص, é, Φ)
- The above rules are enforced by the compiler. If not followed, a compilation error will result.
- The naming standard indicates that:
  - A class name must begin with a capital letter.
  - If the class name is composed of multiple words (ex: invoice detail) then the beginning of each word after the first one must begin with a capital letter (ex: InvoiceDetail).
  - A class name should not be plural.
  - The naming standard rule is not enforced by the compiler, but any code not following the standard will be considered unacceptable.

The constructor

```
public Student()
{
 //do nothing
  name= "";
}
```

- Same name as class
- public➔ can be used from any other class
- Used to give an initial value to instance variables.
- Parameters, if needed, are passed in the parentheses.
- Code between CURLY BRACKETS
- ; to terminate a command
- // and /* */ used for comments.

Method Declaration

```
public void sayHello()
{
 System.out.prinln("Hello");
 }
```

- Again, ***public*** means that objects outside this ***package*** can called this method on instances of the class.
- Any ***method*** must have a return type. If the method does not return any value, it returns ***void***, which the Java way of saying nothing.
- **System.out** is a special object that represent the output stream of your system, which is usually the screen.
- We call a ***method*** on the **System.out** object called ***println***. This method is used to display a line of characters on the screen.
- Note that a String is delimited by two double quotes.

Variable Name

- The name of a variable must begin with either:
  - a letter (a, b, c,…A, B, C….)
  - _
  - $
- The name of the variable can contain (after the first character)
  - a letter (a, b, c,…A, B, C….)
  - a digit (1, 2, 3, …)
  - _
  - $
- Upper and lower case characters are distinct. A variable named “count” is different from another named “COUNT”.
- The name of a variable CAN NOT contain
  - a space
  - any other special character (*, &, %)
  - any non English letter (ص, é, Φ)
- The above rules are enforced by the compiler. If not followed, a compilation error will result.
- The naming standard indicates that:
  - A variable must begin with a small letter.
  - If the variable name is composed of multiple words (ex: average student grade) then the beginning of each word after the first one must begin with a capital letter (ex: averageStudentGrade).
  - The naming standard rule is not enforced by the compiler, but any code not following the standard will be considered unacceptable.

Method Name

- The name of a method must begin with either:
  - a letter (a, b, c,…A, B, C….)
  - _
  - $
- The name of the method can contain (after the first character)
  - a letter (a, b, c,…A, B, C….)
  - a digit (1, 2, 3, …)
  - _
  - $
- Upper and lower case characters are distinct.
- The name of a method CAN NOT contain
  - a space
  - any other special character (*, &, %)
  - any non English letter (ص, é, Φ)
- The above rules are enforced by the compiler. If not followed, a compilation error will result.
- The naming standard indicates that:
  - A method must begin with a small letter.
  - If the method name is composed of multiple words (ex: get average student grade) then the beginning of each word after the first one must begin with a capital letter (ex: getAverageStudentGrade( )).
  - The naming standard rule is not enforced by the compiler, but any code not following the standard will be considered unacceptable.

Java Starting Point

- Up till now, we declared a class, which is like a factory that fabricates objects, we do not have any object yet, nor a program.
- Java has a special method , called main, that is the starting point of all client applications. It must be declared within a class. Inside the method main, we will construct all the objects (our building blocks) and we will call the methods on these objects (perform actions).

```
public class MyFirstProgram
{
  public static void main(String args[])
 {
    Student object = new Student();
    object.sayHello();
  }
}
```

The main

public static void main(String args[])

- The syntax and the parameters of the main method can not be modified or the main function will not act as the starting point of the program. Let's look at these parameters.
- First of all, the method must be public.
- The method does not return any value, thus it returns ***void***.
- The method can be called even if there is no instance of the class created. These types of methods are called ***class method***, and they are declared with the keyword ***static***.
- The ***main*** method takes an array of ***String*** as a parameter. This represents the line arguments that were passed to the program when we the program was called.

Creating our first object

```
Student object = new Student();

object.sayHello();
```

- In Java, every variable must be declared with a specific type. In the above line, we declare a variable of type *Student*, which is the class that we created in the previous step.
- The variable name is *object*. Again, we can use any combination of letters or numbers, but it must begin with a letter. As a convention, a variable name must begin with a small letter.
- An instance of a class is created using a keyword ***new***, followed by the ***constructor*** that will be used to create the instance.
- Once the instance is created, we can start invoking methods on it. We use the object name, then a dot, then the method that we want to call.

Using Objects

```
public class MyApp
{
        public static void main(String args[])
        {
        Student s1;
        s1 = new Student();
        s1.setName("Rafik");
        Student s2 = new Student();
        s2.setName("Said");
        }
}
```