Lab X – Inheritance

Objectives:

- **To develop a subclass from a superclass through inheritance**
- **To invoke the superclass's constructors and methods using the super keyword**
- **To use instanceof**

1. Write a class **Employee** that has the following:
   a. **Private attributes:**
      i. A **String** representing the **name**.
      ii. An **integer** representing the **age**.
      iii. A **double** representing the **salary**.

   b. **Public methods:**
      i. **no-arg constructor** that initializes the **String** to "No infos", the **age** to 22 and the **salary** to 800. (invoke the constructor defined in ii.).

      ii. A **constructor** that takes 1 **String**, 1 integer and 1 double and initializes consecutively the attributes to given values. The constructor must invoke the **setEmployee** method defined in iii.

      iii. **setEmployee** method that sets all the attributes to given values. This method must invoke the 3 mutators.

      iv. **Mutators** and **Accessors** for each attribute.

      v. **toString** method that returns the information of the employee as a **String** in the following format:
      <name>
      <age >
      <salary>

```
Name: Walid
Age: 35
Salary: 2000.0
```

2. Write a class **Programmer** that **inherits** from **Employee** and has the following:
   a. **Private attributes:**
      i. A **String** representing the **language**.

   b. **Public methods:**
      i. A **constructor** that initializes all attributes to given values. The constructor must invoke the **setProgrammer** method defined in ii.

      ii. **setProgrammer** method that sets all the attributes to given values. This method

must invoke the mutators.

iii. **Mutators** and **Accessors** for each attribute.

iv. **toString** method that returns the information of the programmer as a **String** in the following format:
<name>
<age >
<salary>
<language>

```
Name: Lina
Age: 37
Salary: 5000.0
Language: Java
```

3. Write a class **Manager** that **inherits** from **Employee** and has the following:
   a. **Private attributes:**
      i. A **String** representing the **department**.
      ii. An **array of references** to **Employee** class, of size 10 representing the **employeesSupervised.**
      iii. An **integer** representing **numWorkersSupervised** initialized to **0**.

   b. **Public methods:**
      i. A **constructor** that initializes all attributes to given values. The constructor must invoke the **setManager** method defined in ii.

      ii. **setManager** method that sets all the attributes to given values. This method must invoke the mutators.

      iii. **Mutator** and **Accessor** for the attribute department.

      iv. **addWorker** method that takes a **reference** to class **Employee**, add an employee to the array **employeesSupervised**, and increment the **counter** of number of workers supervised by this manager.

      v. **toString** method that returns the information of the manager as a String in the following format:
      <name>
      <age >
      <salary>
      <department>
      < numWorkersSupervised>
      <name>

Lab X – Inheritance

```
Name: Maria
Age: 58
Salary: 15000.0
Department: CEO
6 Worker(s) are supervised by this manager
Those workers are:
Walid
Lina
Marie
Maguy
Fares
Khaled

Name: Khaled
Age: 45
Salary: 6100.0
Department: Translation
This manager is not supervising any employees
```

4. Write a test program that
   a. creates an array of **Employee** of size 6 and fill it by objects of classes **Employee**, **Programmer** and **Manager**
      i. emp[0] is an Employee("Walid", 35, 2000);
      ii. emp[1] is a Programmer("Lina", 37, 5000, "Java");
      iii. emp[2] is a Programmer("Marie", 22, 1500, "C++");
      iv. emp[3] is a Manager("Maguy", 50, 7000, "IT");
      v. emp[4] is an Employee("Fares", 30, 3500);
      vi. emp[5] is a Manager("Khaled", 45, 6100, "Translation");

   b. Print all the employees and display if each employee is a Manager, a Programmer or an Employee (using **instanceof**). (Check the output at the end).

   c. Create 3 managers (manager1, manager2 and manager3).
      i. manager1 is a Manager("Maria", 58, 15000, "CEO");
      ii. manager2 is a Manager("Marc", 50, 10000, "Translation Team");
      iii. manager3 is a Manager("Ola", 55, 11000, "Developing Team");

   d. Add the **Employees Lina** and **Marie** from the array as workers supervised by manager3, by invoking the method **addWorker.**

   e. Add the **Employees Walid** and **Fares** from the array as workers supervised by manager2, by invoking the method **addWorker.**

Lab X – Inheritance

f. Add all the **Employees** from the array as workers supervised by manager1, by invoking the method **addWorker** (**Walid**, **Lina**, **Marie**, **Maguy**, **Fares** and **Khaled**)**.**

g. Print manager1, manager2 and manager3. (Check the output at the end).

h. Display if manager1 is a Manager using **instanceof** Manager.
i. Display if manager1 is an Employee using **instanceof** Employee.
j. Display if manager1 is an Object using **instanceof** Object.

Lab X – Inheritance

```
Employee 1:
Name: Walid
Age: 35
Salary: 2000.0

Employee 2 is a Programmer
Name: Lina
Age: 37
Salary: 5000.0
Language: Java

Employee 3 is a Programmer
Name: Marie
Age: 22
Salary: 1500.0
Language: C++

Employee 4 is a Manager
Name: Maguy
Age: 50
Salary: 7000.0
Department: IT
This manager is not supervising any employees

Employee 5:
Name: Fares
Age: 30
Salary: 3500.0

Employee 6 is a Manager
Name: Khaled
Age: 45
Salary: 6100.0
Department: Translation
This manager is not supervising any employees
```

Lab X – Inheritance

```
Name: Maria
Age: 58
Salary: 15000.0
Department: CEO
6 Worker(s) are supervised by this manager
Those workers are:
Walid
Lina
Marie
Maguy
Fares
Khaled

Name: Marc
Age: 50
Salary: 10000.0
Department: Translation Team
2 Worker(s) are supervised by this manager
Those workers are:
Walid
Fares

Name: Ola
Age: 55
Salary: 11000.0
Department: Developing Team
2 Worker(s) are supervised by this manager
Those workers are:
Lina
Marie

Maria is a manager

Maria is an Employee

Maria is an object
```