TYPE 241

1. (1.5p) a) Explain what is abstraction within object-oriented programming.
b) Which of the following statements are true? Give short explanations about your answers.
S1) An instance method requires the creation of an object before it can be invoked.
S2) Static methods can access both static and non-static methods of the class.
S3) The abstract modifier is used for classes, methods, and variables, allowing them to be incomplete and requiring implementation in derived classes.
S4) A final modifier can appear before a class or a variable but not before a method.

2. (1p) a) What do you think can be improved in the next source code, without modifying the underlying functionalities? b) What is going to be printed after the execution of the main method? Explain it.

```
import java.util.ArrayList;
class Vehicle {
    public String toString() {
        return "Vehicle";
    }
}

class Car extends Vehicle {
    public String toString() {
        return "Car Vehicle";
    }
}

class Moto extends Vehicle {
    public String toString() {
        return "Moto Vehicle";
    }
}
```

```
class Bus extends Vehicle {
    public String toString() {
        return "Bus Vehicle";
    }
}

class Test{
    public static void main(String[] args) {
        ArrayList<Vehicle> vehicles;
        vehicles = new ArrayList<Vehicle>(12);
        vehicles.add(new Car());
        vehicles.add(new Moto());
        vehicles.add(new Bus());
        for (int i = 0; i < vehicles.size(); i++){
            System.out.println(vehicles.get(i));
        }
    }
}
```

3. (1p) Considering the source code given below, answer the following questions and properly explain your answer. a) What is going to be printed after the execution of Line 2?
b) Are we allowed to modify Line1 as if follows: Student c = new Player(“MyPlayer”); without generating a compilation error?

```
interface Person {
    String getType(); }

class Student implements Person {
    private String name;
    public Student(String n) {
        name = getType() + n;
    }
    public String getType() {
        return "Student “;
    }
    public String getName() {
        return name;
    }}
```

```
class Player extends Student {
    private String name;
    public Player(String n) {
        super(n);
    }
    public String getType() {
        return "Player “;
    }}

class Main {
    public static void main(String argv[]) {
        Player c = new Player("MyPlayer"); //Line1
        System.out.println(c.getName());   //Line2
    }}
```

4. (1p) Given the next class

```
        class Employee {
              private String firstName, lastName;
              public Employee(String firstName, lastName) { // . . . }}
```

Instantiate an object that contains Employee objects and print the kept employees. Inside the instantiated object we cannot store more than once the same employee. If needed, add new services inside the given class.

5. (0.75p) What is going to be printed after the execution of the next code? Properly explain your answer.

```
class NewString {
       public static void main(String args[]) {
               String myString="Hello ";
               myString.concat("from Netflix");
               System.out.println(myString); }}
```

6. (0.75p) What is going to be printed after the execution of the next code? Properly explain your answer.

```
class Another {
    private int value;
    public Another(int value) { this.value = value; }

    public void updateValue(int newValue) { this = new Another(newValue); }

    public String toString() { return "Another " + value; }

    public static void main(String[] args) {
        Another obj = new Another(10);
        obj.updateValue(15);
        System.out.println(obj);
    }
}
```

7. (0.75p) Present an example showing a situation in which we are allowed to change the content of a **static final** attribute defined inside a class. Describe why changing the content of the attribute is possible, despite being a static final attribute.

8. (0.75p) In the next example does **Line*** generate a compilation error? If it does, present why it is not possible to compile the given code and if it does not present how doSomething() can be used by a client via a polymorphic call.

```
class ExOne extends Exception {}
class ExTwo extends Exception {}
class A {
       public void doSomething() throws ExOne {}}
class B extends A {
       public void doSomething() throws ExTwo {} //Line*
}
```

9. (0.5p) What is going to be printed after the execution of the next sequence of code? Motivate the answer.

```
String[] myStrings = new String[10];
for (int i = 0; i <= myStrings.length; i++) {
    System.out.println(myStrings[i].toString()); }
```

10. (1p) We have a class named **Items** that encapsulates a collection whose type is LinkedList. In this collection we want to add many furniture objects such as a **Table**, a **Chair**, and an **Armchair**. For this purpose we use the following methods for adding the objects:

    public void add (Table t)
    public void add (Chair c)
    public void add (Armchair ac)

In the near future we may want to add other types of furniture such as a **Sideboard** and we would add in the **Items** class the method

    public void add (Sideboard sb).

Present a solution for adding new furniture object types into the collection so that adding a new furniture type (like Sideboard) does not involve changes inside the **Items** class.