TYPE C251

1. (1.5p) a) Explain what is Class Inheritance. Provide a short example.
b) Does the next source code exhibit problems?

```
class BaseClass {
        public void performService() {
                System.out.println("Hello from BaseClass");
        }}

class DerivedClass extends BaseClass {
        public void performService() throws RuntimeException {
                System.out.println("Hello from DerivedClass");
        }}
```

c) We need to instantiate three books: the first book has 17 pages, the second has 117 pages and the last one has 1117 pages. How many classes are necessarily in order to instantiate the mentioned books? Write the source code.

2. (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 1? b) What is going to be printed after the execution of Line 2? c) What is going to be printed after the execution of Line 3?

```
class MyClass {
    private int myInt;
    private String myString;

    public void setMyString(String myString) {
        this.myString = myString; }

    public void setMyInt(int myInt) {
        this.myInt = myInt; }

    public void changeValue(MyClass c) {
        myInt = c.myInt;
        myString = new String(c.myString);
        System.out.println(myString == c.myString);
        c = new MyClass();
        c.setMyInt(45);
        c.setMyString("XYZ");}

    public String toString() {
        return myInt + " " + myString;}

    public static void main(String[] argv) {
            MyClass b1 = new MyClass();
            MyClass b2 = new MyClass();

            b1.setMyInt(15);
            b1.setMyString("ABC");
            System.out.println(b1); //Line1

            b2.setMyInt(25);
            b2.setMyString("MNP");

            b1.changeValue(b2); //Line2
            System.out.println(b2); //Line3
    }}
```

3. (1p) What is going to be printed after the execution of the next sequence of code? Motivate the answer.

```
String[] myStrings = new String[5];
myStrings[0] =  "AAA";
myStrings[1] =  "BBB";
myStrings[2] =  "CCC";
for (int i = 0; i <= myStrings.length; i++) {
        System.out.println(myStrings[i].toString());
}
```

4. (1p) Given the next class,

```
class Person {
        private String firstName, lastName;
        // … constructor, toString,
}
```

instantiate an object that contains instances of Person and print its content. Each existing Person is stored only once. You are allowed to modify the given class.

5. (1p) Present the mechanism that allows us to define methods that inform their clients about special situations that may occur inside without returning/modifying variables or without printing on the screen messages related to the encountered special situations. Give an example that shows a benefit of the existing mechanism.

6. (0.5p) Comment the next statement:

```
class A { public A() { }}
class B extends A { }
```

The previous code cannot be compiled because class A has a constructor and, consequently, we have to define a constructor in class B in order to be able to compile the code.

7. (0.5p) Define the meaning of abstraction within the object-oriented paradigm.

8. (1p) We need to model two types of shapes: circles and rectangles. Do you think the next class fulfil our needs? Properly explain your answer.

```
class Figure {
	private int shape;
	//these fields are used if figure is a rectangle
	//shape=1
	private double length;
	private double width;

	//this field is used if figure is a circle
	//shape=2
	private double radius;

	//constructor for Circle
	public Figure(double radius) {
		shape = 1;
		this.radius =  radius;
	}

	//constructor for Rectangle
	public Figure(double length, double width) {
		shape = 2;
		this.length = length;
		this.width = width;
	}
}
```

9. (0.5p) Remove the duplication among the existing toString methods in the next source code. Present the performed changes.

```
class Person {
	protected String firstName, lastName;
	//constructor, setters, services

	public String toString() {
		return firstName + " " + lastName;
	}
}

class Student extends Person {
	private String universityName;
	//constructor, setters, services

	public String toString() {
		return firstName + " " + lastName + " " + universityName;
	}
}
```

10. (1p) Define a class StudentManager that has a collection that contains only instances of the previous class Student. The services provided by StudentManager are:

- public boolean add(Student o) - ensures that this collection contains the specified element
- public int size() - returns the number of students in this collection
- public String to String - returns a string representation of the collection

Add the necessarily code that ensures the class cannot have more than one instance - e.g. every time students are added to the same collection.