MIDTERM SAMPLE

1. A data field in a class is required to be available to other classes within the same package and only the same package. In this case, the data field should be marked:

A) It should be marked private
B) It should be marked protected
C) It should not be marked anything
D) It should be marked public

2. Suppose Fruit, Apple, GoldenDelicious, and McIntosh are defined in the following inheritance hierarchy: Apple inherits from Fruit, GoldenDelicious and McIntosh both inherit from Apple. Fruit class implicitly inherits from Object class.

Assume the following code is given:

Fruit fruit = new GoldenDelicious();

Which of the following Java codes results produces false.

A) fruit instanceof Fruit
B) fruit instanceof Apple
C) fruit instanceof GoldenDelicious
D) fruit instanceof McIntosh.

3. Which of the following is true regarding polymorphism:
A. that data fields should be declared private
B. that a class can extend another class
C. that a variable of supertype can refer to a subtype object
D. that a class can contain another class

4. Explain all the access modifiers in Java.

5. Explain what overloading and overriding a method is. Write Java code for both cases.

6. Write one method called filter( ) that takes an **unspecified** number of integers as parameters and returns an array consisting of input numbers that are divisible by 2 or 3 but not both.

filter (1, 2, 3, 5, 6); //should return [2, 3]

filter (8, 9, 11); //should return [8, 9]

filter (10, 20, 6, 30, 40, 60); //should return [10, 20, 40]

7. Validate the following regular expressions

STRING	Examples
Begin with a single letter, at least 1-3 numbers, followed by at least another letter	A1abc, D123ac, n12aak
Date of type mm-dd-yyyy	01-04-1999, 02-29-.2004
Albanian cell phone number +355 6(7 or 8 or 9) 2 numbers 2 numbers 3 numbers	+355 69 11 22 333, +355 68 12 12 123

8. The following exercise will be separated into sub-sections. Be aware that all data fields should be encapsulated.

- Create a class called Animal with fields of *id(int)*, *name (String)* and *birthdate (java.util.Date).* Write getters and setters for all fields. Create one single constructor that initializes all fields.

- Override toString method to return the name of the animal as a string representation.
- Create a class called Monkey which inherits from Animal. It has an encapsulated field *tail (boolean)*. Create one single constructor with all data fields.
- Override equals method to return true if and only if the compared object is a Monkey and the id is equal to the id of comparing object.
- Create a class called Rainforest, with an array of Monkey objects and an index to keep track of how many monkeys are living inside rainforest. The Rainforst has 2 constructors:
    - The first constructor creates an array of default size 5.
    - The second creates an array with size equal to the parameter given.

  Create a method inside this class that will add a new Monkey to the Rainforest array. **If there is no more capacity, you should provide a solution to enter the new Monkey.** A monkey with no tail, is not allowed to enter the rainforest (do not add them into the array). Return true if the monkey entered the rainforest, otherwise false.
  You are not allowed to use ArrayList class to solve this part.
- Create inside the Rainforest class a method called searchMonkey() that takes as a parameter the name of a monkey (String) and returns the first monkey with that name found in the rainforest. If no monkey found, return null.
- Create inside the Rainforest class a method called sorted () that returns an array of all the Monkey objects inside the class sorted according to their birthdate in ascending order.

9. (Partially bonus) What is the output of the below code:

```
public class First {
    protected static int tracker;
    private int i = 1;
    public String course = "SWE";

    public First() {
        i = --tracker;
        System.out.println(course + i);
    }

    public First(String s) {
        this.i--;
        System.out.println("Method->"+s);
    }

    public int mk(int s) {
        i += (tracker + s) + 10;
        return i--;
    }
    public String toString() {
        return course + "[i:" + i + "]";
    }
}
```

```
public class Second extends First {
    private String i;

    public Second() {
        System.out.println("CEN: " + tracker);
    }

    public Second(String x) {
        super(x);
        i = x + course;
        System.out.println(i + tracker--);
    }

    public int process() {
        i = this.toString();
        System.out.println(i + super.mk(tracker));
        return tracker++;
    }
}
```

```
public class Test {
    public static void main(String[] args) {
        int i = 0;
        First[] k = {
                new First(),
                new Second("215"),
                new First("Java"),
                new Second()
        };
        for (Object x : k) {
            if (x instanceof Second)
                i = ((Second)x).process();
            else
                i = ((First) x).mk(i);
        }
    }
}
```

Output: