FIRST PARTIAL EXAM
ALGORITHMS AND DATA STRUCTURES
27th FEBRUARY 2014

Surname, Name:	
Group	
Model	A

Some rules:

- Do not forget to write your name and surname.
- Read each exercise carefully.
- Use additional sheets to prepare your answer. Once you are sure about the answer to an exercise, write down your answer in the corresponding place in the exam sheets. Professors will only collect these exam sheets.
- There can be more than one correct answer for a given question. You must mark every correct answer.
- Wrong answers are valued as 0, do not decrease valuation.
- **Time: 30 minutes**

1. Given the following code:

```java
public class Student {

	private String name;

	/** getName method **/
	public String getName(){
		return name;
	}

	/**
	 * @Method: takeExam
	 * @description: shows when an student takes an exam
	 * for a given course
	 * @param course
	 * @return: true or false in a random way
	 */
	public boolean takesExam(AfterSchoolCourse course){
		if(Math.random()>0.5) return true;
		else return false;
	}
}
```

Choose the right answer/s:

1. **(0,5) In order to be able to use the *takeExam* method in the class Student in the *main* method of a class called *Test,* and supposing that the class *AfterSchoolCourse* is implemented somewhere:**

a) It is enough to define a variable of type `Student` (s) and another one of type `AfterSchoolCourse` (e) before writing the following invocation:

```
s.takeExam(e);
```

b) It is enough to define a variable of type `Student` (s) and another one of type `AfterSchoolCourse` (e) and get an instance of both before writing the following invocation:

```
s.takeExam(e);
```

c) It is enough to define a variable of type `AfterSchoolCourse` (e) and making the following invocation:

```
Student.takeExam(e);
```

d) It is enough to make an invocation with the following code:

```
Student.takeExam(ExtraCourse);
```

2. (0,5) Talking about constructors in class Student:

a) The given code is not correct because it is always needed to create a constructor method in any class. If not constructor is provided, the class cannot be instantiated.

b) The given code is correct, but a constructor must be provided in order to be able to get an instance of the `Student` class.

c) The given code is correct and to get an instance for the `Student` class it is enough to invoke the default constructor. An example is:

```
Student s = new Student("Daniel");
```

d) The given code is correct and to get an instance for the Student class it is enough to invoke the default constructor. An example is:

```
Student s = new Student();
```

3. (0,5) Talking about constructors in class `Student`. Is it possible to create two constructors in the same class?

a) If and only if the names of those constructors are different.

b) If and only if the number of parameters in each constructor is the same.

c) If and only if the number or data types of constructors parameters are different.

d) No, never, constructors cannot be overloaded.

4. (0,5) Given the following interfaces and classes definitions, mark those code lines with the right instantiations:

```
public interface IPerson {...}
public abstract class Employee implements IPerson {...}
public class Professor extends Employee {...}
```

a) IPerson p = new Employee();

b) IPerson p = new Professor();

c) Employee e = new Professor();

d) Professor p = new Employee();

5. (0,5) Given the following code and supposing that the class `Professor` has been implemented somewhere, mark the right options:

```
Professor p1 = new Professor();
Professor p2 = p1;
```

a) An object of type `Professor` has been created and both p1 and p2 variables reference this object.

b) Two objects of type `Professor` have been created, one of them referenced by the p1 variable and another one referenced by p2 variable.

c) The code is not correct (and a compilation error should appear) because objects of type `Professor` are not correctly created.

d) The code is not correct because the assignment in the second line should be the opposite (p1 = p2).

6. (0,5) Talking about access modifiers to attributes and methods and taking into account the following code, mark the correct options:

```
public abstract class Employee{
        protected int v1;
        …
}

public class Professor extends Employee {...}
```

a) It is possible to have direct access to v1 variable to query or modify its value from class `Employee`. It is not needed to use get/set methods to access the v1 variable.

b) It is possible to have direct access to v1 variable to query or modify its value from class `Professor`. It is not needed to use get/set methods to access the v1 variable.

c) The v1 variable would only be accessed from `Professor` class if the v1 variable would be defined as *private*.

d) It is NOT possible to access directly to the v1 variable from class `Professor` to query or modify its value. Using get/set methods to access the v1 variable is a must (if they are defined).

7. (0,5) Given the following code:

```
public class A extends B implements C {…}
```

a) The code is correct if B and C are both defined as interfaces.

b) The code is correct if and only if B is an abstract class and C is an interface

c) The code is correct if C is an interface and B is defined as a class.

d) All previous options are false.

8. (0,5) Given the following code:

```
public class A {

   public void printValue(){
         System.out.println("A");
   }

}

public class B extends A {

   public void printValue(){
         System.out.println("B");
   }

}

public class C extends B {

   public void printValue(){
         System.out.println("C");
   }
}

public class Test {

   public static void main(String[] args) {
          A obj = new B();
          obj.printValue();
          obj = new A();
          obj.printValue();
          obj = new C();
          obj.printValue();
   }

}
```

The output when the Test class is run would be:

a) A C B

b) B A C

c) A A A

d) A compilation error should be shown.

9. (0,5) Given the following code:

```
public class A extends B {

  public int met(int x, String s) {…}

  public int met(int x, String s, double h) {…}

}
```

a) Class B cannot have a met method

b) These methods cannot have the same return type.

c) `met` method is an overloaded method.

d) Only methods in the superclass can be overloaded.

10. (0,5) Given the following code:

```
public class Car {

   static int numberOfWheels = 4;

   public int getWheels(){
          return numberOfWheels;
   }

}
```

```
public class Test {

   public static void main(String[] args) {
          Car myCar;
          Car.numberOfWheels = 4;
   }

}
```

Mark the correct options:

a) A compilation error should be shown because `numberOfWheels` variable is defined as static

b) The right way to access the numberOfWheels variable should be `myCar.numberOfWheels`

c) It is a valid code although the `myCar` variable is not used.

d) `numberOfWheels` variable could only be accessed through the `getWheels` method.

e) All previous options are false.