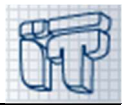


FIRST NAME:
LAST NAME:
NIA:
GROUP:

First midterm exam

Second Part: Problems (7 points out of 10)

Duration: 70 minutes
Highest score possible: 7 points
Date: March 24, 2021

Overall instructions for the exam:

- Books, notes, mobile phones, as well as other electronic devices are not allowed during the exam. Breaking this rule may result in expulsion from the examination
- Complete your personal information before starting the exam.

Problem 1 (2.5 / 7 points)

A vehicle dealer needs an application to manage its vehicle stock. Each vehicle is identified by a unique ID, which is assigned automatically and incrementally every time a new vehicle is delivered to the vehicle dealer and it is ready to be sold. Vehicles also need to store their production cost (`cost`) and their colour (`colour`). The vehicle dealer only sells red, yellow, blue and white vehicles, so those are the only possible colours (consider using constants to model this).

The dealer can sell cars and trucks. All the above features are common to both types of vehicles. The cars also need to store the maximum number of passengers (`passengers`) and the maximum speed (`maxSpeed`). For the trucks it is important to know the number of axles (`axles`) and whether it can have a trailer or not (`trailer`).

All vehicles will have a method to estimate its selling price (`estimateSellingPrice()`). This estimation differs from one type of vehicle to another. For example, for the trucks a combination of the number of axles and whether it can have a trailer or not are used for the estimation. If the number of axles is less than two it will be 20% higher than the production cost. Otherwise, if it can have a trailer the percentage will rise to 40%. But if it cannot, it will be 30%. For the cars the estimation will be influenced by the number of passengers and its maximum speed, but this is not relevant for this exercise.

Section 1.1 (1.5 points)

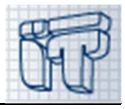
Write the code for the classes `Vehicle`, `Car` and `Truck`, except for the method `estimateSellingPrice()` in classes `Truck` and `Car`. Declare all constants, attributes and methods required to meet the above requirements. Write the code for a constructor with all attributes for each class.

Section 1.2 (0.5 points)

Write the code for the method public double `estimateSellingPrice()` in class `Truck`.

Section 1.3 (0.5 points)

Write a method to test `public void testEstimateSellingPrice()` in class `Truck`. The test must pass (positive test).



Problem 2 (3 / 7 points)

A leading logistics company has asked you to develop a program for managing a warehouse. One of the classes which is relevant for this program is `Person`. It refers to any relevant person for the management of a warehouse. It can be a customer of the warehouse, the employee that manages the order, or the contact person for a provider. To identify a person, we need his/her identification number (`id`), first name (`firstName`), last name (`lastName`) and contact email (`email`). Assume all getters and setters are already implemented for this class.

For the management of the warehouse, we need another class named `Employee`. The `Employee` will have a boss which is another object of class `Employee`. This is relevant to know the boss of the employee who manages the order. All employees are identified by their identification number, which must be unique. So, when creating a new employee and assigning the boss (`boss`) the id numbers must be different, otherwise an exception must be thrown (`EmployeeException`). If an employee has not any boss defined, null will be assigned to boss reference. Identification numbers must be assigned automatically and incrementally every time a new employee is created to avoid repetitions..

Section 2.1 (0.5 points)

Declare the class `EmployeeException` and write the code for its constructor.

Section 2.2 (1.5 points)

Write the code for the class `Employee` which must be a specialization of class `Person`. Declare its attributes, its setter methods (only setters, getters are not required) and write the code for two constructors, one with no arguments and the other with all arguments needed to initialize all its attributes.

Section 2.3 (1 point)

Write the code for the method `public int countSubordinates(Employee employees[], Employee boss)` which returns the number of employees in the array that have the same boss.

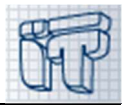
Problem 3 (1.5 / 7 points)

Write the code for a recursive method whose signature is

```
public int sumMultiplesOfNumber(int number, int min, int max)
```

The method will return the sum of the numbers between `min` and `max` which are multiples of the parameter `number`. For example, if `number` is 10, `min` is 20 and `max` is 50, the method must return 140 (20+30+40+50).

Note: Your solution **must be recursive**. No iterative solutions will be allowed.



REFERENCE SOLUTIONS (several solutions are possible)

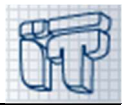
PROBLEM 1

Section 1.1 (1,5 points)

```
public abstract class Vehicle {

	public static final char RED = 'R';
	public static final char YELLOW = 'Y';
	public static final char BLUE = 'B';
	public static final char WHITE = 'W';

	protected static int numVehicles=0;

	protected int vehicleID;
	protected double cost;
	protected char colour;

	public abstract double estimateSellingPrice();

	public Vehicle(double cost, char colour) {
		numVehicles++;
		vehicleID=numVehicles;
		this.cost = cost;
		this.colour = colour;
	}

}

public class Car extends Vehicle {

	private int passengers;
	private double maxSpeed;

public Car(double price, char colour, int passengers, double maxSpeed) {
		super(price, colour);
		this.passengers = passengers;
		this.maxSpeed = maxSpeed;
	}
}

public class Truck extends Vehicle {

	private int axles;
	private boolean trailer;

	public Truck(double price, char colour, int axles, boolean trailer) {
		super(price, colour);
		this.axles = axles;
		this.trailer = trailer;
	}
}
```

Evaluation criteria:

- 0 if the code makes no sense.



- 1 for class Vehicle: 0,25 for the constants, 0,25 for attributes (maximum 0,1 if numVehicles is not static), 0,25 for the constructor and 0,25 for declaring estimateSellingPrice and the class as abstract. If class Vehicle is not abstract then -0,2
- 0,25 for each other class OK (0,1 for extending from Vehicle and declaring the attributes and 0,15 for creating the constructor, -0,1 if super() is not called).
- Significant errors are subject to additional penalties.

Section 1.2 (0,5 points)

```
public double estimateSellingPrice() {

		double result;

		if (axles > 2)
			if (trailer)
				result = cost * 1.4;
			else
				result = cost * 1.3;
		else
			result = cost * 1.2;

		return result;
	}
```

Evaluation criteria

- 0 if the code makes no sense.
- 0,2 for if-else conditions (0,1 each)
- 0,2 for calculating the price correctly.
- 0,1 if result return is correct.
- Significant errors are subject to additional penalties

Section 1.3 (0,5 points)

```
@Test
	public void testEstimateSellingPrice() {
		Truck truck=new Truck(20000, 'B', 3, true);
		assertEquals(truck.estimateSellingPrice(),30000,0.1);
	}
```

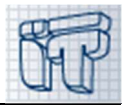
Evaluation criteria

- 0 if the code makes no sense.
- 0,1 for @test annotation
- 0,2 for creating the object Car
- 0,2 for calling assertEquals with the right arguments.
- Significant errors are subject to additional penalties

PROBLEM 2

Section 2.1 (0,5 points)

```
public class EmployeeException extends Exception {
    public EmployeeException(String msg) {
```



```
        super(msg);
    }
}
```

Evaluation criteria

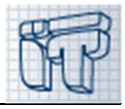
- 0 if the code makes no sense.
- 0,25 if class declaration is OK.
- 0,25 if constructor is OK.
- Significant errors are subject to additional penalties

Section 2.2 (1,5 points)

```
public class Employee extends Person {
            private static int genID = 0;
            private Employee boss;

            public Employee() throws EmployeeException {
                  this("Not defined", "Not defined", "xxxx@xxxx.com", null);
            }

            public Employee(String firstName, String lastName, String email,
Employee boss)
                        throws EmployeeException {

                  genID++;
                  setId(genID);
                  setFirstName(firstName);
                  setLastName(lastName);
                  setEmail(email);
                  setBoss(boss);

            }

            public void setBoss(Employee boss) throws EmployeeException {
                  if (boss != null)
                        if (boss.getId() == this.getId())
                              throw new EmployeeException("Duplicate ID");

                  this.boss = boss;
            }

}
```

Evaluation criteria

- 0 if the code makes no sense.
- 0,25 if class and attributes declarations are OK. If static attribute not declared, then 0,1 as maximum grade.
- 0,25 if the constructor with no attributes is OK.
- 0,4 if constructor with all attributes is OK (0,2 for the exception management and 0,2 for assigning the values to the attributes)
- 0,6 if setBoss method is OK (0,25 for the exception management and 0,25 for ID's comparison, and 0,1 for setting the boss)
- If setGenID is declared, then -0,25
- Significant errors are subject to additional penalties



Section 2.3 (1 point)

```
public int countSubordinates(Person employees[], Person boss){
                    int total=0;

                    for(int i=0;i<employees.length;i++)
                          if(boss.getId() == employees[i].getBoss().getId())
                                total++;

                    return total;
              }
}
```

Evaluation criteria

- 0 if the code makes no sense.
- 0,3 for going through the array properly.
- 0,5 for comparing the ID's correctly
- 0,2 for returning the proper value.
- Significant errors are subject to additional penalties.

PROBLEM 3

```
public int sumMultiplesOfNumber(int number, int min, int max) {

              int result = 0;

              if (min <= max) {

                    if (min % number == 0) // Multiple found
                          result = min;

                    result = result + sumMultiplesOfNumber(number, min + 1, max);

              }

              return result;
       }
```

Evaluation criteria

- 0 if the code makes no sense.
- 0,5 if the stop condition is correct.
- 0,25 if the detection of the multiple is correct
- 0,5 if the recursive call is correct
- 0,25 if the return of the result is correct.
- Significant errors are subject to additional penalties.