* 1. **Final Exam**

**Exercise 1**

Consider the following UML class diagram:



Class**Vehicle**

* *ATTRIBUTES*:

**-ID:** the ID of the vehicle.

**-rented:** true if the vehicle is rented (checked out).

**- nbHours:** duration of the rental in hours.

**- rentalAmount:** rental amount of the vehicle.

* *METHODS*:
* **Vehicle(id:int)**: constructor
* **display()**. this method displays all the attributes of the Vehicle object.
* **computeRentalAmount()**. computes and sets the **rentalAmount** of the vehicle if the vehicle is rented.
* For the **Truck** class, **rentalAmount** is: (hourlyRate\* nbHours)
* For the **Car** class, **rentalAmount** is: (dailyRate \* (nbHours/24)+ mileage \* 0.4)

Class **Car**

* *ATTRIBUTES*:
* **dailyRate:** the daily rental rate.
* **mileage:** thenumber of kilometers drove by the last renter.
* *METHODS*:
* **Car(id:int, rate:double, mileage:int):** Constructor.
* **getDailyRate().** Returns the daily rate of the Car.
* **getMileage().** Returns the mileage of the Car.
* **display().** This method displays all the attributes of the Car.

Class**Truck**

* *ATTRIBUTES*:
* **hourlyRate:** the hourly rate of the Truck.
* *METHODS*:
* **Truck(id:int, rate:double):** Constructor.
* **getHourlyRate().** Returns the hourly rate of the Truck.
* **display().** This method displays all the attributes of the Truck.

**Question:**Translate into Java code the classes**Vehicle** and **Car**.

**Exercise 2**

Consider the following UML class diagram:



Class **Branch**

* *ATTRIBUTES*:
* **name:** the name of the vehicle rental company branch.
* *METHODS*:
* **Branch(n: string, size: int).** Constructor **.** If the size is less or equal than zero, the constructor throws an exception with the following message “Invalid Size”.
* **addVehicle(veh: Vehicle).** Adds a vehicle to the branch. It returns true if the vehicle is added, false otherwise**.**
* **sumRentedCars(mil: int).** Computes and returns the sum of the rental amounts of all rented cars having a mileage less than ***mil***.
* **saveToFile(nbH:int, dailyR:double).** This methods saves the vehicles of the branch as follows:
* The **Car** objects with **dailyRate** equals to ***dailyR*** are savedin the file “cars.data”.
* The **Truck** objects with **nbHours** greater than ***nbH*** are savedin the file “trucks.data”.

**Question:**Translate into Java code the class **Branch**.

**Solution Final Exam**

8 pts

**public** **abstract** **class** Vehicle ---------------- 1

{

 **private** **int** ID;

 **protected** **boolean** rented;

 **protected** **int** nbHours;

 **protected** **double** rentalAmount;

 **public** Vehicle(**int** id)

 {

 ID = id; ---------------- 1

 rented = **false**;

 nbHours = 0;

 rentalAmount = 0.00;

 }

 **public** **int** getNbHours()

 {

 **return** nbHours; ---------------- 1

 }

 **public** **abstract** **void** computeRentalAmount(); ---------------- 1

 **public** **void** display()

 {

 System.*out*.println("ID: " + ID); ---------------- 1

 System.*out*.println("Rented: " + rented); ---------- 1

 System.*out*.println("Nb Hours: " + nbHours); ------- 1

 System.*out*.println("Rental Amount: " + rentalAmount);-- 1

 }

}

10 pts

**public** **class** Car **extends** Vehicle **implements** Serializable ----------- 1

{

 **private** **double** dailyRate;

 **private** **int** mileage;

 **public** Car(**int** id, **double** rate, String mileage)

 {

 **super**(id); ----------- 1

 dailyRate = rate; ----------- 1

 this.mileage = mileage; ----------- 1

 }

 **public** **double** getDailyRate()

 {

 **return** dailyRate; ----------- 1

 }

 **public** **int** getMileage()

 {

 **return** mileage; ----------- 1

 }

 **public** **void** computeRentalAmount()

 {

 rentalAmount = dailyRate \* (nbHours / 24) + mileage \* 0.4; ------- 1

 }

 **public** **void** display()

 {

 **super**.display(); ----------- 1

 System.*out*.println("Daily Rate: " + dailyRate); ----------- 1

 System.*out*.println("Mileage: " + mileage); ----------- 1

 }

}

30 pts

**import** java.io.\*;

**public** **class** Branch

{

 **private** String name;

 **private** Vehicle vehicles[]; ----------- 1

 **private** **int** nb; ----------- 1

 **public** Branch(String n, **int** size) **throws** Exception ----------- 1

 {

 **if**(size <= 0) ----------- 1

 {

 **throw** **new** Exception("Invalid Size"); ----------- 1

 }

 name = n; ----------- 1

 nb = 0; ----------- 1

 vehicles = **new** Vehicle[size]; ----------- 1

 }

 **public** **boolean** addVehicle(Vehicle veh)

 {

 **if**(nb < vehicles.length) ----------- 1

 {

 vehicles[nb] = veh; ----------- 1

 nb++; ----------- 1

 **return** **true**; ----------- 0.5

 }

 **return** **false**; ----------- 0.5

 }

 **public** **double** sumRentedCars(**int** mil)

 {

 **double** sum = 0.00; ----------- 1

 **for**(**int** i=0; i<nb; i++) ----------- 1

 {

 **if**(vehicles[i] **instanceof** Car ----------- 1

 && vehicles[i].rented ----------- 1

 && ((Car)vehicles[i]).getMileage() < mil ) ---------- 1

 {

 vehicles[i].computeRentalAmount();

 sum += vehicles[i].getRentalAmount(); ----------- 1

 }

 }

 **return** sum; ----------- 1

 }

**public** **void** saveToFile(**int** nbH, **double** dailyR)

 {

 **Try** ----------- 1

 {

 File f1 = **new** File("cars.data"); ----------- 1

 FileOutputStream fos1 = **new** FileOutputStream(f1); ---- 1

 ObjectOutputStream os1 = **new** ObjectOutputStream(fos1); -- 1

 File f2 = **new** File("trucks.data");

 FileOutputStream fos2 = **new** FileOutputStream(f2); ----------- 1

 ObjectOutputStream os2 = **new** ObjectOutputStream(fos2);

 **for**(**int** i=0; i<nb; i++) ----------- 1

 {

 **if**(vehicles[i] **instanceof** Car ----------- 1

 && ((Car)vehicles[i]).getDailyRate() == dailyR) --- 1

 {

 os1.writeObject(vehicles[i]); ----------- 1

 }

 **else** **if**(vehicles[i].getNbHours() > nbH) ----------- 1

 {

 os2.writeObject(vehicles[i]); ----------- 1

 }

 }

 }

 **catch**(Exception e)

 {

 }

 }

}

* 1. **Midterm 2**

**King Saud University**

**College of Computer & Information Sciences**

**Computer Science Department**

**CSC 113**

**Question 1:**



***Donation class:***

* + ATTRIBUTES:
	+ ***amount***: the amount of the donation.
	+ ***location***: the location wherethe donation wasreceived.
	+ ***donorName***:name of the donor.
	+ METHODS:
	+ ***Donation(amount :double, location : string, donorName: string):*** constructor**.**
	+ ***getDonorName(): string:*** returns the donor name*.*
	+ ***getAmount: double:*** returns the amount of the donation. This method throws an exception if the amount is negative.
	+ ***display():*** displays the values of the attributes of the donation.

***Cash class:***

* + ATTRIBUTES:
	+ ***currency***: the currency of the cash donation.
	+ METHODS:
	+ ***Cash(amount: double, location: string, donorName: string, currency:string):***constructor.
	+ ***getCurrency():string:*** returns the currency.
	+ ***display()***: displays all the attributes of the **Cash** object.

***Check class:***

* + ATTRIBUTES:
	+ ***bankName***: the name of the bank of the check.
	+ METHODS:
	+ ***Check(amount: double, location: string, donorName: string, bankName:string):***constructor.
	+ ***getBankName():string:*** returns the name of the bank of the check.

***IOInterfaceinterface:***

* + METHODS:
	+ ***saveToFile(fileName:string, donor:string):***This method saves all the **Cash** donations received from the donor **donor** into the object file **fileName**.
	+ ***loadFromFile(fileName:string, arrCheck: Check[]):*** This method reads the **Donation** objects from the file **fileName** and stores the **Check** donations **only** in the array **arrCheck**.

***CharityAssociation class:***

* + ATTRIBUTES:
	+ ***name***: the name of the association.
	+ METHODS:
	+ ***CharityAssociation (name:string, size: int):*** constructor***.***
	+ ***addDonation(d: Donation):boolean:*** adds a donation **d** to the charity association. It returns true if done. False otherwise.
	+ ***avgCashDonation(cur:string):double:*** calculates the average amount of the cash donations which currency is **cur**. The donations with negative amount should not be considered.
	+ ***getCheck (bName:string): Donation:*** returns a **Check** donation which bank name is **bName**.

**QUESTION 1:**

Translate into Java code the class ***Donation***, and the class ***Cash*.**

**QUESTION 2:**

Translate into Java code the interface ***IOInterface*** and the class ***CharityAssociation***.

**QUESTION 3:**

Give the output of the following program if the inputs are: five, 600, 12.

**public** **class** A {

 **protected** **int** xA;

 **public** A() { System.*out*.println("Constructor of A is invoked"); xA = 1;}

 **public** A(**int** y) {

 System.*out*.println("Constructor 2 of A is invoked");

 xA = y;

 }

}

**public** **class** B **extends** A {

 **private** **int** xB;

 **public** B(){ System.*out*.println("Constructor 1 of B is invoked"); }

 **public** B(**int** x) **throws** ArithmeticException {

 **if** (x > 100) {

 **throw** **new** ArithmeticException();

 }

 xB = x;

 System.*out*.println("Constructor 2 of B is invoked");

 System.*out*.println("xA = " + xA);

 System.*out*.println("xB = " + xB);

 }

 **public** **void** process(**int** a) **throws** ArithmeticException {

 **if** (a > 100) **throw** **new** ArithmeticException("Wrong value");

 xA = a/2;

 xB = a/3;

 System.*out*.println("xA = " + xA);

 System.*out*.println("xB = " + xB);

 }

}

**public** **class** App {

 **public** **static** **void** main(String[] args) **throws** Exception {

 Scanner input = **new** Scanner(System.*in*);

 String s = **null**;

 B b = **new** B(10);

 **for**(**int** i=0; i<3; i++) {

 System.*out*.println("Enter string number " + (i+1) + ": ");

 s = input.next();

 **try** {

 b.process(Integer.*parseInt*(s));

 }

 **catch**(NumberFormatException e){ System.*out*.println("Wrong format"); }

 **catch**(Exception e) { System.*out*.println(e.getMessage()); }

 }

 }

}

Hint: Integer.*parseInt*(x: String): int. This method converts a String to an int. If the String does not contain an integer, this method throws a NumberFormatException.

* 1. **Solution Mid Term 2**

**EXERCISE 1**

*Donation:* …………………………………………………………………………. 6

**public** **class** Donation **implements** Serializable …………. 0.5

{

 **private** **double** amount;

 **private** String location;

 **private** String donorName;

 **public** Donation(**double** amt, String loc, String dnr) { …………………………………………. 0.5

 amount = amt;

 location = loc;

 donorName = dnr;

}

 **public** Donation(Donation d) { …………………………………………. 1

 amount = d. amount;

 location = d. location;

 donorName = d. donorName;

}

 **public** String getDonorName() { ……………………. 0.5

 **return** donorName;

 }

 **public** **double** getAmount() **throws** Exception { ……………………. 1

 **if**(amount < 0) ……………………………………. 0.5

 {

 **throw** **new** Exception("Error, amount < 0"); ……………………. 1

 }

 **return** amount; …………………………………………. 0.5

 }

 **public** **void** display() { ………………. 0.5

 System.*out*.println("Amount: " + amount);

 System.*out*.println("Location: " + location);

 System.*out*.println("Donor: " + donorName);

 }

}

*Cash:* …………………………………………………………………………. 7

**public** **class** Cash **extends** Donation …………. 0.5

{

 **private** String currency;

 **public** Cash(**double** amt, String loc, String don, String cur)

 {

 **super**(amt, loc, don); …………. 1

 currency = cur; …………. 1

 }

 **public** Cash(Cash csh) …………. 0.5

 {

 **super**(csh); …………. 1

 currency = csh.currency; …………. 1

 }

 **public** String getCurrency()…………. 0.5

 {

 **return** currency;

 }

 **public** **void** display()

 {

 **super**.display(); …………. 1

 System.*out*.println("Currency: " + currency); …………. 0.5

 }

}

**EXERCISE 2**

*Interface:* …………………………………………………………………………. 3

**public** **interface** IOInterface …………………………………………………………………………. 1

{

 **public** **void** saveToFile(String filename, String donor ); ……………………………. 1

 **public** **void** loadFromFile(String filename, Check arrCheck[]);…………………. 1

}

*CharityAssociation:* …………………………………………………………………………. 36

**import** java.io.\*;

**public** **class** CharityAssociation **implements** IOInterface …………. 1

{

 **private** String name;

 **private** Donation arDonation[]; …………. 0.5

 **private** **int** nb; …………. 0.5

 **public** CharityAssociation(String name, **int** size)

 {

 **this**.name = name;

 arDonation = **new** Donation[size]; …………. 0.5

 nb = 0;

 }

 **public** **boolean** addDonation(Donation d) .....

 {

 **if**(nb < arDonation.length) **…………. 1**

 {

 **if**(d **instanceof** Cash) …………. 1

 {

 arDonation[nb] = **new** Cash((Cash) d); …………. 1

 }

 **else**

 {

 arDonation[nb] = **new** Check((Check) d); …………. 1

 }

 nb++; …………. 0.5

 **return** **true**; …………. 0.5

 }

 **return** **false**; …………. 0.5

 }

 **public** **double** avgCashDonation(String cur) {

 **double** sum = 0.00; …………. 0.5

 **double** amnt = 0.00;

 **for**(**int** i=0; i<nb; i++) { …………. 0.5

 **if**( arDonation[i] **instanceof** Cash && …………. 1

 ((Cash)arDonation[i]).getCurrency().equals(cur) ) { …………. 1

 **try {** …………. 0.5

 amnt = arDonation[i].getAmount(); …………. 0.5

 sum += amnt; …………. 0.5

 }

 **catch**(Exception e) { }

 }

 }

 **if**(nb != 0) { …………. 0.5

 **return** sum/nb; …………. 0.5

 }

 **return** sum; …………. 0.5

 }

 **public** Donation getCheck(String bName) {

 **for**(**int** i=0; i<nb; i++) {

 **if**( arDonation[i] **instanceof** Check && …………. 1

 ((Check)arDonation[i]).getBankName().equals(bName) ) { ……. 1

 **return** arDonation[i]; …………. 0.5

 }

 }

 **return** **null**; …………. 0.5

 }

 **// saveToFile: : Solution 1 ………… 9.5**

 **public** **void** saveToFile(String fileName, String donor) throws IOException {

 File f = **new** File(fileName); …………. 0.5

 FileOutputStream fo = **new** FileOutputStream(f); …………. 1

 ObjectOutputStream os = **new** ObjectOutputStream(fo); …………. 1

**try {** …………. 1

 **for**(**int** i=0; i<nb; i++) { …………. 1

 **if**(arDonation[i] **instanceof** Cash && …………. 1

 **arDonation[i].getDonorName().equals(donor)**) { 1

 os.writeObject(arDonation[i]); …………. 1

 }

 }

 }

 **catch**(IOException e) …………. 1

 {

 System.*out*.println("Error in saveToFile");

 }

 os.close(); …………. 1

 }

 **// saveToFile: : Solution 2 ……………… 9.5.**

 **public** **void** saveToFile(String fileName, String donor) throws IOException {

 File f = **new** File(fileName); …………. 0.5

 FileOutputStream fo = **new** FileOutputStream(f); …………. 1

 ObjectOutputStream os = **new** ObjectOutputStream(fo); …………. 1

 Cash cs[] = **new** Cash[nb]; …………. 0.5

 **int** j = 0; …………. 0.5

**try** …………. 0.5

 {

 **for**(**int** i=0; i<nb; i++)

 {

 **if**(arDonation[i] **instanceof** Cash && …………. 1

 **arDonation[i].getDonorName().equals(donor)**) { 1

 cs[j] = (Cash)arDonation[i]; …………. 0.5

 j++; …………. 0.5

 }

 }

 os.writeObject(cs); …………. 1

 }

 **catch**(IOException e) …………. 0.5

 {

 System.*out*.println("Error in saveToFile");

 }

 os.close(); …………. 1

 }

**// loadFromFile: Solution 1 ……….. 8.5**

**public** **void** loadFromFile(String fileName, Check arrCheck[])throws IOException {

 File f = **new** File(fileName);

 FileInputStream fi = **new** FileInputStream(f); …………. 1

 ObjectInputStream is = **new** ObjectInputStream(fi); …………. 1

Donation d;

 **int** j = 0; …………. 0.5

 **try**

 {

 while (true){ …………. 0.5

 d = (Donation) is.readObject(); …………. 1

 **if**( d **instanceof** Check && …………. 1

 j < arrCheck.length){ // this is optional

 arrCheck[j] = (Check) d; …………. 1

 j++; …………. 0.5

 }

 }

}

 **catch**(Exception e) …………. 1

 {

 System.*out*.println("Error in loadFromFile ");

 }

 is.close(); …………. 1

 }

**// loadFromFile: Solution 2 …………… 8.5**

 **public** **void** loadFromFile(String fileName, Check arrCheck[])

 {

 **try**

 {

 File f = **new** File(fileName);

 FileInputStream fi = **new** FileInputStream(f); …………. 1

 ObjectInputStream is = **new** ObjectInputStream(fi); …………. 1

 Donation array[] = (Donation[]) is.readObject(); …………. 1

 is.close(); …………. 1

 **int** j = 0; …………. 0.5

 **for**(**int** i=0; i<array.length; i++)

 {

 **if**(array[i] **instanceof** Check) …………. 1

 {

 arrCheck[j] = (Check) array[i]; …………. 1

 j++;

 }

 }

 }

 **catch**(IOException e) …………. 1

 {

 System.*out*.println("Error in loadFromFile ");

 }

 **catch**(Exception e) …………. 1

 {

 System.*out*.println("Error in loadFromFile ");

 }

 }

}

**EXERCISE 3** …………………………………………………………………………. 8 marks

Constructor of A is invoked …………………………………………………………………………. 1

Constructor 2 of B is invoked …………………………………………………………………………. 1

xA = 5 …………………………………………………………………………. 0.5

xB = 10

Enter string number 1: five …………………………………………………………………………. 0.5

Error code 1 …………………………………………………………………………. 1

Enter string number 2: 600 …………………………………………………………………………. 0.5

Error code 2 …………………………………………………………………………. 1

Enter string number 3: 87 …………………………………………………………………………. 0.5

xA = 3 …………………………………………………………………………. 1

xB = 7 …………………………………………………………………………. 1