COMP 249: Object Oriented Programming II

Tutorial 5:

Exception Handling

Question 1

Match each situation in the first column with an item in the second column.

1. int[] A;
   A[0] = 0;

1. The Java VM starts running your program, but the VM can't find the Java platform classes. (The Java platform classes reside in classes.zip or rt.jar.)

1. A program is reading a stream and reaches the end of stream marker.

1. A program tries to use a FileWriter instance to open a read-only file.

1. Error

2.Checked exception

3. Compile Error

4. No exception

Question 2

Match each situation in the first column with an unchecked exception in the second column.

1.
```
int a = 30, b =0;
int c = a/b;
```

2.
```
String a;
System.out.print(a.charAt(0));
```

3.
```
class Car{...}
class BMW extends Car{...}
class Mercedes extends Car{...}
Car car = new BMW();
Mercedes mercedes = (Mercedes) car;
```

4.
```
int array[] = new int[5];
array[6] = 9;
```

1.NullPointerException

2.ArrayOutOfBoundsException

3. ClassCastException

4.ArithmeticException

Question 3

Modify the following cat method so that it will compile:

```
public static void cat(File named) {
    RandomAccessFile input = null;
    String line = null;
    try {
        input = new RandomAccessFile(named, "r");
        while ((line = input.readLine()) != null) {
            System.out.println(line);
        }
        return;
    } finally {
        if (input != null) {
            input.close();
        }
    }
}
```

Question 4

Modify the following method so that it will compile:

```
package data;

import java.io.File;
import java.io.IOException;
import java.sql.SQLException;

public class BadIO {
	public static void cat(File named) {
		BadIO obj_IO = new BadIO();

		try{
			obj_IO.fileBlowUp();
			obj_IO.databaseBlowUp();
		} //INSERT CODE HERE
	}

	void databaseBlowUp() throws SQLException {
		throw new SQLException();
	}

	void fileBlowUp() throws IOException {
		throw new IOException();
	}
}
```

Question 5

Given the following CreditCard Class:

```
package tutorial6;
import tutorial6.Exception.*;
import java.util.Date;

class CreditCard {
        private String name; // Card owner name
        private double balance; // current amount of money lent to the card owner

        private final static double creditLimit = 1500; //
        private final static Date expiryDate = new Date(2023,12,31); // you may change this date to test your exception class


    /** constructor */
    public CreditCard(String name, double balance) {
                this.name = name;
                this.balance = balance;
    }

    /** constructor */
    public CreditCard(String name) {
                this(name,0);
    }

    /** return balance */
    public double getbalance() {
                return balance;
    }
    …
```

Question 5 (cont.)

Given the following CreditCard Class:

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

    public String toString() {
                return "Name: " + name + "\n" + "balance: " + balance ;
    }

   /** make a purchase with the creditCard */
    public void makeAPurchase(double purchaseAmount) throws CreditCardException {
        // TODO
    }
    /** pay off the creditCard */
    public void payOff(double paymentAmount) throws CreditCardException {
        // TODO
    }
```

Question 5 (cont.)

1. Create the following Exception classes:
    1. CreditCardException
    2. InvalidAmountException extends CreditCardException
    3. CardExpiredException extends CreditCardException
    4. AvailableCreditException extends CreditCardException
    5. OverpaymentException extends CreditCardException
2. Implement the makeAPurchase(double purchaseAmount) method which:
    1. Throws InvalidAmountException if purchaseAmount is non positive
    2. Throws AvailableCreditException if purchaseAmount is greater than the available credit
    3. Throws CardExpiredException if the card is expired
    4. Otherwise, process the purchase
3. Implement the payOff(double paymentAmount) method which:
    1. Throws InvalidAmountException if paymentAmount is non positive
    2. Throws OverpaymentException if the paymentAmount is greater than the balance
    3. Throws CardExpiredException if the card is expired
    4. Otherwise, process the payment

Question 5 (cont.)

Create a driver class to test your code:

- Create 1 credit card
- Test the makePurchase and payOff methods in multiple scenarios