Previous Quizzes 2
(CSC 113)

Question 1: Given the java code below:

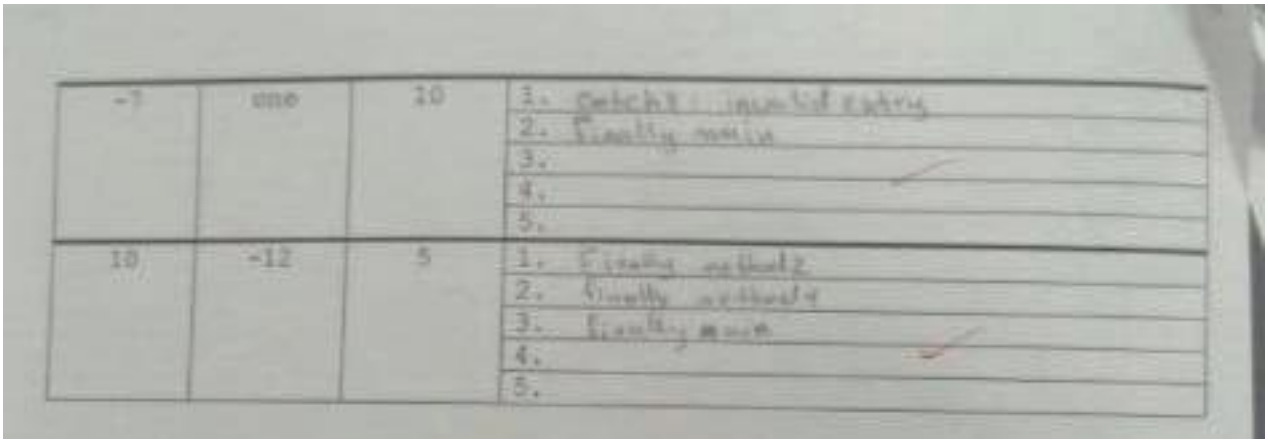
```
import java.util.*;

class MyException extends Exception
{
   public MyException(String message)
   {
      super(message);
   }
}//end class MyException

public class Test
{
   public static void method1(int x, int y, String s) throws Exception
   {
      try
      {
         if(x<0)
            throw new IllegalArgumentException("x is less than 0");
         double result=x/y;
         method2(s);
      }
```

```
    catch( IllegalArgumentException e)
    {
       System.out.println("method1 catch: "+e.getMessage());
    }
    finally
    {System.out.println("finally method1"); }

}//end method1

public static void method2( String s) throws MyException
{
    try
    {
       int x= Integer.parseInt(s);

       if(x>50)
         throw new IllegalArgumentException("x is greater than 50");
    }
    catch(NumberFormatException e)
    {
       System.out.println("catch methd2");
       throw new MyException("NumberFormatException") ;
          }
    finally
    { System.out.println("Finally method2");}
}

public static void main (String args[])
{
    Scanner scan =new Scanner (System.in);

    try
    {
       int a=scan.nextInt();
       int b=scan.nextInt();
       String c=scan.next();

       method1(a,b,c);
    }//end try
    catch(ArithmeticException e)
    {
    System.out.println("catch1:Arithmetic Exception");
    }
```

```
catch (MyException e)
{
   System.out.println("catch2:"+e.getMessage());
}
catch(InputMismatchException e)
{
   System.out.println("catch3: invalid entry");
}
catch(Exception e)
{
   System.out.println("catch4:"+e.getMessage());
}
finally {System.out.println("Finally main");}

}
}
```

Assume the user's input for the three declared variables `a`, `b` and `c` will be as shown in the following table, show the program output for each user input such that each run will output no more than 5 lines: **(10 marks)**

a	b	c	output
-3	2	4	1. method1 catch: x is less than 0 2. finally method1 3. Finally main 4. 5.
30	20	75	1. Finally method2 2. method1 catch: x is greater than 50 3. Finally method1 4. Finally main 5.
3	0	200	1. finally method1 2. catch1: Arithmetic Exception 3. Finally main 4. 5.
200	8	one	1. catch method2 2. Finally method2 3. finally method1 4. cath 2: NumberFormatException 5. Finally main



-7	one	10	1. Switch : invalid expression 2. Finally main 3. 4. 5.
10	-12	5	1. Finally without 2 2. Finally without 4 3. Finally main 4. 5.

King Saud University
College of Computer and Information Sciences
Computer Science Department

CSC 113 **Second Semester 1436-1437**

Quiz 2

Student Name	Student ID	Section Number	Serial Number

Q1:Trace the following Java program

```
publicclass testExceptions{
static Boolean flag=true;
publics tatic int simplefy (int n){
while(flag){
       try{
       if(n>= 10)
          thrownew ExceptionB  ();
       else
           return n;
       }
       catch(ExceptionBexp){ System.out.println(exp);
          return n/10;}
       finally{ System.out.println("finally simplefy "+ n);}}
          return 0;
       }

       Public static boolean validate(int num)throwsExceptionC{
       try{
       int number = simplefy(num);
         if(number> 0)
       throw new ExceptionC();
       return false;}
          catch(ExceptionA exp){ System.out.println(exp);
       return true;}}

       public static void Test(int num)throws ExceptionA{
       try{
           if(validate(num)) throw new ExceptionA("ExceptionA");
          }
       finally{ System.out.println("finally Test ");}
       }

Public static void main(String[] args){
       try{
       Test(100);}
             catch(Exception e){
               System.out.println(e + " last catch ");}}}

class ExceptionA extends Exception  {
       public ExceptionA (String m){
          super(m);} }

class ExceptionB extends ExceptionA {
             public ExceptionB (){
       super("ExceptionB");} }

       class ExceptionC extends ExceptionA {
          public ExceptionC (){
             super("ExceptionC");}  }
```

Output:

```
ExceptionB
finally simplefy 100
ExceptionC
finally Test
ExceptionA last catch
```

Q2:**Completethe following code that simulate the *battleship game***

The programcreates an object of type **Ship**with (100,300) as the maximum values for itscoordinates.Then, the ship coordinates **x** and **y** are set. Finally, the program generates randomly***valX***and***valY*** and tries to hit the ship.The program includes 2 user defined exception classes:

- **ShipDestroyed**: is *an Unchecked exception*, thrown **if the ship ishit** .
- **InvalidInput:** is *a checked exception*, thrown **if the value of x or y is negative or greater than x_max or y_max**respectively.

These user defined exceptions are handled in main in addition to any other type of exceptions:

- If ***InvalidInput*** is caught:assign the half of x_max to x and half ofy_max to y .
- If ***ShipDestroyed*** is caught: the exception message is displayed.
- Any other exception, print its message
- In all cases , display the values of x and y, even if the program terminates safely.

Note: complete the code where it's suitable.

```
Class ShipDestroyed extends RuntimeException {
ShipDestroyed() {
super("You have been hit! the ship is destroyed");
}
}
Class InvalidInput extends Exception
{
invalidInput(){
super("the value you entered is invalid");
}
}
```

```
Class Ship {
int x ,y , x_max ,y_max;

ship(intx_max , inty_max){
this.x_max=x_max;
this.y_max=y_max;
}

// the ship is hit if vX == x and vY==y
Public void shipHit (intvX,intvY) {
if (x==vX&& y==vY)
throw new ShipDestroyed()}

// setX will check the value of x before assigning it
Publicvoidsetx(intx) throws  invalidInput {
if (x>x_max || x < 0)
throw new invalidInput();
this.x=x;}

// sety will check the value of y before assigning it
Public voidsetY(inty) throws invalidInput {
if (y>y_max || y < 0 )
throw new invalidInput();
this.y=y;
}
```

```
class test{
public static void main(String[] args){

Random randomGenerator = new Random();
Scanner read=new Scanner(System.in);

Ship usership = new Ship(100,300);// Creates a ship

try{

int x=  read.nextint();
int y = read.nextint();
 usership.setX(x);
 usership.setY(y);
int valX= randomGenerator.nextint(101); //Generates a random number from 0 to 100
int valY= randomGenerator.nextint(301); //Generates a random number from 0 to 300
   usership.shipHit(valX,valY);//Hit the ship
}

catch ( InputMismatchException e) {
System.out.println(e.getMessage());}
catch (InvalidInput e){
usership.setX(100/2);  // or usership.setX(50);
usership.setY(300/2);  // or usership.setY(150);
}
catch (ShipDestroyed e) {
System.out.println(e.getMessage());}
finaly{
System.out.println("value of x " + x +"value of y " + y);}
}}
```

Q1:Trace the following Java program (4.5)

```
public class testExceptions{
static int number=3;
public static int  m1(int a){
try{
if(a%2==0) throw new ExceptionA  ("even !");
if(a++>0) throw new ExceptionA("m1 throws an ExceptionA");
number++;
}catch(ExceptionB  e)
{System.out.println(e.getMessage()+" ** catch m1");}
finally
{ System.out.println("finally m1");
}
return 50;}

public static String m2(){

try{
number = m1(number);
if(number >0) throw new ExceptionB("more than zero");
return "try";
}
catch(ExceptionB  exp)
{ System.out.println(exp.getMessage()+" ** catch1 m2");
return "catch1";}

catch(ExceptionA  exp)
{ System.out.println(exp.getMessage()+" ** catch2 m2");
return "catch2";}

finally{ System.out.println("finally m2 "+ number); }
}
```

```
public static void m3(){

try{
int num=Integer.parseInt(m2());
}catch (NumberFormatException exp)
{System.out.println("non-numeric String  ");}
}

public static void main(String args[]){
try{m3();}
catch(Exception e){ System.out.println("last catch");}}}

class ExceptionA  extends RuntimeException  {
   public ExceptionA (String m){
      super(m);} }
 class ExceptionB  extends ExceptionA {
  public ExceptionB (String m){
      super(m);} }
```

Output:

finally m1 (1)
m1 throws an ExceptionA ** catch2m 2 (1)
finally m2 3 (1
non-numeric string (1

Q2: Complete code (5.5)

Complete the following program that will read from user a 4 digit integer number as his/her year of birth. The program include 2 user defined exception classes

E1: is a checked exception, thrown if year entered is less than 4 digits Y < 1000

E2: is a checked exception, thrown if year entered is less than 1900 Y < 1900

These user defined exceptions are to be handled in main in addition to any other type of exception

- **If E1 is caught** : fix the year to make it 4 digits then test it if the fixed number in range (**>= 1900**).
- **If E2 is caught** : the exception message is displayed.
- **In any case**, display the year entered.
- finally

```
// method that will test if year entered is 4 digit number
public void test_year(int age) throws E1
{
If( x < 1000)
throw new E1("num should be >1000");
}
```

```
Class E1 extends Exception{
public E1(String description){
super(description);
}}
```

```
// method that will read year from user then call method test_year to check if it is 4 digit number
public int get_year() throws E1
{
Scanner read=new Scanner (System.in);
int x = read.nextInt();
test_year(x);
}
```

```
Class E2 extends Exception{
public E2(String description){
super(description);
}}
```

```
public static void main(String [] args) {
try {
int age = get_year ();
}
Catch ( InputMismatchException E3 )
{system.out.println("enter a correct value");}

Catch ( E1 e1 ){
   age=age+1000;
   try{
      if (age < 1900)
      throw new E2 ("num should be >=1900")
   }
Catch ( E2 e2 )
{System.out.println(e2.getMessage());}

} finally { System.out.println("Year is :" +age );}

}
```

← from get_year (int x = read.nextInt())

```
public class Bakery {
	private char t;
	private int p;

	Bakery(char t) {
		if ((t != 'M') && (t != 'B'))
			throw new RuntimeException("Unable to create the object");
		this.t = t;
		System.out.println("In the constructor, t= "+this.t);

	}

	public void setP(int p, int minP) throws Exception {
		try {
			if (p < minP)
				throw new UserException("Limit violation.");
			if (p >= 3 * minP)
				this.p = p / minP;
			else
				this.p = p;
		} catch (ArithmeticException e) {
			System.out.println("ArithmeticException in setP: " +
e.getMessage());
			this.p = 1;
		} catch (RuntimeException e) {
			System.out.println("RuntimeException in setP: " +
e.getMessage());
			this.p = minP;
			throw new RuntimeException("Runtime Exception in setP");
		} finally {
			System.out.println("Finally of setP, p = "+this.p);
		}
		System.out.println("End of setP");

	}

}
```

```
public class UserException extends RuntimeException{
   public UserException(String m) {
      super(m);
   }
}
```

Input	Trace
1 1 M	In the constructor, t= M Finally of Method insert Inserted:true Finally of setP, p = 1 End of setP 1 objects in the array
1 1 K	Catch 5: Unable to create the object 0 objects in the array
1 3 M B B	In the constructor, t= M Finally of Method insert Inserted:true In the constructor, t= B Finally of Method insert Inserted:true In the constructor, t= B Catch inside method insert Finally of Method insert Catch 4: Limit exceeded 2 objects in the array
6 1 M	In the constructor, t= M Finally of Method insert Inserted:true RuntimeException in setP: Limit violation. Finally of setP, p = 6 Catch 5: Runtime Exception in setP 1 objects in the array
0 1 B	In the constructor, t= B Finally of Method insert Inserted:true ArithmeticException in setP: / by zero Finally of setP, p = 1 End of setP 1 objects in the array

King Saud University
College of Computer and Information Sciences
Computer Science Department

CSC 113 **Second Semester 1436-1437**

Quiz 3

Student Name	Student ID	Section Number	Serial Number

```
import java.util.*;
public class Test {
	static Bakery[] l = new Bakery[2];
	static int cpt;

	public static boolean insert(Bakery b) {
			try {
			l[cpt] = b;
			cpt++;
			return true;
		} catch (RuntimeException e) {
			System.out.println("Catch inside method insert");
			throw e;
		} finally {
			System.out.println("Finally of Method insert");
		}
	}

	public static void main(String[] args) {
		Scanner scan = new Scanner(System.in);
		try {
			int minP = scan.nextInt();
			int nbInp = scan.nextInt();
			for(int i=0;i<nbInp;i++){
				Bakery b = new Bakery(scan.next().charAt(0));
				boolean temp =  insert(b);
				System.out.println("Inserted:" + temp);
			}
			for (int i = 0; i < cpt; i++)
				l[i].setP(i + 1, minP);

		} catch (NumberFormatException e) {
			System.out.println("Catch 1: NumberFormatException");
		} catch (InputMismatchException e) {
			System.out.println("Catch 2: InputMismatchException");
		} catch (ArithmeticException e) {
			System.out.println("Catch 3: ArithmeticException");
		} catch (IndexOutOfBoundsException e) {
			System.out.println("Catch 4: Limit exceeded");
		} catch (Exception e) {
			System.out.println("Catch 5: " + e.getMessage());
		} finally {
			System.out.println(cpt + " objects in the array");
		}
	}
}
```

King Saud University
College of Computer and Information Sciences
Computer Science Department

First Semester 1432-1433 **CSC 113**

QUIZ # 2

Student Name:	**Serial Number:**
Student Id:	**Section Number:**

Question 1: [3 marks] What is the output of the following Java code?

```
public class Quiz2 {
public static void f1() throws Exception {
int a=100, b=200;
System.out.println("1");
try {
    System.out.println("2");
    f2(a,b);
    System.out.println("3");
}
catch (Exception e) {
   System.out.println("4");
   throw e;
}
finally {
    System.out.println("5");
}

System.out.println("6");
}

//-----------------------------------------------
---------
public static void f2 (int x, int y) throws Exception {
      if (x<y) throw new Exception();
}
//-----------------------------------------------------------
public static void main(String s[]) {
try {
   f1();
}
catch (Exception e){
     System.out.println ("Exception in method f1()caught in main");}
}}
```

Output

1 ½ pt
2 ½ pt
4 ½ pt
5 ½ pt
Exception in method f1() caught in main 1 pt

Question 2: [7 marks] 10 students in a class deserve a bonus grade. You are given a text file **"Student_Data.txt"**, where each line in the file includes the student's first name, last name, current grade and bonus grade. For example the first line of the file will look like this:

Sara Ali 73.5 2

Assume you have the following class declaration

```
public class Student implements Serializable {
  public String firstName, lastName;
  public double grade; }
```

Complete the program below such that it will do the following:

1. Using a **Scanner** object, read the data of each student from the file **"Student_Data.txt"**, and store it in an object of type **Student**. The **`grade`** of the Student that you will store in the object should be the new **`grade`** (after adding the bonus).
2. Store each **Student** object in a binary object file called **"Student_File.obj"**.
3. Declare an array of 10 students called **studArray**. Read the data of the students from the object file **"Student_File.obj"**, and store it in the array.

```
import java.util.*;
import java.io.*;

class Student implements Serializable{
public String firstName, lastName;
public double grade;

}
public class Quiz2FileTest{
public static void main(String [] args) throws IOException,  ¼ pt
ClassNotFoundException  ½ pt {

Scanner scanner= new Scanner (new File("Student_Data.txt")); ½ pt

File studFile= new File("Student_File.obj"); ½ pt

FileOutputStream outFileStream = new FileOutputStream (studFile);½ pt
ObjectOutputStream outObjStream= new ObjectOutputStream (outFileStream);  ½ pt


for (int i=0; i<10; i++){ ¼ pt

      Student obj= new Student(); ¼ pt
      obj.firstName= scanner.next(); ¼ pt
      obj.lastName= scanner.next(); ¼ pt
      obj.grade= scanner.nextDouble()+scanner.nextDouble(); ½ pt
      outObjStream.writeObject(obj); ½ pt
}

Student [] studArray= new Student[3];


FileInputStream inFileStream = new FileInputStream (studFile); ½ pt

ObjectInputStream inObjStream= new ObjectInputStream (inFileStream); ½ pt

for (int i=0; i<10; i++){ ¼ pt

    studArray[i]= (Student) inObjStream.readObject(); 1 pt
}
}
}
```