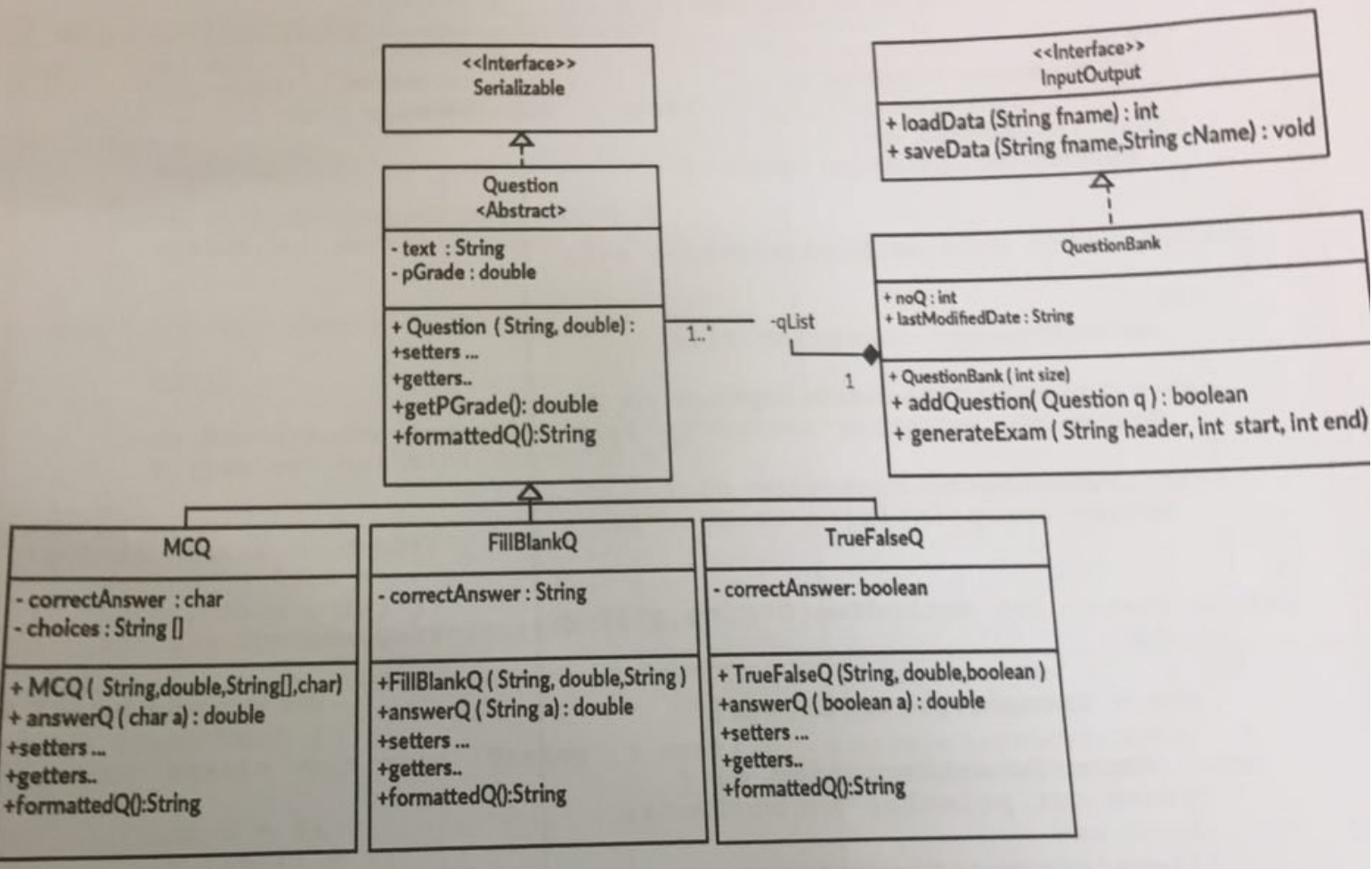
Mid2 (37)

Question#2: Code Question *(13 pts.)*

Consider the following UML and the corresponding method descriptions.



The above UML simulate online exam for given course, Question Bank contains list of questions of different type.

Interface InputOutput

- **void saveData (String fname,String cName):** save objects to a file, possibly throw exception
- **int loadData(String fname):** read objects from file, possibly throw exceptions

Class Question

- String text: question text
- double pGrade: possible grade
- String **formattedQ():** Returned formatted string depends on the type of question
- double **getPGrade():** return possible grade for question

Question [] **qList**: array of question

boolean **addQuestion**(Question q)

- Add question to qList and return true, False otherwise

int loadData(String fname)

- String *fname:* String contains the file name where data should be read from.

- This method reads object by object from object file ***fname*** and adds them into the array **qList**. The method ends when no more object left for reading. Then returns the number of objects successfully added from the file.
- **Hint**: Use the method **addQuestion**

- **void saveData (String fname,String cName)**
  - String *fname:* String contains the file name where data should be saved.
  - String *cName*: String contains the type of question should be written in to files.

  - This method writes object by object in object file ***filename*** all questions that are of type cName. If cName is an empty string then all questions should be written into the file

- **void generateExam (String header, int start, int end):**
  - String *header*: String contains the exam information and it should appear at the beginning of the file.
  - int *start*: start index
  - int *end*: end index

  - This method writes into a **text file** named "***Exam.txt***" the header of exam. Then write **set of questions** starting from the specified index (start) till the specified index (end) of the **qList**. At the end, the total **grades** of the exam should be written into this file. In case of any RunTimeException "*** Question Error***" will be written instead.
  - **Example**:
  - if **start** =2, **end** =5 and **noQ** =10 then header, questions at index 2, index 3, index 4,index 5 from qList and total grades will be written into the file.
  - if **start** =2 and **end** =5 **noQ** =5 then header, questions at index 2, index 3, index 4, from qList and "*** Question Error***" and total grades will be written into the file.
  - **Note**: 1) Use the method **formattedQ()** to write the question in the file in proper format. 2) start and end contains any integer value but start <=end

KING SAUD UNIVERSITY COLLEGE OF COMPUTER AND INFORMATION SCIENCES COMPUTER SCIENCE DEPARTMENT		
CSC 113: Computer Programming II	Midterm 2 (Duration: 1:45)	2nd Semester 1438-1439

Answer Sheet

Student Name (Arabic)	Student ID	Section Number	Serial Number

Question#1: Multiple Choice Questions *(7 pts.)*

For each statement there is a list of options. Choose the option that would be the valid one.

(1)	(2)	(3)	(4)	(5)	(6)	(7)

(8)	(9)	(10)	(11)	(12)	(13)

Question#2:

1. Write the interface InputOutput. (*3 pts*)

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

2. Write the following methods from QuestionBank class:

a) loadData(String fname) *(4 pts)*

public void loadData(String fname).....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

b) saveData (String filename,String cName) *(2 pts)*

public void saveData (String fname,String cName).....
{
FileOutputStream fout=new FileOutputStream(new File(filename));
.....
if (cName.equals("")) //empty string
for (int i=0;i<noQ;i++)
.....
else
for (int i=0;i<noQ;i++){
.....
.....
}.....}

c) generateExam (String header, int start, int end): (*4 pts*)

public void generateExam (String header, int start, int end)

Question#2: Consider the method *getBonus* below:

```
public class Employee
{
   private double salary;
   ......
   public void getBonus (String bonus)
   {
      for ( int i = 0; i < bonus.length(); i++)
         if (! Character.isDigit(bonus.charAt(i)))
         {
            System.out.println("Salary is not numeric. Cannot set it!");
            System.exit(1);
         }
      salary += Double.parseDouble(bonus);
   }
}
```

The method *getBonus* in the class *Employee* accepts the bonus as a string and adds it to the salary. As you know, throwing an exception is a much better approach than terminating the program. Modify the method *getBonus* to handle the exception thrown and repeatedly allow the user to re-enter the bonus again until a valid bonus is entered.

```
public void getBonus (String bonus)
   {
      Scanner input = new Scanner (System.in);
      ..........
      ..........
      ..........
            salary += Double.parseDouble(bonus);
            ..........
      ..........
            System.out.println("Salary is not numeric. Cannot set it!");
            ..........
}
```

Question#2:

1. **Write the interface InputOutput.** (*3 pts*)
2. **Write the following methods from QuestionBank class:**
   a) **loadData** (*4 pts*)
   b) **complete implementation of method saveData** (*2 pts*)
   c) **generateExam** (*4 pts*)

Question 1

1. What is the output of the following code (if any)?

```
public class Test {
   public static void main (String [] args) {
      try {
         return;
      }
      finally {
         System.out.print("finally");
   } } }
```

(a) Finally
(b) It compiles successfully but no output
(c) It causes a compilation error

2. The following code will ...

```
public class Test {
   public static void main (String [] args)
{
        myMethod(200);
   }
   public static void myMethod (int n)
throws
   BigNumberException
   {
      if (n > 100)
         throw new BigNumberException();
      System.out.println(n*2);
   }
}
class BigNumberException extends Exception
{
/*Constructor*/
}
```

(a) throw BigNumberException
(b) print 400
(c) print 400 and throw BigNumberException
(d) cause a compilation error

3. Primitive data types cannot be written to an object file.

(a) Yes
(b) No

4. No conversion for a specific data type is needed when the Scanner class is used to read from a text file.

(a) Yes
(b) No

5. What is the output of the following program?

```
public class Test {
{
    public static void main (String [] args)
        try {
            int i, sum;
            sum = 10;
            for (i = -2; i < 1 ; i++) {
                sum = (sum / i);
                System.out.print(sum + " ");
            }
        }
        catch(ArithmeticException e) {
            System.out.print("0");
        }
}
```

(a) -5, 5,
(b) -5, 5, 0
(c) 0
(d) No output

6. What is the output of the following program?

```
public class Test {
   public static void main (String [] args) {
      try {
         method( "hi" );
         System.out.println( "after method" );
      }
      catch (NumberFormatException ex) {
         System.out.println( "main" );
      }
   }
   public static void method(String val) {
      try {
         int i = Integer.parseInt(val);
      }
      finally {
         System.out.println( "finally" );
      }
      System.out.println( "Bye" );
   }
}
```

(a) finally
main

(b) finally
bye
main

(c) finally
main
after method

d) finally
bye
main
after method

7. The following code will ...

```
import java.util.*;
public class Test {
   public static void main (String [] args) {
         int i = 0;
         method(i);
   }
   public static void method(int n) {
      if (n == 0)
         throw new ArithmeticException();
      else if (n < 0)
         throw new IllegalArgumentException();
}
```

(a) compile successfully
(b) cause a compilation error (missing throws clause in method's header)

is the output of the following program (if any)?

```
public class Test {
    public static void main (String [] args) {
        try {
            method();
            System.out.println( "end" );
        }
        catch (StringIndexOutOfBoundsException e)
        {
            System.out.println( "catch in main" );
        }
}
public static void method() {
    try {
        String str = "Hello" ;
        System.out.println(str.charAt(6));
    }
    catch (IndexOutOfBoundsException e)
    {
        System.out.println( "catch in method" );
} } }
```

(a) catch in method
(b) catch in main
(c) catch in method
end
(d) catch in main
end

output of the following program?

```
public class Test {
   static int i = 5;
   public static void main (String [] args) {
      try {
         i = myMethod();
      }
      catch (Exception e) {
         i--;
      }
      System.out.println(i);
   }
   public static int myMethod() throws
Exception {
      try {
         if (i >= 5)
            throw new Exception();
         i++;
         return i;
      }
      catch (RuntimeException e) {
         i = 0;
      }
      finally {
         i = i * 2;
      }
      return i; } }
```

(a) 0
(b) 9
(c) 10
(d) 11

10. Java allows a class to implement more than one interface.

(a) Yes
(b) No

What is the output of the following program (if any)?

```
import java.util.*;
public class Test {
   public static void main (String [] args) {
      try {
         m1();
      }
      catch (InputMismatchException e) {
         System.out.println( "catch in main" );
      }
   }
   public static void m1() {
      try {
         m2();
      }
      catch(NumberFormatException e) {
         System.out.println( "catch in m1" );
      }
   }
   public static void m2() {
      String [] list =
      { "Love" , "Java" , "(2)" };
      int x = Integer.parseInt(list[4]);
} }
```

(a) catch in m1
(b) catch in main
(c) No output, it causes a compilation error
(d) No output, exception handled by system

... following code will ...

```
public class Test {
    public static void main (String [] args) {
        try {
            int i = 0;
            if ((5/i) > 1)
                throw new ExcepOne();
            if (i >= 0)
                throw new ExcepTwo();
        }
        catch (ExcepOne e) {
            e.printStackTrace();
        }
        catch (ExcepTwo e) {
            e.printStackTrace();
        }
    }
}
class ExcepOne extends Exception {
/*constructor*/
}
class ExcepTwo extends Exception {
/*constructor*/
}
```

(a) throw ExcepOne
(b) throw ExcepTwo
(c) throw ArithmeticException ✓
(d) not throw any exception

13. What is the output of the following program?

```
public class Test {
   public static void main (String [] args)
{
      try {
         methodOne( );
      }
      catch (Exception e) {
         System.out.println( );
      }
   }
   public static void methodOne(String str)
   {
      try {
         methodTwo(str.substring(2));
      }
      catch(IndexOutOfBoundsException e) {
         System.out.println( );
      }
      catch(NumberFormatException e) {
         System.out.println( );
      }
   }
   public static int methodTwo(String str) {
      int num;
      try {
         num = Integer.parseInt(str);
      }
      catch(NumberFormatException e) {
         System.out.println( );
         throw new
         IllegalArgumentException();
      }
      return num;
   } }
```

(a) methodOne
(b) methodTwo
(c) methodTwo
main
(d) methodTwo
methodOne
main

```
import java.io.*; //0.25
public interface InputOutput //.25
{
   public void saveData(String fname) throws
IOException;
//1
   public int loadData(String fname) throws
IOException, ClassNotFoundException; //1.5
}
```

```
public int loadData(String fileName) throws IOException, ClassNotFoundException
{
// declaring stream --> 1
  File F= new File(fileName); 0.25
  FileInputStream FIS = new FileInputStream (F); 0.25
  ObjectInputStream OIS = new ObjectInputStream(FIS); 0.5
  int cpt=0; .25
  boolean flag= true;
  while (flag){ 25
  try{ .25
    Question q=(Question)(OIS.readObject()); 0.5
  if (addQuestion(q))0.5
      cpt++ ; 0.25
  }catch(EOFException e1) .25
  {flag= false;.25
    }
  }
  OIS.close();0.25
  return cpt;0.25
  }
```

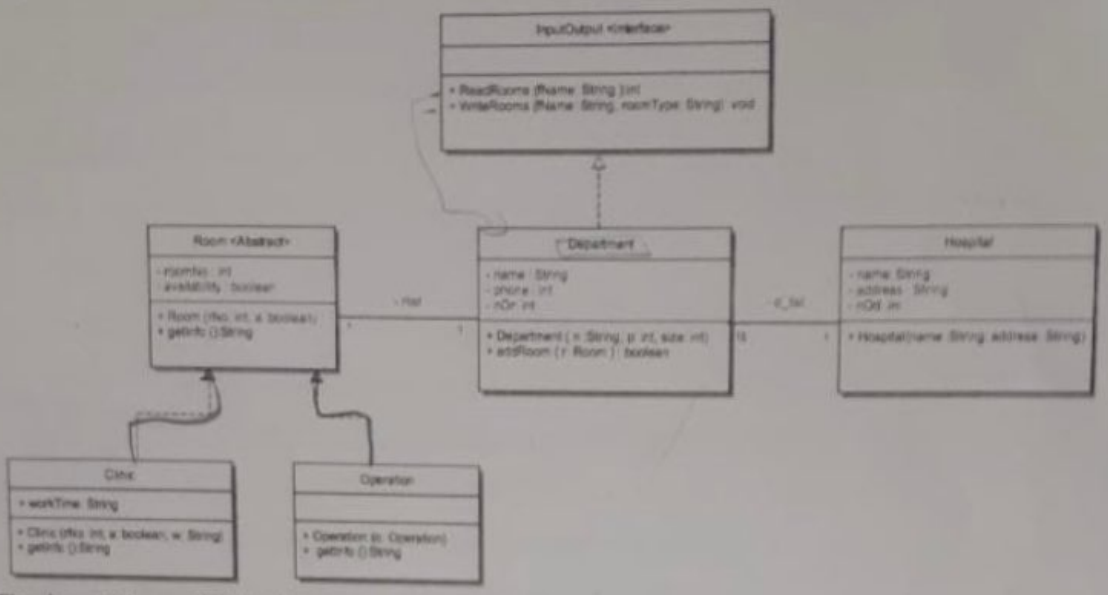
```
public void saveData (String fname,String cName) throws IO
{
FileOutputStream fout=new FileOutputStream( new File(file
ObjectOuputStream Oos = new ObjectOuputStream (fout);
if (cName.equals("")) //empty string
   for (i=0;i<noQ;i++)
     Oos.writeObject(qList[i]);  //0.25
else
   for (i=0;i<noQ;i++){
    if(qList[i].getClass.getName().equals(cName)) 0.5
    Oos.writeObject(qList[i]);  0.25
    }
Oos.close();  0.5
}
```

```
void generateExam (String header, int start, int end)throws IOException .25
{
        File file = new File("Exam.txt"). 25
        PrintWriter pw = new PrintWriter( new FileOutputStream(file)).5
        double total=0; 25
        pw.println(header); 0.25
        for (int i = start;i<=end;i++).5
        {
        try{0.25
        Question q=qList[i];

        pw.println(q.formattedQ()).5
        total+=q.getPossibleGrades();.25

        }
       catch (RunTimeExcept:n ex) 25
        {
         pw.println("*** Question Error***");.25

        }
}
 pw.println ("Total grades" + total);0.25
pw.close(). 0.25

}
```

Mid2 (38)

Name:

Hospital System UML for Question #3



The descriptions of UML classes:

Department class:

- Attributes:
  - ***nOr***: number of rooms in the department.
- Methods:
  - ***addRoom(r:Room):boolean:*** this method adds a room (r) to the department *if possible* and returns true. Otherwise, the method will display an appropriate message and return false.

  - ***ReadRooms (String fName):int***
    This method reads Room objects from an **object file** ***fName*** and **adds** them to the department (Hint: add to array ***rList***). The rooms should be added in a specific order. First all clinic rooms should be added, then followed by all operation rooms.

    The method **returns** the number of Room objects successfully added from the file. When the end of file is reached, an appropriate exception is caught in order to close the file.

  - ***WriteRooms (String fName, String roomType): void***
    This method writes all Room objects of a specific type ***roomType*** to a **text file** ***fName***.
    - If ***roomType*** is "Clinic" then all clinic rooms should be written.
    - If ***roomType*** is "Operation" then all operation rooms should be written.
    - If ***roomType*** is "All" then all rooms should be written.
    - If the ***roomType*** is none of the above, an appropriate exception with the message "Invalid Room Type" should be thrown.

    Use the following format:
    - each two consecutive lines correspond to one Room.
    - If the room is an operation, the method writes:
      roomNo
      Operation room available? ***availability.***
    - If the room is a clinic, the method writes:
      roomNo
      Clinic room available? ***availability*** , at time: ***worktime.***
    - Where ***roomNo***, ***availability*** and ***workTime*** are replaced by their values.

Name:

Room class:

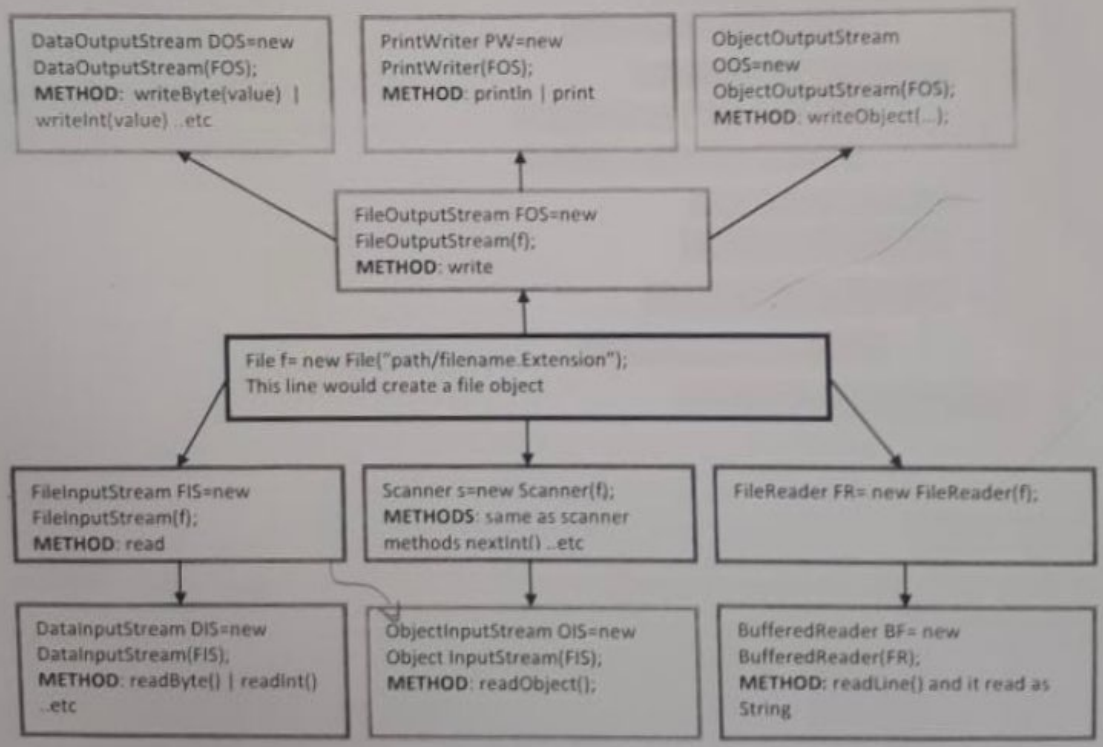
- Attributes:
  - ***roomNo***: the room number.
  - ***availability:*** indication if the room is available to receive a patient or not.
- Methods:
  - ***getInfo(): String:*** returns a String formed of:
    - Operation room available? ***availability.***

    or
    - Clinic room available? ***availability*** , at time: ***worktime.***

Where ***availability*** and ***workTime*** are replaced by their values.

Some Helpful Information to use in the exam:

A. Files and Streams



	Q1	Q2	Q3								TOTAL
	10.5	3	16.5								30
SCORE	10.5	2.5	16								29

excellent!

Question #1: Trace *(10.5 pts)*

A. Trace the code and answer the questions for the cases of nums[]:

```
1 import java.util.*;
2 public class MidTestQ1A {
3 public static void main(String[] args) {
4   int nums[] = {?? ?? ?? ??};
5   int num;
6   try {
7     System.out.println("trying");
8     validateNum(nums[0]);
9     for (int i = 1; i <= 4; i++) {
10      num = 50 / nums[i];
11      System.out.println(num);
12    }
13    System.out.println("Succeeded");
14  }
15  catch (IndexOutOfBoundsException e) {
16    System.out.println("Caught e!");
17  }
18  catch (NullPointerException e) {
19    System.out.println("Caught null!");
20  }
21  System.out.println("finally done");
22 } //main
23 static void validateNum(int num){
24   System.out.println("checking: " + num);
25   if (num == 0)
26     throw new NullPointerException();
27   System.out.println("Valid");
28 } }// validateNum, class MidTestQ1A
```

a. What exception will be thrown?
b. At which line?
c. Will it be caught by main? (Yes / No)
   if not, what happens?
d. What is the output of main?

	1. int nums[] = {0,5,10,50};	2. int nums[] = {50,5,0,-5};	3. int nums[] = {10,10,10,10};
a.	NullPointerException	ArithmeticException	ArrayIndexOutOfBoundsException
b.	26	10	10
c.	yes.	No, It will compile successfully but will cause a run time Exception	yes. (ArrayIndexOutOfBound is a subclass of IndexOutOfBo
d.	trying checking: 0 Caught null finally done	trying valid checking 50 10 5 Stack Trace for ArithmeticException (Divide by zero)	trying checking: 10 valid 5 5 5 Caught e! finally done

Name:

B. What is the output of main for the three cases :

Assume: `NotMeException extends runtimeException`
`NotDoubleException extends IOException`

4.25

```
import java.util.*; import java.io.*;
public class MidTestQ1C {
public static void main(String[] args){
try {
  System.out.println("trying");
  writeMe(?? ?? ??);
  System.out.println("Succeeded");
}
catch (Exception e) {
  System.out.println("exception");
  System.out.println(e.getMessage());
}
finally {
  System.out.println("finally");
}
System.out.println("Done!!");
} // main
public static void writeMe(String s)
throws Exception {
  String sub = s.substring(0,2);
  if (sub.equals("Me")){
    System.out.println(s);
    writeDouble(s);
  }
  else
    throw new NotMeException ("Not me");
  System.out.println("Me Done");
} // writeMe
public static void writeDouble(String d)
throws Exception {
  int mid = d.length()/2;
  String half1=d.substring(0,mid);
  String half2=d.substring(mid,d.length());
  if (half1.equals(half2))
    System.out.println("double");
  else
    throw new NotDoubleException
                ("Not double");
  System.out.println("Double Done");
}} // writeDouble // class MidTestQ1C
```

Substring method of Strings

- ***str.substring (int beginIndex, int endIndex)***

This method extracts a substring from ***str*** starting from the position ***beginIndex*** (the first parameter) till ***endIndex***-1 (the second parameter).

1. `writeMe("Meow");`

Trying
Meow
exception
Not Double
Finally
Done!

2. `writeMe("WuffWuff");`

Trying
exception
Not me
Finally
Done!

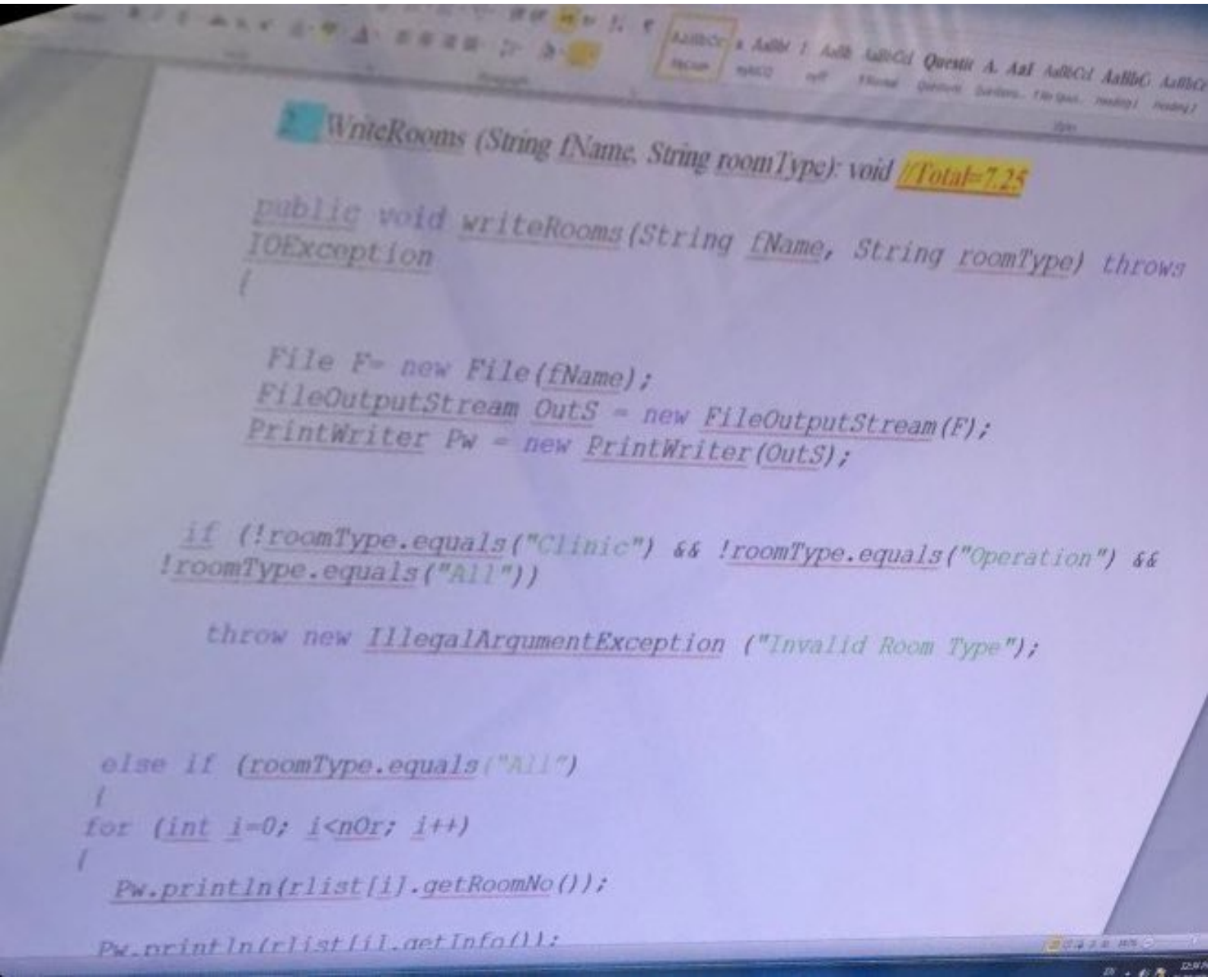
3. `writeMe("MeMe");`

Trying
MeMe
double
Double Done
Me Done
Succeeded
finally
Done!

Note: It is important that you make use of existing methods.

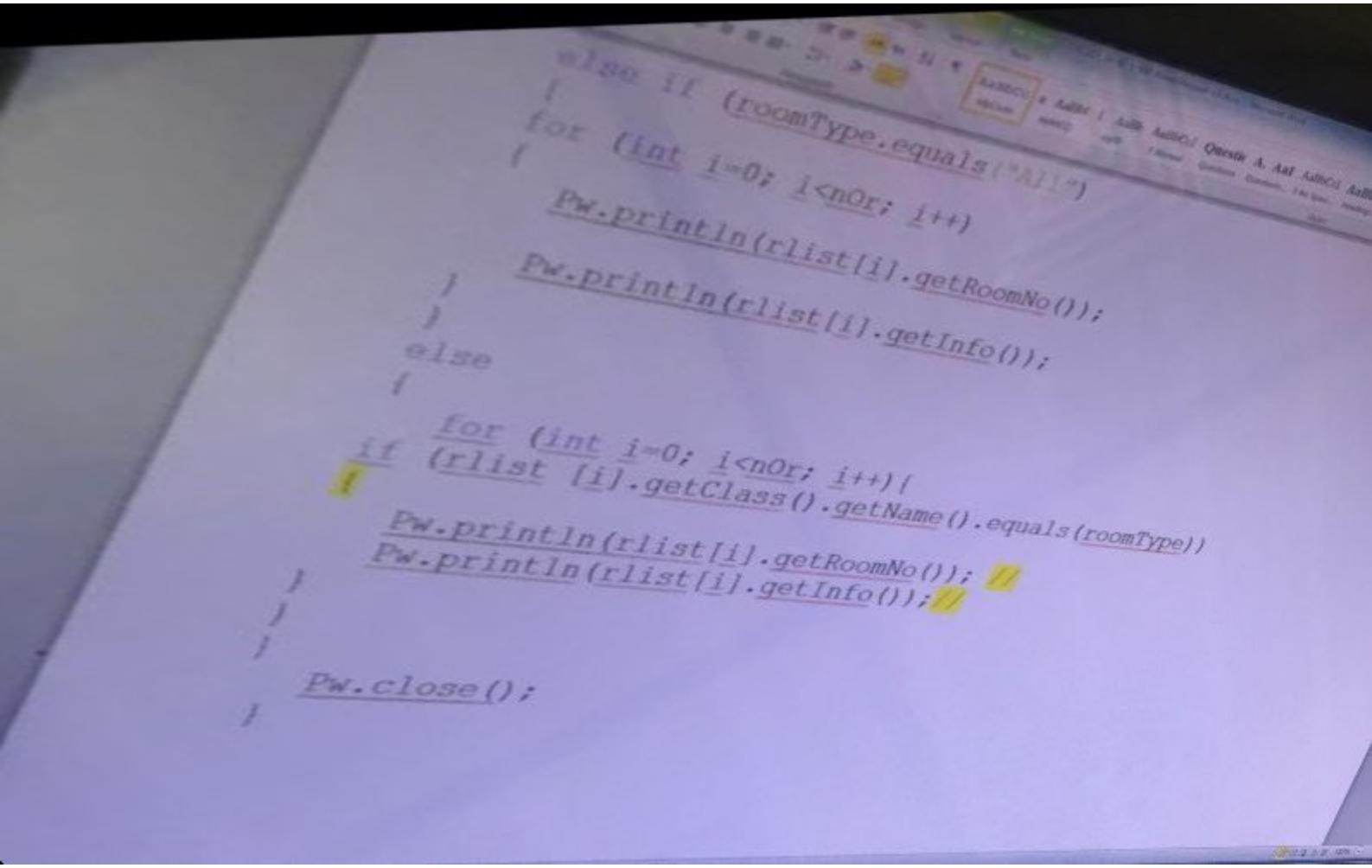
1. ReadRooms (String fName):int //Total 9.25

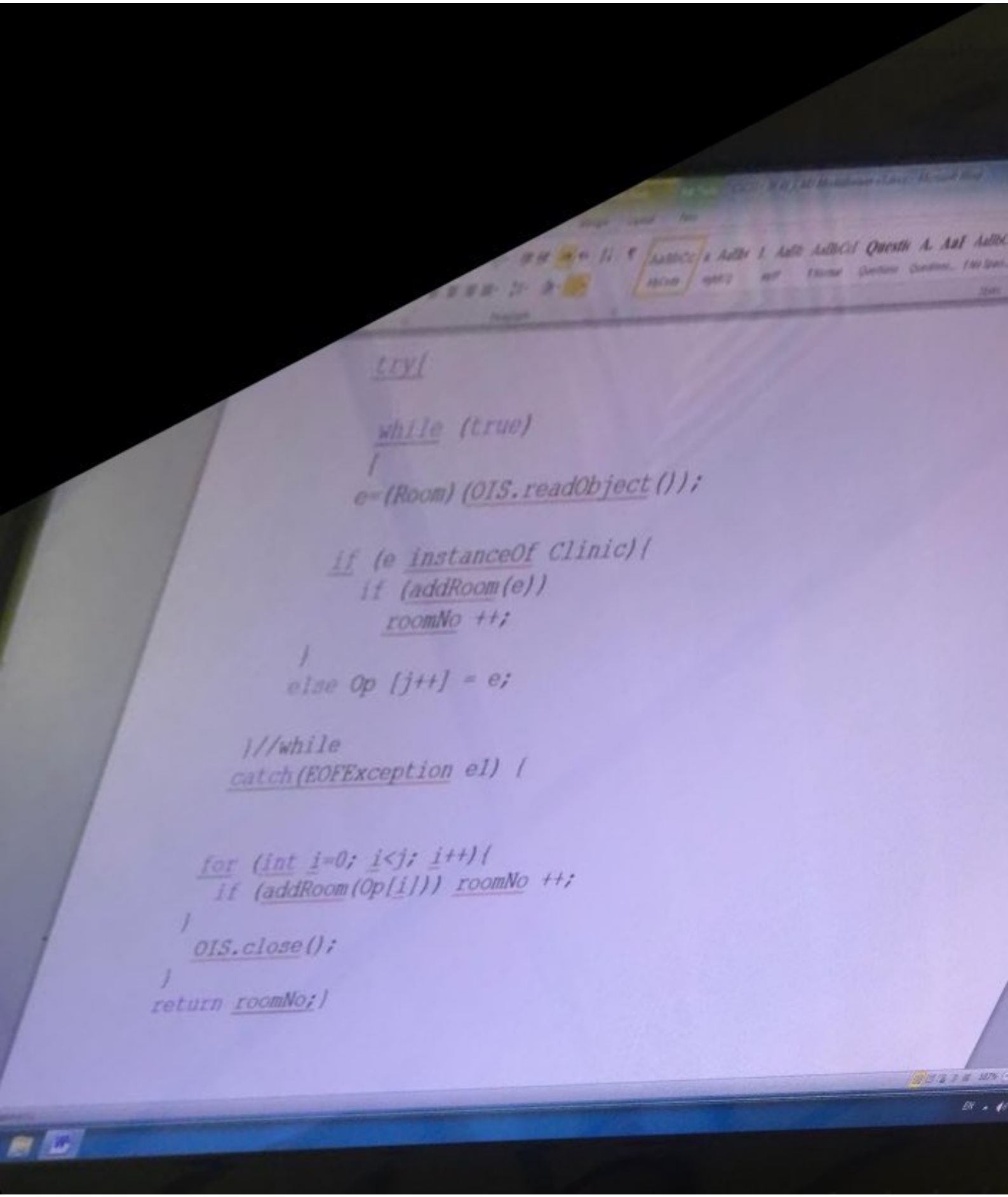
```
public int readFromFile(String fileName) throws IOExceptio
ClassNotFoundException

  File F= new File(fileName);
  FileInputStream FIS = new FileInputStream (F);
  ObjectInputStream OIS = new ObjectInputStream(FIS);

 int roomNo=0;
 int j=0;
 Room e;
 Operation [] Op = new Operation [rlist.lenght()-nOr];

try{

 while (true)
 {
e=(Room)(OIS.readObject());

if (e instanceOf Clinic){
  if (addRoom(e))
    roomNo ++;
```



writeRooms (String fName, String roomType): void //Total=7.25

```
public void writeRooms(String fName, String roomType) throws
IOException
{

 File F= new File(fName);
 FileOutputStream OutS = new FileOutputStream(F);
 PrintWriter Pw = new PrintWriter(OutS);


  if (!roomType.equals("Clinic") && !roomType.equals("Operation") &&
!roomType.equals("All"))

    throw new IllegalArgumentException ("Invalid Room Type");


 else if (roomType.equals("All")
 {
for (int i=0; i<nOr; i++)
{
  Pw.println(rlist[i].getRoomNo());

 Pw.println(rlist[i].getInfo());
```





Name:

Question #2: Find the errors *(3 pts)* 2.5

Consider the two programs and find 2 errors in each and correct them:

1.

```
public class MidQ2-1 {
public static void main (String [] args) {
      try {
        String str ="";
        ...
        if (str.equals("Error"))
           throw new ExcepOne();
      }
1—    catch (Exception e) {
         System.out.print (e.getMessage());
      }
      catch (ExcepOne e) {
         e.printStackTrace();
      }
   }//main
}

2 class ExcepOne extends
//Checked user defined exception.
}
```

excellent!

Error 1

Catching Exception befor excepOne will cause ~~Error~~ Compilation Error. [Exception is the super class Exception for all Exceptions] 0.75

Correction: Catch ExcepOne First Then Exception.

Error 2

Class ExcepOne is a checked Exception and java.util.* should be imported. 0.25

Correction: import java.util.*; in The head of The program.

2.

```
public void method2() {
   try {
        int x=5/0;

2       try
        {
          String str ="Java";
          char c = str.charAt(5);
        }
   }
1  catch (Exception e) {
        throw new Exception();
   }

   finally {
   System.out.print("Finally");
   }
```

which is a checked exception

Error 1

The Catch(Exception e) Throws a new Exception. It must be handeled (by another Try Catch) or we must Change The method into a probegator.

Correction: Public void method2 Throws Exception { 0.75

Error 2

The inner Try has neither a nor a Catch

Finally block.

Correction: example:

Try{ ... }

Catch (IndexOutOfBoundsException){ } 0.75

*Or we can remove The inner Try since OutOfBoundsException is an unchecked Exception