****
WRITE YOUR EXAM NUMBER HERE:

****
IMPORTANT READ CAREFULLY
****

1- There are nine questions in this paper. Answer ALL NINE questions.
2- Edit your code in java files in the BlueJ project, NOT in this file
3- SAVE your work regularly. It is your responsibility to save it.
4- DO NOT use JOptionPane in the exam.
5- DO NOT change any of the code apart from what is specified in a question. Failing to comply may result in heavy penalties.
6- DO NOT IMPORT additional packages.
7- NEVER change method names and the type and order of parameters specified in the question: if the question asks to write a method with signature
public void Foo(int n, double d)
then writing: public void foo(int n, double d) will be wrong (f should be capital) and public void Foo(double d, int n) will also be wrong (parameter int should be before parameter double)
8- Check carefully that the output of your program is EXACTLY the one required by the exercises. For example if asked to print a string s (e.g. "Wrong argument"), print exactly that string i.e. use System.out.println("Wrong argument"), do not modify the output string.
9- NEVER write two classes in the same java file.
10- If a question provides some code to test your code, to pass those tests is not a guarantee that you will get marks;
the tests usually only guarantee that you have made no syntax errors
11- java api documentation is available by clicking on the browser icon.
The Appendix at the end of this file contains help for some methods that may help in the exam.
12- ONLY use System.exit(0) if the question says so. Never use System.exit with any other argument
e.g. NEVER use System.exit(-1)

****
Question 1 [6 Marks]
****

In the following class some of the instance variables are wrongly declared:
Modify only the types in declarations so that the class will compile
(do not modify the name of the variables or their initial values):

```
public class Ex1 {

private Student y='h';     //modify only the type
private float x="hello";      //modify only the type
```

```
private Account c =false;    //modify only the type
private int s ="double";       //modify only the type

   public void getValuesA(){
   System.out.println(x+" "+y+" ");
   }

   public void getValuesB(){
   System.out.println(s+" "+c);
   }

}
```

****
Question 2 [6 Marks]
****

Insert in the following class a constructor that will initialise the instance variables name and status.
The constructor has two arguments: a string personName and an integer mark; name is initialized by personName.
About the instance variable status: if mark is over 40 the constructor initializes the instance variable status to “pass”
otherwise the constructor initializes the instance variable status to “fail”.
For example
the statement (new Ex2("John",40)).getValues() will return "John pass”
the statement (new Ex2("John",15)).getValues() will return "John fail”

```
public class Ex2 {

private String name;
private String status;


   public String getValues(){
      return ""+name+" "+status;
   }
   public static void main(String arg[]){ // you can use the main method to test your code
      System.out.println((new Ex2("John",40)).getValues());
   }

}
```

****
Question 3 [9 Marks]
****

Consider the following Accumulator class with missing code for the method 'getOverM(int m)'.
getOverM should return the last element of the array A if the value of the last element is greater than m and should return -1
if the value of the last element of the array A is less or equal to m.

For example if A is the array {2,4,3,5,8} then
getOverM(3) will return 8
getOverM(2) will return 8
getOverM(11) will return -1
getOverM(30) will return -1

(Hint: the length of an array A is given by A.length)

Insert the code for the body of the method getOverM.

```
public class Accumulator {
   private int[] A;

   public Accumulator(int[] X) {
      A= new int[X.length];
      for (int i=0; i<X.length; i++)
         A[i] = X[i];
   }

   public int getOverM(int m) {

   }

    public static void main(String args[]){ // you can use the main method to test your code

     int[]  A = {2,4,3,5,8};

     int r=new Accumulator(A).getOverM(3); //change argument to test different cases
     System.out.println(r);
   }

}
```

****
Question 4 [7 Marks]
****

Fill in the class method main of class Ex4 with code that multiplies up to three

command line arguments
if these are numbers of type double.
The user should provide at least one command line argument.
If the user provides only one command line arguments then that element should be printed.
If the user provides two or three command line arguments then the product of the command line arguments should be printed.
if the user provides less than one or more than three command line arguments the program should print
"number of arguments invalid" and terminate (you can test for the number of arguments with args.length for this purpose).
If one of command line arguments is not a number the program should print “invalid operation” and terminate (you can
use NumberFormatException for this purpose).

For example
java Ex4 3 will print 3.0
java Ex4 3 2 2 will print 12.0
java Ex4 5 1 will print 5.0
java Ex4 A B C D will print the string “invalid operation”
java Ex4 3 2 2 5 6 will print ”number of arguments invalid"
java Ex4 will print ”number of arguments invalid"

```
public class Ex4 {

  public static void main(String[] args) {

  }
}
```

****
Question 5 [12 Marks]
****

Consider the following class:

```
public class Assessment
{
   private String student;
   private double mark;

   public Assessment(String sstudent, double mmark) {
      student =sstudent;
      mark =mmark;
    }

   public String getStudent(){
      return student;
    }
```

```
    public double getMark(){
      return mark;
    }

    public void setMark(double newmark){
      mark= newmark;
    }
}
```

Write two subclasses of the Assessment class, one called Coursework, the other called FinalExam

The constructor of Coursework has signature
  public Coursework(String sstudent, double mmark, double ppercentage, int ctype)
this constructor will call the superclass constructor on its arguments

The class Coursework has a private instance variable of type double named percentage and initialised to the value of ppercentage.
Coursework has also a private instance variable of boolean type called hurdle.

This variable is initialised in the constructor Coursework as follows:

if ctype has value 0,1 or 2 then hurdle is set to false, else hurdle is set to true.

The class Coursework also has getters for the variables percentage and hurdle called getPercentage and getHurdle respectively.

Hence for example for an instance of Coursework created with
new Coursework("John",50,0.2, 3)
getHurdle() would return true

For an instance of Coursework created with
new Coursework("John",50,0.2, 1)
getHurdle() would return false

About the class FinalExam.

The constructor of FinalExam has signature
  public FinalExam(String sstudent, double mmark, double ppercentage, boolean rresit)
this constructor will call the superclass constructor on its arguments

The class FinalExam has a private instance variable of type double named percentage and initialised to the value of ppercentage.
FinalExam has also a private instance variable of type bool named resit and initialised to the value of rresit.

The class FinalExam has getters for percentage and resits called getPercentage and getResit respectively.

The method getMark() is overriden in FinalExam as follows: if resit is false then it returns mark, however if resit is true
then it returns mark if mark is below 40 and returns 40 if mark is above 40.

Hence for example for an instance of FinalExam created with
new FinalExam("John",50,0.2, true)
getMark() would return 40

For an instance of FinalExam created with
new FinalExam("John",50,0.2, false)
getMark() would return 50

****
Question 6 [13 Marks]
****

In the following program, the code in the method combineAssessments is missing. Write the code for this method, which takes as arguments: ArrayList<Coursework> cw,ArrayList<FinalExam> exam, and returns an ArrayList<Assessment> object, where each student has the marks combined. In this exercise we assume percentage is 0.5, resit is false and ctype is 0.

For example if "John Smith" is a student having coursework mark 60 in cw and exam mark 70 in exam and both coursework and final exam have percentage 0.5
then in combined "John Smith" will have mark 65 (because 0.5*60+0.5*70=65)

For example if cw consists of "John Smith" with mark 60 and "Mary Green" with mark 50, and exam consists of "John Smith" with mark 70 and "Mary Green"
with mark 80 and percentage is 0.5 for all then combineAssessments will return an ArrayList<Assessment> object consisting of the assessment objects
"John Smith" with mark 65 and "Mary Green" with mark 65

The method should also:

- handle the the case of any of the arguments being null. In that case the method should print "Error" and return null

- handle the case of a student being present in one ArrayList but not in the other ArrayList, e.g. a student present in the list cw but not in the list exam.
In that case the method should print "Missing Student" and return null

```
import java.util.*;

public class Ex6 {

    public static ArrayList<Assessment>  combineAssessments
(ArrayList<Coursework> cw,ArrayList<FinalExam> exam) {
        return combined;
```

```
   }

 public static void main(String[] args){ // you can use this main method to test your
code
       ArrayList<Coursework> cw = new ArrayList<Coursework>();
       cw.add(new Coursework("John Smith",60,0.5,0));
       cw.add(new Coursework("Mary Green",50,0.5,0));
       ArrayList<FinalExam> exam = new ArrayList<FinalExam>();
       exam.add(new FinalExam("John Smith",70,0.5,false));
       exam.add(new FinalExam("Mary Green",80,0.5,false));
       ArrayList<Assessment> combined = combineAssessments ( cw, exam);
       for (int i=0; i<exam.size();i++){
          System.out.print(combined.get(i).getStudent()+" "+combined.get(i).getMark()
+" ");
       }
   }
}
```

****
Question 7 [15 Marks]
****

In the class Ex7 write the code for the method readAssessment(String fName). The method will return an ArrayList<Assessment> object
whose elements are Assessment objects which are populated by the names and marks from each line of the file fName.
Each line of the file fName has the following structure:
First-name Family-name mark

As an example, if "data.txt" is
   John Smith 60
   Anna Green 80
then
   readAssessment("data.txt") will return an ArrayList<Assessment> object whose elements are two Assessment objects, the first created with
new Assessment(“John Smith”, 60) and the second created with new Assessment(“Anna Green”, 80)

However,

- if an IOException occurs, the method should print "IO Error" and return null

To test your program, use the file "data.txt" (available in your current directory), each line of which contains three strings as in the above example.

(Hint:
   1) the statement: String[] lines=line.split(" ");
   will put in the array lines the word in the string line separated
   by an empty space, e.g. if line is

"John Doe 2000" then after String[] lines=line.split(" ");
lines[0] is "John", lines[1] is "Doe" and lines[2] is "2000")

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

public class Ex7 {

   public static ArrayList<Assessment> readAssessment(String fName){

   }

}
```

****
Question 8 [12 Marks]
****

In the following program, the code in the method namesFail is missing.
Write the code for this method, which takes as argument the ArrayList of Assessment assessList and returns the names of students
for which getMark() returns a value below 40

```
   public static String namesFail(ArrayList<Assessment>  assessmentList)
```

For example, if assessmentList contains three assessment objects the first with name John and mark 60, the second with name Paul
and mark 30, the third with name Ali and mark 35 then
namesFail(ArrayList<Assessment> assessmentList)
will return the string
“Paul Ali "

```
import java.util.*;
 public class Ex8 {

   public static String namesFail(ArrayList<Assessment>  assessmentList){

   }
  public static void main(String[] args){ //you can use this main method to test your code

       ArrayList<Assessment>  assessmentList= new ArrayList<Assessment> ();
       assessmentList.add(new Assessment("John",60));
       assessmentList.add(new Assessment("Paul",30));
       assessmentList.add(new Assessment("Ali",35));
       System.out.print(namesFail(assessmentList) );
```

```
    }
}
```

****
Question 9 [20 Marks]
****

Consider the following interface

```
public interface SpecialAssessment{
    public void setExtraTime(int time);
    public int getExtraTime();
}
```

a)
Write a subclass of Assessment called SpecialFinalExamAssessment implementing the interface SpecialAssessment.
The constructor SpecialFinalExamAssessment has signature
public SpecialFinalExamAssessment(String sstudent, double mmark, double ppercentage, boolean rresit, int condition)

overall SpecialFinalExamAssessment is the same as FinalExam but has also a getter for the instance variable extratime (called getExtraTime())
whose initial value is determined by the value of the variable condition.
If condition has value 1 or 2 then extratime is set to 10, if condition has value 3 or 4 then extratime is set to 20,
in all other cases extratime is set to 0

(this part is worth 8 marks)

b)
Write a class called FilterSpecialAssessments with a class method with signature

```
    public static String filterSpecial (ArrayList<Assessment> assessList)
```

this method takes the ArrayList<Assessment> assessList and return a string containing the names of the students with extra time.
Hence the method look for elements in the ArrayList<Assessment> assessList which have type SpecialAssessment and if such an object
has extratime >0 the name of that student will be appended to the result

For example if assessList contains an Assessment object with name “John”, a SpecialFinalExamAssessment with name ”Mark" and condition 1, a SpecialFinalExamAssessment with name ”Tom" and condition 3 and a SpecialFinalExamAssessment with name ”Tom" and condition 5 then filterSpecial(assessList) will return the string “Mark Tom”

(this part is worth 12 marks)

********** APPENIDX **********
********** DOCUMENTATION **********

for the java api documentation: click on the browser icon

****
SOME USEFUL ARRAYLIST METHODS
boolean add(E e)
Appends the specified element to the end of this list.
void add(int index, E element)
Inserts the specified element at the specified position in this list.

get(int index)
Returns the element at the specified position in this list.

size()
Returns the number of components in this list.

****
To check if an object is an instance of a class c use the operator instanceof
o instanceof c will return true if o is an instance of c and false otherwise.

****
Integer.parseInt(String s)
Parses the string argument as an integer. Throws NumberFormatException if the string does not contain a parsable integer.

****
% is the Remainder operator. (n % m) returns the remainder of the division of n by m, e.g. 3 % 2 = 1

****
To generate a double random number between 0 and 1 you can use Math.random()

****
To get an integer number that shows the number of elements in an array a, you can use a.length

****
SOME FILE READING CONSTRUCTORS AND METHODS

FileReader(String fileName)
Creates a new FileReader, given the name of the file to read from.

BufferedReader(Reader in)
Create a buffering character-input stream that uses a default-sized input buffer.

String readLine()
Read a line of text.

to read a file until its last line you should check if the line is null, e.g. if the current line of the file is in the String line then you can use

while(line != null) ...

****
SOME USEFUL STRING METHODS

char charAt(int index)
Returns the character at the specified index.

boolean equals(Object anObject)
Compares this string to the specified object.

int indexOf(int ch)
Returns the index within this string of the first occurrence of the specified character.

int lastIndexOf(String str)
Returns the index within this string of the rightmost occurrence of the specified substring.

int length()
Returns the length of this string.

String substring(int beginIndex, int endIndex)
Returns a new string that is a substring of this string.

String trim()
Returns a copy of the string, with leading and trailing whitespace omitted