Student's name:..... **ID:** ..... **Section:** .....

Question 1: What is the output of the following program?

```
public abstract class Sh {
  protected String name;
  public Sh() {
    name = "Sh";
  }
  public abstract float area();
  public String toString() {
    return "Hello";
  }
}
```

```
public class Rec extends Sh {
  float width;
  float length;
  public Rec(String n, float w, float l) {
    if (name == null) name = n;
    width = w;
    length = l;
  }
  public float area() {
    return width * length;
  }
  public String toString() {
    return  "Name:" + name + " Width:" + width +
            " Length:" + length + " Area:" + area();
  }
}
```

```
public class Sq extends Sh {
  float side;
  public Sq(String n, float s) {
    name = n;
    side = s;
  }
  public float area() {
    return side * side;
  }
  public String toString() {
    return  super.toString() + " Side:" + side + " Area:" + area();
  }
}
```

```
public class TestSh {
  public static void main(String[] args) {
    Sh[] sh = new Sh[4];
    sh[0] = new Sq("Sq1", 5);
    sh[1] = new Rec("Rec", 2, 3);
    sh[2] = new Sq("Sq2", 2);
    sh[3] = new Rec("Sq2", 3, 4);
    try {
      for (int i=1; i <= sh.length; i++)
        System.out.println(sh[i].toString());
      }
      catch (ArrayIndexOutOfBoundsException e) {
        System.out.println("know your limits");
      }
    }
  }
}
```

Solution:

6.0

1.5

Name:Sh Width:2.0 Length:3.0 Area:6.0

1.5

Hello Side:2.0 Area:4.0

1.5

Name:Sh Width:3.0 Length:4.0 Area:12.0

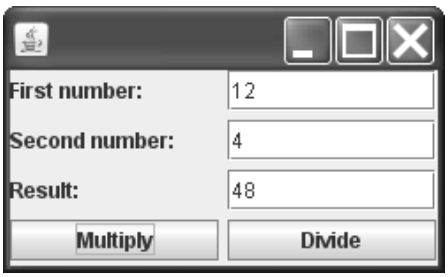
1.5

know your limits

Question 2: Given the class Calculator, which defines two methods: multiply() and divide() to perform the corresponding arithmetic integer binary operations, write a class CalcGUI that provides a graphical user interface to access these two functionalities.

```
public class Calculator {
  public int multiply(int x, int y) {
    return x * y;
  }

  public int divide(int x, int y) throws ArithmeticException {
    return x / y;
  }
}
```

The interface might, but not necessary, look like the window drawn aside.



If the requested operation cannot be done, use exception handling to show the word *"Error"* in the result box.

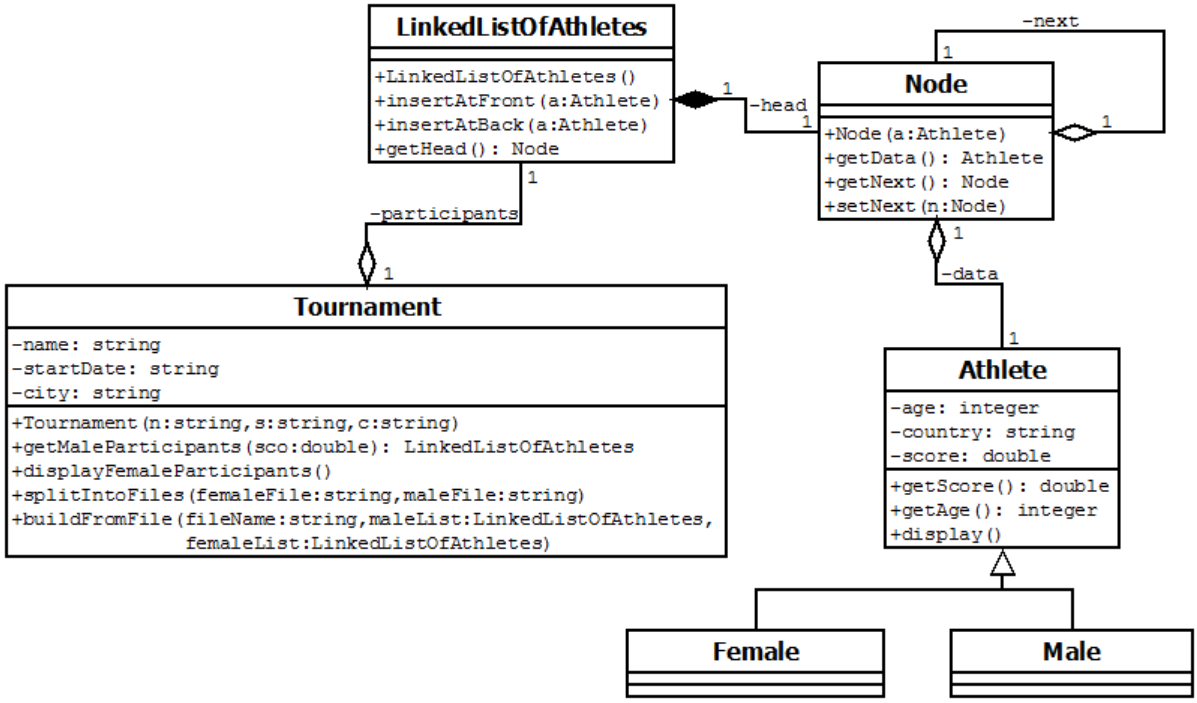
Solution

10.0

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

public class CalcGUI extends JFrame implements ActionListener {

  Calculator calc;

  JLabel lblNum1, lblNum2, lblResult;
  JTextField tfNum1, tfNum2, tfResult;
  JButton btnMul, btnDiv;

  public CalcGUI() {

    calc = new Calculator();                                    0.5

    setSize(250, 150);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    Container cp = getContentPane();                            0.5

    cp.setLayout(new GridLayout(4,2, 5, 5));

    lblNum1 = new JLabel("First number:");                      0.5
    cp.add(lblNum1);

    tfNum1 = new JTextField();                                  0.5
    cp.add(tfNum1);

    lblNum2 = new JLabel("Second number:");                     0.5
    cp.add(lblNum2);

    tfNum2 = new JTextField();                                  0.5
    cp.add(tfNum2);
```

```
        lblResult = new JLabel("Result:");                          0.5
        cp.add(lblResult);

        tfResult = new JTextField();                                0.5
        cp.add(tfResult);

        btnMul = new JButton("Multiply");                           0.5
        cp.add(btnMul);

        btnMul.addActionListener(this);                             0.5

        btnDiv = new JButton("Divide");                             0.5
        cp.add(btnDiv);

        btnDiv.addActionListener(this);                             0.5
      }

      public void actionPerformed(ActionEvent ae) {
        try {

          int x = Integer.parseInt(tfNum1.getText());               0.5
          int y = Integer.parseInt(tfNum2.getText());               0.5

          if (ae.getSource() == btnMul) {
            int z = calc.multiply(x, y);                            0.5
            tfResult.setText(z+"");                                 0.5
          }

          if (ae.getSource() == btnDiv) {
            try {                                                   0.5

              int z = calc.divide(x, y);                            0.5
              tfResult.setText(z+"");                               0.5
            }

            catch (ArithmeticException e) {                         0.5
              tfResult.setText("Err");
            }
          }

        }
        catch (NumberFormatException nfe) {}
      }
    }
```

Question 3: Consider the following UML Class Diagram.



A ***Tournament*** is composed of **many** ***Athletes***. Each ***Athlete*** obtains a ***score*** in the tournament.

We assume that the athletes are inserted in the list in ascending order by age, from youngest to oldest.

Description of the methods of the class Tournament

- ***getMaleParticipants(sco: double) : LinkedListOfAthletes***

  This method returns a list of male athletes who obtained a score greater than **sco**.

- ***displayFemaleParticipants ( )***

  This method displays the list of female participants in descending order by age, from oldest to youngest.

Write the class ***Tournament*** and implement **only** the constructor and the following methods:

- ✓ ***getMaleParticipants***
- ✓ ***displayFemaleParticipants***

Solution:

10.0

```
import java.io.*;
public class Tournament {
```

```
  private String name;                                              0.25
  private String startDate;                                         0.25
  private String city;                                              0.25
  LinkedListOfAthletes participants;                                0.25
```

```
  public Tournament(String n, String s, String c) {
    name = n;                                                       0.25
    startDate = s;                                                  0.25
    city = c;                                                       0.25
    participants = new LinkedListOfAthletes();                      0.25
  }
```

```
  public LinkedListOfAthletes getMaleParticipants(double sco) {
    LinkedListOfAthletes mList = new LinkedListOfAthletes();        0.5

    Node n = participants.getHead();                                0.5

    while (n != null) {                                             0.5

      Athlete a = n.getData();                                      0.5

      if (a instanceof Male && a.getScore() > sco)                  0.5

        mList.insertAtBack(a);                                      0.5

      n = n.getNext();                                              0.5
    }

    return mList;                                                   0.5
  }


public void displayFemaleParticipants() {
    LinkedListOfAthletes reverse = new LinkedListOfAthletes();      0.5

    Node n = participants.getHead();                                0.5

    while (n != null) {                                             0.25

      reverse.insertAtFront( n.getData() );                         0.5

      n = n.getNext();                                              0.5
    }

    n = reverse.getHead();                                          0.5

    while (n != null) {                                             0.25

      n.getData().display();                                        0.5

      n = n.getNext();                                              0.5
    }
  }

  ...
}
```

Question 4: Let's Consider the UML Class Diagram of the question 3.

Description of the methods of the class Tournament

- ***splitIntoFiles(femaleFile: String, maleFile: String)***

  This method receives two object file-names as parameters. It saves the female-athletes objects into the object-file femaleFile, and stores also the male-athletes objects into the object-file maleFile.

- ***buildFromFile(filename:String, maleList:LinkedListOfAthlete, femaleList: LinkedListOfAthlete)***

  This method receives three parameters: an object-file nam e containing only objects of type Athlete and two linked lists of Athletes called respectively ***maleList*** and ***femaleList***. This method **reads the objects** from the object-file and **stores** the **junior** (age less than 16) **male**-athletes **objects** in the ***maleList***. It also **stores** the **senior** (age greater than 18) **female**-athletes **objects** in ***femaleList***.

Write **only** the following methods of the class ***Tournament***:

- ✓ ***splitIntoFiles***
- ✓ ***buildFromFile***

Solution:

14.0

```
public void splitIntoFiles(String femaleFile, String maleFile)
            throws IOException {                                          0.50

  File mf = new File(maleFile);                                           0.25
  FileOutputStream mfos = new FileOutputStream(mf);                       0.25
  ObjectOutputStream moos = new ObjectOutputStream(mfos);                 0.50

  File ff = new File(femaleFile);                                         0.25
  FileOutputStream ffos = new FileOutputStream(ff);                       0.25
  ObjectOutputStream foos = new ObjectOutputStream(ffos);                 0.50

  Node n = participants.getHead();                                        0.50

  while (n != null) {                                                     0.50
    if (n.getData() instanceof Female)                                    0.50

      foos.writeObject( n.getData() );                                    0.50

    if (n.getData() instanceof Male)                                      0.50

      moos.writeObject( n.getData() );                                    0.50

    n = n.getNext();                                                      0.50
  }
  foos.close();
  moos.close();
}

public void buildFromFile(String fileName,
                          LinkedListOfAthletes maleList,
                          LinkedListOfAthletes femaleList)
              throws IOException, ClassNotFoundException {
                                                                          0.50
  File f = new File(filename);                                            0.25
  FileInputStream fis = new FileInputStream(f);                           0.25
  ObjectInputStream ois = new ObjectInputStream(fis);                     0.50

  try  {                                                                  0.50

    while (true) {                                                        0.50

      Athlete a = (Athlete)ois.readObject();                              1.00

      if (a instanceof Male && a.getAge() < 16)                           0.50

        maleList.insertAtBack(a);                                         0.50

      if (a instanceof Female && a.getAge() > 18)                         0.50

        femaleList.insertAtBack(a);                                       0.50
    }
  }

  catch (EOFException e) {}                                               0.50
  ois.close();
}
```