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

```java
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

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

Hello Side: 2.0 Area: 4.0

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

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

```
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();
        setSize(250, 150);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        Container cp = getContentPane();
        cp.setLayout(new GridLayout(4,2, 5, 5));
        lblNum1 = new JLabel("First number:");
        tfNum1 = new JTextField();
        cp.add(lblNum1);
        cp.add(tfNum1);
        
        lblNum2 = new JLabel("Second number:");
        tfNum2 = new JTextField();
        cp.add(lblNum2);
        cp.add(tfNum2);
        
        JButton btnMul = new JButton("Multiply");
        btnMul.addActionListener(this);
        cp.add(btnMul);
        
        JButton btnDiv = new JButton("Divide");
        btnDiv.addActionListener(this);
        cp.add(btnDiv);
    }
    
    public void actionPerformed(ActionEvent e) {
        String str = e.getActionCommand();
        String num1 = tfNum1.getText();
        String num2 = tfNum2.getText();
        int a, b;
        try {
            a = Integer.parseInt(num1);
            b = Integer.parseInt(num2);
            if (str.equals("Multiply")) {
                tfResult.setText(calc.multiply(a, b) + " error");
            } else if (str.equals("Divide")) {
                if (b == 0) {
                    throw new ArithmeticException("Division by zero");
                } else {
                    tfResult.setText(calc.divide(a, b) + " error");
                }
            }
        } catch (NumberFormatException e) {
            tfResult.setText("Invalid input");
        } catch (ArithmeticException e) {
            tfResult.setText("Error: "+e.getMessage());
        }
    }
}
```
lblResult = new JLabel("Result:");
cp.add(lblResult);

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

btnMul = new JButton("Multiply");
btnMul.addActionListener(this);
btnMul.add(btnMul);

tfResult.setText(tfResult.getText());

tfResult.add(tfResult);

btnDiv = new JButton("Divide");
btnDiv.addActionListener(this);
btnDiv.add(btnDiv);

tfResult.setText(tfResult.getText());

tfResult.add(tfResult);

public void actionPerformed(ActionEvent ae) {
    try {
        int x = Integer.parseInt(tfNum1.getText());
        int y = Integer.parseInt(tfNum2.getText());

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

        if (ae.getSource() == btnDiv) {
            try {
                int z = calc.divide(x, y);
                tfResult.setText(z + "
            } catch (ArithmeticException e) {
                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:

```java
import java.io.*;
public class Tournament {
    private String name;
    private String startDate;
    private String city;
    LinkedListOfAthletes participants;

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

    public LinkedListOfAthletes getMaleParticipants(double sco) {
        LinkedListOfAthletes mList = new LinkedListOfAthletes();
        Node n = participants.getHead();
        while (n != null) {
            Athlete a = n.getData();
            if (a instanceof Male && a.getScore() > sco) {
                mList.insertAtBack(a);
            }
            n = n.getNext();
        }
        return mList;
    }

    public void displayFemaleParticipants() {
        LinkedListOfAthletes reverse = new LinkedListOfAthletes();
        Node n = participants.getHead();
        while (n != null) {
            reverse.insertAtFront(n.getData());
            n = n.getNext();
        }
        n = reverse.getHead();
        while (n != null) {
            n.getData().display();
            n = n.getNext();
        }
    }
}
```
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 name 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:

```java
public void splitIntoFiles(String femaleFile, String maleFile) throws IOException {
    File mf = new File(maleFile);
    FileOutputStream mfos = new FileOutputStream(mf);
    ObjectOutputStream moos = new ObjectOutputStream(mfos);
    File ff = new File(femaleFile);
    FileOutputStream ffos = new FileOutputStream(ff);
    ObjectOutputStream foos = new ObjectOutputStream(ffos);
    Node n = participants.getHead();
    while (n != null) {
        if (n.getData() instanceof Female)
            foos.writeObject( n.getData() );
        if (n.getData() instanceof Male)
            moos.writeObject( n.getData() );
        n = n.getNext();
    }
    foos.close();
    moos.close();
}

public void buildFromFile(String fileName, LinkedListOfAthletes maleList, LinkedListOfAthletes femaleList) throws IOException, ClassNotFoundException {
    File f = new File(fileName);
    FileInputStream fis = new FileInputStream(f);
    ObjectInputStream ois = new ObjectInputStream(fis);
    try {
        while (true) {
            Athlete a = (Athlete)ois.readObject();
            if (a instanceof Male && a.getAge() < 16)
                maleList.insertAtBack(a);
            if (a instanceof Female && a.getAge() > 18)
                femaleList.insertAtBack(a);
        }
    }
    catch (EOFException e) {}
    ois.close();
}
```