Open Ended exercise

Q. Write a program that creates user interface to perform integer division. The user enters two numbers in textfields. Num1 & Num2. The division of Num1 & Num2 is displayed in result field when divide button is clicked. If Num1/Num2 were NOT an integer, the program would throw NumberFormatException, if Num2 were zero, the program would throw ArithmeticException. Display the exception in message dialog box.

```
import java.lang.*;
import java.awt.*;
import java.awt.*;

public class IntegerDivisionApp{
    public static void main(String xx[])
    {
        JFrame Frame = new JFrame("Integer Division");
        Frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(400,200);
        frame.setLayout(new GridLayout(4,2,10,10));

        JLabel labelNum1=new JLabel("Num1:");
        JTextfield textNum1=new JTextfield(5);
        JLabel labelNum2=new JLabel("Num2:");
        JTextfield labelResult=new JLabel("Result:");
        JTextfield textResult=new JTextField();
        textResult.setEditable(false);
```

```
JButton divideButton = new JButton("Divide");

Frame.add(labelNum1);
Frame.add(textNum1);
Frame.add(labelNum2);
Frame.add(textNum2);
Frame.add(labelResult);
Frame.add(textResult);
Frame.add(new JLabel());
Frame.add(divideButton);

divideButton.addActionListener(new ActionListener()
{ public void actionPerformed(ActionEvent e){
try{
int num1 = Integer.parseInt(textNum1.getText());
int num2 = Integer.parseInt(textNum2.getText());
if(num2==0)
{ throw new ArithmeticException("Cannot divide by zero");
}
int res = num1/num2;
textResult.setText(String.valueOf(result));
}
catch(NumberFormatException ex){
JOptionPane.showMessageDialog(Frame," Please Enter valid
integers for Num1 & Num2","Number
Format Error", JOptionPane.ERROR_MESSAGE);
}
```

```
Catch (ArithmeticException ex){
    JOptionPane.showMessageDialog(Frame, ex.getMessage(),
    "Arithmetic Error", JOptionPane.ErrorMessage);
    }}
});
  Frame.setVisible(true);
  }
}
```

Week 10 - Open Ended Exercise - II

Q. Demonstrate Inter Process communication & deadlock

```
//deadlock
class A{
    Synchronized void Foo(B b){
        String name = Thread.currentThread().getName();
        System.out.println(name + " entered A.Foo");
        try
        {  Thread.sleep(1000);
        }
        catch(Exception e)
        {    System.out.println("A interrupted");
        }
        System.out.println(name + " trying to call B.last()");
        b.last();
    }
    Synchronised void last(){
        System.out.println("Inside A.last");
    }
}
```

```
class B{
    synchronized void bar(A a){
        String name = Thread.currentThread().getName();
        System.out.println(name + " entered B.bar");
        try
        {
            Thread.sleep(1000);
        }
        catch(Exception e){
            System.out.println("B interrupted");
        }
        System.out.println(name + " trying to call A.last()");
        a.last();
    }

    synchronized void last()
    {   System.out.println(" Inside B.last");
    }
}

class Deadlock implements Runnable
{   A a = new A();
    B b = new B();
    Thread t;
```

```
Deadlock()
  {
    Thread.currentThread.setName("Main Thread");
    t=new Thread(this, "Racing Thread");
  }
void deadlockstart()
{ t.start();
  a.Foo(b);
  System.out.println("Back in main Thread");
}
public void run(){
    b.bar(a);
  System.out.println("Back in other thread");
  }
  public static void main(String xx[]){
    Deadlock dl=new Deadlock();
    dl.deadlock.start();
  }
}
```

```
//Interprocess communication
Class Q
{
    int n;
    boolean valueSet = false;
  Synchronized int get(){
      while(!valueSet){
       try{wait();}
      catch(InterruptedException e)
      { System.out.println(e.getMessage());
      }
      System.out.println("Got : "+n);
      valueSet=false;
       notify();
      return n;
  }

Synchronised void put(int n){
      while(valueSet){
       try{ wait();}
      catch(InterruptedException e){
            System.out.println("Int. excep caught");
          }
      this.n=n;
      valueSet=true;
      System.out.println("Put : "+n);
      notify();}
```

```
class Producer implements Runnable
{
    Q q;
    Thread t;
    producer(Q q)
    { this.q=q;
      t=new Thread(this, "Producer");
    }
    public void or.run(){
      int i=0;
      while(true)
      { q.put(i++);
      }
    }
}
class consumer implements Runnable
{ Q q;
  Thread t;
  consumer(Q q)
  { this.q=q
    t=new Thread(this, "consumer");
  }
  public void run()
  { while(true){
      q.get();
  }
}
```