You will implement a game of rock-paper-scissors,**(JAN-KEN-PO)**and save the results of every match as well as the overall results into a file.

The interface is up to you but it MUST be a USABLE interface. If the user knows how to play Jan-Ken-Po, then there should be no need for instructions but if needed be instructions should be available... just in case.

Your game will implement the rock-paper-scissors game featuring the user Vs the Computer. Submit the following:

* The Interface design of your game is very important. Please test it with friends. Have them play without telling them what the program is about or how to play. If they can do it then your interface is correct. If you have to explain where to click etc then you should modify it.
* You should keep score. User - Computer - Ties **(accurate scoring 150 points)**
* You will write the results of each match as well as the final scores into a file. This**file**should be called ***scores.txt***. **(50 points)**. Make sure to let the user know where the scores file is located. Failure to give the user the file location will incur in 8 points deduction.
* Your grade will depend heavily on the Graphical User Interface design. Be sure to set up the appropriate size so that your GUI looks at its best.  The interface should be user-friendly as explained in the first bullet. **(150 points)**
* You must follow the Java coding standards and comment appropriately as needed.**(50 points)**
* The interfaces with best usability will earn extra credit.**(up to 50 points) <---- TEST your interface!!!**

**Important things to do when programming a GUI**

* Test your design, like I did in the video. Have someone pretend to play in a paper design. Have the person tell you where they would click first, etc this way you get feedback before coding.
* Test your code. Ask someone to play your game AND give them NO instructions. A good interface should be understood by anyone without instructions. If your user asks questions that means your interface is NOT good and you need to fix it.
* You may also do "monkey testing" which means to click buttons without even looking at the keyboard and then open your eyes and verify that nothing unwanted happened. YES, I will do this too. (you may look at the definition of[Monkey Testing in wikipedia](https://en.wikipedia.org/wiki/Monkey_testing))

You must call your driver class**RPS.java,** any other name will result in points deduction. If have other classes you may name them anything you want. The driver is the only one that must have the name as shown. You do not need to submit the scores.txt file. It should be generated as the user plays.

***If your code is incomplete or it does not compile you will get at least 60% deduction.***

***Please***[***watch this video***](https://youtu.be/3G9ifQUWWj8)***so you see my design process and my code running. I expect your GUIs to look much better than mine.***Here is the direct link just in case: <https://youtu.be/3G9ifQUWWj8>

**BookGUIFile.java**

import javax.swing.\*;  //contains graphic elements  
import java.awt.\*;  
import java.awt.event.\*;  //allows graphic elements to work  
import java.io.\*; //file  
import java.util.\*; //scanner  
/\*  
Add actionlistener to list all  
create the action to list all  
check that "ok" cant be clicked if the fields are blank  
\*/  
public class BookGUIFile extends JFrame{  
   
   private JLabel label = new JLabel(" Welcome to the ICS211 Library ");  
   private JLabel lp = new JLabel (" Enter File Name: ");  
   private JButton buttonOkFile = new JButton ("ok file");  
   private int count = 0;    
   private TextArea ta = new TextArea(10, 40);  
   private TextField tfFile = new TextField(40);  
   private final int WIDTH = 600;  
   private final int HEIGHT = 350;  
     
   //file variables  
   String name;  
   //File file = new File (name);  
   Scanner readFromFile = null;  
            // readFromFile = new Scanner(file);  
   //    
     
   Book[ ] bArr = new Book[20];  //array of books //////  
   int i = 0; //to keep the book count and array position//////  
     
   public BookGUIFile( ){  
      this.setSize(WIDTH, HEIGHT);    
      this.setTitle("BOOKS");  
      this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  
      this.setVisible(true);  
      this.setResizable(false);  
   }  
   
   /\* puts the elements in the screen where they belong  
      adds listeners to elements that will monitor events  
      \*/  
   public void initializeGUI( ) throws Exception{  
      //add action listeners to graphic elements  
      //add action listeners to buttons only  
      ActionListener ears = new MyListener( );  
      buttonOkFile.addActionListener(ears);  
         
      JPanel top = new JPanel( );  
      top.setLayout(new BorderLayout( ));  
      top.add("North", label);  
        
      JPanel bPanel = new JPanel( );    
      bPanel.add(lp);  
      bPanel.add(tfFile);  
      bPanel.add(buttonOkFile);  
      top.add("South", bPanel);     
          
      JPanel center = new JPanel( );  
      ta.setText("");  
      center.add(ta);    
      top.add("Center", center);  
      this.add(top);  
         
       tfFile.requestFocus( );       
   }  
   
  //inner class  
   private class MyListener implements ActionListener{  
      public void actionPerformed(ActionEvent event) {  
         String s = "";  
         String titleL = "";  
         String isbnL="";  
         int isbni = 0;  
         Book b;  
         if(event.getSource( )==buttonOkFile){  
            try{  
               name = tfFile.getText( );  
               File file = new File (name);  
               readFromFile = new Scanner(file);  
               
               while(readFromFile.hasNext( )){  
                  titleL = readFromFile.nextLine( );  
                  isbnL = readFromFile.nextLine( );  
                  isbni = Integer.parseInt(isbnL);  
                  b = new Book(isbni, titleL);    
                  bArr[i] = b;  
                  i++;     
                  ta.append(b.toString( )+ "\n=========\n");  
               }  
               tfFile.setText("");  
            }  
            catch(FileNotFoundException fnf){  
               JOptionPane.showMessageDialog(new JFrame( ),  
                               "File "+ name+ " does not exist. Try again",  
                               "ICS211-Exception",  
                                JOptionPane.ERROR\_MESSAGE);  
               tfFile.setText("");  
               return;  
                 
            }  
            catch(NumberFormatException nfe){  
               JOptionPane.showMessageDialog(new JFrame( ),  
                               "File does not contain a number in the ISBN field",  
                               "ICS211-Exception",  
                                JOptionPane.ERROR\_MESSAGE);  
               return;                   
            }  
            catch(BookException be){  
               JOptionPane.showMessageDialog(new JFrame( ),  
                               "ISBN not in range or Book title too short",  
                               "ICS211-Exception",  
                                JOptionPane.ERROR\_MESSAGE);  
               return;  
            }            
         }  
       
      }  
        
   }  
}

import java.io.\*;  
public class UseOutFiles{  
   public static void main(String[ ] arg) throws Exception{  
      File f = new File("TestFile.txt");  
      FileWriter fw = new FileWriter(f);  
      PrintWriter pw = new PrintWriter(fw);  
      for (int i = 0; i<=20; i++){  
         pw.println("line: " + i);  
      }  
      pw.close( );  
   } //closes main method  
} //closes class