Jeopardy Review Game

Daniel McGuigan
IB Computer Science HL-Y2

Table of Contents

A1-Analysis:

The Problem

Many teachers try to review for a test in class. However, the best method to do so often eludes them. Some try and quiz the students, some give sample problems, and some play a ‘review game’ that challenges students and brings out their competitive side, while forcing them to remember critical information to win. A common game that I’ve experience over the years is a Jeopardy-type knockoff. It has also been my experience that this works rather well in reviewing because you can ask a broad variety of questions and include a large number of people.

Expanding on this idea, what if there was a way for students to use this format of review to study individually? They could have questions that are specific to a chapter, or perhaps a course review, such as would be good for an AP class before the exam. The potential for student success would greatly increase, and there wouldn’t be a need to have a large group of people to review, nor would valuable class time be taken up to review.

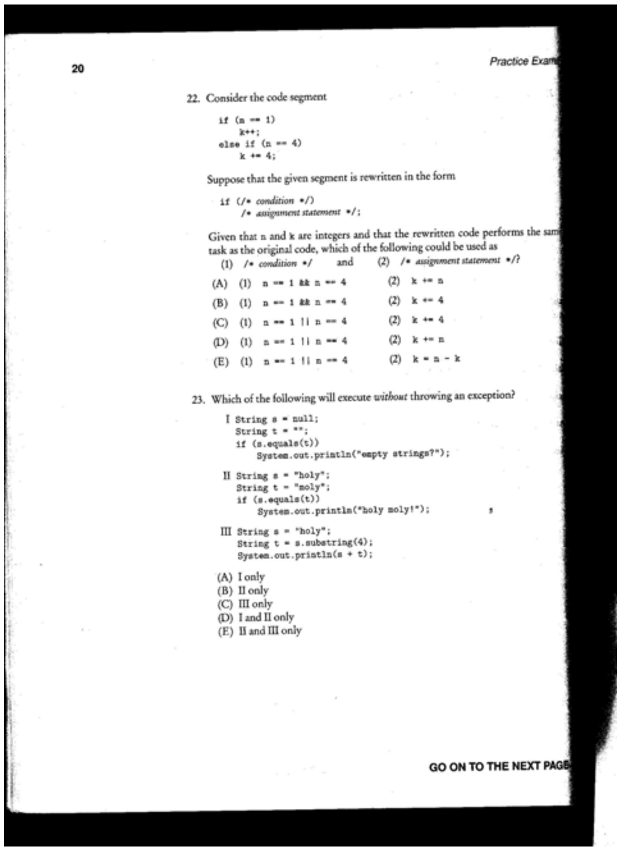
Though it would be intended for individual use, there will also be a ‘multiplayer’ type setting, allowing students to review together and challenge each other. This setting would allow up to two friends to join the user and have a traditional Jeopardy like competition.

In the future, it could be possible to expand beyond just one subject. The program will be compatible to add questions/answers for a broad subject variety, as in the true Jeopardy style.

The program would open with a window to allow the user to select the number of players, as well as a subject, if applicable. They would then be taken to the game screen, which would show the title of the set of questions and a grid of squares, each of which having a question that the user would then answer.

The initial version of this game will include questions to prepare for the AP Computer Science courses. In *collaboration* with Mr. Jason Parker, the course instructor for our school, he has provided me with a book of sample questions to use for the program. The book, which is a review book that he currently uses, contains both multiple choice sample tests, as well as sample free response questions. This program will only be including multiple choice type questions.

Sample Data



22. Consider the code segment

```
if (n == 1)
    k++;
else if (n == 4)
    k += 4;
```

Suppose that the given segment is rewritten in the form

```
if (/* condition */)
    /* assignment statement */;
```

Given that n and k are integers and that the rewritten code performs the sam task as the original code, which of the following could be used as

(1) `/* condition */` and (2) `/* assignment statement */`?

(A) (1) `n == 1 && n == 4` (2) `k += n`
(B) (1) `n == 1 && n == 4` (2) `k += 4`
(C) (1) `n == 1 || n == 4` (2) `k += 4`
(D) (1) `n == 1 || n == 4` (2) `k += n`
(E) (1) `n == 1 || n == 4` (2) `k = n - k`

23. Which of the following will execute *without* throwing an exception?

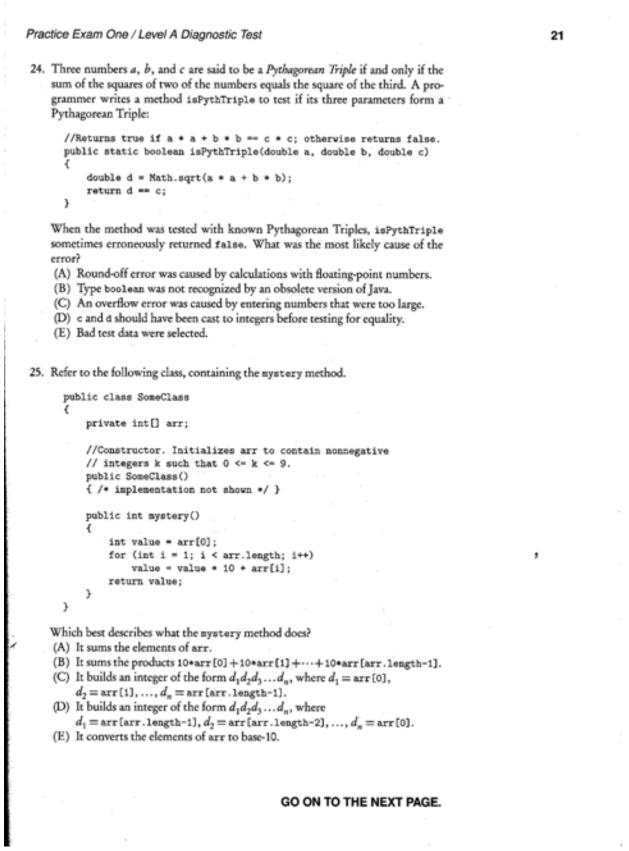
```
I   String s = null;
    String t = "";
    if (s.equals(t))
        System.out.println("empty strings?");

II  String s = "holy";
    String t = "moly";
    if (s.equals(t))
        System.out.println("holy moly!");

III String s = "holy";
    String t = s.substring(4);
    System.out.println(s + t);
```

(A) I only
(B) II only
(C) III only
(D) I and II only
(E) II and III only

GO ON TO THE NEXT PAG



24. Three numbers *a*, *b*, and *c* are said to be a *Pythagorean Triple* if and only if the sum of the squares of two of the numbers equals the square of the third. A programmer writes a method `isPythTriple` to test if its three parameters form a Pythagorean Triple:

```
//Returns true if a * a + b * b == c * c; otherwise returns false.
public static boolean isPythTriple(double a, double b, double c)
{
    double d = Math.sqrt(a * a + b * b);
    return d == c;
}
```

When the method was tested with known Pythagorean Triples, `isPythTriple` sometimes erroneously returned `false`. What was the most likely cause of the error?
(A) Round-off error was caused by calculations with floating-point numbers.
(B) Type `boolean` was not recognized by an obsolete version of Java.
(C) An overflow error was caused by entering numbers that were too large.
(D) `c` and `d` should have been cast to integers before testing for equality.
(E) Bad test data were selected.

25. Refer to the following class, containing the `mystery` method.

```
public class SomeClass
{
    private int[] arr;

    //Constructor. Initializes arr to contain nonnegative
    // integers k such that 0 <= k <= 9.
    public SomeClass()
    { /* implementation not shown */ }

    public int mystery()
    {
        int value = arr[0];
        for (int i = 1; i < arr.length; i++)
            value = value * 10 + arr[i];
        return value;
    }
}
```

Which best describes what the `mystery` method does?
(A) It sums the elements of `arr`.
(B) It sums the products `10*arr[0]`+`10*arr[1]`+⋯+`10*arr[arr.length-1]`.
(C) It builds an integer of the form $d_1d_2d_3 \ldots d_n$, where d_1 = `arr[0]`, d_2 = `arr[1]`, …, d_n = `arr[arr.length-1]`.
(D) It builds an integer of the form $d_1d_2d_3 \ldots d_n$, where d_1 = `arr[arr.length-1]`, d_2 = `arr[arr.length-2]`, …, d_n = `arr[0]`.
(E) It converts the elements of `arr` to base-10.

GO ON TO THE NEXT PAGE.

This is how the class currently prepares for tests, by simply taking pre-released tests as well as sample questions prepared by the Barron’s company (pictured above).

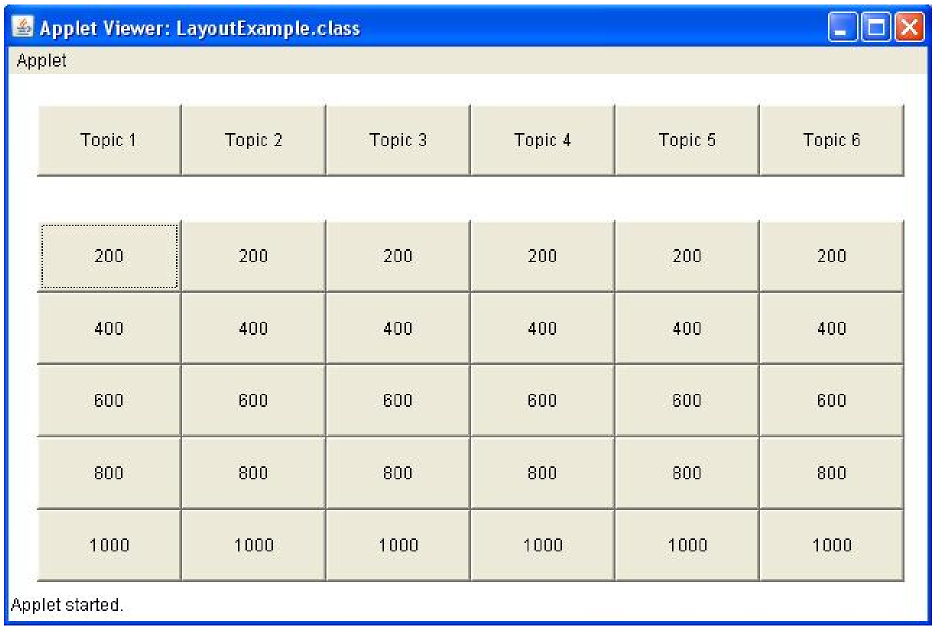
<u>A2-Criterion for Success</u>

An easy to use, straightforward interface.
The interface needs to function without much confusion, leaving the user with just the burden of choosing options and entering the answer. If the user cannot easily determine what is required of them and they don’t know how to do what they are supposed to do, then they will simply not use the program.

A smooth random question generator that doesn’t repeat itself continually.
There needs to be minimal repetition in order for the game to be effective. The point of the game is lost if half of the questions are repeated continuously. This will be obtained by selecting the questions at the beginning of the game and they will be set for the remainder of the game. It will be choosing randomly, and after a question is chosen, it’s ‘line index’ will not be available for adding.

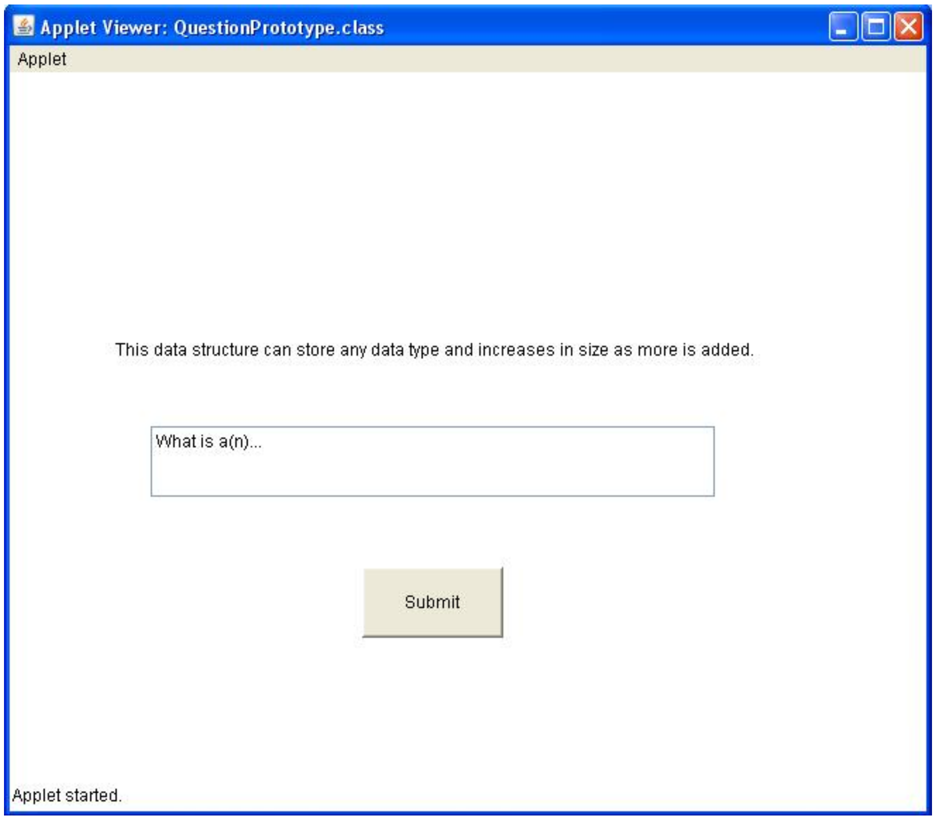
Usable by teacher for large scale review by a class.
The game should not be limited to small groups of up to 3 people per game. The teacher should be able to use it in the classroom setting, whether they divide the group into 3 teams, or have an alternate setting for teacher that allows them to specify a number of teams for the game.

<u>A3-Prototype Solution</u>



Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6
200	200	200	200	200	200
400	400	400	400	400	400
600	600	600	600	600	600
800	800	800	800	800	800
1000	1000	1000	1000	1000	1000

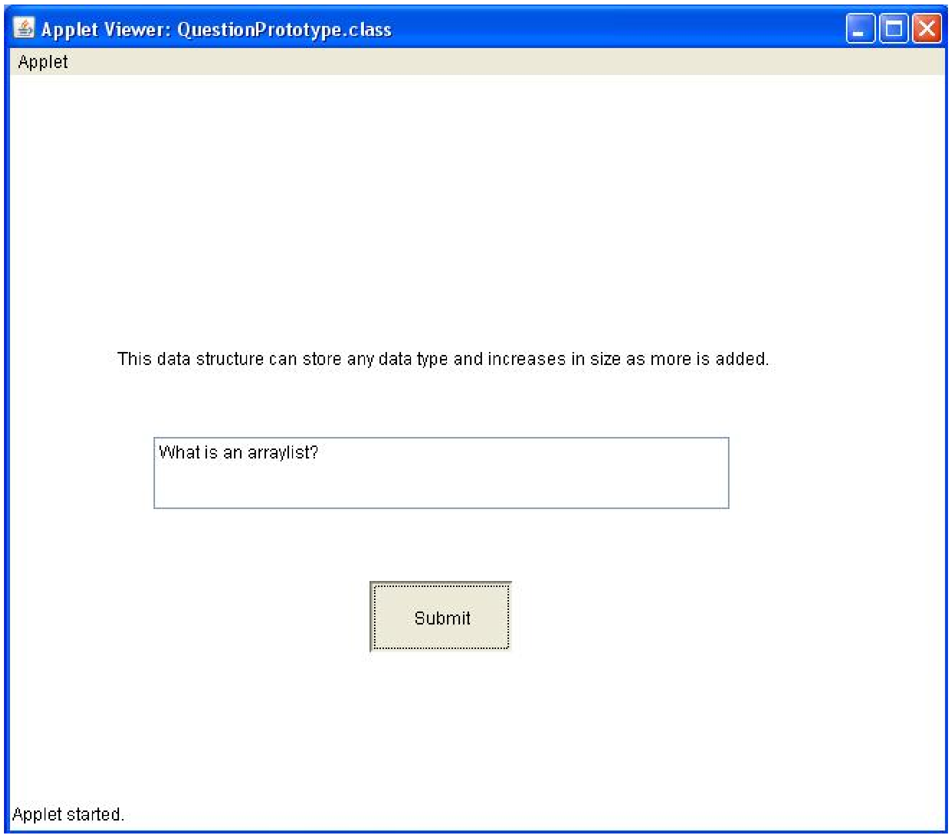
Above is a screenshot of what the question grid will look like when the Prototype is initially loaded. The user will select one of the amounts, much like in the TV game show. After clicking on the amount, the user will be presented with a new window:



This data structure can store any data type and increases in size as more is added.

What is a(n)...

They will then enter their answer in the text field beneath the question. And following that, they would hit the



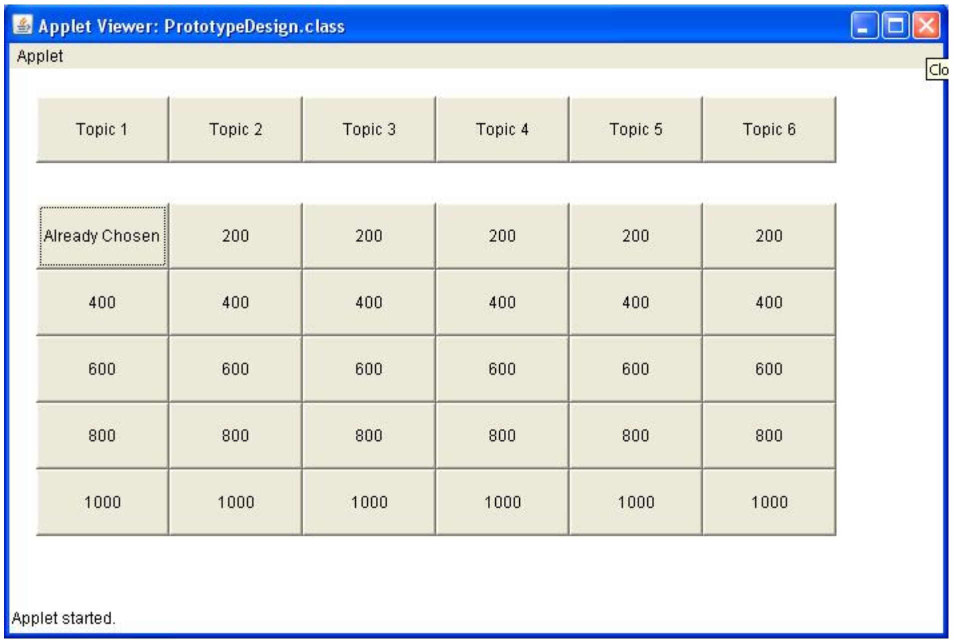
This data structure can store any data type and increases in size as more is added.

What is an arraylist?

Applet started.

submit button. Critical to getting the question correct, is answering in the correct form. Though not set in stone yet, it will be in the Jeopardy game standard, by answering in question form.

After answering the question, the grid will be updated to reflect that the option has been chosen to prevent repetition issues.



Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6
Already Chosen	200	200	200	200	200
400	400	400	400	400	400
600	600	600	600	600	600
800	800	800	800	800	800
1000	1000	1000	1000	1000	1000

Applet started.

B1-Data Structures

Arrays: I am using 3 arrays in the main game class. One of these will store the buttons that make up the GUI portion, while the other 2 will be parallel and contain questions in one, and answers in the other. These are loaded at the beginning of the game, and pulled from randomly throughout the game.

0	1	…	24
Question 1	Question 2	…	Question 25

0	1	…	25
Answer 1	Answer 2	…	Answer 25

Text Files: I will be using text files to hold the questions and answers for the game. Each time a new game is begun, it will randomly select from that file questions and their respective answer, which will be loaded into the aforementioned arrays. I am also contemplating on including a question writer, which will allow the user (most likely a teacher in this case) to create new questions and/or edit existing ones. With the new questions, they will be able to either create a new set of questions that can be played in a game, or they can add it to an existing set of questions that are already usable.

```
question
answer
question
answer
question
answer
question
answer
etc.
```

Swing: I am using the javax.swing packages to create the GUI for the program. This allows me to use buttons, text fields, etc. and also allows me to modify text files, and read from them as well.

B2-Algorithms

Load Interface: This is a rather self explanatory algorithm. It will assign questions and their answers to buttons on the game board, creating the data structure and filling it. At this point, the 2-D Array will be created and initialized, and filled with the arrays of question answer combos. Also, it will initialize the GUI, projecting the

```
try{
    deathToTrees = new Scanner(new File(fileName));
    int counter = 0;
    int index = 1;
    while(counter < questions.length){
        questions[counter] = deathToTrees.nextLine();
        answers[counter] = deathToTrees.nextLine();
        counter++;
    }
}catch(IOException io){
    JOptionPane.showMessageDialog(panel, "File not found.", "Uh Oh......",
    JOptionPane.ERROR_MESSAGE);
    return;
}
```

Display Question: This method will, quite ironically, display the question, with an answer dialog box. It will be triggered by the clicking of a corresponding button. It will then open a new window, containing the question and a spot for the user to enter their answer.

```
Create window;
SOP question from the selected slot;
Read answer;
checkAnswer();
updateBoard;
```

Check Answer: Compare answer given to correct answer. If the user enters the correct answer, then it will return true, add the respective number of points to their total, and display a message telling them it was correct. If wrong, it will return false, subtract the correct amount of points, and then display the correct answer.

```
If(answer.equals(this)){          //if answer and the given answer are the same
    addPoints to total;
    display message;
    return true;
}
subtractPoints from total;
display correct answer;
return false;
```

Record High Score: Keeps track of high scores, storing top 10 scores in a text file. Compares player score at end of game to recorded scores and then updates if necessary. (Not Yet Implemented, still unsure.)

```
ArrayList <String[]> hiScores = new ArrayList <String[]>();
While(file.hasNext() && file.hasNext().hasNext()){
        hiScores.add(new String[] = {next(), nextInt()});
}
While(n < hiScores.size()){
        String[] temp = hiScores.get(n);
        if(this.compareTo(temp[1]) > 0)
        {
                Prompt for name;
                Store name and score in file;
                Return;
        }
        Else if(this.compareTo(temp[1]) == 0)
        {
                Prompt for name;
                store name and score;
                return;
        }
}
Display high score list;
```

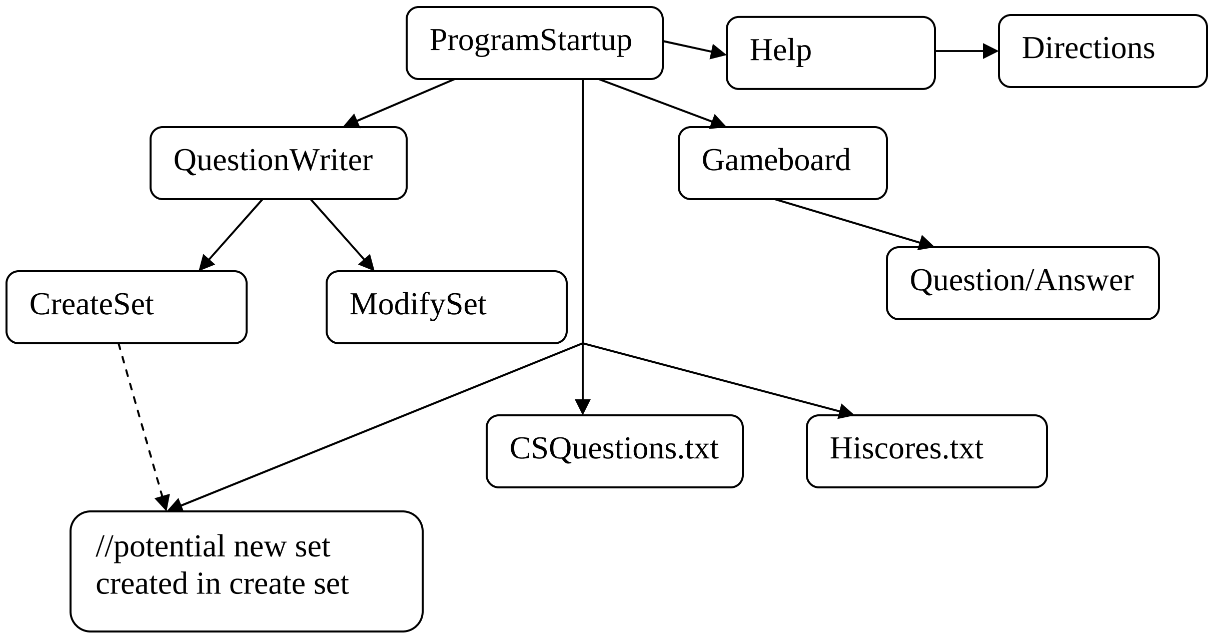
Update Board: updates board with still available questions. If I button has been clicked, then its face is set to “Already Chosen”, signifying that nothing will happen if it is selected. Once a button has been clicked and completed, then once it has been clicked, a message will display, informing the user to select a different option. It also updates the counter for the choice of the next question.

```
if(temp.getLabel().equals("Already Chosen")){
        JOptionPane.showMessageDialog(panel, "That option has already been chosen.  Please
        choose another.", "Option Unavailable, nubber.....",
        JOptionPane.INFORMATION_MESSAGE);
}
else{
        QuestionBox question = new QuestionBox(questions[choice], answers[choice],
        questionValue, score);
        temp.setLabel("Already Chosen");
        random = (int)(Math.random() * 25);
        for(int n = 0; n < used.size(); n++){
                if(random == used.get(n)){
                        random = (int)(Math.random() * 25);
                }
        }
        choice = random;
}
```

Game Over: checks to see if the game has been completed. Displays a message if true and then closes the game window. Else, game continues on.

```
public static boolean gameOver(){
    for(JButton button: field){
        if(!(button.getLabel().equals("Already Chosen"))){
            return false;
        }
    }
    return true;
}
```

B-3: Modular Organization



ProgramStartup: Essentially a "Welcome" screen. Will prompt the user to select a set of questions and number of players before beginning a game.
QuestionWriter: Contains universal methods for creating and modifying sets. Task completed with aid of sub-classes.
CreateSet: contains methods used to create a new set of questions. Creates a new file and stores questions and answers within it.
ModifySet: Prompts user to select a set to modify. They will then be able to add or remove questions, which are stored in a text file.
Gameboard: creates the game board. Contains the game board data structure and keeps track of player points.
HiScores.txt: Stores the top 10 hi scores in the game, updated when necessary following each game.
CSQuestions.txt: stores the questions and answers for this particular set. Can be modified using the modify function.
Still uncreated sets: stores any new sets of questions created using the question writer function. Will only be available for use if there are 30 questions in the set.

C-1: The Code

Class: Startup

```
/**
 * Review Game: Startup Class
 * Author: McGuigy
 * Date: many days
 * School: Valencia HS (OC)
 * Computer: #19 (Dell Intel QuadCore)
 * IDE: Blue J
 * Purpose: Launch the program
 */


import java.awt.*;
import java.awt.event.*;
import java.util.*;
import javax.swing.*;
import java.io.*;

public class Startup extends JFrame implements ItemListener{
    private Toolkit toolkit;
    private JPanel panel;
    private JLabel title;
    private JComboBox gameChoices;
    private JButton start;
    private JButton help;
    private JButton quit;
    private final String[] gameOptions = {"Computer Science A", "Computer Science AB"};
    private String gameChoice;
    private String gameFile;
    private int gameChoiceNum;


    /**
     * Constructor
     */
    public Startup(){
        setTitle("Welcome to Jeopardy");
        setSize(800,700);

        setDefaultCloseOperation(EXIT_ON_CLOSE);

        panel = new JPanel();
        getContentPane().add(panel);

        panel.setLayout(null);

        /**
         * creates title of program object
         */
        title = new JLabel("JEOPARDY");
```

```
title.setFont(new Font("Arial Bold", Font.PLAIN, 28));
title.setVisible(true);
title.setBounds(50,50,500,250);

/**
 * creates selection box of game choices
 */
gameChoices = new JComboBox(gameOptions);
gameChoices.setSelectedIndex(-1);
gameChoices.setBounds(50,250,250,50);
gameChoices.addItemListener(this);

/**
 * creates button that will launch the selected game
 */
start = new JButton("Begin");
start.setBounds(50,400,100,50);
start.addActionListener(new ActionListener(){
   public void actionPerformed(ActionEvent eve){
      try{
         GameBoard game = new GameBoard(gameChoice, gameFile);
      }catch(Exception bob){
         JOptionPane.showMessageDialog(panel,"You need to select a game to play!","Selection
            Error",JOptionPane.ERROR_MESSAGE);
      }
   }
});

/**
 * creates button that will launch the help menu
 */
help = new JButton("HELP");
help.setBounds(200, 400, 100, 50);
help.addActionListener(new ActionListener(){
   public void actionPerformed(ActionEvent ev){
      Help menu = new Help();
   }
});

/**
 * creates button that will cease the program execution
 */
quit = new JButton("QUIT");
quit.setBounds(500, 400, 100, 50);
quit.addActionListener(new ActionListener(){
   public void actionPerformed(ActionEvent event){
      System.exit(0);
   }
});

/**
```

```
         * creates button that will launch the question writer
         */
        JButton writer = new JButton("Question Editor");
        writer.setBounds(50, 475, 150,50);
        writer.addActionListener(new ActionListener(){
            public void actionPerformed(ActionEvent event){
                QuestionWriter bob = new QuestionWriter();
            }
        });

        /**
         * adds all GUI elements to window
         */
        panel.add(title);
        panel.add(gameChoices);
        panel.add(start);
        panel.add(help);
        panel.add(quit);
        panel.add(writer);

        //sets the window to be displayed on the screen
        setVisible(true);
    }

    /**
     *
     * @param event the occurance that will trigger the method
     * @return
     */
    public void itemStateChanged(ItemEvent event){
        JComboBox bob = (JComboBox) event.getSource();
        int index = bob.getSelectedIndex();
        gameChoiceNum = index;
        gameChoice = (String)bob.getItemAt(index);
        gameFile = gameChoice + ".txt";
    }

    /**
     * creates instantiation when called
     *
     * @param args array of arguments
     * @return
     */
    public static void main(String[] args){
        Startup test = new Startup();
    }
}
```

Class: GameBoard

```
/**
 * Review Game: GameBoard Class
 * Author: McGuigy
 * Date: many moons
 * School: Valencia HS (OC)
 * Computer: #19 (Dell Intel QuadCore)
 * IDE: Blue J
 * Purpose: create game board GUI and functionality
 */



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

import java.awt.Dimension;
import java.awt.Toolkit;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.*;


public class GameBoard extends JFrame{
   private static JButton[] field = new JButton[25];        //stores the buttons in the game board
   private String[] questions = new String[25];             //stores questions for current game
   private String[] answers = new String[25];               //stores answers for question array data
   private Toolkit toolkit;                                  //yeah
   private int num;                                          //used multiple times for various counting iterations
   private int choice;                                       //the selected random number to select the question
   private int random;                                       //the variable used to find a random number for
                                                                 question selection
   private static int score;                                //the score      duh
   private ArrayList<Integer> used;                         //stores random numbers once used to select a question
   private static JTextArea scoreboard;                    //displays current score
   private JPanel panel;                                    //contains the things the user sees
   private int bounds = 0;                                  //modifier for the x direction
   private int questionValue = 0;                           //int version of the value of the question
   private int modifier;                                     //modifier for the y direction

   public GameBoard(String gameTopic, String fileName){
      toolkit = getToolkit();

      setTitle("Jeopardy: " + gameTopic);
      setSize(toolkit.getScreenSize());
```

```
    setDefaultCloseOperation(DISPOSE_ON_CLOSE);

    panel = new JPanel();
    getContentPane().add(panel);

    panel.setLayout(null);

    JButton help = new JButton("Help Menu");
    help.setBounds(250,800,100,50);
    help.addActionListener(new ActionListener(){
       public void actionPerformed(ActionEvent event){
          Help meun = new Help();
       }
    });

    score = 0;//------------------------------------------------------------------------------>sets default score to 0

    scoreboard = new JTextArea();
    scoreboard.setBounds(20, 800, 100, 50);
    scoreboard.setEditable(false);
    scoreboard.setText(score + "");

    //fill questions/answers arrays with questions/answers
    Scanner deathToTrees;
    try{
       deathToTrees = new Scanner(new File(fileName));
       int counter = 0;
       int index = 1;
       while(counter < questions.length){
          questions[counter] = deathToTrees.nextLine();
          answers[counter] = deathToTrees.nextLine();
          counter++;
       }
    }catch(IOException io){
       JOptionPane.showMessageDialog(panel, "File not found.", "Uh Oh......",
            JOptionPane.ERROR_MESSAGE);
       return;
    }

    used = new ArrayList<Integer>();


    // initialize game board
    num = 0;
   random = returnRandom();

    while(num < field.length){
       if(num < 5){
          questionValue = 200;
          modifier = 0;
       }
```

```
        else if(num > 4 && num < 10){
            questionValue = 400;
        }
        else if(num > 9 && num < 15){
            questionValue = 600;
        }
        else if(num > 14 && num < 20){
            questionValue = 800;
        }
        else if(num > 19 && num < 25){
            questionValue = 1000;
        }

        field[num] = new JButton(questionValue + "");
        if(num == 5){
            bounds = 0;
            modifier += 100;
        }
        else if(num == 10){
            bounds = 0;
            modifier += 100;
        }
        else if(num == 15){
            bounds = 0;
            modifier += 100;
        }
        else if(num == 20){
            bounds = 0;
            modifier += 100;
        }
        field[num].setBounds(100 + bounds, 200 + modifier, 200, 100);
        field[num].addActionListener(new ActionListener(){
            public void actionPerformed(ActionEvent event){
                JButton temp = (JButton)event.getSource();
                int index = 0;
                while(!(temp.equals(field[index]))){
                    index++;
                }
                if(index < 5){
                    questionValue = 200;
                }
                else if(index > 4 && index < 10){
                    questionValue = 400;
                }
                else if(index > 9 && index < 15){
                    questionValue = 600;
                }
                else if(index > 14 && index < 20){
                    questionValue = 800;
                }
                else if(index > 19 && index < 25){
                    questionValue = 1000;
                }
                if(temp.getLabel().equals("Already Chosen")){
                    JOptionPane.showMessageDialog(panel, "That option has already been chosen.  Please choose
                        another.", "Option Unavailable, nubber.....", JOptionPane.INFORMATION_MESSAGE);
                }
                else{
                    QuestionBox question = new QuestionBox(questions[random], answers[random], questionValue,
                        score);
                    temp.setLabel("Already Chosen");
                    random = returnRandom();
                }
                if(gameOver()){
                    dispose();
                }
            }
        });
        num++;
        bounds += 200;
    }

    num = 0;
    while(num < field.length){
        panel.add(field[num]);
        num++;
    }
    panel.add(help);
    panel.add(scoreboard);
    setVisible(true);

}

//checks to see if random choice selector has used that number before
private boolean contains(Integer num){
    int k = 0;
    while(k < used.size()){
        if(num.equals(used.get(k))){
            return true;
        }
    }
    return false;
}

private int getQuestionValue(){
    return questionValue;
}

//kind of self explanatory....
public static void updateScore(int newScore){
    score = newScore;
```

```
        scoreboard.setText(score + "");
    }

    //checks to see if all questions have been completed
    public static boolean gameOver(){
        for(JButton button: field){
            if(!(button.getLabel().equals("Already Chosen"))){
                return false;
            }
        }
        return true;
    }

    //returns a random, unchosen integer for question selection
    private int returnRandom(){
        int rand = (int)(Math.random() * 25);
        int counter = 0;
        while(counter < used.size()){
            if(used.get(counter) == rand){
                rand = (int)(Math.random() * 25);
                counter = 0;
            }
            counter++;
        }
        used.add(rand);
        return rand;
    }
}
```

Class: QuestionBox

```
/**
 * Review Game: QuestionBox Class
 * Author: McGuigy
 * Date: many moons
 * School: Valencia HS (OC)
 * Computer: #19 (Dell Intel QuadCore)
 * IDE: Blue J
 * Purpose: create GUI for question and answer
 */

import java.awt.*;
import java.util.*;
import java.io.*;

import java.awt.Dimension;
import java.awt.Toolkit;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JTextField;
import javax.swing.*;

public class QuestionBox extends JFrame{
   private Toolkit toolkit;
   private String actualQ;
   private String actualA;
   private String entry;
   private JPanel panel;
   private JTextField text;
   private boolean statsu;
   private boolean answered;
   private int value;                    ///////////////point value of question
   private int score;                    ///////////////current game score

   public QuestionBox(String question, String answer, int pointValue, int currentScore){
      setTitle("Question: " + pointValue +"");
      setSize(600, 500);

      toolkit = getToolkit();
      Dimension size = toolkit.getScreenSize();
      setLocation((size.width - getWidth())/2, (size.height - getHeight())/2);

      setDefaultCloseOperation(DISPOSE_ON_CLOSE);

      panel = new JPanel();
      getContentPane().add(panel);
```

```
panel.setLayout(null);

actualQ = question;
actualA = answer;

value = pointValue;
score = currentScore;

JLabel theQuestion = new JLabel(actualQ);
theQuestion.setFont(new Font("Georgia", Font.PLAIN, 14));
theQuestion.setVisible(true);
theQuestion.setBounds(50, 50, 600, 400);

text = new JTextField(50);
text.setBounds(50, 350, 200, 30);
text.addActionListener(new ActionListener(){
   public void actionPerformed(ActionEvent event){


   }
});

//creates SUBMIT button
JButton submit = new JButton("SUBMIT");
submit.setBounds(50,400,80,30);
submit.addActionListener(new ActionListener() {
   public void actionPerformed(ActionEvent event){
      entry = text.getText();
      boolean result = checkAnswer(entry, actualA);
      if(result == true){
         JOptionPane.showMessageDialog(panel, "Correct Answer", "Good Job!",
               JOptionPane.INFORMATION_MESSAGE);
         GameBoard.updateScore(score + value);
      }
      else{
         JOptionPane.showMessageDialog(panel, "Wrong Answer.  The Correct Answer was: " + actualA,
               "Woe is you....", JOptionPane.INFORMATION_MESSAGE);
         GameBoard.updateScore(score - value);
      }
      if(GameBoard.gameOver()){
         dispose();
         JOptionPane.showMessageDialog(panel, "Congratulations, you've completed the game! Your score
               was " + score, "w00t", JOptionPane.INFORMATION_MESSAGE);
      }
      dispose();
   }
});


panel.add(theQuestion);
```

```
        panel.add(text);
        panel.add(submit);

        setVisible(true);

    }

    private boolean checkAnswer(String input, String correct){
        if(input.equalsIgnoreCase(correct))
            return true;
        return false;
    }

    public boolean isCorrect(){
        if(statsu == true)
            return true;
        return false;
    }

    public boolean isAnswered(){
        if(answered == true){
            return true;
        }
        return false;
    }
}
```

Class: Help

```
/**
 * Review Game: Help Class
 * Author: McGuigy
 * Date: 02/19/10-Present
 * School: Valencia HS (OC)
 * Computer: #19 (Dell Intel QuadCore)
 * IDE: Blue J
 * Purpose: Display a help menu
 */

import java.awt.*;
import java.awt.event.*;
import java.util.*;
import javax.swing.*;
import java.io.*;

public class Help extends JFrame{
    private Toolkit toolkit;
    private JPanel panel;

    /**
     * constructor for help menu window
     */
    public Help(){
        setTitle("Help Menu");
        setSize(800,700);

        setDefaultCloseOperation(DISPOSE_ON_CLOSE);

        panel = new JPanel();
        getContentPane().add(panel);

        panel.setLayout(null);

        /*
         * creates direction for window
         */
        JLabel intro = new JLabel("Please select a help topic:");
        intro.setFont(new Font("Arial", Font.PLAIN, 22));
        intro.setVisible(true);
        intro.setBounds(20,20,500,200);

        /*
         * creates button to open Directions
         */
        JButton directions = new JButton("Directions");
        directions.setBounds(50,200,100,50);
        directions.addActionListener(new ActionListener(){
            public void actionPerformed(ActionEvent event){
                Directions dirs = new Directions();
            }
        });

        /*
         * creates button to open WriterDirections
         */
        JButton writer = new JButton("Q. W. Help");
        writer.setBounds(200,200,100,50);
        writer.addActionListener(new ActionListener(){
            public void actionPerformed(ActionEvent event){
                WriterDirections qw = new WriterDirections();
            }
        });

        /*
         * creates button to handle closing function
         */
        JButton close = new JButton("Close");
        close.setBounds(500,300,100,50);
        close.addActionListener(new ActionListener(){
            public void actionPerformed(ActionEvent event){
                dispose();
            }
        });

        /*
         * adds objects to window pane
         */
        panel.add(intro);
        panel.add(directions);
        panel.add(writer);
        panel.add(close);

        /*
         * sets window to be displayed on screen
         */
        setVisible(true);
    }

    /**
     * creates instantiation when invoked
     *
     * @param args array of arguments
     * @return
     */
    public static void main(String[] args){
        Help bobert = new Help();
    }
}
```

Class: Directions

```
/**
 * Review Game: Directions Class
 * Author: McGuigy
 * Date: 02/24/10-0226/10
 * School: Valencia HS (OC)
 * Computer: #19 (Dell Intel QuadCore)
 * IDE: Blue J
 * Purpose: Display directions for program in help menu
 */

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

public class Directions extends JFrame{
    private Toolkit toolkit;
    private JLabel info;
    private JPanel panel;
    private String directions;

    /**
     * Constructor for Direction window
     */
    public Directions(){
        setTitle("Jeopardy Help Desk");
        setSize(600,500);

        setDefaultCloseOperation(DISPOSE_ON_CLOSE);

        directions = "<html> DIRECTIONS: <br>" +
                     "1. Select a topic <br>" +
                     "2. Click Begin <br>" +
                     "3. In game area, select a point value <br>" +
                     "4. Answer the question in the textfield and click submit. <br>" +
                     "5. After the game, if your score is in the top 10, then you will be prompted to enter your
                         initials for the records. <br>";

        panel = new JPanel();
        panel.setLayout(null);

        /*
         * creates the text of the window
         */
        info = new JLabel(directions);
        info.setBounds(20,20,500,200);
        info.setFont(new Font("Georgia", Font.PLAIN, 14));

        /*
```

```
         * creates button for close function
         */
        JButton close = new JButton("Close");
        close.setBounds(20,300,100,50);
        close.addActionListener(new ActionListener(){
            public void actionPerformed(ActionEvent event){
                dispose();
            }
        });

        /*
         * adds objects to window pane
         */
        panel.add(info, BorderLayout.CENTER);

        getContentPane().add(panel);

        panel.add(close);

        /*
         * sets window to be displayed on screen
         */
        setVisible(true);
    }

    /**
     * creates instantiation when invoked
     *
     * @param args array of arguments
     * @return
     */
    public static void main(String[] args){
        Directions menu = new Directions();
    }
}
```

Class: WriterDirections

```
/**
 * Review Game: Directions Class
 * Author: McGuigy
 * Date: 02/24/10-0226/10
 * School: Valencia HS (OC)
 * Computer: #19 (Dell Intel QuadCore)
 * IDE: Blue J
 * Purpose: Display directions for program in help menu
 */

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

public class WriterDirections extends JFrame{
    private Toolkit toolkit;
    private JLabel info;
    private JPanel panel;
    private String directions;

    /**
     * Constructor for Direction window
     */
    public WriterDirections(){
        setTitle("Jeopardy Help Desk");
        setSize(600,500);

        setDefaultCloseOperation(DISPOSE_ON_CLOSE);

        directions = "<html> DIRECTIONS: <br>" +
                     "1. Enter a name for a file. <br>" +
                     "2. Enter the question and answer into relative spot. <br>" +
                     "3. Click 'ADD' <br>" +
                     "4. Continue until you have added all desired questions. <br>" +
                     "5. Click 'DONE' when finished. <br>";

        panel = new JPanel();
        panel.setLayout(null);

        /*
         * creates the text of the window
         */
        info = new JLabel(directions);
        info.setBounds(20,20,500,200);
        info.setFont(new Font("Georgia", Font.PLAIN, 14));

        /*
         * creates button for close function
         */
        JButton close = new JButton("Close");
        close.setBounds(20,300,100,50);
        close.addActionListener(new ActionListener(){
            public void actionPerformed(ActionEvent event){
                dispose();
            }
        });

        /*
         * adds objects to window pane
         */
        panel.add(info, BorderLayout.CENTER);

        getContentPane().add(panel);

        panel.add(close);

        /*
         * sets window to be displayed on screen
         */
        setVisible(true);
    }

    /**
     * creates instantiation when invoked
     *
     * @param args array of arguments
     * @return
     */
    public static void main(String[] args){
        WriterDirections menu = new WriterDirections();
    }
}
```

Class: QuestionWriter

```
/**
 * Review Game: QuestionWriter Class
 * Author: McGuigy
 * Date: 03/01/10-Present
 * School: Valencia HS (OC)
 * Computer: #19 (Dell Intel QuadCore)
 * IDE: Blue J
 * Purpose: create interface to create review game files
 */

import javax.swing.*;
import java.util.*;
import java.awt.*;
import java.awt.event.*;
import java.io.*;

public class QuestionWriter extends JFrame{
    private Toolkit toolkit;
    private JPanel panel;
    private JLabel fileTitle;
    private JLabel enterQuestion;
    private JLabel enterAnswer;
    private JTextField question;
    private JTextField answer;
    private JTextField fileName;
    private JButton save;
    private JButton close;
    private String name;

    /**
     * constructor for question writer objects
     */
    public QuestionWriter(){
        setTitle("Welcome to the Question Generator!");
        setSize(800,700);

        setDefaultCloseOperation(DISPOSE_ON_CLOSE);

        panel = new JPanel();
        getContentPane().add(panel);

        panel.setLayout(null);

        /*
         * creates directions for title addition
         */
        fileTitle = new JLabel("Please create a title for the set: ");
        fileTitle.setFont(new Font("Arial", Font.PLAIN, 14));
        fileTitle.setVisible(true);
fileTitle.setBounds(50, 50, 300,50);

/*
 * creates field for user to input file name
 */
fileName = new JTextField(50);
fileName.setBounds(380, 50, 250, 30);
fileName.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent event){
        name = fileName.getText();
    }
});

/*
 * creates label for user instruction
 */
enterQuestion = new JLabel("Please enter your question here ----> ");
enterQuestion.setFont(new Font("Arial", Font.PLAIN, 14));
enterQuestion.setBounds(50,150,250,50);
panel.add(enterQuestion);

/*
 * creates field for user to input question
 */
question = new JTextField(200);
question.setBounds(325,150,200,30);
question.setVisible(true);
panel.add(question);

/*
 * creates label for user instruction
 */
enterAnswer = new JLabel("Now enter your answer here ----> ");
enterAnswer.setFont(new Font("Arial", Font.PLAIN, 14));
enterAnswer.setBounds(50,300,250,50);
panel.add(enterAnswer);

/*
 * creates field for user to input answer
 */
answer = new JTextField(200);
answer.setBounds(325,300,200,30);
answer.setVisible(true);
panel.add(answer);

/*
 * creates button to save input question and answer combo
 */
save = new JButton("Save Question");
save.setBounds(50,500,125,50);
save.addActionListener(new ActionListener(){
```

```
            public void actionPerformed(ActionEvent event){
                try{
                    PrintWriter output = new PrintWriter(new FileWriter(fileName.getText() + ".txt", true));
                    if(question.getText().equals("")){
                        JOptionPane.showMessageDialog(panel, "Question ERROR", "ENTER a QUESTION!",
                            JOptionPane.ERROR_MESSAGE);
                    }
                    if(answer.getText().equals("")){
                        JOptionPane.showMessageDialog(panel, "Answer ERROR", "ENTER an ANSWER!",
                            JOptionPane.ERROR_MESSAGE);
                    }
                    else{
                        output.println(question.getText());
                        output.println(answer.getText());
                        output.close();
                        question.setText("");
                        answer.setText("");
                    }
                }catch(IOException err){
                    JOptionPane.showMessageDialog(panel, err.getMessage(), "ERROR",
                        JOptionPane.ERROR_MESSAGE);
                }
//                  dispose();
            }
        });

        /*
         * creates button to close window
         */
        close = new JButton("Finished");
        close.setBounds(200, 500, 125, 50);
        close.addActionListener(new ActionListener(){
            public void actionPerformed(ActionEvent event){
                dispose();
            }
        });

        panel.add(fileTitle);
        panel.add(fileName);
        panel.add(save);
        panel.add(close);

        setVisible(true);
    }

    ///////READ ONE QUESTION AT A TIME THEN ADD

    public static void main(String[] args){
        QuestionWriter author = new QuestionWriter();
    }
}
```

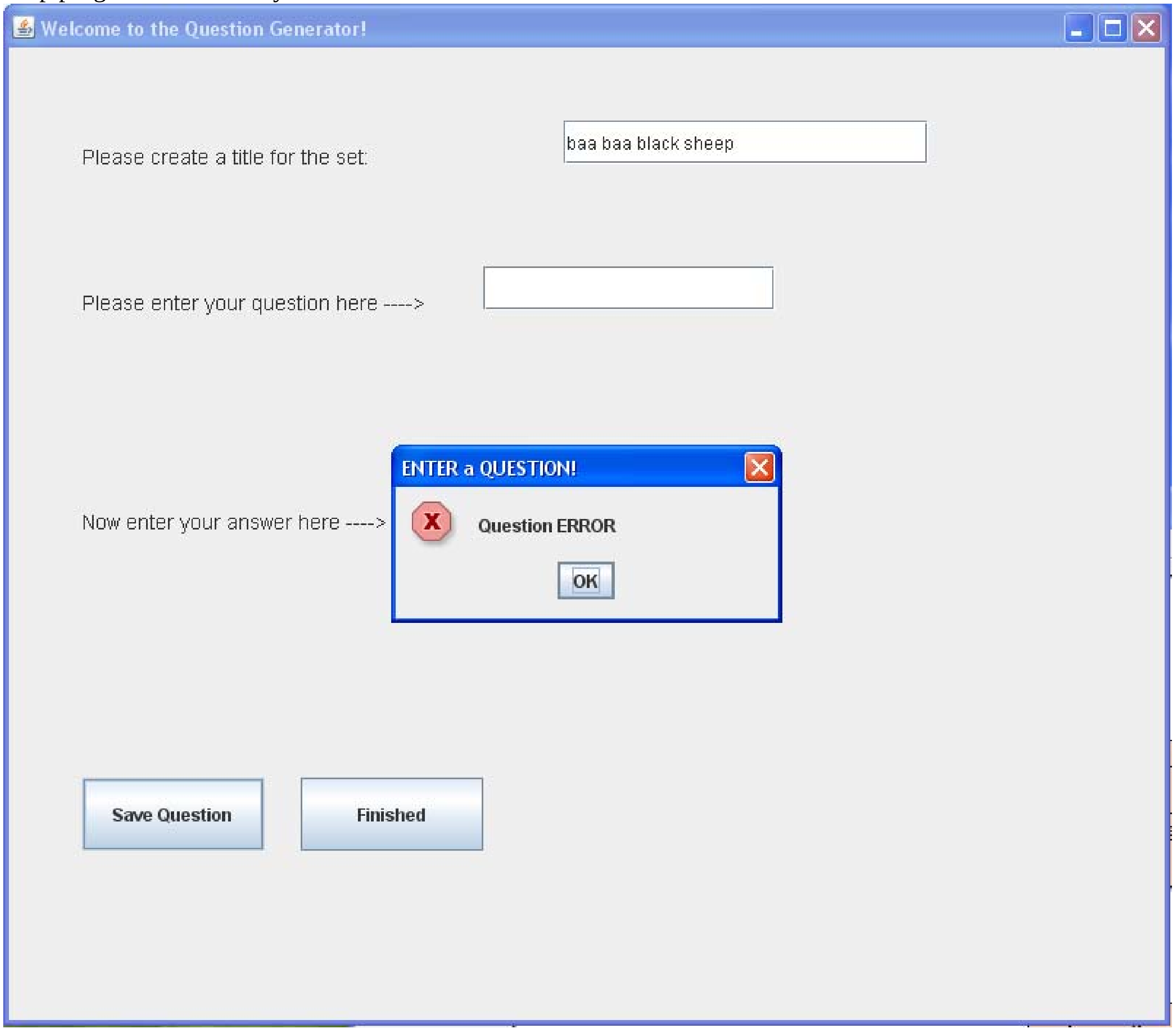
File: Computer Science A (Test Data Set)
Name a data structure with a fixed length
Array
what data structure removes the last thing added?
stack
Define syntax
rules that govern a language
Instantiate an array that will contain Strings
String[]
What is the variable type that stores Ture/False options?
boolean
What variable type stores integer objects?
int
What variable type stores decimal numbers?
double
what would be returned by 6 % 4?
2
Convert 255 to hexidecimal.
FF
What is 10011001 in decimal?
153
What is the most inefficient search algorithm?
bubble sort
which searching algorithm uses recursion?
Merge Sort
What is it called when a method calls itself in the code?
recursion
What is IB collaboration?
cheating
What line of code returns a random number?
Math.random()
True or False: Is "Object bob = new String();" a correct initialization?
true
What is Mr. Jenkins' primary class in WoW?
Rogue
Can an abstract class be instantiated?
no
True or False: Arraylists can store ints.
false
What method allows you to find the length of an arraylist?
size()
How do we count nodes?
very carefully
What kind of loop is this: ____(int n = 0; n < 3; n++)
for loop
What does Karel put down on the grid?
beepers
What line of code do you put down at end of Karel programs?
turnOff()
h

```
fajiads
jfsdkla
jfdklasf
jfdlasj
\jfdklasj
jfdklsajf
fdsjklafja
jhfldasjf
jhflsdajf
```

NOTE: The random keystroke lines were simply to fill up lines so that game would work and compile. I have actually checked and none are repeated during gameplay.

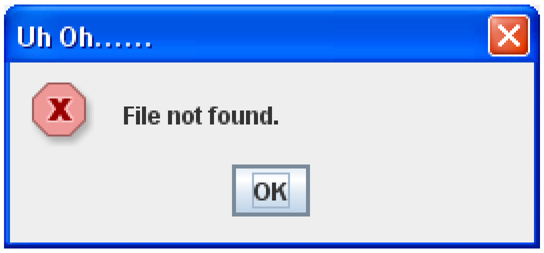
C-2: Handling Errors

While system errors are possible, the greatest chance of errors is from user input. As such, I have included many information windows that will prevent users from proceeding with out entering info when it is required to keep program functionality.


Welcome to the Question Generator!
Please create a title for the set:
baa baa black sheep
Please enter your question here ---->
Now enter your answer here ---->
ENTER a QUESTION!
Question ERROR
OK
Save Question
Finished

Here, this message appears when you have failed to enter a question when you click “Save Question”. Because of the way that questions and answers are stored, failing to enter one will cause problems later on in the game. As such, the code will prevent a user from entering a question without an answer, or an answer without a question. Also, it will prevent users from being able to add a blank question/answer combo.

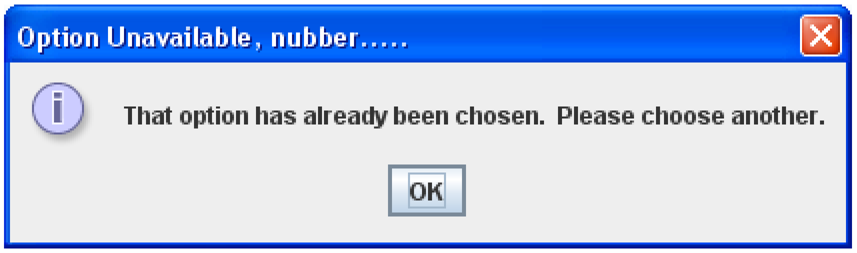
The code also catches errors where the file cannot be found. If this occurs, then it will display a message stating such, and allowing the user to choose a different topic or to solve the missing file issue. Also, if a user attempts to answer a question that has already been chosen, it will stop it and tell the user to pick another question.



Uh Oh......

File not found.

OK



Option Unavailable, nubber.....

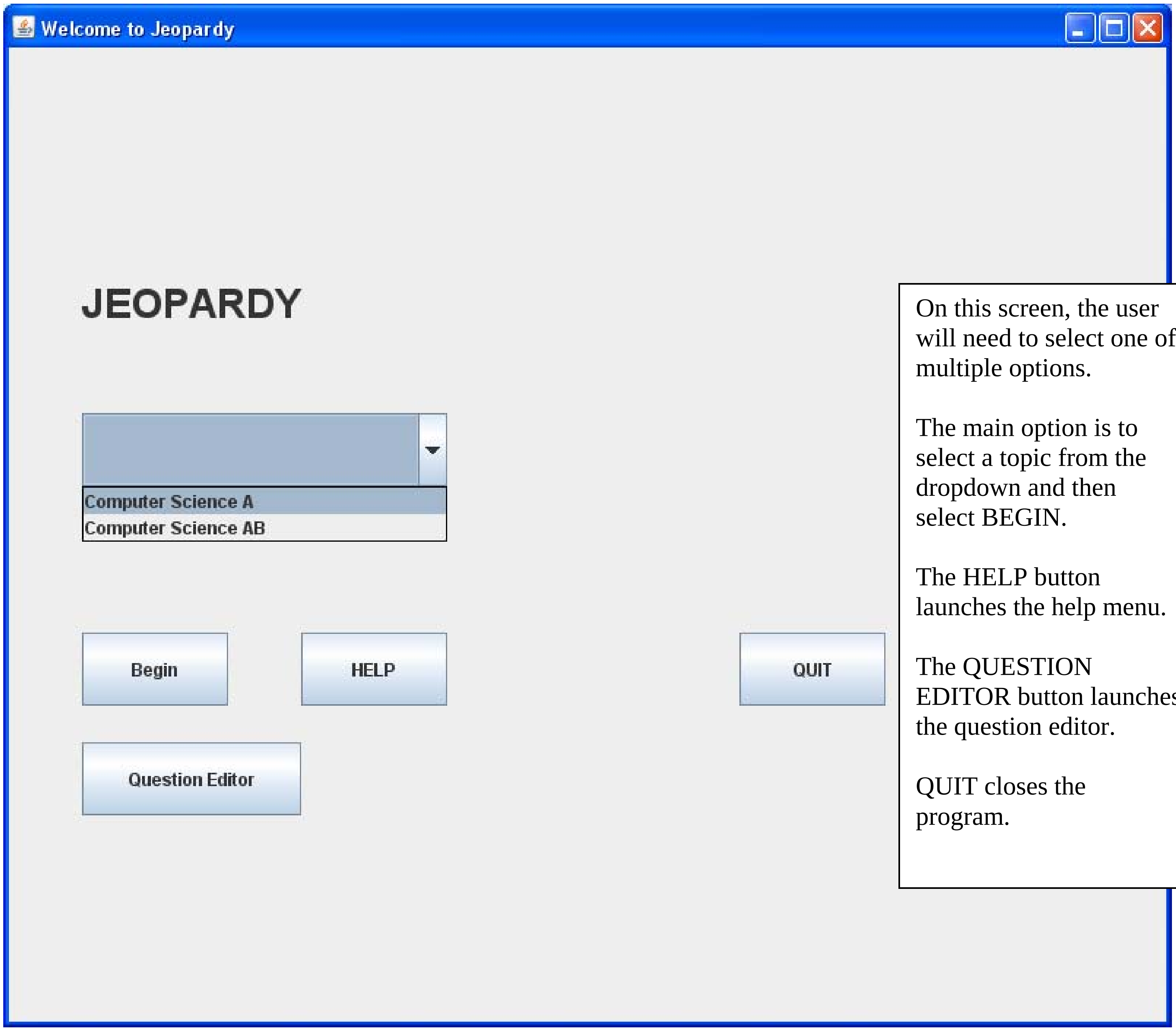
That option has already been chosen. Please choose another.

OK

C-3: Success of Program

Objectives	Evidence
An easy to use, straightforward interface 1. Easy to maneuver thru program 2. All user needs to do is pick a question and answer	1. see annotations in D-1. (rather straightforward) 2. see page 36
A smooth random question generator that doesn't repeat itself continually.	See page 19; [private int returnRandom()]
Usable by teacher for large scale review by a class.	Due to usage by a single individual, it can easily be used on a projector or smartboard system for a large scale review.

D-1: Test Run (Annotated)



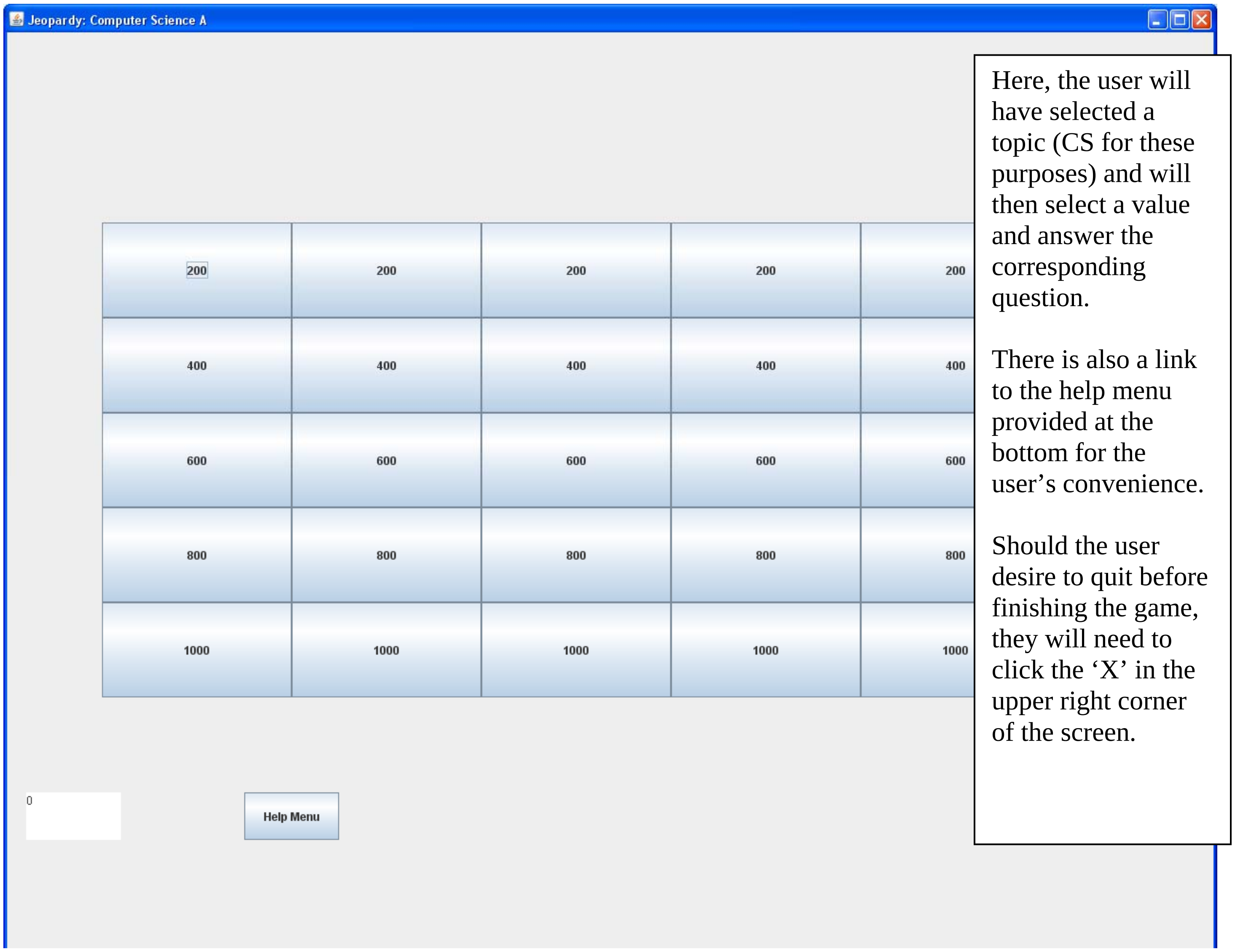
On this screen, the user will need to select one of multiple options.

The main option is to select a topic from the dropdown and then select BEGIN.

The HELP button launches the help menu.

The QUESTION EDITOR button launches the question editor.

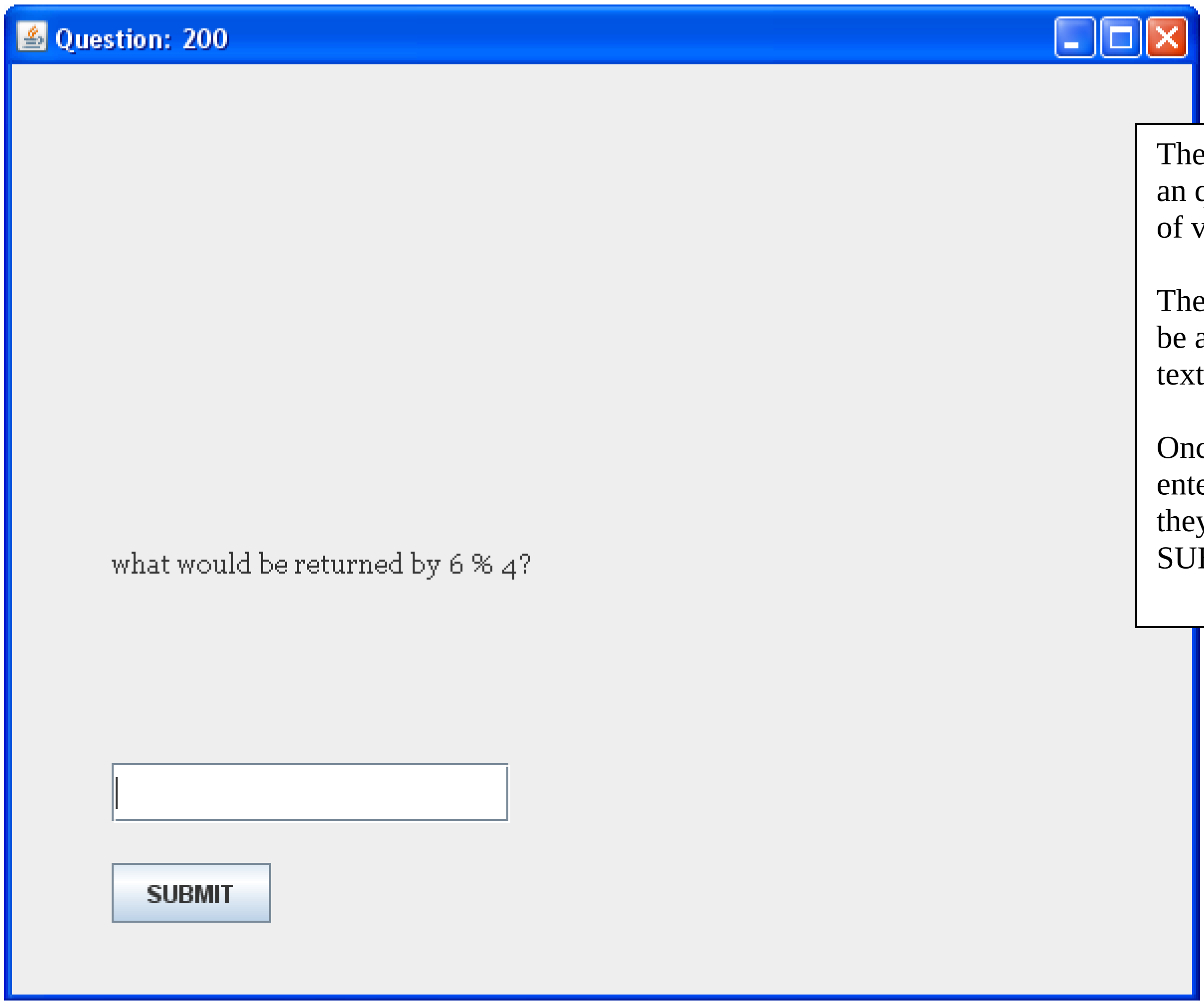
QUIT closes the program.



Here, the user will have selected a topic (CS for these purposes) and will then select a value and answer the corresponding question.

There is also a link to the help menu provided at the bottom for the user's convenience.

Should the user desire to quit before finishing the game, they will need to click the 'X' in the upper right corner of the screen.

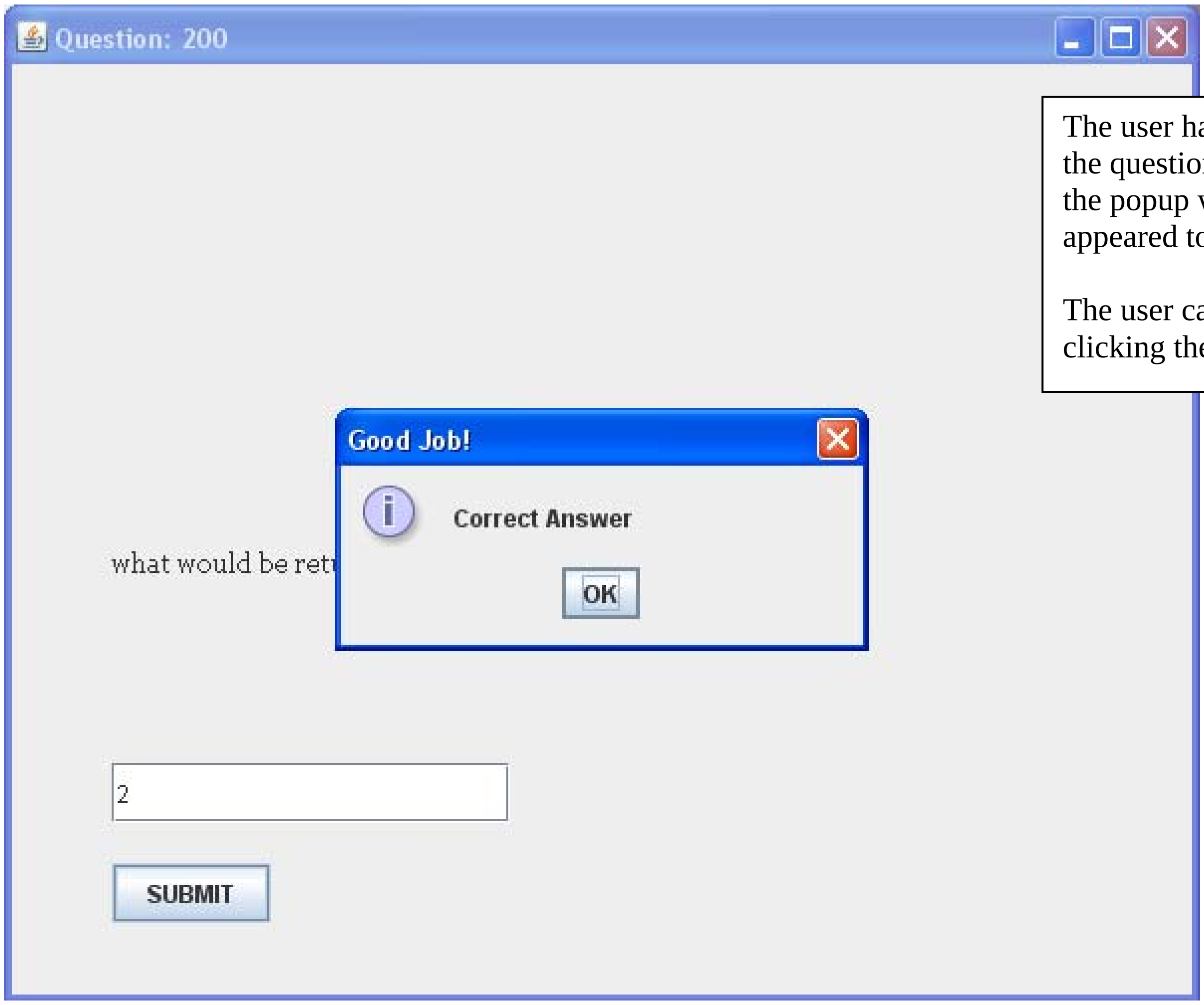


what would be returned by 6 % 4?

The user has chose an question, here of value 200.

The question is to be answered in the text field provided.

Once the user has entered an answer, they will click the SUBMIT button.



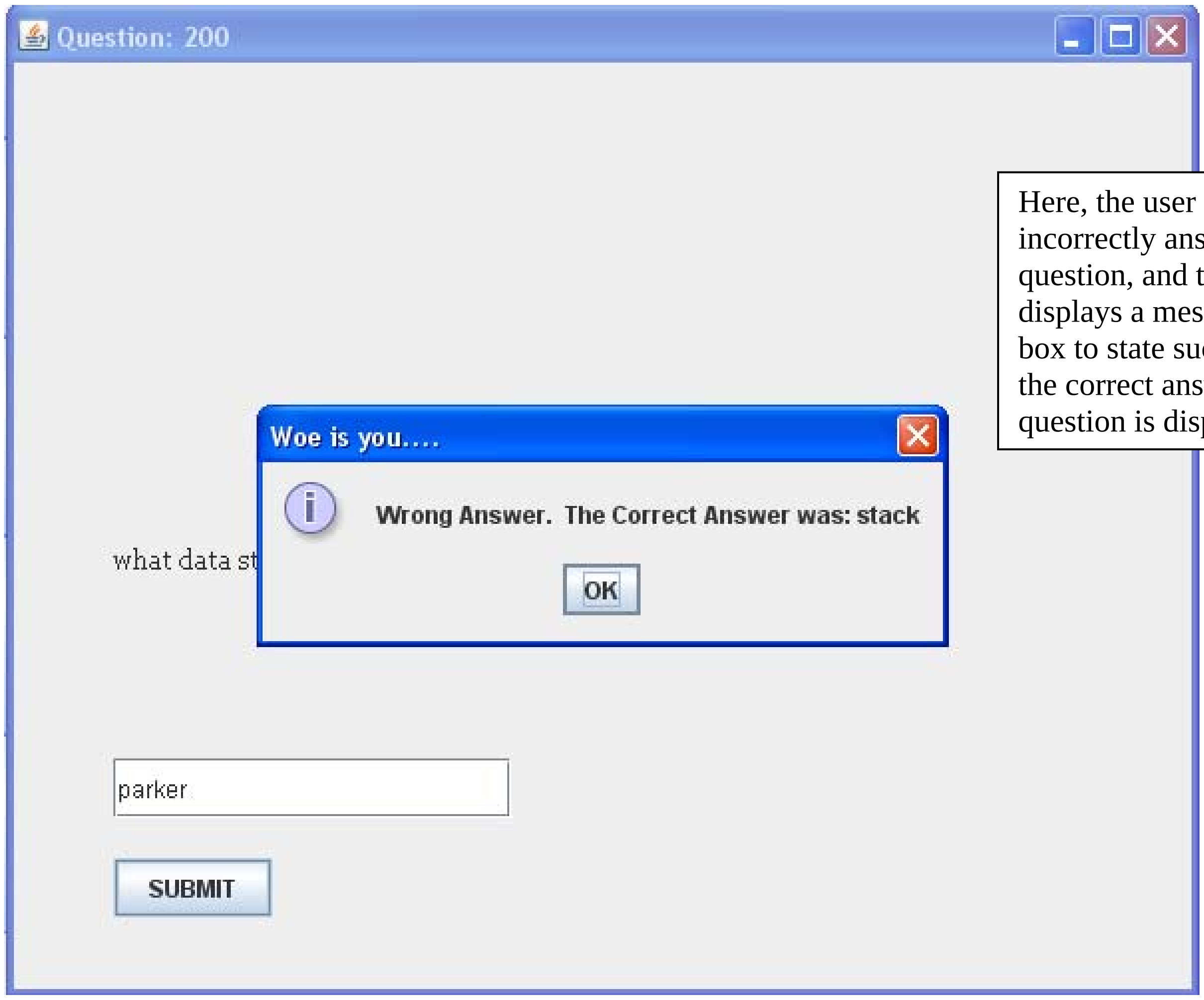
Correct Answer

what would be ret

2

The user has here answered the question correctly, and the popup window has appeared to state such.

The user can continue by clicking the OK button.

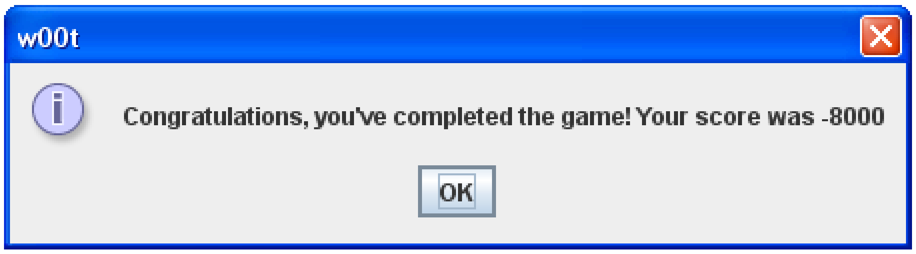


Wrong Answer. The Correct Answer was: stack

what data st

parker

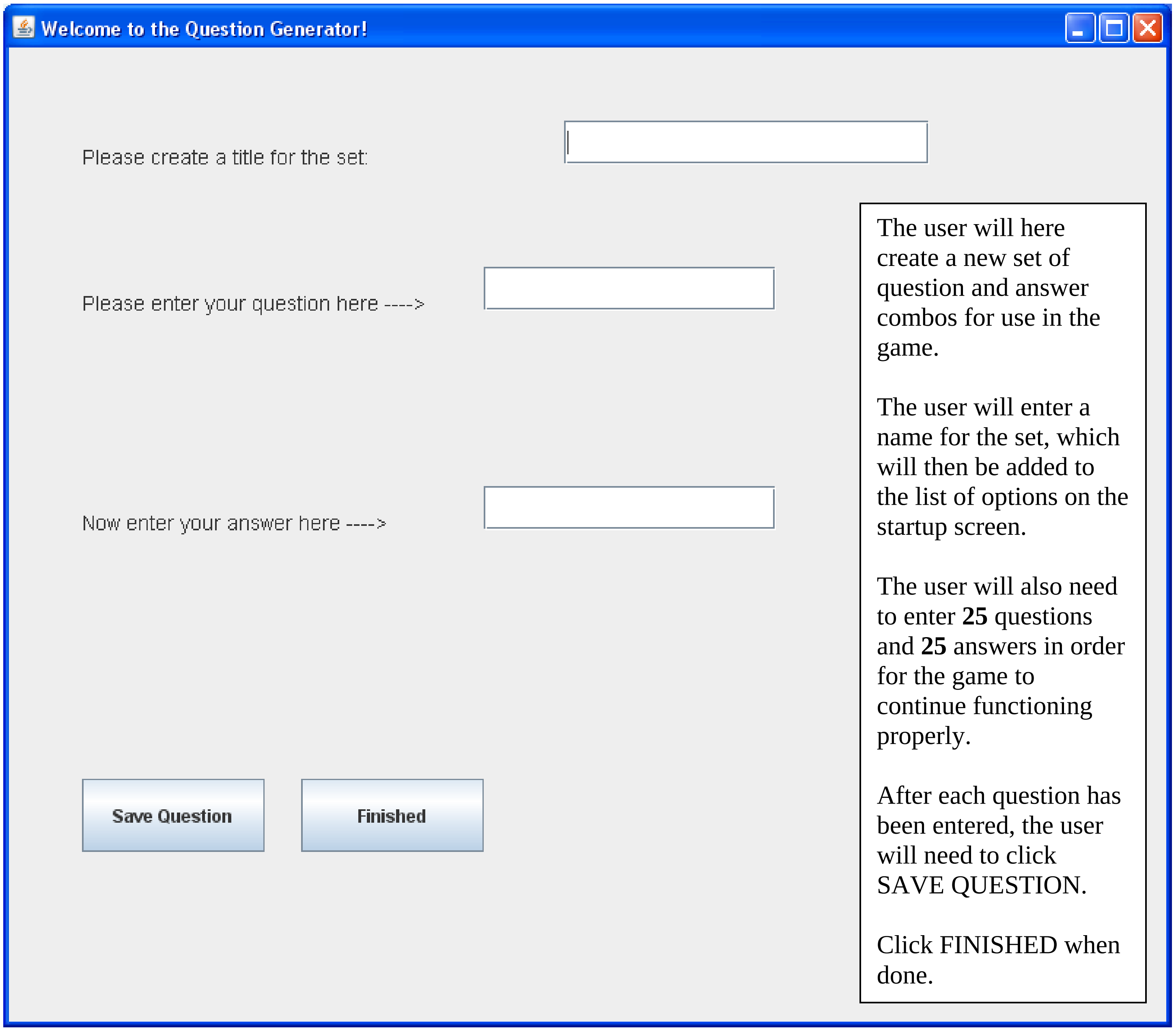
Here, the user has incorrectly answered a question, and the program displays a message dialog box to state such. Also, the correct answer to the question is displayed.



w00t

Congratulations, you've completed the game! Your score was -8000

This window pops up after EVERY question has been answered.
It tells you what your final score was at the end of the game.
Click OK to continue.



Welcome to the Question Generator!

Please create a title for the set:

Please enter your question here ---->

Now enter your answer here ---->

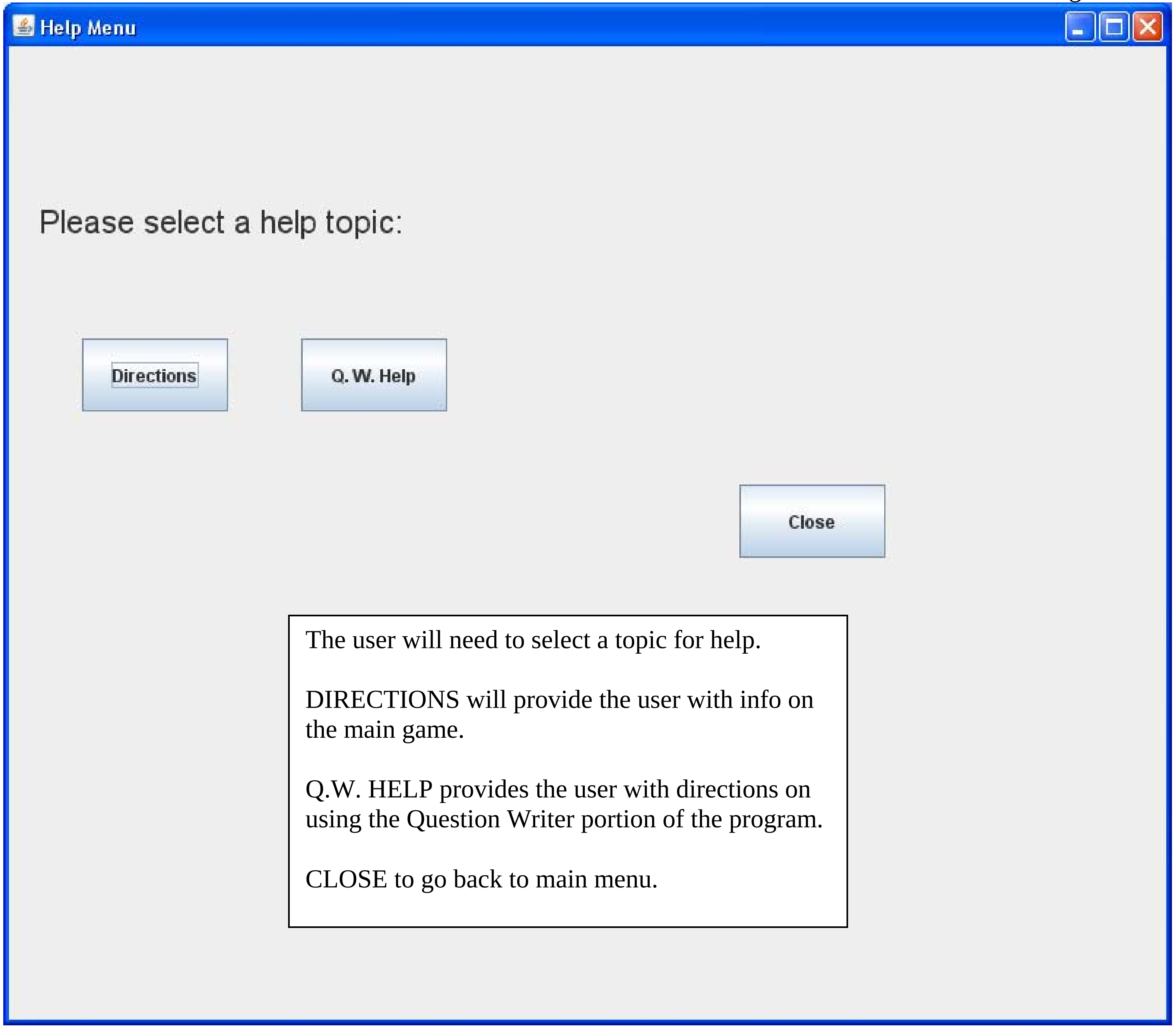
The user will here create a new set of question and answer combos for use in the game.

The user will enter a name for the set, which will then be added to the list of options on the startup screen.

The user will also need to enter **25** questions and **25** answers in order for the game to continue functioning properly.

After each question has been entered, the user will need to click SAVE QUESTION.

Click FINISHED when done.

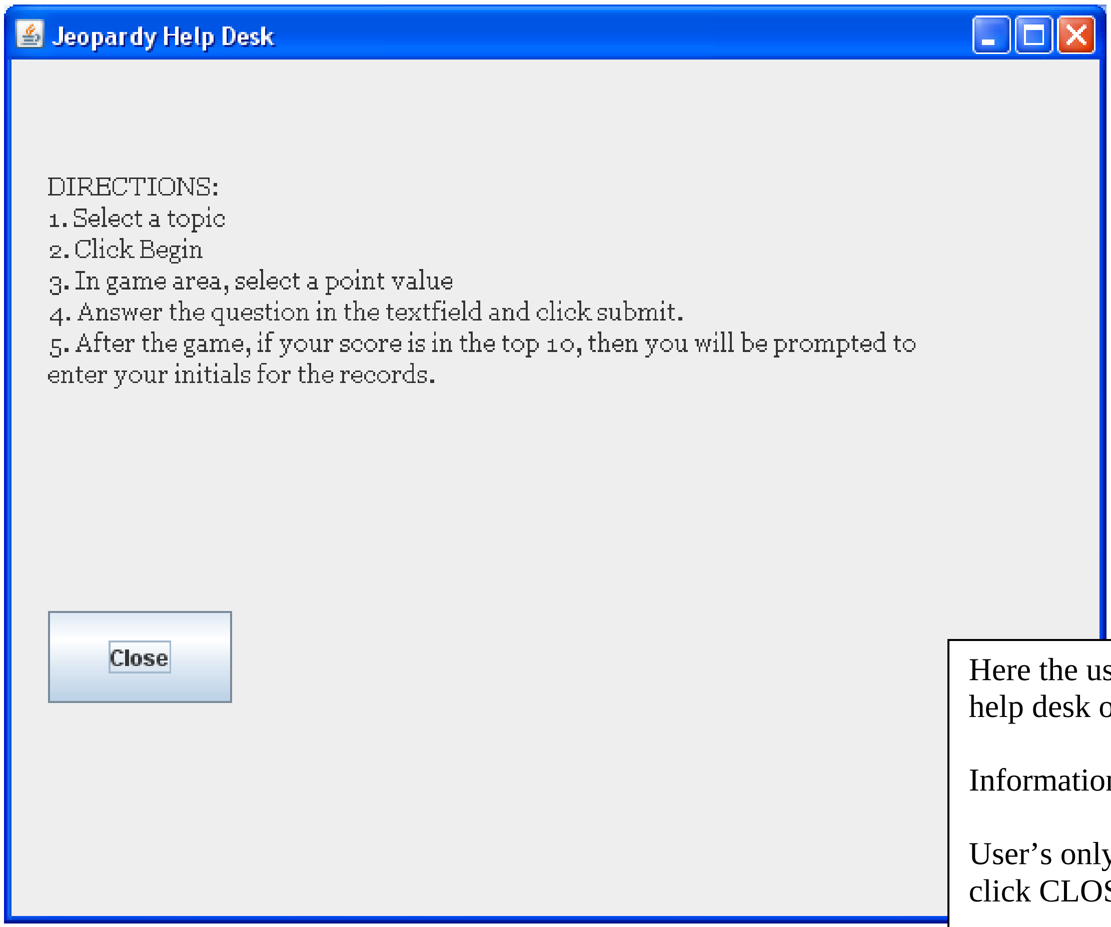


The user will need to select a topic for help.

DIRECTIONS will provide the user with info on the main game.

Q.W. HELP provides the user with directions on using the Question Writer portion of the program.

CLOSE to go back to main menu.

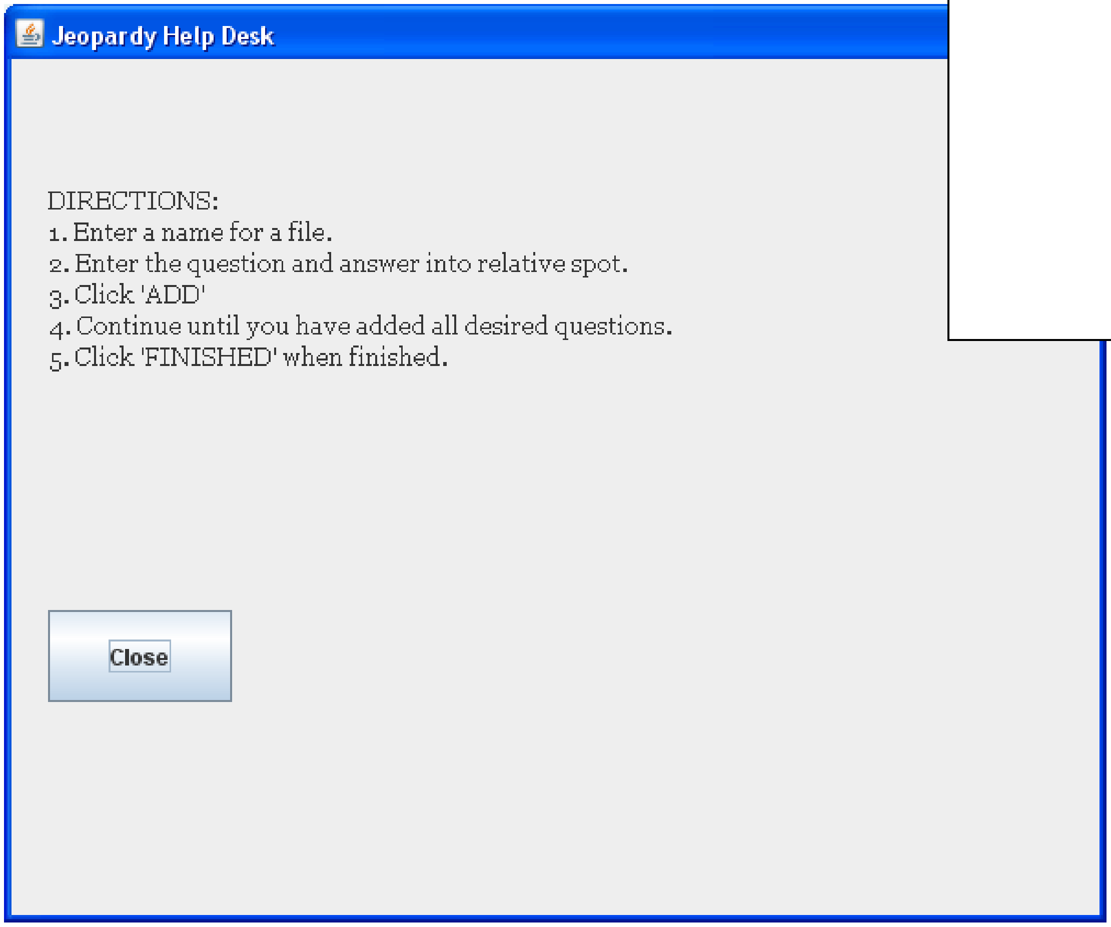


DIRECTIONS:
1. Select a topic
2. Click Begin
3. In game area, select a point value
4. Answer the question in the textfield and click submit.
5. After the game, if your score is in the top 10, then you will be prompted to enter your initials for the records.

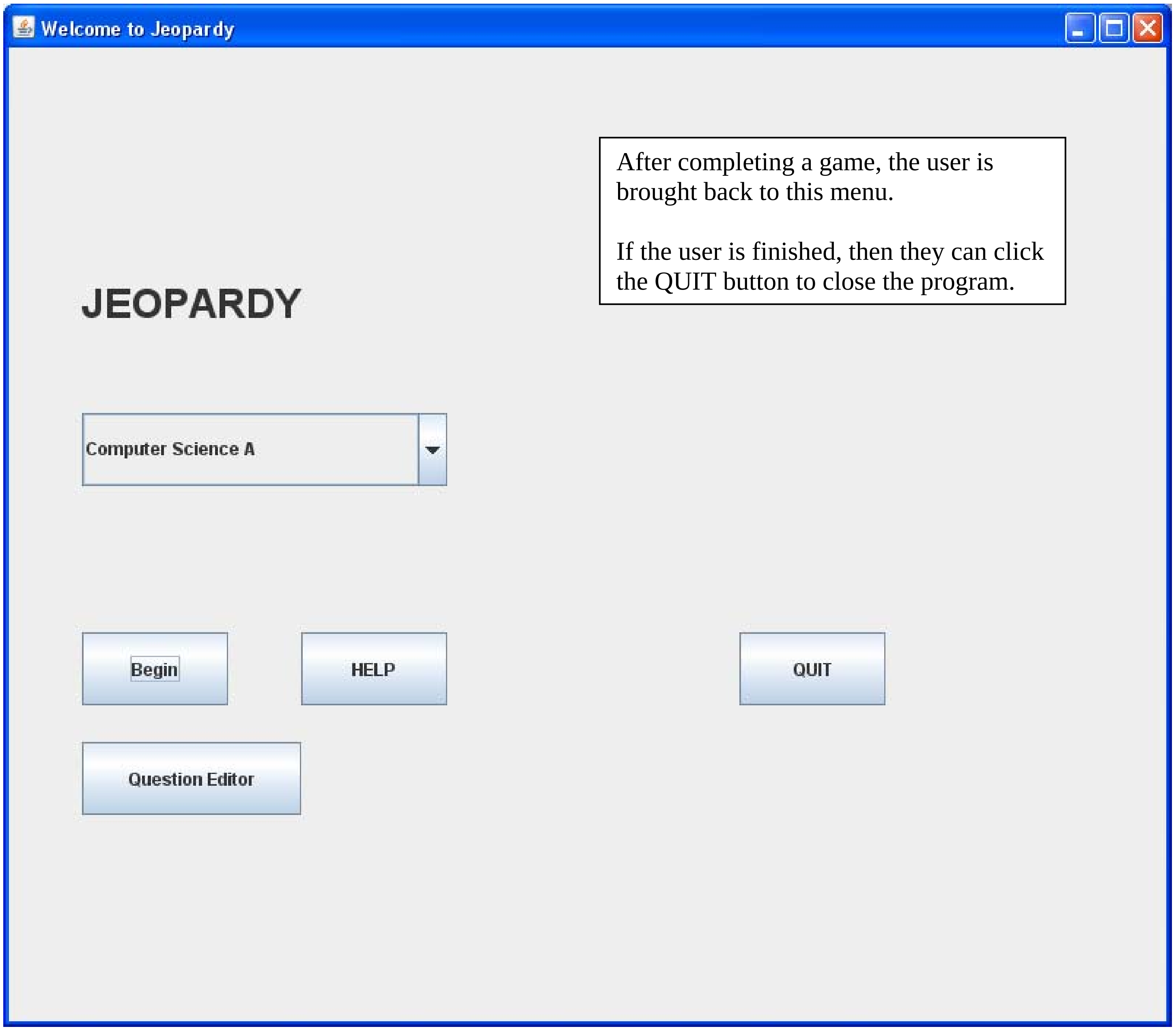
Here the user has selected a help desk option.

Information is provided.

User’s only option is to click CLOSE once finished.



DIRECTIONS:
1. Enter a name for a file.
2. Enter the question and answer into relative spot.
3. Click 'ADD'
4. Continue until you have added all desired questions.
5. Click 'FINISHED' when finished.

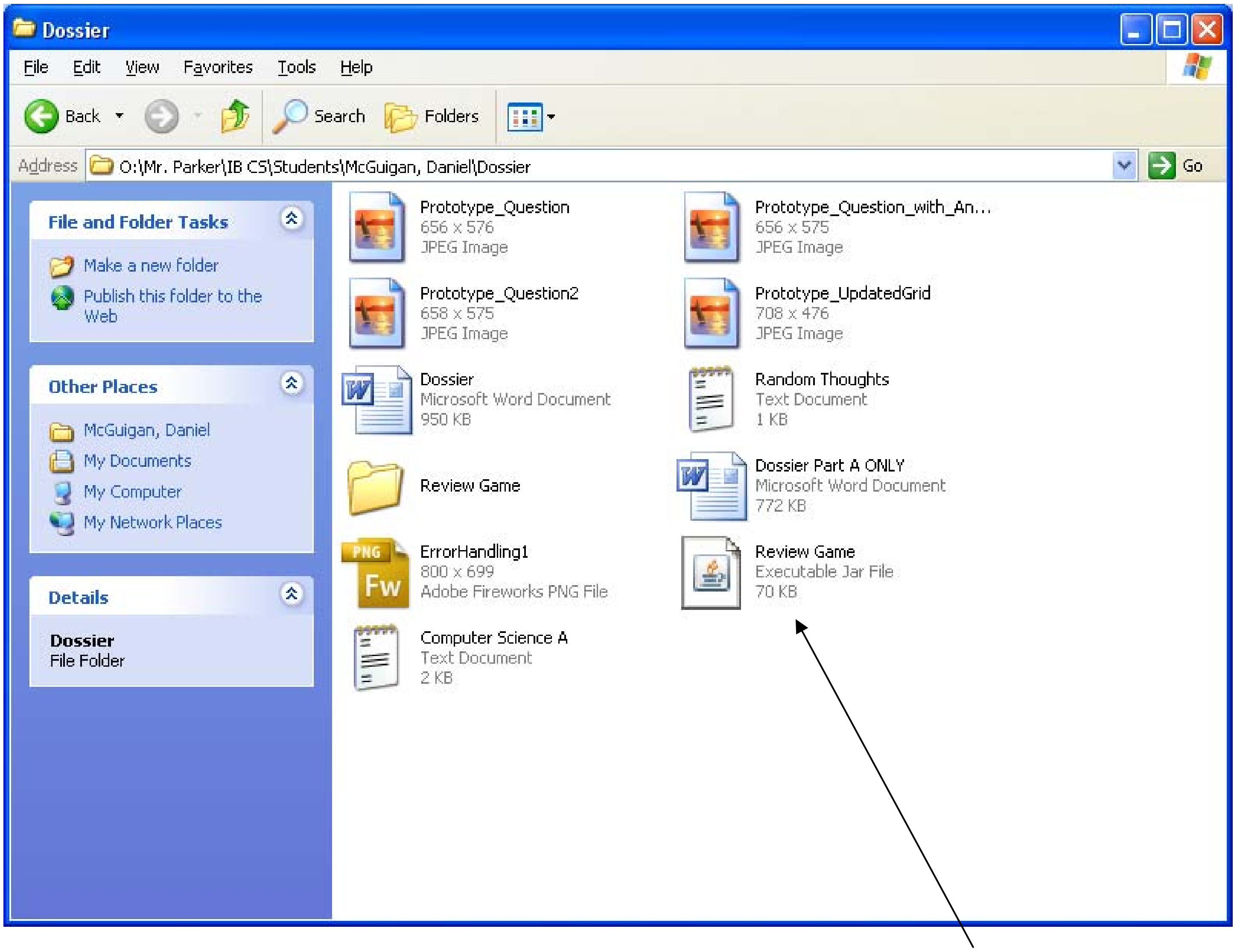


After completing a game, the user is brought back to this menu.

If the user is finished, then they can click the QUIT button to close the program.

D-2: Startup

To start up the program, the user will need to navigate to the folder that it is currently stored in.



O:\Mr. Parker\IB CS\Students\McGuigan, Daniel\Dossier

Dossier
File Folder

Prototype_Question 656 x 576 JPEG Image	Prototype_Question_with_An... 656 x 575 JPEG Image
Prototype_Question2 658 x 575 JPEG Image	Prototype_UpdatedGrid 708 x 476 JPEG Image
Dossier Microsoft Word Document 950 KB	Random Thoughts Text Document 1 KB
Review Game	Dossier Part A ONLY Microsoft Word Document 772 KB
ErrorHandling1 800 x 699 Adobe Fireworks PNG File	Review Game Executable Jar File 70 KB
Computer Science A Text Document 2 KB	

At this point, the user will need to double click on the Executable Jar File titled "Review Game". This will launch the program. This will bring up the startup window, which will then require the user to pick a game topic and then begin (See Test Run for screenshots on following references).

D-2: Conclusion/Evaluation

Successful?
The program does successfully execute and run fine. It begins running smoothly, with a simple double click on the shortcut, launching the main menu. From there, the user has free reign to use the program to all its intended capabilities.

Criterion Addressed?
It did successfully meet the criterion that were determined at the beginning of this process. Because of the way it has been implemented, it can be used both individually and as a group thing, allowing for variety in game play. Also, everything is rather clearly labeled so that even the slowest of kretins will still be able to use the program without much difficulty. At the very least, they will be able to find the help directions. Also, the questions are displayed randomly, and not repeated, thanks to a simple code algorithm. It generates a random integer, and then checks to see if it has already been used. If it has, then it generates a new number.

Was Success Limited?
So far as my testing has shown, the program has been able to handle a variety of data sets. It is simply a review quiz type program, so it reads in String files from a text file. What it is in the file is irrelevant; it is converted and stored as a string variable in the program.

Any Limitations?
Yes, there are some limitations. Because it is a 'free-response' style quiz, exactly identical answers are required in order to get credit for answering correctly. If one word is missing or incorrect, then credit will not be awarded.

Also, because of the way the Question Writer portion is written, it does not yet require the user to enter a minimum number of questions before being usable in the game. This occurrence will cause a problem with the game, but I am in the process of finding an effective means of preventing this that I can add at a later date.

Additional Features?
One thing that I had contemplated putting in that didn't make it was a record of the top 10 high scores. However, this missed being added in the current 'final' version due to time constraints and the need to eliminate the occurrences of exceptions.

Also, I am planning on creating a new and improved version of the Question Writer portion of the program. Its current condition works fine, but there is plenty of room for error that could be avoided by modifications in the coding.

Initial Design Appropriate?
Well, since I was initially using Applets, I'd say that it's a pretty safe bet that it wasn't. However, the overall idea lasted all the way to the final version. There is a startup window. From there, there is the game, and then from the game board, the user is taken to individual windows for each question. The final version is essentially an improved and expanded version of the initial design idea.

Future Enhancements?
There will most definitely be quite a few. As I mentioned above, the Question Writer class will have a priority in revamping. It really needs it.

Also, I am considering making the game follow a more multiple choice style game play, instead of free response. This will make it easier to ensure that people who know what they are talking about actually get credit for knowing the correct answer.