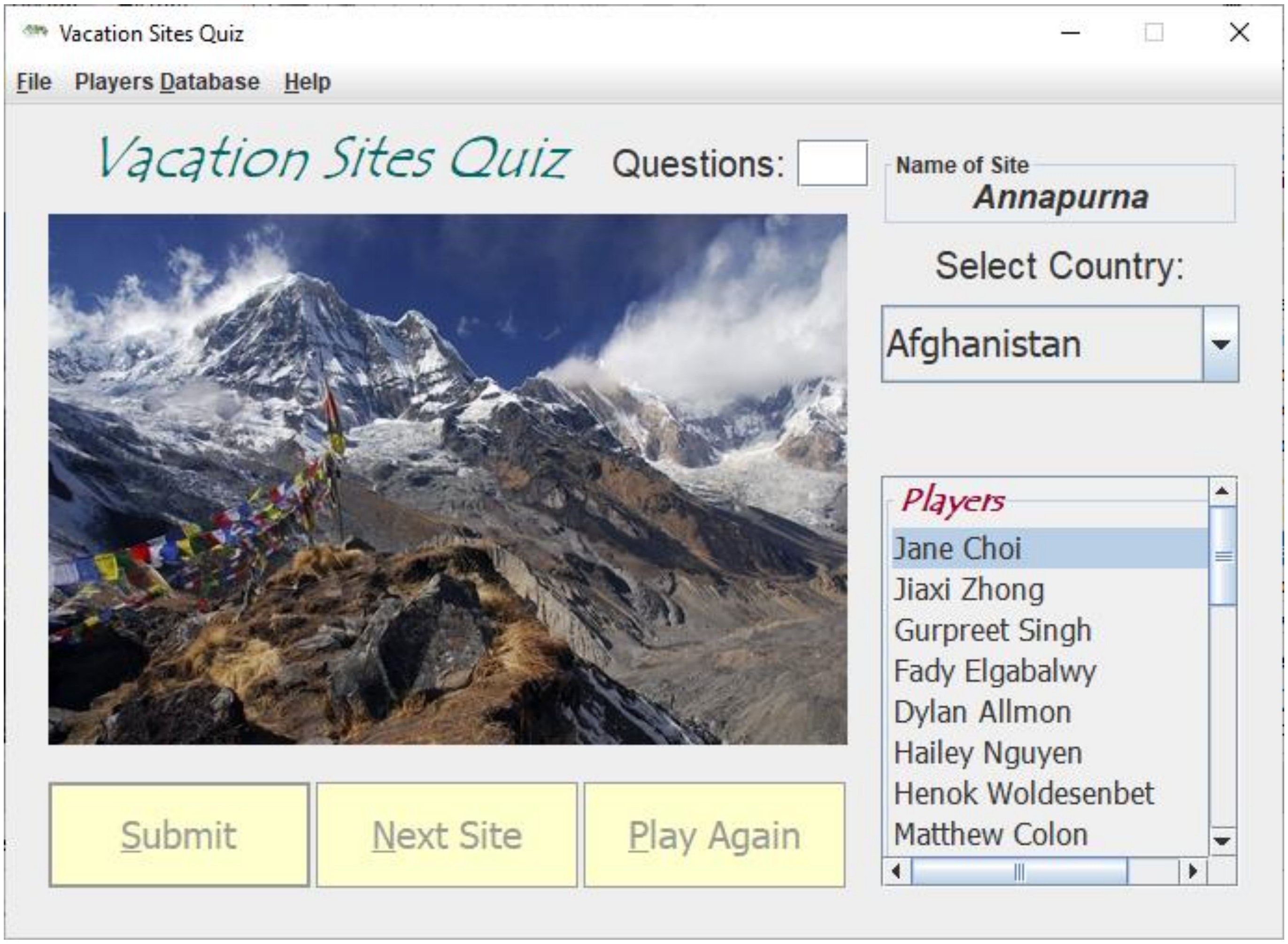
DUE: 10 Oct 2023

Project #1—Vacation Sites Quiz

Write a Java Swing GUI-based program (similar to the one shown below) to test the user's knowledge of location of famous vacation sites. The application displays a random picture of a vacation site, its name, and asks from the user to select country where it is located. The user is to enter a valid number of questions for the quiz and to select the correct country from a number of choices displayed in a countries JComboBox.



This project aims to review and reaffirm familiarity with some very important ADT (abstract Data Types): ArrayList, LinkedList, HashMap, TreeSet, BST and others. It is important that you review these classes before attempting this project. You will find the video recordings for Lab #8 and #10 from last quarter most helpful.

Initially, as the program loads, information about the vacation sites is read from comma-delimited text file, Sites.txt that contains the following private five fields of a Site class object: name, country, populations (in millions), capital, and area (in square miles). For each line read from this file a Site object is created and added into a LinkedList (sitesList) and a HashMap (sitesHashMap) maps the name (key) of each vacation site to the country (value) where the Sites is located. A partial list of this file might look like this:

```
The Blue Mosque,Afghanistan,34.66,Kabul,652864.0
Great Barrier Reef,Australia,24.85,Canberra,7692024.01
Schönbrunn Palace,Austria,8.8,Vienna,83879.0
Tiwanaku,Bolivia,11.2,Sucre,1098581.0
Christ the Redeemer,Brazil,208.6,Brasília,5767.0
Rila Monastery,Bulgaria,7.1,Sofia,110993.6
…
```

As the program starts, the user is asked to enter number of questions for the quiz (in need to be verified) after which a random image of a vacation site and its name are displayed. The user is asked to select the correct country where the vacation site is located and the answer is checked. The program counts and displays the number of correct answers.

Also when the program starts, players who are to take this quiz are read from an external text file, Players.txt. The comma-delimited player's in this file are: name, age, number of correct answers, and total number of questions form the required instance variables for a Player class, a subclass of an abstract Person class. Here is partial content of the Players.txt file; the names are the students in this class but the scores are fictitious--you could create one with names and ages you desire):

```
Jane Choi,19,7,9
Jixi Zhong,19,8,12
Gurpreet Singh,23,6,10
Fady Elgabalwy,18,1,4
Dylan Allmon,21,7,7
Hailey Nguyen,19,6,8
Henok Woldesenbet,19,1,6
…
```

Initially all players have 0 for both the number of questions and answers but as the quiz is completed the score for the player who took the quiz is updated and saved to the Players.txt file. The player can submit an answer only once and move to the next questions but can take the quiz multiple times with only the last take recorded (i.e., no history of the quiz taking is required).

Typical Splash screen, About form, PrintUtilities file for printing capabilities, File menu selection for new set of sites and players, players database management (Add, Edit, Search, Delete, Player Details and Site Details), Help (with About) and with multitude of classes specified below are all required for this project.

Project Requirements: Your program must contain and have the following features, components, classes and functionality:

1. Comma-delimited text file, Sites.txt, of vacation sites with five comma-delimited fields. The program starts with reading of this file, it creates Sites objects from the Site class and stores these Site class objects in a LinkedList. The text file Sites.txt looks like this:

```
The Blue Mosque,Afghanistan,34.66,Kabul,652864.0Great Barrier
Reef,Australia,24.85,Canberra,7692024.01
Schönbrunn Palace,Austria,8.8,Vienna,83879.0
Tiwanaku,Bolivia,11.2,Sucre,1098581.0
Christ the Redeemer,Brazil,208.6,Brasília,5767.0
Rila Monastery,Bulgaria,7.1,Sofia,110993.6
...
```

Create a second smaller Sites_1.txt file for the New menu option and for easier debugging.

The following line must appear in your program (in the main GUI class) as a declaration of main SitesGUI class instance variable:

```
private List<Sites> SitesList = new LinkedList<Sites>();
```

2. Comma-delimited text file of players, who will take the quiz. This file is also read as the program starts with Player objects from the derived Person class saved in a binary search tree of Player object. The text file Players.txt looks like this:

```
Jane Choi,19,7,9
Jiaxi Zhong,19,8,12
Gurpreet Singh,23,6,10
Fady Elgabalwy,18,1,4
Dylan Allmon,21,7,7
…
```

3. The database management for the Player objects (Add, Edit, Search, and Delete) is done by methods in a required binary search tree data type. The following line must also appear in the main SitesGUI class as a declaration of main class instance variable:

```
        private BinarySearchTree playersTree = new BinarySearchTree();
```

4. The names of the Players only are displayed in a playersJList. The user of the quiz selects number of questions for the quiz and a player from this JList before the quiz starts. This selection cannot change until the quiz is completed at which stage the score is recorded and saved for the selected player. If Clear is selected from the File menu before the quiz finishes, the quiz is cancelled with no recording for the incomplete attempt and the playersJList is enabled for another player to be selected.
5. The following are required classes (your project may contains others, especially if you attempt the extra credit options):
    a. SitesGUI: main JFrame class acting as a driver and quiz user interface.
    b. Site class with the following instance variables, default, overloaded and copy constructors, setters, getters, toString, and equal methods:

```
private String name;
private String country;
private float population;
private String capital;
```

```
 private float area;
```

This class should implement the Comparable interface and override the compareTo() method in order to use a TreeSet which will contains unique and sorted countries (to be copied in the countriesJComboBox).

c. This class should also implement the Serializable interface in order to read/save Sites objects from/to file (optional, extra credit).
d. Splash Screen with a large image, and % progress bar.
e. About JDialog form with type Utility. It should describe in great detail the project in a non-editable JTextArea.
f. PrintUtilities class to print the main form as a GUI.
g. Validation class with at least one static boolean isInteger() method to verify the user entry for number of questions is valid (> 0 and <= number of sites in the text file). You may start with an existing Validation class from the CS 142 labs.
h. Person abstract class to act a parent to the Player class. It should have the following instance variables:

    ```
    protected String name;
    protected int age;
    ```

    In addition, the Person class needs two constructors, setters and getters, an overridden equals and toString methods. The Person class must be abstract even though it need not have an abstract method. For the purpose of saving a Player in the Player.txt file correctly consider the following simple line for the toString method:

    ```
    return name + "," + age;
    ```

i. Player class, subclass of the Person class that implements the Comparable interface. It should have the following instance variables:

    ```
    private int correct;  //number of correct answers
    private int totalQuestions;
    ```

    The Player class needs three constructors, setters and getters, calculatePercent, an overridden equals, toString, and compareTo method. And again, for the purpose of saving a Player in the Player.txt file correctly consider the following simple line for the Player’s toString method:

    ```
    return super.toString() + "," + correct + "," + totalQuestions;
    ```

j. PlayerDetails JDialog form that displays all fields of the selected player, including the percent of correct answers and an image of the Player.
k. SiteDetail JDialog form that displays the contents of a text file that all fields of the shown site, and an image of the site’s country.
l. AddPlayer JDialog form to add a new player to the database. Do not allow duplicate players to be added in the DB and validate all of the required fields.
m. EditPlayer JDialog form to edit the selected player. As with the AddPlayer, the edited player should be saved in the database with validated fields and without duplication.

n. BinarySearchTreeNode class—same BST Node class as in lesson 19 examples with modification to include Player for the data instead of integer.

o. BinarySearchTree—data structure for the Players. Again use but modify the BST from lesson 19 examples by replacing all int data references to Player data type with appropriate change for comparisons.

   1. The BST class should contain at least these instance variables:
      `private BinarySearchTreeNode root;`
      `StringBuilder buffer = new StringBuilder("");`
   2. In addition to all the methods already in the BST class, add at least the following Boolean method: `public boolean contains(Player player)`—a boolean method to determine if a player is in the BST. It relies on the method `nodeWith(Player data, BinarySearchTreeNode node)` of the BST:

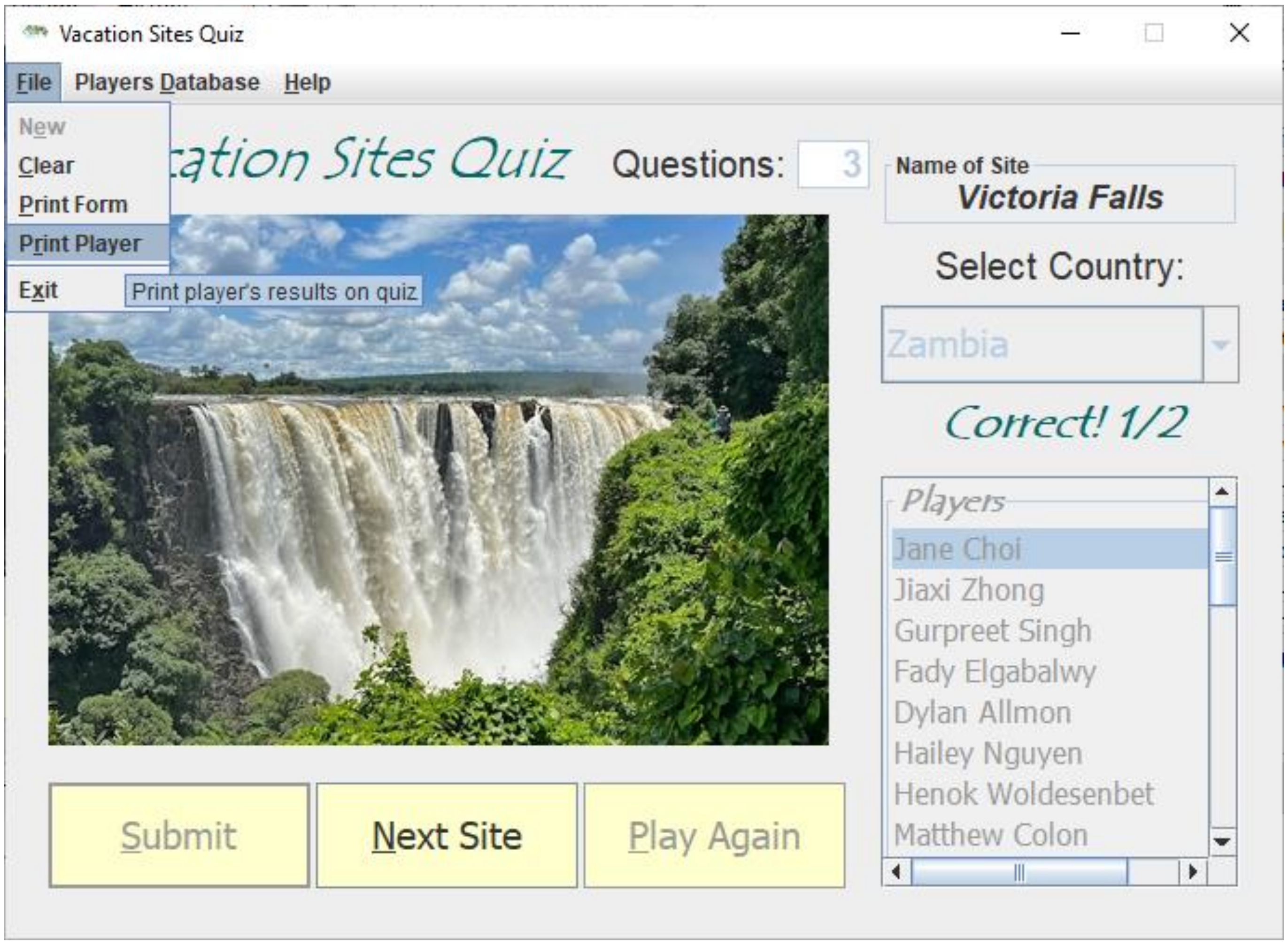
```
    public boolean contains(Player player)
    {
      if(nodeWith(player, root) == null)
          return false;
      else
      {
          BinarySearchTreeNode foundPlayer =
               nodeWith(player, root);
          return (foundPlayer.data).equals(player);
      }
    }
```

   3. `public String getBuffer()`—a method to return the buffer instance variable as a String.

6. To determine if the player correctly selects the country for the given vacation site, use a HashMap that maps the name of the site (key) to the country where it is located (value). The following must be used as class instance variable in the SitesGUI class:
   `private Map<String,String> SitessHashMap = new HashMap<String,String>();`
   The keys will be the sites (String) and corresponding mapped values will be the countries (String). Thus, when a key is given to the HashMap (randomly and without repetition for the duration of the quiz), the country will be easily determine with the HahsMap's get method. How to obtain an unused random state is one of the main challenges for this project (and solved for you in Lab #10 last quarter)!

7. After the selected player enters a valid number for number of questions (needed to be validated) on the quiz, a random site is selected from the LinkedList (use the get method of the LinkedList with the same random number generated), picture of it is displayed, its name is shown and the Submit button is enabled. After the player selects a country from the countriesJComboBox and presses the Submit button, you might get something like this (with a different vacation site and picture, most likely). Note that the result with Correct/Incorrect is displayed with a count for correct out of

total questions answered thus far. Note how the submit button triggers the decision for correct or incorrect answer, disables itself and enables the Next Site button when submit is clicked; likewise, the Next Site button will display a different picture of another unused vacation site and its name, disables itself and enables the submit. This continues until the end of the last question when the play again button comes to life and the other two are disabled.



Hints and advice:

1. Two print capabilities are required: print the form as a GUI and a second print with Player's details.
2. Appropriate classes and data structures as required above (see discussion on classes above). Specifically, you are NOT to use any arrays and you must use the LinkedList of Site class objects, BinarySearchTree for Players, and HashMap with name/country pairs as keys/values for determining correct answers.
3. Javadocs, description of the program, and comments everywhere.
4. Menus that synchronize with corresponding buttons and with at least the following menu choices:

- File menu with menu items: New (JFileChooser to select a new set of players or select new vacation sites with countries data), Clear (cancel and reset current quiz), Print Form, Print Player, and Exit menu items.
- Players Database with the functionalities to Add New Player, Edit Player, Delete Player, Search Player, Player Details and Site Details menu items.
- Help with About menu item for a meaningful and detailed About form.

5. Read the vacation sites and players text files, create appropriate objects and fill the LinkedList with vacation site objects and the BinarySearchTree with Player objects. A JFileChooser should be the most appropriate control to deal with which files to display (you might want to set up different directories for sites files and players files).
6. Consider using a JFormattedTextField for the number of quiz questions with keyTyped event handler that will allow only digits, backspace, and the delete key. Valid values for number of questions for any given quiz are integers in the range [1, number of vacation sites in the file], inclusive.
7. The AddPlayer JDialog form should only create a player with valid fields—use the Validation class for this purpose. The program should not allow duplicate players in the database.
8. Include ToolTip text for the number of questions, menus, buttons and menus at least, explaining their function and restrictions.
9. Confirm deletion of the selected player. Deleted player should be removed from the Players BST and the Players.txt file.
10. Provide appropriate icons and pictures that tie all forms together.
11. The forms should not be resizable or maximizable (although one should be able to minimize the main form).

Additional Hints and Consideration:

1. This project has an objective to review some of very important abstract data types examined in detail in CS 142: our own built Binary Search Tree from Lesson 19 (and the book's variation in Chapter 25), LinkedList and HashMap from Lesson 20 (and in chapters 20, 21 and 24), and TreeSet from Lesson 20 (and in chapter 21) . A solution to a very similar problem and helpful for this project, Lab 10—Famous Artists Quiz, is available and ***you are strongly urged to re-examine it***. Note however, even though this project can be easier programmed with arrays and ArrayLists, in this project no arrays or ArrayLists for the players and sites data structures are allowed. The only ArrayLists you may use are the Boolean and Integer ArrayLists used in selecting a unique random integer which in turn determines which state is selected.
2. The following are required data structures to be used in this project—the names are suggestive of the data they contain:

```
private BinarySearchTree playersTree = new BinarySearchTree();
private List<Sites> sitesList = new LinkedList<Sites>();
private Map<String,String> sitesHashMap = new HashMap<String,String>();
private ArrayList<Boolean> sitesUsedArrayList = new ArrayList<Boolean>();
private ArrayList<Integer> numbersArrayList = new ArrayList<Integer>();
```

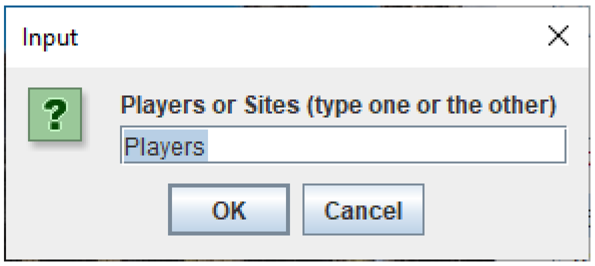
3. The crucial part for successful quiz simulation is to select a vacation site randomly and not to repeat it for the duration of the quiz. You must use a HashMap that maps the vacation sites (keys) to the countries (values) in order to determine if the player selected a correct answer. The difficulty is that unlike List classes, the HashMap (and HashSets) class does not provide any methods in which we can get the elements using their index. This is the reason why you are required to use LinkedList for the vacation sites; LinkedLists are indexed.
4. In Lab 10—Famous Artists Quiz from last quarter the selection of such unique random integer was accomplished in two different ways:
    a. with a “blind” method getUniqueRandomNumber which used parallel boolean array that kept track of which indices where already chosen so not to repeat, or
    b. with a more efficient method getUniqueAlternate that used an array of integers with the last available index replacing the selected index and reducing the range of random integers.
5. You may use either one of these two method by replacing the arrays with ArrayList or you can use the following third method:
    a. Once you create the vacation sites HashMap (required), create a set of keys with something like this:
       ```
       Set keys = vacation sites.keySet();
       ```
    b. Create a temporary LinkedList (legal because it is a LinkedList and not an array or ArrayList) that will contain the keys, like this:
       ```
       List list = new LinkedList();
       list.addAll(keys);    //make sure you clear list first
       ```
    c. Then select a random state with a random integer, `randomNum,` in the range 0-number of sites on file - 1 on the list (which has an index):
       ```
       String randomState = list.get(randomNum.toString();
       ```
    d. Or, alternatively, pass to the list the shuffle method of the Collections class (but making sure to remove the first element after each selection):
       ```
       Collections.shuffle(list);
       String randomState = list.get(0).toString();
       ```
6. Once initially read (a call to a `readSites(SitessFileName);` method in the constructor), the LinkedList of Sites objects does not need to be modified. Because it is indexed, the LinkedList should be used to determine which state is chosen randomly and without repetition for duration of the quiz.
7. The BST that contains the players on the other hand needs to be updated and changed often. After the players are initially read and stored in the BST (again with a call to a method `readPlayers(playersFileName);` in the constructor), the `playersTree` BST is updated when a new player is added or edited, when the quiz ends (to record player’s score). You should provide a method to save players for these updates not only in the BST but also in the external Players.txt file: `private void savePlayers(String playersFile).`
8. The `savePlayers(playersFileName);` method should be called after:
    a. Deleting a player.

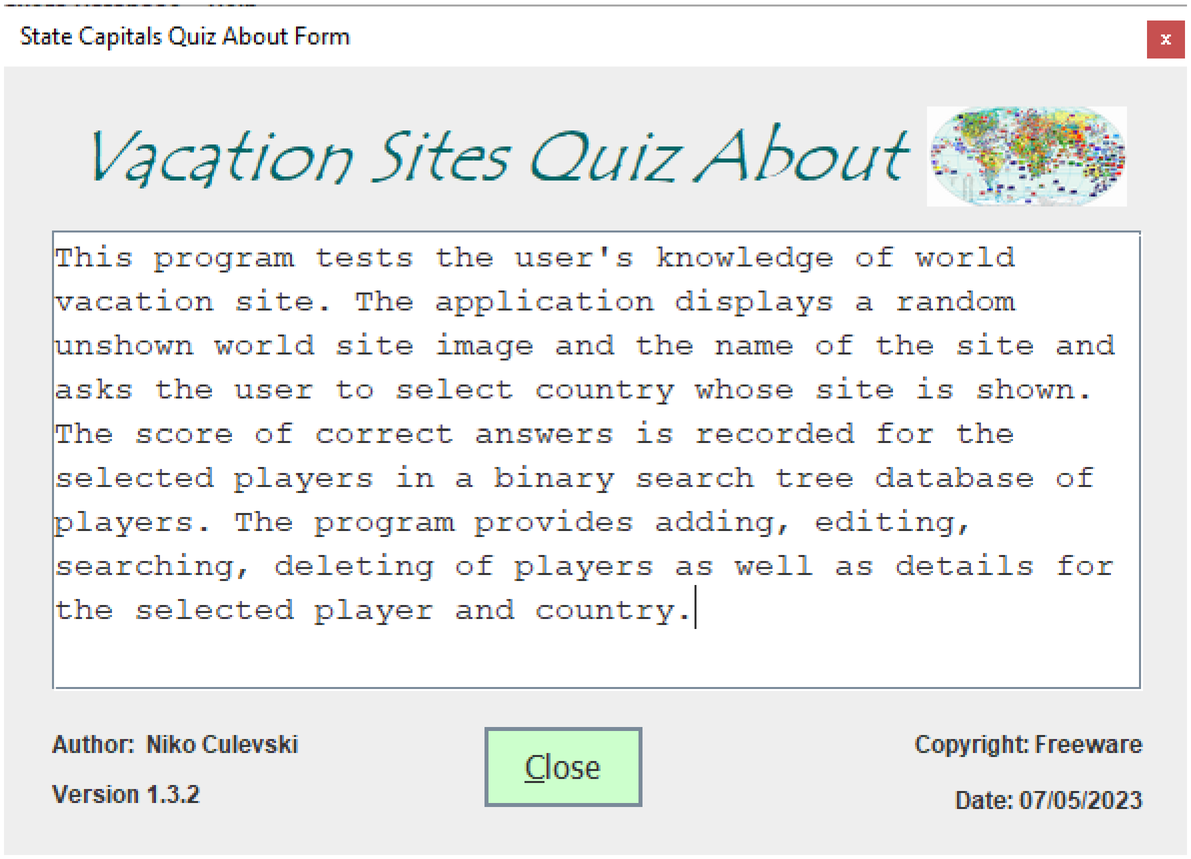
   b. Play Again JButton’s event handler when the selected player’s score is updated.
   c. Add and Edit players.
9. The following changes/additions are strongly suggested for the TreeNode and Tree classes from Lesson 19 Examples (in Canvas):
   a. Rename the two files to `BinarySearchTreeNode` and `BinarySearchTree`.
   b. Change the data type from `int` to Player and make appropriate changes in all methods that use the data.
   c. Add a class instance variable buffer to be used for updates to the Players.txt file: `StringBuilder buffer = new StringBuilder();`Provide get and set methods for this buffer variable.
   d. Write a `public void buildBuffer(BinarySearchTreeNode node)` method to recursively build the buffer which will contain the text of the Players.txt file.
10. Special problem arises for filling up the countriesJComboBox: after you read the Sites.txt text file and create a LinkedList of Site objects you will need to cycle through this SitesList, select the county only and add it to the countriesJComboBox. The problem is that there are several vacation sites in the same country and you do not want repeat a country in the countriesJComboBox. Furthermore, it is more convenient if the countries are listed alphabetically. Hence, creating a TreeSet of countries is ideal solution: as a set it allows adding only unique entries and it sorts them! So I suggest creating a fillComboBox() method called also in the constructor of SitesGUI after reading the `readSites(SitessFileName);` and `readPlayers(playersFileName);` methods.
11. Finally, here is an ordered algorithmic path for what to do:
    a. Design the main SitesGUI JFrame. Build and add all the support and utility classes: Person, Player, Sites, Splash, About, PrintUtilities, Validation, AddPlayer, EditPlayer, PlayerDetails, BinarySearchTreeNode, and BinarySearchTree.
    b. Add the data text files: Players.txt and Sites.txt and place the m in separate packages. For test purposes, have a Players_1.txt and Sitess_1.txt files with only few lines. Add images of the vacation sites and images of players (fictitious) for players’ details.
    c. In the constructor, read the vacation sites (save in a LinkedList and create a HashMap in the same read method) and read players (save in BST) and populate the data structures. Call fillComboBox method to populate the countriesComboBox with unique countries in alphabetical order (TreeSet comes very handy). Also disable all three buttons and set focus to number of question, numberJFormattedTextField.
    d. In the numberJFormattedTextField’s ActionPerformed event handler, validate the input, enable certain components (submit button, answersJTextField, playersJList) and call displaySites method to display a random unused picture of a vacation site and show its name.

e. The displaySites method calls getUniqueRandomNumber or getUniqueAlternate method to generate a random number (0-number of sites in file -1) that is used to get a vacation site from the SitesList LinkedList. The sitesHashMap produces the correctCountry (class variable) and the image of that site is placed into the sitesJLabel.
f. The user then has a chance to select a player, select the answer for the countriesJComboBox whose site's name and picture are shown. The Submit button is enabled for the player to commit to an answer.
g. The checking for correct answer is done in the Submit button event handler: the player's answer is compared to the sitesHashMap's get method for the selected site.
h. The rest of operations, including enabling and disabling components, updated counters, when the quiz ends, and so on is pretty much the same as in Lab 10—Famous Artist Quiz. However, the player's score needs to be updated at the end of the quiz. You can do this by removing the old player from the BST and inserting another version of the same player with updated scores—a very inefficient way of updating the database, a problem to be addressed with our project 2 which will use SQL and JDBC.
i. Find below a screen capture of methods and event handlers in just the SitesGUI class in my solution to the project—it might prove helpful.

Extra Credit: Provide a Statistic class which displays the mean, median, and standard deviation of the percents of correct answers of all the players who took the quiz (and omit those who have not). Upon request, the same Statistic class with provide the details of the top 3 Players (wit highest correct percents). Additional extra credit points can be earned by replacing the text files reading and writing with binary files by reading and writing objects such as LinkedList and BST. For more extra credit, decide on a good strategy of what to do if a player retakes a quiz: do you keep record of all attempts and how to do so. Do you replace the previous attempt and only if the score is better?

Screen Captures:





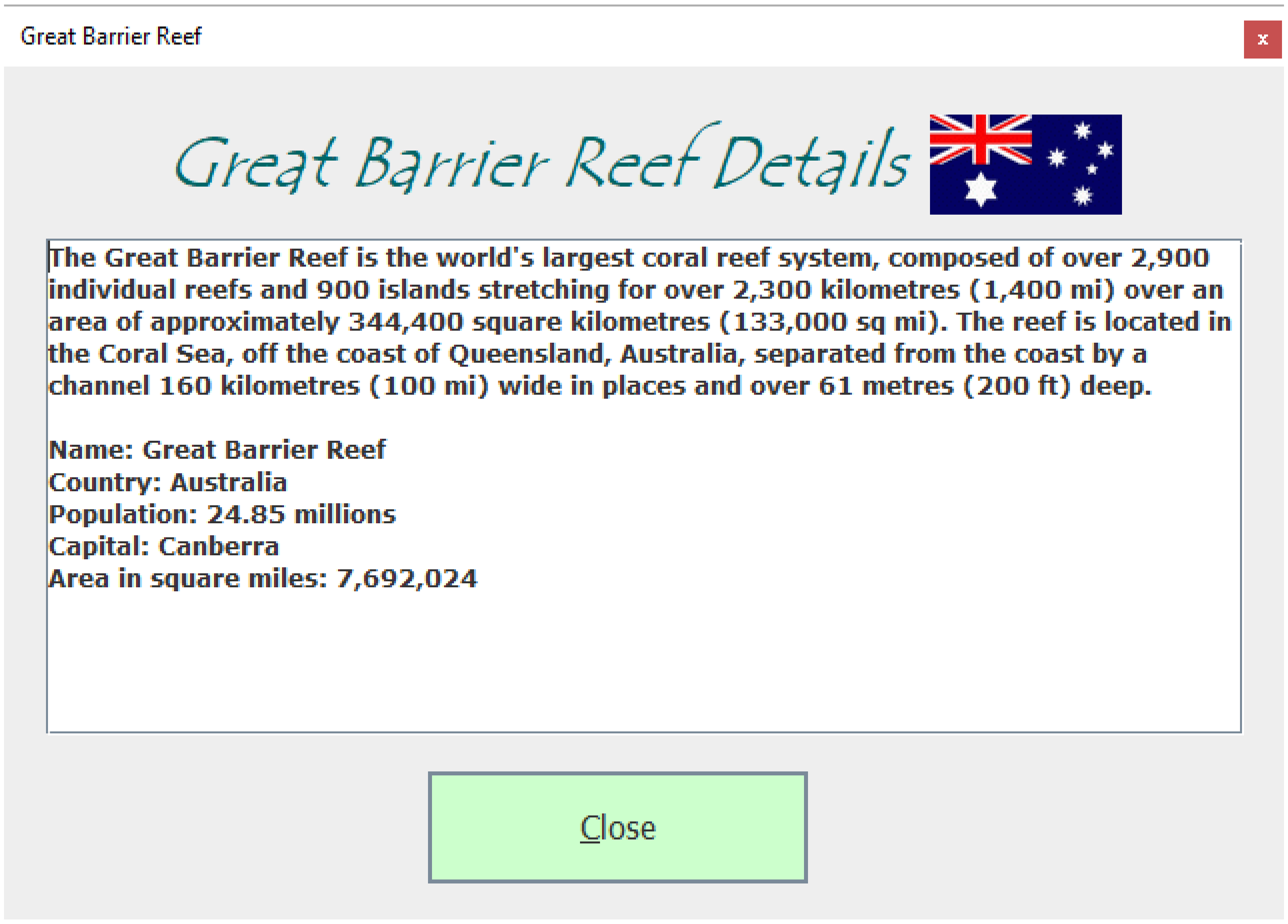
This program tests the user's knowledge of world vacation site. The application displays a random unshown world site image and the name of the site and asks the user to select country whose site is shown. The score of correct answers is recorded for the selected players in a binary search tree database of players. The program provides adding, editing, searching, deleting of players as well as details for the selected player and country.

Author: Niko Culevski

Version 1.3.2

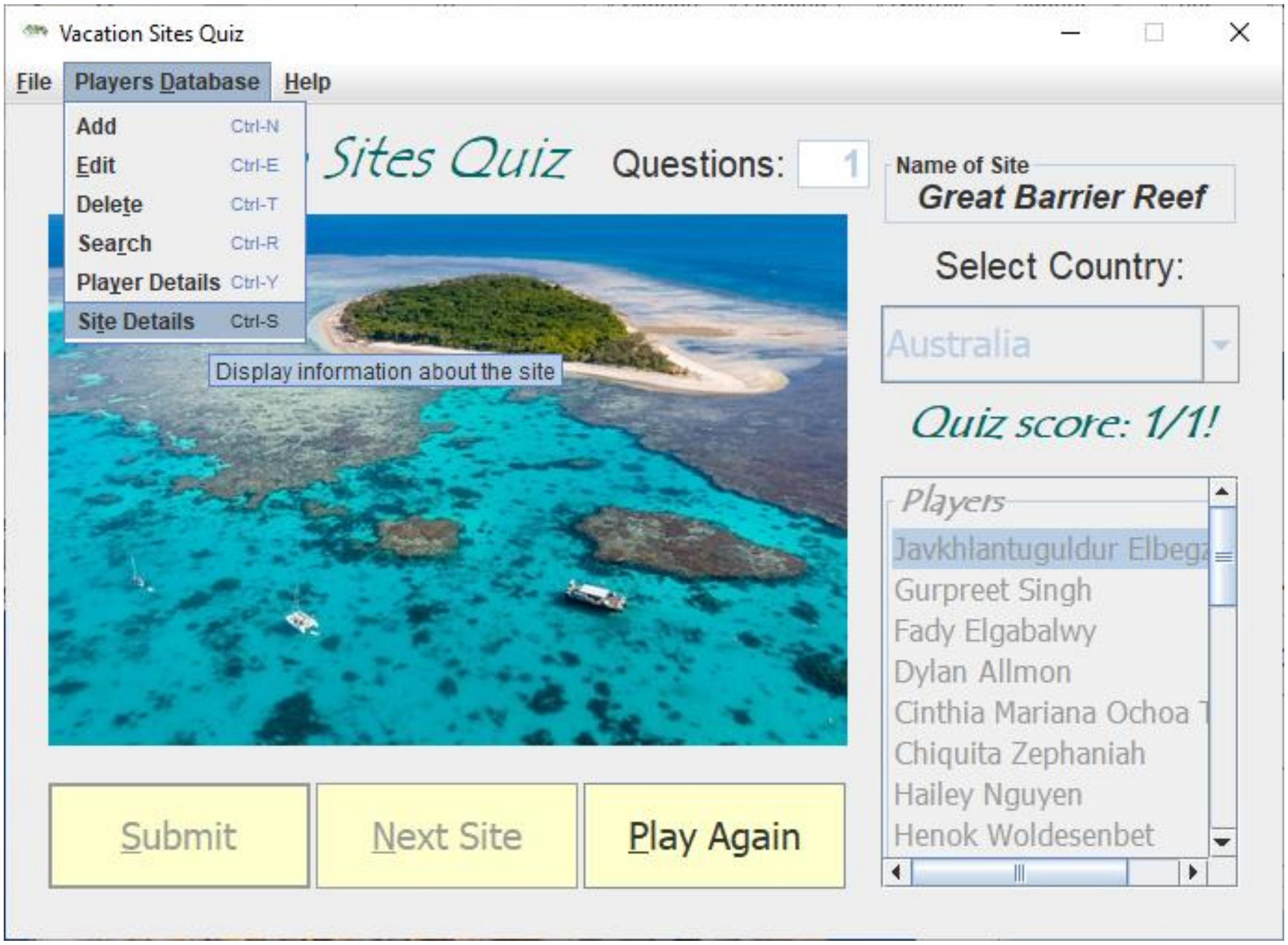
Copyright: Freeware

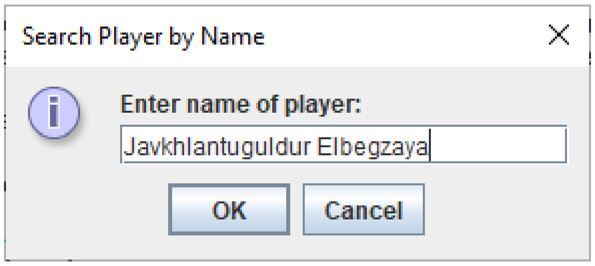
Date: 07/05/2023

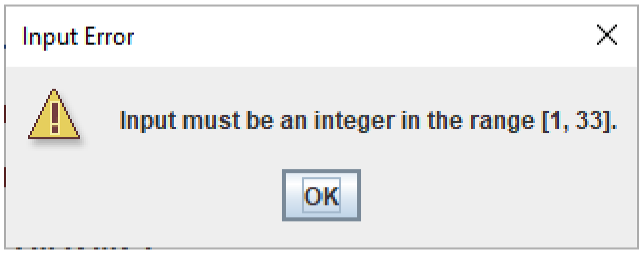


The Great Barrier Reef is the world's largest coral reef system, composed of over 2,900 individual reefs and 900 islands stretching for over 2,300 kilometres (1,400 mi) over an area of approximately 344,400 square kilometres (133,000 sq mi). The reef is located in the Coral Sea, off the coast of Queensland, Australia, separated from the coast by a channel 160 kilometres (100 mi) wide in places and over 61 metres (200 ft) deep.

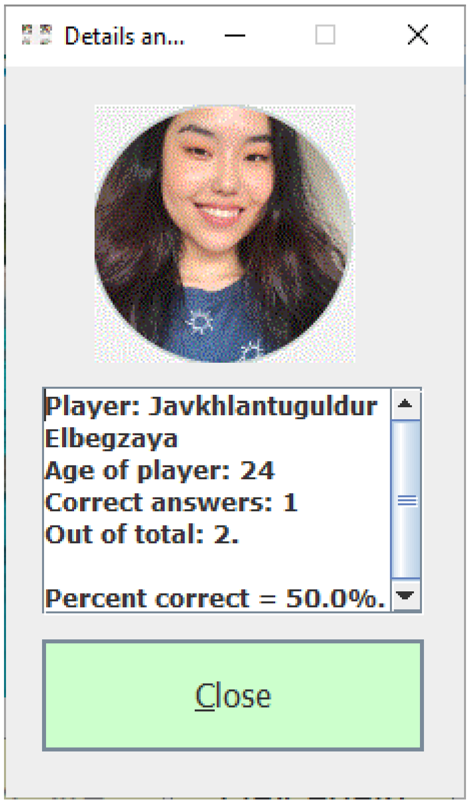
Name: Great Barrier Reef
Country: Australia
Population: 24.85 millions
Capital: Canberra
Area in square miles: 7,692,024





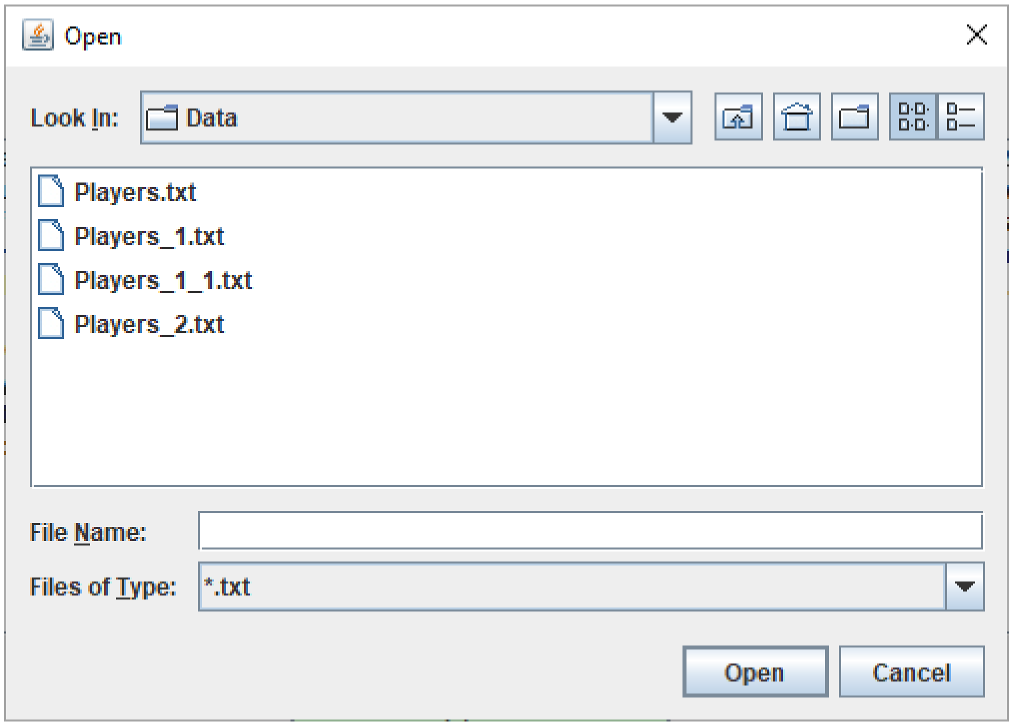


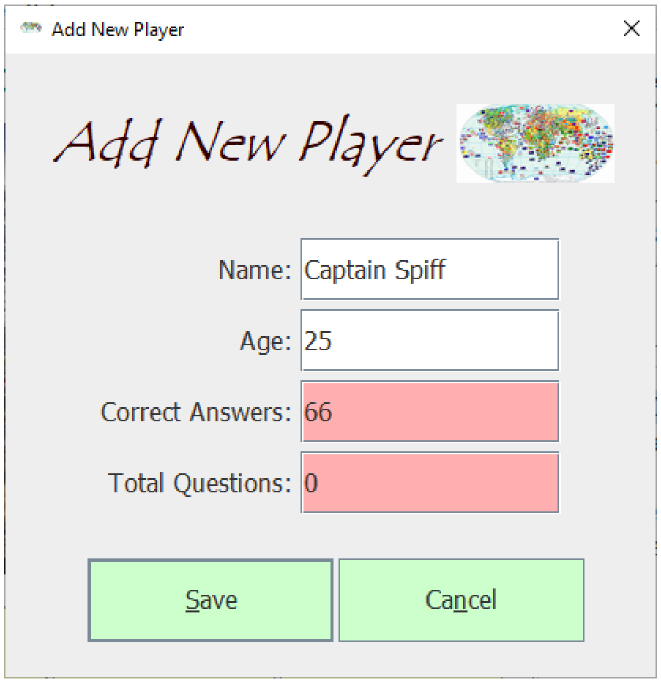
Input must be an integer in the range [1, 33].

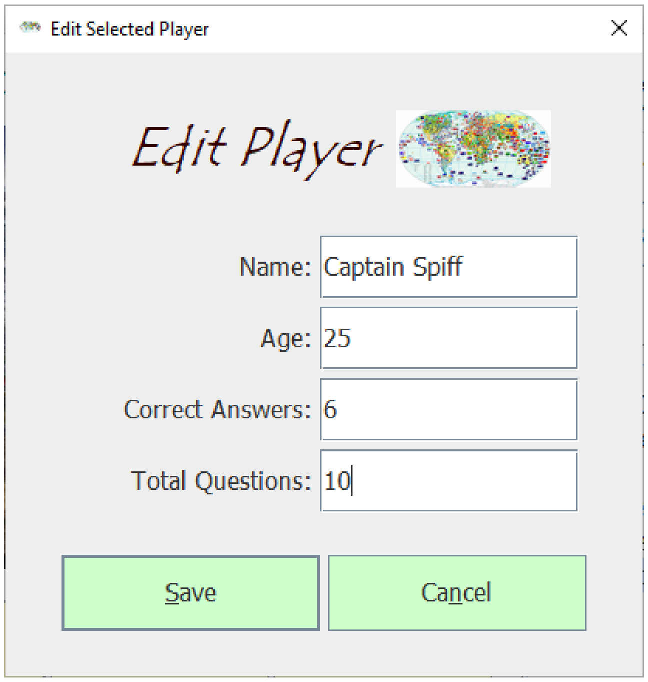


Player: Javkhlantuguldur Elbegzaya
Age of player: 24
Correct answers: 1
Out of total: 2.

Percent correct = 50.0%.







You might find these partial comments worthwhile—each counts as a -1 point.

```
/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
'Comments by the prof:
'Great effort.  Here are suggestions for improvement:
'1. Use Javadoc comments throughout the program for all classes and generate them.
'2. Missing some of the required classes.
'3. Need two printings: printing of form and printing of player.
'4. Must use HashMap to determine if user answer is correct.
'5. Make the info JTextArea in About form read-only.
'6. Validate input--do not allow illegal input.
'7. It is good style to tab code inside a method.
'8. Confirm deletion of player.
```

```
'9. Declare all constants as final.
'10. Read vacation sites and players from external text files.
'11. Missing LinkedList, HashMap, TreeSet or BST class.
'12. Name the classes and project appropriately.
'13. Missing PlayerDetail class with picture and details of player.
'14. Unable to Add new player or new Player is not saved to file.
'15. Unable to Edit selected player.
'16. Include a separate Validation class for validating input.
'17. Enable menu choices that synchronize with buttons' functionality.
'18. Not allowed to use arrays.
'19. Illegal use of ArrayLists.
'20. Disable maximization or resizing of all forms.
'21. Add a meaningful About JDialog form of type Utility which describes the project.
'22. Add a Splash screen with % progress bar displayed in center.
'23. Add icons/images to relevant forms.
'24. Missing site details form.
'25. Do not change the player in the middle of a quiz.
'26. Clear should reset the quiz.
'27. Select a new set of players and sites with JFileChooser when File->New is
selected.
'28. Not selecting unique random vacation site--they repeat on a given quiz.
'29. Enabling and disabling of JButtons and menus is not correct.
'30. Search for players does not work.
'31. Player's statistics are incorrect.
'32. Player's results are not saved to file.
'33. Program crashes on empty.
'34. Do not allow adding duplicate players.
'35. Do not allow duplicate countries in the countriesJComboBox.
'36. Unable to load new set of Vacation sites.
'37. Incomplete.
'38. Late.
'
'The ones that apply to your project are:
'2, 13, 30, 35
'
'26+1=27/30
'~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*/
```