AssessedExerciseTemplate-miniproject-lvl7-Copy1

December 3, 2020

1 Miniproject Exercise ‘Pointless’ Answer Quiz

2 Level 7

2.1 Ifthy Fairoze

2.2 25 November 2020

2.3 Version 1

2.4 Summary of the Question

Write a short answer quiz procedural program where the player gets points based on the number of members of the public (out of 100 people asked) who gave that answer (for the purposes of testing you can make up the response numbers). The aim is to choose an answer that as few people as possible thought of so gain as FEW points as possible (eg Name UK olympic track and field gold medal winners. … Suppose 78 people named Dina Asher-Smith, 42 named Sebastian Coe, and 2 named Fatima Whitbread, then an answer of FatimaWhitbread gains 2 points and is the best answer. Wrong answers gain 100 points.)

Level 7 - Contains all the criteria from Level 1 (F-) to Level 7 (A+).

2.5 The literate program development

Building on the what I have learned in from the ECS401U Jupyter Notebook, to answer the question, I created twenty four methods for this miniproject. For each method, I will explain about the method, what is does and how I tested it and provide the code of the method itself. Next, I show how to run the solution to the question. Each code fragment can be run in Jupyter notebooks, here. Finally, I show the full program that can only be run outside of notebooks, along with instructions on how to do this and a summary of the structure of the full program.

2.5.1 inputString

What it does The method inputString allows the code to ask a question to the user where they can answer it in the next line. It then stores what is inputted by the user to be used later on in the code.

Implementation (how it works) The first line in the code defines the method and calls it inputString. This method returns a value as it is not void. The string message parameter allows the programmer to write a question or message when calling the method. In the next line we link the keyboard to the code to input data from the real world (by using the keyboard). We then declare a variable called answer as string type. We then print out the message variable that the programmer writes when calling the method as an argument in the brackets. Then in the next line nextLine() allows the user to enter the answer where it reads the answer and stores it in variable answer. Then at the end of the method we return the answer variable and the input data held in it. Note that you have to import scanner in the full program.

[1]:
```java
/* This allows the programmer to call this method and ask a question or message
as its argument and allow the user to answer the question in the next line.
Then the input data can be used further on in the program when returned. */
public static String inputString(String message)
{
    Scanner scanner = new Scanner(System.in);
    String answer;
    System.out.println(message);
    answer = scanner.nextLine();
    return answer;
} // END inputString
```

Testing

[2]:
```java
System.out.println(inputString("What is the answer?"));
```

```
What is the answer?
I don't know
I don't know
```

2.5.2 inputInt

What it does This method prints out a question to the user, then the user types an integer response and returns that response as a result of this method.

Implementation (how it works) It sets up a local scanner to the method, linked to the keyboard from the first line. This then makes avaliable the method scanner.nextLine(). Then the question is printed out to the user. Then the user is able to type an String response of an integer using nextLine. This response is read and stored into textInput variable as a String value. Then the value in textInput is then converted to a integer using Integer.parseInt and stored in numberInput. Then the value in NumberInput is returned as a result.

[3]:
```java
/* This method asks a question to the user and then waits for the user to type␣
 ↪an integer response which is
returned as a result of the method. */
public static int inputInt(String message)
{
    Scanner scanner = new Scanner(System.in);
    String textInput;
    int numberInput;
    System.out.println(message);
    textInput = scanner.nextLine();
    numberInput = Integer.parseInt(textInput);
    return numberInput;
} // END inputInt
```

Testing

[4]:
```
System.out.println(inputInt("What is the answer?"));
```

```
What is the answer?
3
3
```

2.5.3 createQuestion

What it does This method creates the question records of the new type QuizQuestion to get the required field data of the new question, the answers and their point scores.

Implementation (how it works) A new class is needed to create the new type QuizQuestion and the required fields which are question, answer_1, point_score_1, answer_2, point_score_2, answer_3 and point_score_3. So this method is given the question, answers and their points as arguments storing them in the parameters shown in the code. It first creates a new QuizQuestion value storing it in a variable called question_1. Using the dot notation it directly sets the fields in the local record question_1 to the values stored in the parameters when given as arguments. It then returns the record made by returning the value stored in the variable, question_1. This is a initialise method to set the question record.

[5]:
```
// This is a new type QuizQuestion to sort the questions and answers and points␣
 ↪into records.
class QuizQuestion
{
    String question;
    String answer_1;
    int point_score_1;
    String answer_2;
    int point_score_2;
    String answer_3;
    int point_score_3;
}

/* This method creates and intialises a question record by setting the question␣
 ↪for that record,
the answers and their points. It then returns that record stored in a variable␣
 ↪by returning
the value stored in that variable. */
public static QuizQuestion createQuestion(String question1, String answer1, int␣
 ↪point_score1,
                                          String answer2, int point_score2,␣
 ↪String answer3, int point_score3)
```

```
{
    QuizQuestion question_1 = new QuizQuestion();
    question_1.question = question1;
    question_1.answer_1 = answer1;
    question_1.point_score_1 = point_score1;
    question_1.answer_2 = answer2;
    question_1.point_score_2 = point_score2;
    question_1.answer_3 = answer3;
    question_1.point_score_3 = point_score3;
    return question_1;
} // END createQuestion
```

Testing

[6]:
```
QuizQuestion question = createQuestion("What is the answer?", "answer1", 25,␣
 ↪"answer2", 15, "answer3", 60);
// This creates a question record stored in variable question with its relevant␣
 ↪fields (stored in heap not stack).
// Nothing is supposed to be outputted. If variable question is printed it will␣
 ↪print the reference.
```

2.5.4 getQuestion

What it does This method is an accessor method which gets the question out of the question field from a question record of type QuizQuestion and returns it as a result.

Implementation (how it works) This accessor method works by first using question1 record as a parameter to get the fields from it. The actual question record data is passed through via argument when calling this method. This then returns the question string value from the question field of the record as a result of the method.

[7]:
```
// This method retrieves the string question from a question record from its␣
 ↪question field.
public static String getQuestion(QuizQuestion question1)
{
    return question1.question;
} // END getQuestion
```

Testing

[8]:
```
System.out.println(getQuestion(question)); // Gets the question string from␣
 ↪record question from question field.
```

```
What is the answer?
```

2.5.5 getAnswer1

What it does This method is an accessor method which gets the first answer out of the answer_1 field from a question record of type QuizQuestion and returns it as a result.

Implementation (how it works) This accessor method first uses local question1 record as a parameter to get the fields from it. The actual question record data is passed through via argument when calling this method. This then returns the first answer string value from the answer_1 field of the record as a result of the method.

[9]:
```
// This method retrieves the string answer 1 from a question record from its␣
↪answer_1 field.
public static String getAnswer1(QuizQuestion question1)
{
    return question1.answer_1;
} // END getAnswer1
```

Testing

[10]:
```
System.out.println(getAnswer1(question));
```

```
answer1
```

2.5.6 getAnswer2

What it does This method is an accessor method which gets the second answer out of the answer_2 field from a question record of type QuizQuestion and returns it as a result.

Implementation (how it works) This accessor method first uses local question1 record as a parameter to get the fields from it. The actual question record data is passed through via argument when calling this method. This then returns the second answer string value from the answer_2 field of the record as a result of the method.

[11]:
```
// This method retrieves the string answer 2 from a question record from its␣
↪answer_2 field.
public static String getAnswer2(QuizQuestion question1)
{
    return question1.answer_2;
} // END getAnswer2
```

Testing

[12]:
```
System.out.println(getAnswer2(question));
```

```
answer2
```

2.5.7 getAnswer3

What it does This method is an accessor method which gets the third answer out of the answer_3 field from a question record of type QuizQuestion and returns it as a result.

Implementation (how it works) This accessor method first uses local question1 record as a parameter to get the fields from it. The actual record data is passed through via argument when calling this method. This then returns the third answer string value from answer_3 field of the record as a result of the method.

[13]:
```java
// This method retrieves the string answer 3 from a question record from its␣
↪answer_3 field.
public static String getAnswer3(QuizQuestion question1)
{
    return question1.answer_3;
} // END getAnswer3
```

Testing

[14]:
```java
System.out.println(getAnswer3(question));
```

```
answer3
```

2.5.8 getPoints1

What it does This method is an accessor method which gets the points of the first answer out of the point_score_1 field from a question record of type QuizQuestion and returns it as a result.

Implementation (how it works) This accessor method first uses local question1 record as a parameter to get the fields from it. The actual record data is passed through via argument when calling this method. This then returns the points of answer 1 integer value from point_score_1 field of the record as a result of the method.

[15]:
```java
// This method retrieves the integer points of answer 1 from a question record␣
↪from its point_score_1 field.
public static int getPoints1(QuizQuestion question1)
{
    return question1.point_score_1;
} // END getPoints1
```

Testing

[16]:
```java
System.out.println(getPoints1(question));
```

```
25
```

2.5.9 getPoints2

What it does This method is an accessor method which gets the points of the second answer out of the point_score_2 field from a question record of type QuizQuestion and returns it as a result.

Implementation (how it works) This accessor method first uses local question1 record as a parameter to get the fields from it. The actual record data is passed through via argument when calling this method. This then returns the points of answer 2 integer value from the point_score_2 field of the record as a result of the method.

[17]:
```
// This method retrieves the integer points of answer 2 from a question record␣
↪from its point_score_2 field.
public static int getPoints2(QuizQuestion question1)
{
    return question1.point_score_2;
} // END getPoints2
```

Testing

[18]:
```
System.out.println(getPoints2(question));
```

```
15
```

2.5.10 getPoints3

What it does This method is an accessor method which gets the points of the third answer out of the point_score_3 field from a question record of type QuizQuestion and returns it as a result.

Implementation (how it works) This accessor method first uses local question1 record as a parameter to get the fields from it. The actual record data is passed through via argument when calling this method. This then returns the points of answer 3 integer value from the point_score_3 field of the record as a result of the method.

[19]:
```
// This method retrieves the integer points of answer 3 from a question record␣
↪from its point_score_3 field.
public static int getPoints3(QuizQuestion question1)
{
    return question1.point_score_3;
} // END getPoints3
```

Testing

[20]:
```
System.out.println(getPoints3(question));
```

```
60
```

2.5.11 fileOutputQuestion1

What it does This method creates a text file and writes the first question and answers and their points to that file and saves it to be used later and and returns the file name string value as a result. This is an example of file output.

Implementation (how it works) Note that in the full program importing the relevant libraries is needed (java.io). Firstly when defining this method (on the header line) a throws IOException code is needed first to deal with file IO errors such as the filesystem might be full so it needs to throw an exception to pass the error. Then question1 is defined and intialised to get string value called “actors.txt”. This is the file name used to create a text file with that name. Then to write to that file the code creates an output stream targeted at that file. This is done in the second line of the method. This creates the text file called actors.txt and creates an output stream connecting variable outputStream with the file called actors.txt. FileWriter here links it to a FileWriter object in the program, so the program can write to it. This makes outputStream.println() and outputStream.close avaliable. Then to write to it the code uses outputStream.println() method with the string you want to write as the argument. So this code prints the question in the first line and then the answers and their points each line after to the text file. Then the file is closed using outputStream.close() method. Finally as a result we return the file name string value which is stored in variable question1 so we can hold on to it so we can read the file later.

[21]:
```
/* This method creates and writes the first question on actors and its
answers and points on a text file and saves it and returns the file name as a␣
 ↪result. */
public static String fileOutputQuestion1() throws IOException
{
    String question1 = "actors.txt";
    PrintWriter outputStream = new PrintWriter(new FileWriter(question1));
    outputStream.println("Out of the three options: Option 1: Rami Malek,␣
 ↪Option 2: Adam Sandler, " +
                "Option 3: Denzel Washington, Who is an Oscar winning actor?");
    outputStream.println("Rami Malek");
    outputStream.println("20");
    outputStream.println("Adam Sandler");
    outputStream.println("5");
    outputStream.println("Denzel Washington");
    outputStream.println("75");
    outputStream.close();
    return question1;
} // END fileOutputQuestion1
```

Testing

[22]:
```
String filename = fileOutputQuestion1();
System.out.println(filename);
/* Creates a text file which holds question, answers and their points and␣
 ↪stores file name in variable filename,
which is printed out. */
```

```
actors.txt
```

2.5.12 fileOutputQuestion2

What it does This method creates a text file and writes the second question and answers and their points to that file and saves it to be used later and returns the file name string value as a result.

Implementation (how it works) Firstly throws IOException is needed at the header line to deal with file IO errors in order to throw an exception to pass the error. Then question2 is initialised and declared as a string type which stores the file name of the question which is "presidents.txt" as a string value. Then the second line in the method creates the text file with that name from using the variable as an argument in the method. To then write to that file the code creates an output stream targeted to that file. This creates an output stream connecting variable outputStream with the file called presidents.txt. FileWriter here links it to a FileWriter object in the program, so the program can write to it. This makes outputStream.println() and outputStream.close() avaliable. Then to write to the file the code uses outputStream.println() method with the string you want to write as the argument. So this code like in method fileOutputQuestion2 prints the question in the first line and then the answes and their points each line after to the text file. The file is then closed using outputStream.close() method. Finally as a result the file name string value stored in variable question2 is returned so we can hold on to it in order to read the file later.

[23]:
```
/* This method creates and writes the second question on presidents and its
answers and points on a text file and saves it and returns the file name as a␣
 ↪result. */
public static String fileOutputQuestion2() throws IOException
{
    String question2 = "presidents.txt";
    PrintWriter outputStream = new PrintWriter(new FileWriter(question2));
    outputStream.println("Out of the three options: Option 1: Benjamin␣
 ↪Franklin, Option 2: George Washington, " +
            "Option 3: Barack Obama, Who is an American president?");
    outputStream.println("Benjamin Franklin");
    outputStream.println("12");
    outputStream.println("George Washington");
    outputStream.println("28");
    outputStream.println("Barack Obama");
    outputStream.println("60");
    outputStream.close();
    return question2;
} // END fileOutputQuestion2
```

Testing

[24]:
```
String filename2 = fileOutputQuestion2();
System.out.println(filename2);
```

```
presidents.txt
```

2.5.13 fileOutputQuestion3

What it does This method creates a text file and writes the third question and answers and their points to that file and saves it to be used later and returns the file name string value as a result.

Implementation (how it works) Firstly throws IOException is needed at the header line to deal with the file IO errors in order to throw an exception to pass the error. Then question3 is intialised and declared as a string type which stores the file name of the question which is "cities.txt" as a string value. Then the second line in the metod creates the text file with that name using the variable (question3) as ab argument in the method. To then write to that file the code creates an output stream targeted to that file. This creates an output stream connecting variable outputStream with the file called cities.txt. FileWriter here links it to a FileWriter object in the program, so the program can write to it. This makes outputStream.println() and outputStream.close() available. Then to write to the file the code uses outputStream.println() method with the string you want ot write as the argument. So this code like in method fileOutputQuestion3 prints the question in the first line and then the answers and their points each line after to the text file. The file is then closed using outputStream.close() method. Finally as a result the file name string value in variable question3 is returned so we can hold on to it in order to read this file later.

[25]:
```
/* This method creates and writes the third question on cities and its
answers and points on a text file and saves it and returns the file name as a␣
↪result. */
public static String fileOutputQuestion3() throws  IOException
{
    String question3 = "cities.txt";
    PrintWriter outputStream = new PrintWriter(new FileWriter(question3));
    outputStream.println("Out of the three options: Option 1: Agra, Option 2:␣
↪Mumbai, " +
            "Option 3: Hyderabad, Which city is India?");
    outputStream.println("Agra");
    outputStream.println("25");
    outputStream.println("Mumbai");
    outputStream.println("63");
    outputStream.println("Hyderabad");
    outputStream.println("12");
    outputStream.close();
    return question3;
} // END fileOutputQuestion3
```

Testing

[26]:
```
String filename3 = fileOutputQuestion3();
System.out.println(filename3);
```

```
cities.txt
```

2.5.14 fileInputQuestion

What it does This method reads the file which contain the question, answers and points in order to create a record of them of type QuizQuestion and return that record as a result.

Implementation (how it works) Firstly parameter question_file (of type string) is used to allow code to pass through the file name via argument when calling this method in order to access the file so it can be read. Throws IOException is needed at the end of the header line in order to deal with file IO errors (this is always needed when dealing with file IO). Then the first line allows the text file of file name of whatever value is in variable question_file to be read. This creates an input stream connected to the file called whatever the value in variable question_file. It links variable inputStream to the file so that FileReader can get the characters from the file and BufferedReader putting the characters back into Strings (it buffers the data) until it is ready to be taken via the variable inputStream. To read a line of text from the file you use inputStream.readLine method and in the first line contains the question which is stored as a string value in variable question. Then the second and third line contains the answer and the points of that answer and that is read using inputStream.readLine method and stored in string type variable answer1 and integer type variable score1. Using inputStream.readLine method produces a string value to store the score in score1 variable as an integer type we use Integer.parseInt to convert the string value into an integer value. This is done for fourth and fifth lines of the text file for variables answer2 and score2 and then sixth and seventh lines of the text file for variables answer3 and score3. To then close the file to stop reading we then use inputStream.close() method. We then create a record of type QuizQuestion which holds the question, answers and their corresponding points or score using method createQuestion and the arguments which are the question variable holding the question string value, answer1 and score1 variables holding their answer string value and score integer value and so on. As a result we then return the question record created holding the data.

[27]:
```
/* This method reads the file containing the question, answers and points in␣
↪order to create a record holding
that data and return that record as a result. */
public static QuizQuestion fileInputQuestion(String question_file) throws␣
↪IOException
{
    BufferedReader inputStream = new BufferedReader(new␣
↪FileReader(question_file));
    String question = inputStream.readLine();
    String answer1 = inputStream.readLine();
    int score1 = Integer.parseInt(inputStream.readLine());
    String answer2 = inputStream.readLine();
    int score2 = Integer.parseInt(inputStream.readLine());
    String answer3 = inputStream.readLine();
    int score3 = Integer.parseInt(inputStream.readLine());
    inputStream.close();
    QuizQuestion questionRecord = createQuestion(question, answer1, score1,␣
↪answer2, score2, answer3, score3);
    return questionRecord;
} // END fileInputQuestion
```

Testing

[28]:
```
QuizQuestion Question_Actors_record = fileInputQuestion(filename);
/* Creates question record holding the
data of the question on actors (data meaning question,answers and their points).
↪ */
```

2.5.15 createBank

What it does This creates a question bank by storing the question records in an array of type QuizQuestion and returning that array as a result.

Implementation (how it works) Firstly throws IOException is needed at the end of the header line as file IO is part of the code. Then it intialises and declares 3 variables (question1_file, question2_file and question3_file) which are string that hold the file names of the text files created and written which hold question, answers and their points of each question using fileOutputQuestion1, fileOutputQuestion2 and fileOuputQuestion3 methods corresponding to their variables. Then records are created which are type QuizQuestion called question1, question2 and question3 which hold the data on each question. This is done by calling fileInputQuestion with the corresponding file name string type variable as the argument, so for record question1 the argument would be question1_file and so on. This method fileInputQuestion reads the text files that were just created and stores the data read into records and returns that record. Then an array is created called Question_Bank of size 3 of type QuizQuestion and stores record question1 in position 0 and record question2 in position 1 and record question3 in position 2. Then returns the array Question_Bank as a result of the method.

[29]:
```java
/* This creates a question bank storing the question records
which hold the questions and their data into an array and returning that array␣
↪as a result. */
public static QuizQuestion[] createBank() throws IOException
{
    String question1_file = fileOutputQuestion1();
    String question2_file = fileOutputQuestion2();
    String question3_file = fileOutputQuestion3();
    QuizQuestion question1 = fileInputQuestion(question1_file);
    QuizQuestion question2 = fileInputQuestion(question2_file);
    QuizQuestion question3 = fileInputQuestion(question3_file);
    QuizQuestion [] Question_Bank = new QuizQuestion[3];
    Question_Bank [0] = question1;
    Question_Bank [1] = question2;
    Question_Bank [2] = question3;
    return Question_Bank;
} // END createBank
```

Testing

[30]:
```java
QuizQuestion [] QuestionBank = createBank();
// Array containing the question records
```

2.5.16 Question_score

What it does This method keeps a total of the user's score when they have answered a question and tells the user how many points they receive for that answer. The score is returned as a result.

Implementation (how it works) This method first uses parameters question1 of type QuizQuestion which is the question record and answer of type string which holds the answer the user inputted from the question. These data are passed through via arguments when calling this method. Then it declares the variable score2 which is an integer type. Then if-then-else statements are used which first test if the string value in variable answer equals to the first answer from the question. This is taken from the question record using accessor method getAnswer1 with the question record as the argument. If this test is true then variable score2 gets the points of getting that first answer which is also taken from the question record using accessor method getPoints1 with the question record as the argument. Then it prints out to the user how many points they got from this answer using concatentation of score2 to output the points. This is repeated in two more tests that ask the same but this time if the answer equals to the second answer or third answer (using accessor methods getAnswer2 and getAnswer3) and then uses accessor methods getPoints1 or getPoints2 to add to the score depending if the user's answer equals to the corresponding answer 2 or 3. Then prints out how many points they got from that answer. Otherwise if all other tests are false then the else body is executed which is when the user's answer does not equal to any of the options given and score2 gets 100 as its value and printing to the user that their answer is not one of the options and telling them how many points they get from this answer. Finally score2's value is returned as a result.

[31]:
```
/* This method stores the total score the user gets when they have answered a␣
↪question and tells the user how
points they recieve for that answer and return the score value as a result. */
public static int Question_score(QuizQuestion question1, String answer)
{
    int score2;
    if (answer.equals(getAnswer1(question1)))
    {
        score2 = getPoints1(question1);
        System.out.println("This answer gives you " + score2 + " points.");
    }
    else if (answer.equals(getAnswer2(question1)))
    {
        score2 = getPoints2(question1);
        System.out.println("This answer gives you " + score2 + " points.");
    }
    else if (answer.equals(getAnswer3(question1)))
    {
        score2 = getPoints3(question1);
        System.out.println("This answer gives you " + score2 + " points.");
    }
    else
    {
        score2 = 100;
        System.out.println("This is not one of the options");
        System.out.println("This answer gives you 100 points.");
    }
    return score2;
} // END Question_score
```

Testing

[32]:
```
String user_answer = "answer1";
int user_score = Question_score(question, user_answer);
```

```
This answer gives you 25 points.
```

2.5.17 printQuestion

What it does This method will ask the user what question to answer (either 1, 2 or 3) and if they choose an option then the question will be printed out asking the user for an answer and then stores the score they get from the user's answer. Otherwise it prints to the user that the option they inputted was not an option. Then it returns the score as a result.

Implementation (how it works) This method first gets parameter question_bank which is an local array that holds the quetion records of type QuizQuestion. This bank data is passed through via argument when calling this method. Then it first declares and intialises the variable score (type integer) which holds value 0. Then it asks the user out of questions 1, 2 or 3 which one they want to answer and gets a response from them using method inputInt and stores their response in variable choice of type integer. Then an if statement in used where the test asks if the value in choice is greater than or equal to 1 and that choice is also less than and equal to 3 (giving options 1, 2 or 3). If this test is true then the question record used will be the one in the question_bank array at position value in variable choice - 1 since the array starts at position 0. Then the user will be asked the question from that record using accessor method getQuestion (using the question record used as the argument) and get a response from the user using inputString method call and storing the answer in variable answer of type string. Then the score is calculated using method call Question_score (using the question record used and answer as the arguments). Otherwise if the test is false then it will print to the user that their response was not an option. Finally it returns the value in variable score as a result of the method.

[33]:
```
/* This method asks the user which question they want to answer and gets a␣
 ↪response from the user and then
user answers the question and stores the score they get and returns the score␣
 ↪as a result. */
public static int printQuestion(QuizQuestion [] question_bank)
{
    int score = 0;
    int choice = inputInt("Out of question 1, 2 or 3, which one do you want to␣
 ↪answer?");
    if (choice <= 3 & choice >= 1)
    {
        QuizQuestion question_used = question_bank[(choice - 1)];
        String answer = inputString(getQuestion(question_used));
        score = Question_score(question_used, answer);
    }
    else
    {
        System.out.println("That is not an option");
    }
    return score;
} // END printQuestion
```

Testing

[34]:
```
int user_score2 = printQuestion(QuestionBank);
```

```
Out of question 1, 2 or 3, which one do you want to answer?
1
Out of the three options: Option 1: Rami Malek, Option 2: Adam Sandler, Option
3: Denzel Washington, Who is an Oscar winning actor?
Rami Malek
This answer gives you 20 points.
```

2.5.18 createScore

What it does This method is a primitive method of ADT TotalScore which creates a record holding user's name and total score in the required fields and returning that record as a result.

Implementation (how it works) Firstly a new ADT type is created in a new class which is defined and called TotalScore (the type name). Then the required fields are declared which are name and score which are both string type. Then in the method paramters given_name and given_score are needed which are string type. These will hold the data of the user's name and their score which is passed through to the method via arguments when calling this method. Then local record named player is created of type TotalScore. Here new means it makes a reference in order to add values to the heap. Then the name field of record player is assigned the name of the user from the value in given_name variable. Then score field of record player is assigned the score of the user from the value in given_score variable. Then the record player is returned as a result of the method.

[35]:
```
/* This is a new abstract data type TotalScore which holds the name and score␣
 ↪of user who have done the quiz.
They have two fields which are name and score both string type. */
class TotalScore
{
    String name;
    String score;
}

/* This method is a primitive method of ADT TotalScore which creates a record␣
 ↪holding name and score of user doing
the quiz and returns that record as a result. */
public static TotalScore createScore(String given_name, String given_score)
{
    TotalScore player = new TotalScore();
    player.name = given_name;
    player.score = given_score;
    return player;
} // END createScore
```

Testing

[36]:
```
String player_name = "User 1";
String player_score = "60";
TotalScore user_details = createScore(player_name, player_score);
// user_details is a record created of type TotalScore holding user's name and␣
↪score which is User 1 and 60.
```

2.5.19 AddToArray

What it does This primitive method of ADT TotalScore adds to an existing string type array which holds the name and score of each user who is participating in the quiz.

Implementation (how it works) This first uses parameters scoreArray which is an array of type string (this is the exisiting array passed through to this method via argument when calling this method which holds the each user's name and score) and position in the array it is being added to (passed through to this method via argument) which is type integer and the record holding the user name and score data of type TotalScore (also passed through to this method via argument). Then it at scoreArray at positon value in whatever value in variable position it is assigned the string value which holds the user's name and score and returns back to the code from where it was called.

[37]:
```
/* This primitive method of ADT TotalScore add to an existing array that is of␣
↪type
string which holds the name and score of each user. */
public static void AddToArray(String [] scoreArray, int position, TotalScore␣
↪player)
{
    scoreArray [position] = player.name + " has score of " + player.score + "␣
↪points.";
    return;
} // END AddToArray
```

Testing

[38]:
```
String [] array = new String[2];
AddToArray(array, 0, user_details);
AddToArray(array, 1, user_details);
System.out.println(array[0]);
```

```
User 1 has score of 60 points.
```

2.5.20 printScores

What it does This is a primitive method of ADT TotalScore which prints the final total scores and names of users doing the quiz to the user.

Implementation (how it works) This primitive method first uses parameter scoreArray which is an array of string type. This will hold the actual array which contains each of the user's name and score data which is passed through to this method via argument when calling this method. Then it prints scores of each user as the title. Then it uses a for loop to print out each value in the array scoreArray. From position 0 to the length of the array (which is the number of users who particpated in the quiz). Then at the end it print an blank line to seperate what has been printed. Then returns back to the code where this method was called.

[39]:
```
// This primitive method of ADT TotalScore prints the final total scores and␣
↪names of the users doing the quiz.
public static void printScores(String [] scoreArray)
{
    System.out.println("The scores of each user:");
    for (int i=0; i<scoreArray.length; i++)
    {
        System.out.println(scoreArray[i]);
    }
    System.out.println();
    return;
} // END printScores
```

Testing

[40]:
```
printScores(array);
```

```
The scores of each user:
User 1 has score of 60 points.
User 1 has score of 60 points.
```

2.5.21 createTableScore

What it does This creates a leaderboard by using a sorting algorithm to sort each user's scores with the person with the least points as first and user with the highest points is last.

Implementation (how it works) This method first uses parameter board which an array of type integer which hold each of the users particpating their score. This will temporarily hold the actual array that holds the each users scores which is passed through to this method via argument when calling this method. Then we ask the user if they want to see the leaderboard and get a response from them either Y for yes and N for NO. This is done by calling inputString method with the question as the argument and stores the response in variable option which is of type string. Then an if statement is used which test if the value held in option is Y then it will execute the body of this branch. The body instructions will then sort the unordered array (board) which holds each users scores using a bubble sort which compares each position and the position before that using if statement as shown in line 9 of the method. If the position before is higher than the original position then it will swap them around. This is done for each position in the array until the last position. This will then be a pass. This pass repeatedly occurs a value of the number of values in the array board. As shown by the fifth line in the method. Then now the array is sorted. Otherwise if the test of the if statement is false then it will print thank you for taking part in the quiz to the user and exit the program. At the end of the method it returns back to the code to where this method was called.

[41]:
```
/* This method creates a leaderboard of the scores using a sort algorithm where␣
 ↪the user with least points
is first while the user with the most points is last. */
public static void createTableScore(int [] board)
{
    String option = inputString("Would you like to see the leaderboard? (Y/N)");
    if (option.equals("Y"))
    {
        int temporary = 0;
        for (int pass=0; pass<board.length; pass++)
        {
            for (int b=1; b<board.length-pass; b++)
            {
                if (board[b-1] > board[b])
                {
                    temporary = board[b-1];
                    board[b-1] = board[b];
                    board[b] = temporary;
                }
            }
        }
    }
    else
    {
        System.out.println("Thank you for taking part in the Quiz.");
        System.exit(0);
    }
    return;
} // END createTableScore
```

Testing

[42]:
```
int [] game_scores = {45,86,22,12,55,80};
createTableScore(game_scores);
// This will have sorted the array game_scores
```

```
Would you like to see the leaderboard? (Y/N)
Y
```

2.5.22 printTableScore

What it does This method prints the leaderboard of the score of the users who particpated in the quiz. Where first has the least amount of points and last has the most points.

Implementation (how it works) This method first uses parameter board (type integer) which will temporarily hold the array with the sorted scores now. The actual data of the sorted array is passed through to this method via argument when calling this method. Then a for loop is used which will print each of the score values in the array which is already in order. The i + 1 is used to indicate what position it is in the leaderboard. Then it returns back to the code to where this method was called.

[43]:
```
// This method prints the leaderboard of the scores where first has least␣
 ↪points and last has most points.
public static void printTableScore(int [] board)
{
    for (int i=0; i<board.length; i++)
    {
        System.out.println((i + 1) + ": " + board[i] + " points.");
    }
    return;
} // END printTableScore
```

Testing

[44]:
```
printTableScore(game_scores);
// Prints the sorted version of array game_scores as a leaderboard
```

```
1: 12 points.
2: 22 points.
3: 45 points.
4: 55 points.
5: 80 points.
6: 86 points.
```

2.5.23 QuizDetails

What it does This method is the method doing all the work. It asks the user how many people want to do the quiz and then lets each user do the quiz and allows each user to answer another question if they like (each time they have finished answering a question) and then tallys up and records the each users scores and print the scores and prints a leaderboard if the user wants to see it. Then the program is exited.

Implementation (how it works) Note that this method needs throws IOExeception as it handles file IO. Firstly this method creates the question bank by calling the method createBank (which involves file IO) and then stores it into an array of type QuizQuestion called Quiz_questions. This variable Quiz_questions is a final variable as it stays constant throughout the code. Then it asks the user how many people want to do the quiz and gets their response. This is done by calling method inputInt with the question as the argument and then stores the response in variable number_of_times (which is of type integer). This is a final variable since it stays constant throughout the code. Then we create two new arrays, one is called board which is of type integer which will hold the final scores of each user in order to create a leaderboard. The other array is called playerScores which is string type and will hold the scores of each user in order to print out all the user's and their scores. The size of both arrays is the value in final variable number_of_times (which is the number of people doing the quiz) since the number of scores is the number of people doing the quiz. Then a for loop is used to repeatedly do the quiz for each user particpating in the quiz. Then a blank line is printed to seperate what is printed next. Then it will print to the user doing the quiz saying the first person to answer the question from the question bank through concatenation of variable user_name which holds the string value of which person doing the quiz. Then two totals are declared and intialised which are total and total2 which are both integer type and hold value 0. Variable total is used to hold the score the user get from each question they answer and variable total2 will keep a running total of what score they get overall. Then a while loop is used to repeatedly ask the user what question they want to answer and allow them to answer the question they chose and get the score from their answer. This is done by calling method printQuestion with the question bank variable Quiz_questions as the argument. This will then return the score they get from answering that question and store it in variable total. Then it increments total2 by the value in total. Then it ask the user if they want to answer another question with a response of Y or N. This is done by calling inputString with the question as the argument. This response is stored in variable tries. If the user does have a response of Y then it will repeat the loop since the test is true as tries will equal to Y. If the response is N stored in variable tries then it will exit the while loop and then create a record of the score and name of that user who just done the quiz by calling createScore and store it into record player of ADT TotalScore. Then it adds the record data (which contains score and name of the user) to a string array playerScores at position i. Then adds the value of total2 (which is the final score that user got) to array board at position i. Then creates a blank line to again seperate it. This body of the for loop is repeated for the next user to answer the quiz and so on until the value of number_of_times. Then it prints the each value of array playerScores to show the final scores each user got by calling printScores method with array playerScores as the argument. Then it asks the user if they want to see the leaderboard or not. If not it prints a message and exits the program and if they say yes it sorts the array board with the final scores of each user. This is done by calling createTableScore method with array board as the argument. Then it prints the leaderboard with the ranked positions (from least points to highest) by calling method printTableScore with array board as the argument. Then the method returns to the main method.

[45]:
```java
/* This is the method doing all the work which asks the user how many people␣
↪are doing the quiz and then lets
each user do the quiz and ask if they want to answer another question and␣
↪tallys up each users total scores
and prints the scores and also prints a leaderboard if the user wants to see it.
↪ */
public static void QuizDetails() throws IOException
{
    final QuizQuestion [] Quiz_questions = createBank();
    final int number_of_times = inputInt("How many people want to do the Quiz?
↪");
    int [] board = new int[number_of_times];
    String [] playersScores = new String[number_of_times];
    for (int i=0; i<number_of_times; i++)
    {
        System.out.println();
        String user_name = "Person " + (i + 1);
        System.out.println(user_name + " to answer question from question bank:
↪");
        int total = 0;
        int total2 = 0;
        String tries = "Y";
        while (tries.equals("Y"))
        {
            total = printQuestion(Quiz_questions);
            total2 = total2 + total;
            tries = inputString("Do you want to answer another question (Y/N)");
        }
        TotalScore player = createScore(user_name, Integer.toString(total2));
        AddToArray(playersScores, i, player);
        board [i] = total2;
        System.out.println();
    }
    printScores(playersScores);
    createTableScore(board);
    printTableScore(board);
    return;
} // END QuizDetails
```

Testing

[46]:
```
QuizDetails();
```

```
How many people want to do the Quiz?
2

Person 1 to answer question from question bank:
Out of question 1, 2 or 3, which one do you want to answer?
1
Out of the three options: Option 1: Rami Malek, Option 2: Adam Sandler, Option
3: Denzel Washington, Who is an Oscar winning actor?
Rami Malek
This answer gives you 20 points.
Do you want to answer another question (Y/N)
Y
Out of question 1, 2 or 3, which one do you want to answer?
2
Out of the three options: Option 1: Benjamin Franklin, Option 2: George
Washington, Option 3: Barack Obama, Who is an American president?
Barack Obama
```

```
This answer gives you 60 points.
Do you want to answer another question (Y/N)
N


Person 2 to answer question from question bank:
Out of question 1, 2 or 3, which one do you want to answer?
1
Out of the three options: Option 1: Rami Malek, Option 2: Adam Sandler, Option
3: Denzel Washington, Who is an Oscar winning actor?
Adam Sandler
This answer gives you 5 points.
Do you want to answer another question (Y/N)
Y
Out of question 1, 2 or 3, which one do you want to answer?
3
Out of the three options: Option 1: Agra, Option 2: Mumbai, Option 3: Hyderabad,
Which city is India?
Hyderabad
This answer gives you 12 points.
Do you want to answer another question (Y/N)
N

The scores of each user:
Person 1 has score of 80 points.
Person 2 has score of 17 points.

Would you like to see the leaderboard? (Y/N)
Y
1: 17 points.
2: 80 points.
```

2.5.24 Running the program

Run the following call to simulate running the complete program.

[47]:
```
QuizDetails();
```

```
How many people want to do the Quiz?
2

Person 1 to answer question from question bank:
Out of question 1, 2 or 3, which one do you want to answer?
1
Out of the three options: Option 1: Rami Malek, Option 2: Adam Sandler, Option
3: Denzel Washington, Who is an Oscar winning actor?
Denzel Washington
This answer gives you 75 points.
```

```
Do you want to answer another question (Y/N)
N


Person 2 to answer question from question bank:
Out of question 1, 2 or 3, which one do you want to answer?
2
Out of the three options: Option 1: Benjamin Franklin, Option 2: George
Washington, Option 3: Barack Obama, Who is an American president?
George Washington
This answer gives you 28 points.
Do you want to answer another question (Y/N)
Y
Out of question 1, 2 or 3, which one do you want to answer?
1
Out of the three options: Option 1: Rami Malek, Option 2: Adam Sandler, Option
3: Denzel Washington, Who is an Oscar winning actor?
Adam Sandler
This answer gives you 5 points.
Do you want to answer another question (Y/N)
N

The scores of each user:
Person 1 has score of 75 points.
Person 2 has score of 33 points.

Would you like to see the leaderboard? (Y/N)
Y
1: 33 points.
2: 75 points.
```

2.6 The complete program

This version will only compile here. To run it copy it into a file called initials.java on your local computer and compile and run it there.

[49]:
```java
// NAME: Ifthy Fairoze
// DATE: 25/11/2020
// VERSION: 1
/* BRIEF OVERVIEW OF PURPOSE: This program allows multiple users to particpate␣
 ↪in a quiz where they each answer
a question of their choice from the question bank and can answer other␣
 ↪questions and tallys up and records their
scores and prints their scores and a leaderboard (where user with least points␣
 ↪is first and user with most points
is last) if they want to. */

import java.util.Scanner; // Needed to make Scanner available
import java.io.*; // Needed to make file IO avalaible

// This is a new type QuizQuestion to sort the questions and answers and points␣
 ↪into records.
class QuizQuestion
{
    String question;
    String answer_1;
    int point_score_1;
    String answer_2;
    int point_score_2;
    String answer_3;
    int point_score_3;
}

/* This is a new abstract data type TotalScore which holds the name and score␣
 ↪of user who have done the quiz.
They have two fields which are name and score both string type. */
class TotalScore
{
    String name;
    String score;
}

class PointlessQuiz
{
    public static void main (String [] a) throws IOException
    {
        QuizDetails();
        System.exit(0);
    }

    /* This allows the programmer to call this method and ask a question or␣
 ↪message
    as its argument and allow the user to answer the question in the next line.
    Then the input data can be used further on in the program when returned. */
    public static String inputString(String message)
    {
        Scanner scanner = new Scanner(System.in);
        String answer;
        System.out.println(message);
        answer = scanner.nextLine();
        return answer;
    } // END inputString

    /* This method asks a question to the user and then waits for the user to␣
 ↪type an integer response which is
```

```
   returned as a result of the method. */
   public static int inputInt(String message)
   {
       Scanner scanner = new Scanner(System.in);
       String textInput;
       int numberInput;
       System.out.println(message);
       textInput = scanner.nextLine();
       numberInput = Integer.parseInt(textInput);
       return numberInput;
   } // END inputInt

   /* This method creates and intialises a question record by setting the␣
↪question for that record,
   the answers and their points. It then returns that record stored in a␣
↪variable by returning
   the value stored in that variable. */
   public static QuizQuestion createQuestion(String question1, String answer1,␣
↪int point_score1,
                                              String answer2, int point_score2,␣
↪String answer3, int point_score3)
   {
       QuizQuestion question_1 = new QuizQuestion();
       question_1.question = question1;
       question_1.answer_1 = answer1;
       question_1.point_score_1 = point_score1;
       question_1.answer_2 = answer2;
       question_1.point_score_2 = point_score2;
       question_1.answer_3 = answer3;
       question_1.point_score_3 = point_score3;
       return question_1;
   } // END createQuestion

   // This method retrieves the string question from a question record from␣
↪its question field.
   public static String getQuestion(QuizQuestion question1)
   {
       return question1.question;
   } // END getQuestion

   // This method retrieves the string answer 1 from a question record from␣
↪its answer_1 field.
   public static String getAnswer1(QuizQuestion question1)
   {
       return question1.answer_1;
   } // END getAnswer1
```

```
    // This method retrieves the string answer 2 from a question record from␣
↪its answer_2 field.
    public static String getAnswer2(QuizQuestion question1)
    {
        return question1.answer_2;
    } // END getAnswer2

    // This method retrieves the string answer 3 from a question record from␣
↪its answer_3 field.
    public static String getAnswer3(QuizQuestion question1)
    {
        return question1.answer_3;
    } // END getAnswer3

    // This method retrieves the integer points of answer 1 from a question␣
↪record from its point_score_1 field.
    public static int getPoints1(QuizQuestion question1)
    {
        return question1.point_score_1;
    } // END getPoints1

    // This method retrieves the integer points of answer 2 from a question␣
↪record from its point_score_2 field.
    public static int getPoints2(QuizQuestion question1)
    {
        return question1.point_score_2;
    } // END getPoints2

    // This method retrieves the integer points of answer 3 from a question␣
↪record from its point_score_3 field.
    public static int getPoints3(QuizQuestion question1)
    {
        return question1.point_score_3;
    } // END getPoints3

    /* This method creates and writes the first question on actors and its
    answers and points on a text file and saves it and returns the file name as␣
↪a result. */
    public static String fileOutputQuestion1() throws IOException
    {
        String question1 = "actors.txt";
        PrintWriter outputStream = new PrintWriter(new FileWriter(question1));
        outputStream.println("Out of the three options: Option 1: Rami Malek,␣
↪Option 2: Adam Sandler, " +
                "Option 3: Denzel Washington, Who is an Oscar winning actor?");
        outputStream.println("Rami Malek");
        outputStream.println("20");
        outputStream.println("Adam Sandler");
        outputStream.println("5");
        outputStream.println("Denzel Washington");
        outputStream.println("75");
        outputStream.close();
        return question1;
    } // END fileOutputQuestion1

    /* This method creates and writes the second question on presidents and its
    answers and points on a text file and saves it and returns the file name as␣
↪a result. */
    public static String fileOutputQuestion2() throws IOException
    {
        String question2 = "presidents.txt";
        PrintWriter outputStream = new PrintWriter(new FileWriter(question2));
        outputStream.println("Out of the three options: Option 1: Benjamin␣
↪Franklin, Option 2: George Washington, " +
                "Option 3: Barack Obama, Who is an American president?");
        outputStream.println("Benjamin Franklin");
        outputStream.println("12");
        outputStream.println("George Washington");
        outputStream.println("28");
        outputStream.println("Barack Obama");
        outputStream.println("60");
        outputStream.close();
        return question2;
    } // END fileOutputQuestion2

    /* This method creates and writes the third question on cities and its
    answers and points on a text file and saves it and returns the file name as␣
↪a result. */
    public static String fileOutputQuestion3() throws  IOException
    {
        String question3 = "cities.txt";
        PrintWriter outputStream = new PrintWriter(new FileWriter(question3));
        outputStream.println("Out of the three options: Option 1: Agra, Option␣
↪2: Mumbai, " +
                "Option 3: Hyderabad, Which city is India?");
        outputStream.println("Agra");
        outputStream.println("25");
        outputStream.println("Mumbai");
        outputStream.println("63");
        outputStream.println("Hyderabad");
        outputStream.println("12");
        outputStream.close();
        return question3;
    } // END fileOutputQuestion3

    /* This method reads the file containing the question, answers and points␣
↪in order to create a record holding
    that data and return that record as a result. */
    public static QuizQuestion fileInputQuestion(String question_file) throws␣
↪IOException
    {
        BufferedReader inputStream = new BufferedReader(new␣
↪FileReader(question_file));
        String question = inputStream.readLine();
        String answer1 = inputStream.readLine();
        int score1 = Integer.parseInt(inputStream.readLine());
        String answer2 = inputStream.readLine();
        int score2 = Integer.parseInt(inputStream.readLine());
        String answer3 = inputStream.readLine();
        int score3 = Integer.parseInt(inputStream.readLine());
        inputStream.close();
        QuizQuestion questionRecord = createQuestion(question, answer1, score1,␣
↪answer2, score2, answer3, score3);
        return questionRecord;
    } // END fileInputQuestion

    /* This creates a question bank storing the question records
    which hold the questions and their data into an array and returning that␣
↪array as a result. */
    public static QuizQuestion[] createBank() throws IOException
    {
        String question1_file = fileOutputQuestion1();
        String question2_file = fileOutputQuestion2();
        String question3_file = fileOutputQuestion3();
        QuizQuestion question1 = fileInputQuestion(question1_file);
        QuizQuestion question2 = fileInputQuestion(question2_file);
        QuizQuestion question3 = fileInputQuestion(question3_file);
        QuizQuestion [] Question_Bank = new QuizQuestion[3];
        Question_Bank [0] = question1;
        Question_Bank [1] = question2;
        Question_Bank [2] = question3;
        return Question_Bank;
    } // END createBank

    /* This method stores the total score the user gets when they have answered␣
↪a question and tells the user how
    points they recieve for that answer and return the score value as a result.␣
↪*/
```

```
    public static int Question_score(QuizQuestion question1, String answer)
    {
        int score2;
        if (answer.equals(getAnswer1(question1)))
        {
            score2 = getPoints1(question1);
            System.out.println("This answer gives you " + score2 + " points.");
        }
        else if (answer.equals(getAnswer2(question1)))
        {
            score2 = getPoints2(question1);
            System.out.println("This answer gives you " + score2 + " points.");
        }
        else if (answer.equals(getAnswer3(question1)))
        {
            score2 = getPoints3(question1);
            System.out.println("This answer gives you " + score2 + " points.");
        }
        else
        {
            score2 = 100;
            System.out.println("This is not one of the options");
            System.out.println("This answer gives you 100 points.");
        }
        return score2;
    } // END Question_score

    /* This method asks the user which question they want to answer and gets a␣
↪response from the user and then
    user answers the question and stores the score they get and returns the␣
↪score as a result. */
    public static int printQuestion(QuizQuestion [] question_bank)
    {
        int score = 0;
        int choice = inputInt("Out of question 1, 2 or 3, which one do you want␣
↪to answer?");
        if (choice <= 3 & choice >= 1)
        {
            QuizQuestion question_used = question_bank[(choice - 1)];
            String answer = inputString(getQuestion(question_used));
            score = Question_score(question_used, answer);
        }
        else
        {
            System.out.println("That is not an option");
        }
        return score;
    } // END printQuestion

    /* This method is a primitive method of ADT TotalScore which creates a␣
↪record holding name and score of user doing
    the quiz and returns that record as a result. */
    public static TotalScore createScore(String given_name, String given_score)
    {
        TotalScore player = new TotalScore();
        player.name = given_name;
        player.score = given_score;
        return player;
    } // END createScore

    /* This primitive method of ADT TotalScore add to an existing array that is␣
↪of type
    string which holds the name and score of each user. */
    public static void AddToArray(String [] scoreArray, int position,␣
↪TotalScore player)
    {
        scoreArray [position] = player.name + " has score of " + player.score +␣
↪" points.";
        return;
    } // END AddToArray

    // This primitive method of ADT TotalScore prints the final total scores␣
↪and names of the users doing the quiz.
    public static void printScores(String [] scoreArray)
    {
        System.out.println("The scores of each user:");
        for (int i=0; i<scoreArray.length; i++)
        {
            System.out.println(scoreArray[i]);
        }
        System.out.println();
        return;
    } // END printScores

    /* This method creates a leaderboard of the scores using a sort algorithm␣
↪where the user with least points
    is first while the user with the most points is last. */
    public static void createTableScore(int [] board)
    {
        String option = inputString("Would you like to see the leaderboard? (Y/
↪N)");
        if (option.equals("Y"))
        {
            int temporary = 0;
            for (int pass=0; pass<board.length; pass++)
            {
                for (int b=1; b<board.length-pass; b++)
                {
                    if (board[b-1] > board[b])
                    {
                        temporary = board[b-1];
                        board[b-1] = board[b];
                        board[b] = temporary;
                    }
                }
            }
        }
        else
        {
            System.out.println("Thank you for taking part in the Quiz.");
            System.exit(0);
        }
        return;
    } // END createTableScore

    // This method prints the leaderboard of the scores where first has least␣
↪points and last has most points.
    public static void printTableScore(int [] board)
    {
        for (int i=0; i<board.length; i++)
        {
            System.out.println((i + 1) + ": " + board[i] + " points.");
        }
        return;
    } // END printTableScore

    /* This is the method doing all the work which asks the user how many␣
↪people are doing the quiz and then lets
    each user do the quiz and ask if they want to answer another question and␣
↪tallys up each users total scores
    and prints the scores and also prints a leaderboard if the user wants to␣
↪see it. */
    public static void QuizDetails() throws IOException
    {
        final QuizQuestion [] Quiz_questions = createBank();
        final int number_of_times = inputInt("How many people want to do the␣
↪Quiz?");
        int [] board = new int[number_of_times];
        String [] playersScores = new String[number_of_times];
        for (int i=0; i<number_of_times; i++)
        {
            System.out.println();
            String user_name = "Person " + (i + 1);
            System.out.println(user_name + " to answer question from question␣
↪bank:");
            int total = 0;
            int total2 = 0;
            String tries = "Y";
            while (tries.equals("Y"))
            {
                total = printQuestion(Quiz_questions);
                total2 = total2 + total;
                tries = inputString("Do you want to answer another question (Y/
↪N)");
            }
            TotalScore player = createScore(user_name, Integer.
↪toString(total2));
            AddToArray(playersScores, i, player);
            board [i] = total2;
            System.out.println();
        }
        printScores(playersScores);
        createTableScore(board);
        printTableScore(board);
        return;
    } // END QuizDetails
}
```

END OF LITERATE DOCUMENT