|  |  |
| --- | --- |
|  | |
| Array and File Applications | |
| **Updated** | **10.28.2018 2:41 PM** |

# **Problem 1:**

Use a 1-D array to solve the following problem: Write an application that asks the user to input five numbers, each between 10 and 100 (inclusive). As each number is read, display it if and only if it is not a duplicate of a number already read. Provide for the "worst case” in which all five numbers are different. Use the smallest possible array to solve this problem. Display the complete set of unique values input after the user inputs each new value.

# **Approach**

The first task was create and Array of integers and ensure users could add to the array. Once this worked I validated that the user entries are between 10 and 100 (inclusive) and if not the application should throw an error. The next task was to validate that the user entry is not a duplicate of an existing entry in the array. If both of these validations pass then the application reads the integer in to the Array and displays the complete set of unique values input by the user. Finally the entire block of code is wrapped in a Try – Catch for Input Mismatch exceptions.

NOTES: I was unsure from the question if the application should be limited to only 5 total user entries or to 5 valid user entries so I used the second option. Also it was unclear if all the entries should be entered by the user in one go or added one by one – I assumed one at a time. For simplicity I decided not to use the JOptionPane input and focus on the logic code.

# **1D Array Design**

Declare a NumArry of length 5 integers

Declare number of valid integers entered

Declare user number variable

Set Boolean valid to false

Create a new Scanner object

Output Instructions to user to enter integers between 10 and 100

While Valid is not false

And the number of valid numbers is less that array length

Loop

Set the duplicate flag to false

Set the number of user entries remaining

Ask the User to enter a integer

Try following block of code

If user entry is not an integer number

Catch and alert user input is not a valid integer

Set user number variable

Validate that integer is between 10 and 100

If number is not between 10 and 100 (inclusive)

Alert the user they have entered an invalid integer

Else if the integer is valid

Validates that the integer is not a duplicate

If it is a dupe set the duplicate flag to true

Alert the user they have entered a duplicate

If no duplicate exist continue

Enter the number in the Array

Set valid Boolean to True

Increase count of valid numbers

Display the complete set of unique values input

If loops complete tell user that the application is finished

EXIT

# **Testing**

The program was run several times with different inputs to ensure the code behaves as expected for each possible execution scenario including the input of invalid and out of bound parameters.

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Test** | **Expected Result** | **Actual Result** |
| 1 | Create Array and user adds a valid unique integers | Integer is added to Array and display the complete set of unique values input | **Pass** |
| 2 | Create Array and user adds a integer outside 10 and 100 range | Application throws error, does not add integer to Array and informs user that the entry is invalid | **Pass** |
| 3 | Create Array and user adds a integer duplicate of a previous entered integer | Application throws error, does not add integer to Array and informs user that the entry is a duplicate | **Pass** |
| 4 | Create Array and user adds a non-integer to the Array | Application throws an exception and informs user that the entry is not a valid integer | **Pass** |
| 5 | Create Array and user adds 5 valid unique integers | 5 Integers are added to Array and it display the complete set of 5 unique values input | **Pass** |

# **Source Code – Array1D.java**

|  |
| --- |
| /\* mmcmahon\_wk6\_Assignment\_Array1D.java  \* 1-D array to asks the user to input five numbers between 10 and 100 (inclusive) and display the complete  \* set of unique values input after the user inputs each new value.  \*/  **import** java.util.\*; //import java.util package  **public** **class** Array1D{  **public** **static** **void** main(String[] args) {  **int**[] NumArray = **new** **int**[5]; // Declare a NumArry of length 5 integers  **int** ValidNums = 0; // number of valid integers entered  **int** num = 0; // user number variable  **boolean** valid = **false**; // Set boolean valid to false  Scanner scan = **new** Scanner(System.***in***); // Create a new Scanner object    System.***out***.print("\nEnter five integers between 10 and 100"); //Instructions to user  **while** (!valid) {  //loop while the number of valid numbers is less that array length  **while** (ValidNums < NumArray.length) {    **boolean** containsNumber = **false**; //set the duplicate flag to false    **int** Remaining = (NumArray.length - ValidNums); // number of user entries remaining  System.***out***.print("\nYou have " +Remaining+ " integers left - enter a number: "); //User enter a integer    //block of statements to try  **try** {  num = scan.nextInt(); //Set user number variable    //validate that integer is between 10 and 100  **if** (num >= 10 && num <= 100) {  //validates that the integer is not a duplicate  **for** (**int** i = 0; i < NumArray.length; i++) {  **if** (num == NumArray[i]) {  containsNumber = **true**; //set the duplicate flag to true  // ValidNums++; // increase count of valid numbers  System.***out***.print("You have entered a duplicate! Try again.");  }  }  **if** (!containsNumber) { // if no duplicate exist continue  NumArray[ValidNums] = num; //Enter the number in the Array  valid = **true**; // set valid boolean to True  ValidNums++; // increase count of valid numbers    //Display the complete set of unique values input  System.***out***.print("Unique values are: ");  **for** (**int** i = 0; i < ValidNums; i++) {  System.***out***.print(NumArray[i] + " ");  }  System.***out***  .print("\n");  }    } // else number is not between 10 and 100 (inclusive)  **else** {  System.***out***.println("Out of range. Try again");  }  // Catch-User entry is not an integer number  } **catch** (InputMismatchException e) {  System.***out***.println("Not a valid integer. Try again");  scan.next();  }  }  // Tell user that the application is finished  System.***out***.print("\nFive valid integers have now been entered! ");  }  scan.close(); // close the scanner  System.*exit*(0); // exit  }  } |

# **Problem 2:**

Re-write the program above, this time using a collection class derived from Set (e.g. HashSet). As you will have discovered in your Research, sets handle duplicates differently from arrays, so your program's algorithm may be quite different using a set.

# **Approach**

As instructed I used the code from question 1 above and adapted it to use a HashSet. Since a HashSet does not store duplicates this simplified the code considerably. In question 1 I validated that the integer is not a duplicate by looping through the existing array integers and check if there was a duplicate – all this code could be removed.

NOTE: Because HashSet ignores duplicates the set loops only 5 times it does not prompt the user for a unique integer when a duplicate occurs unlike the Array loop in question 1. I did not have time to make both programs operate exactly the same so there is this difference in their operations.

# **1D ArrayHashSet Design**

Declare the max set size

Declare number of valid integers entered

Declare user number variable

Set Boolean valid to false

Create a new Scanner object

Declare the Integer HashSet

Output Instructions to user to enter integers between 10 and 100

While Valid is not false

And the number of valid numbers is less that array length

Loop

Set the duplicate flag to false

Set the number of user entries remaining

Ask the User to enter an integer

Try following block of code

If user entry is not an integer number

Catch and alert user input is not a valid integer

Set user number variable

Validate that integer is between 10 and 100

If number is not between 10 and 100 (inclusive)

Alert the user they have entered an invalid integer

Else if the integer is valid

If not duplicate the number added to HashSet

Set valid Boolean to True

Increase count of valid numbers

Display the complete set of unique values input

If loops complete tell user that the application is finished

EXIT

# **Testing**

The program was run several times with different inputs to ensure the code behaves as expected for each possible execution scenario including the input of invalid and out of bound parameters.

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Test** | **Expected Result** | **Actual Result** |
| 1 | Create Array and user adds a valid unique integers | Integer is added to Array and display the complete set of unique values input | **Pass** |
| 2 | Create Array and user adds a integer outside 10 and 100 range | Application throws error, does not add integer to Array and informs user that the entry is invalid | **Pass** |
| 3 | Create Array and user adds a integer duplicate of a previous entered integer | Application throws error, does not add integer to Array and informs user that the entry is a duplicate | **Pass** |
| 4 | Create Array and user adds a non-integer to the Array | Application throws an exception and informs user that the entry is not a valid integer | **Pass** |
| 5 | Create Array and user adds 5 valid unique integers | 5 Integers are added to Array and it display the complete set of 5 unique values input | **Pass** |

# **Source Code – Array1DHashSet.java**

|  |
| --- |
| /\* mmcmahon\_wk6\_Assignment\_Array1DHashSet.java  \* Re-write the program Array1D.java using a collection class derived from Set.  \*/  **import** java.util.\*; //import java.util package  **public** **class** Array1DHashSet {  **public** **static** **void** main(String[] args) {  **int** SetSize = 5; // Set the size of the HashSet  **int** ValidNums = 0; // number of valid numbers entered  **int** num = 0; // user number variable  **boolean** valid = **false**; // Set boolean valid to false  Scanner scan = **new** Scanner(System.***in***); // Create a new Scanner object  Set<Integer> set = **new** HashSet<Integer>(5); // Declare the Integer Set    System.***out***.print("\nEnter five integers between 10 and 100"); //Instructions to user  **while** (!valid) {  //loop while the number of valid numbers is less that array length  **while** (ValidNums < SetSize) {  **int** Remaining = (SetSize - ValidNums); // number of user entries remaining  System.***out***.print("\nYou have " + Remaining + " integers left - enter a number: "); //User enter a integer  //block of statements to try  **try** {  num = scan.nextInt(); //Set user entered Integer variable    //validate that integer is between 10 and 100  **if** (num >= 10 && num <= 100) {  valid = **true**; // set valid boolean to True  ValidNums++; // increase count of valid numbers  set.add(num); //Add the Integer to the Set  //Display the complete set of unique values input  System.***out***.println("Unique values are: ");  **for** (Integer i : set) {  System.***out***.printf(" %d", i); // output the set  }  }  **else** {  // else number is not between 10 and 100 (inclusive)  System.***out***.println("Out of range. Try again");  }  }  // Catch-User entry is not an integer number  **catch** (InputMismatchException e) {  System.***out***.println("Not a valid integer. Try again");  scan.next();  }  }  // Tell user that the application is finished  System.***out***.print("\nFive valid integers have now been entered! ");  scan.close(); // close the scanner  System.*exit*(0); // exit  }  }  } |

# **Problem 3:**

Write a Java program to read and process a text file. The name of the text file is entered by the user. If the file does not exist, the program displays an error message and exits. Otherwise, the file is opened and read, keeping a count of the number of lines, words, and total number of characters. These three values are displayed in a message box after the whole file has been read. After use, the file is closed. (Note that your character count should not include white spaces).

# **Approach**

He first task was to get a filename from the user and check if the file exists – if so proceed else using a Try / Catch mechanism output an error message and exit the application. The application will loop through the file until the end and count the total lines. Then loop through the lines counting the words using the white space as the word delineator. Then loop every character ignoring white spaces and count total character. Finally close the file and displayed in a message box the three values.

# **Read Text File Design**

Declare total words count variable

Declare total lines count variable

Declare total character count variable

Declare line variable

Request user to enter a file name

Catch exception if the file is not found

Open the input file

Read each line from the input file until null returned

Increase the total lines count

Read each word using white space as delineator

Increase the total word count

Remove white spaces

Increase the char count

The Line, Word and Char count values are displayed in a message box

Close the file

Exit application

# **Testing**

The program was run several times with different inputs to ensure the code behaves as expected for each possible execution scenario including the input of invalid and out of bound parameters.

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Test** | **Expected Result** | **Actual Result** |
| 1 | User enters a valid file name | The application runs and the number of lines, words, and total number of characters values are displayed in a message box. | **Pass** |
| 2 | User enters an invalid file name | The program displays an error message and exits | **Pass** |

# **Source Code – ReadTxtFile.java**

|  |
| --- |
| /\* mmcmahon\_wk6\_Assignment\_Array1DHashSet.java  \* Re-write the program Array1D.java using a collection class derived from Set.  \*/  **import** java.util.\*; //import java.util package  **public** **class** Array1DHashSet {  **public** **static** **void** main(String[] args) {  **int** SetSize = 5; // Set the size of the HashSet  **int** ValidNums = 0; // number of valid numbers entered  **int** num = 0; // user number variable  **boolean** valid = **false**; // Set boolean valid to false  Scanner scan = **new** Scanner(System.***in***); // Create a new Scanner object  Set<Integer> set = **new** HashSet<Integer>(5); // Declare the Integer Set    System.***out***.print("\nEnter five integers between 10 and 100"); //Instructions to user  **while** (!valid) {  //loop while the number of valid numbers is less that array length  **while** (ValidNums < SetSize) {  **int** Remaining = (SetSize - ValidNums); // number of user entries remaining  System.***out***.print("\nYou have " + Remaining + " integers left - enter a number: "); //User enter a integer  //block of statements to try  **try** {  num = scan.nextInt(); //Set user entered Integer variable    //validate that integer is between 10 and 100  **if** (num >= 10 && num <= 100) {  valid = **true**; // set valid boolean to True  ValidNums++; // increase count of valid numbers  set.add(num); //Add the Integer to the Set  //Display the complete set of unique values input  System.***out***.println("Unique values are: ");  **for** (Integer i : set) {  System.***out***.printf(" %d", i); // output the set  }  }  **else** {  // else number is not between 10 and 100 (inclusive)  System.***out***.println("Out of range. Try again");  }  }  // Catch-User entry is not an integer number  **catch** (InputMismatchException e) {  System.***out***.println("Not a valid integer. Try again");  scan.next();  }  }  // Tell user that the application is finished  System.***out***.print("\nFive valid integers have now been entered! ");  scan.close(); // close the scanner  System.*exit*(0); // exit  }  }  } |

# **References:**

*ArrayList Cheat Sheet - Blogs - Java Programming Forum - Learn Java Programming*. (2016). *Java-forums.org*. Retrieved 2 October 2016, from <http://www.java-forums.org/blogs/penguincoder/1566-arraylist-cheat-sheet.html>

*Counting number of lines, words, and characters in a text file*. (2016). *Stackoverflow.com*. Retrieved 2 October 2016, from <http://stackoverflow.com/questions/15239371/counting-number-of-lines-words-and-characters-in-a-text-file>

Deitel, H. & Deitel, P. (2010). *Java* (9th ed.). Upper Saddle River, NJ: Prentice Hall.

*Duplicate Elimination*. (2016). *Stackoverflow.com*. Retrieved 2 October 2016, from <http://stackoverflow.com/questions/1770055/duplicate-elimination>

file, C. (2016). *Counting number of words in a file*. *Stackoverflow.com*. Retrieved 2 October 2016, from <http://stackoverflow.com/questions/4094119/counting-number-of-words-in-a-file>

*how to calculate the length of a string without spaces in java*. (2016). *Stackoverflow.com*. Retrieved 2 October 2016, from <http://stackoverflow.com/questions/29127884/how-to-calculate-the-length-of-a-string-without-spaces-in-java>

*Java Examples | Other Java Examples | Scanner That Only Takes Valid Numbers*. (2016). *Javacodex.com*. Retrieved 2 October 2016, from <http://www.javacodex.com/Other-Java-Examples/Scanner-That-Only-Takes-Valid-Numbers>

*Java One Dimensional Array Help - Java | Dream.In.Code*. (2016). *Dreamincode.net*. Retrieved 2 October 2016, from <http://www.dreamincode.net/forums/topic/271016-java-one-dimensional-array-help/>