Lab Exercises

Lab Exercise 1 — Duplicate Elimination

Name: ______________________ **Date:** ______________________

Section: ______________________

This problem is intended to be solved in a closed-lab session with a teaching assistant or instructor present. The problem is divided into six parts:

1. Lab Objectives
2. Description of Problem
3. Sample Output
4. Program Template (Fig. L 7.1 and Fig. L 7.2)
5. Problem-Solving Tips
6. Follow-Up Questions and Activities

The program template represents a complete working Java program, with one or more key lines of code replaced with comments. Read the problem description and examine the sample output; then study the template code. Using the problem-solving tips as a guide, replace the `/* */` comments with Java code. Compile and execute the program. Compare your output with the sample output provided. Then answer the follow-up questions. The source code for the template is available at `www.pearsonhighered.com/deitel`.

Lab Objectives

This lab was designed to reinforce programming concepts from Chapter 7 of *Java How to Program: 8/e*. In this lab, you will practice:

- Declaring and initializing arrays.
- Comparing input to array elements.
- Preventing array out-of-bounds errors.

The follow-up questions and activities will also give you practice:

- Initializing array sizes during program execution.
- Generalizing programs.

Problem Description

Use a one-dimensional array to solve the following problem: Write an application that inputs five numbers, each of which is between 10 and 100, inclusive. As each number is read, display it 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.

Lab Exercises

Name:

Lab Exercise 1 — Duplicate Elimination

Sample Output

```
Enter number: 11
11
Enter number: 85
11 85
Enter number: 26
11 85 26
Enter number: 11
11 has already been entered
11 85 26
Enter number: 41
11 85 26 41
```

Program Template

```java
// Lab 1: Unique.java
// Reads in 5 unique numbers.
import java.util.Scanner;

public class Unique
{
   // gets 5 unique numbers from the user
   public void getNumbers()
   {
      Scanner input = new Scanner( System.in );

        /* Create an array of five elements*/
      int count = 0; // number of uniques read
      int entered = 0; // number of entered numbers

      while( entered < numbers.length )
      {
         System.out.print( "Enter number: " );
         /* Write code here to retrieve the input from the user */

         // validate the input
         /* Write an if statement that validates the input */
         {
            // flags whether this number already exists
            boolean containsNumber = false;

            // increment number of entered numbers
            entered++;

            /* Compare the user input to the unique numbers in the array using a for
               statement. If the number is unique, store new number */

            /* add the user input to the array only if the number is not already
               in the array */
            if ( !containsNumber )
            {
               /* Write code to add the number to the array and increment
                  unique items input */
            } // end if
```

Fig. L 7.1 | Unique.java. (Part 1 of 2.)

Lab Exercises

Name:

Lab Exercise 1 — Duplicate Elimination

```
               else
                  System.out.printf( "%d has already been entered\n",
                     number );
            } // end if
            else
               System.out.println( "number must be between 10 and 100" );

            // print the list of unique values
            /* Write code to output the contents of the array */
         } // end while
      } // end method getNumbers
   } // end class Unique
```

Fig. L 7.1 | `Unique.java`. (Part 2 of 2.)

```
   // Lab 1: UniqueTest.java
   // Test application for class Unique
   public class UniqueTest
   {
      public static void main( String args[] )
      {
         Unique application = new Unique();
         application.getNumbers();
      } // end main
   } // end class UniqueTest
```

Fig. L 7.2 | `UniqueTest.java`.

Problem-Solving Tips

1. Initialize the integer array `numbers` to hold five elements. This is the maximum number of values the program must store if all values input are unique.
2. Remember to validate the input and display an error message if the user inputs invalid data.
3. If the number entered is not unique, display a message to the user; otherwise, store the number in the array and display the list of unique numbers entered so far.
4. If you have any questions as you proceed, ask your lab instructor for assistance.