Lab 3

Creating and Using Arrays

Objectives

Upon completion of this lab, you should be able to:

- Complete review questions
- Create and use a one-dimensional array
- Use arrays in a loop
- Parse run time arguments
- Create and use two-dimensional arrays

Lab Overview

In this lab, you complete the review questions and four exercises.

- In the first exercise, you create and use one-dimensional arrays of primitive types and of objects.
- In the second exercise, you use loops in various ways to initialize an array.
- In the third exercise, you use the `args[]` array of the `main` method.
- In the fourth exercise, you create and use two-dimensional arrays.

Completing Review Questions

Complete the following questions:

1. The following code is the correct syntax for _____ an array

   ```
   array_identifier = new type [length];
   ```

   a. Declaring

   b. Setting array values

   c. Instantiating

   d. Declaring, instantiating, and setting an array of values

2. State whether the following statements are true or false:

   a. The syntax for declaring, instantiating, and setting an array of values is:
   ```
   type [] array_identifier =
   {comma-separated_list_of_values_or_expressions};
   ```

   b. The syntax for setting array values in an array is:
   ```
   array_identifier[index] = value;
   ```

3. Given the following array declaration, determine whether the following statements are true or false.

```
int [ ] autoMobile = new int [13];
```

   a. `autoMobile[0]` is the reference to the first element in the array.

   b. `autoMobile[13]`is the reference to the last element in the array.

   c. There are 13 integers in the `autoMobile` array.

4. A two-dimensional array is similar to a _____:

   a. Shopping list

   b. List of chores

   c. Matrix

   d. A bar chart containing the dimensions for several boxes

Exercise 1: Creating and Using One-Dimensional Arrays

The objective of this exercise is to write classes that use one-dimensional arrays of primitive data types and of objects.

This exercise consists of two tasks:

- "Task 1 – Creating a Class With a One-Dimensional Array of Primitive Types"
- "Task 2 – Creating a Class With a One-Dimensional Array of Objects"

Preparation

Ensure that the `VacationScaleTest.java` file and the `Shirt.java` file exist in the `SL110/exercises/10_arrays/exercise1` directory. This is your working directory.

Task 1 – Creating a Class With a One-Dimensional Array of Primitive Types

The number of vacation days that an employee at DirectClothing, Inc. receives is based on the number of years that the employee has worked for DirectClothing, Inc. Following is the vacation scale for DirectClothing, Inc.

Number of years of employment	**Number of days of vacation**
Up to 1 year	10
1, 2, or 3 years	15
4 or 5 years	20
6 or more years	25

In this task, you write a class called `VacationScale` containing an array of seven elements. Assume that the number of years of employment is relative to the position of the element in the array. Each element contains the number of days of vacation corresponding to the number of years of employment.

Follow these steps to write your class:

1. Open the terminal window and go to the `SL110/exercises/10_arrays/exercise1` directory.
2. Write a class called `VacationScale` that contains a member variable called `vacationDays` of type `int[]`. Each element will contain the number of days of vacation relative to its index in the array. For example, index 0 represents 0 years of service, so `vacationDays[0]` will contain the number 10, and index 4 represents 4 years of service, so `vacationDays[4]` will contain the number 20.
3. Create a `public` method called `setVacationScale` with no argument and no return type. Initialize `vacationDays` to size seven and set values in all the elements of the `vacationDays` array.
4. Create a `public` method called `displayVacationDays` that returns nothing, accepts an `int` (`yearsOfService`) argument, and displays the number of days of vacation the employee receives. For example, if a 1 is passed to `displayVacationDays`, then the number 15 is displayed.

   Hint: You can use a variable within the square brackets when retrieving values from an array. For example: `vacationDays[yearsOfService]`
5. Compile and execute your class using the provided `VacationScaleTest` class. The output of your program should be similar to:

```
Your vacation is: 15
Your vacation is: 20
Your vacation is: 25
```

Task 2 – Creating a Class With a One-Dimensional Array of Objects

In this task, you write a program that creates and assigns values to an array of `Shirt` objects. Complete the following steps:

1. Go to the `SL110/exercises/10_arrays/exercise1` directory.
2. Write a class called `ShirtArrayTest` that:
   - Creates three `Shirt` objects and set their value using a constructor. The constructor for the `Shirt` class is:

```
public Shirt(int shirtID, String description, char
colorcode, double price, int quantityInStock)
```

- Places each `Shirt` object reference in an array.
- Uses the array to display the values of the `Shirt` object by using the `displayInformation` method of the `Shirt` object.

3. Compile and execute your `ShirtArrayTest` class. Examine the output.

Exercise 2: Using Loops and Arrays

The objective of this exercise is to use a loop for initializing an array.

Preparation

Ensure that the `VacationScaleTwoTest.java` file exists in the `SL110/exercises/10_arrays/exercise2` directory. This is your working directory.

Task – Using a Loop to Initialize an Array

In this task, you rewrite the `setVacationScale` method in the `VacationScale` class to use a loop. You had created `VacationScale` class in the previous exercise. Follow these steps to rewrite your method:

1. Go to the working directory.
2. Make a copy of the `VacationScale.java` file located in the `SL110/exercises/mod10_arrays/exercise1` directory to the current directory, and rename the copied file as `VacationScaleTwo.java`.
3. Edit the `VacationScaleTwo.java` file, and change the class name to `VacationScaleTwo`.
4. Rewrite the method `displayVacationDays` and use a loop to iterate through the `vacationDays` array and display the values stored in each element of the array. The method should not accept any argument.
5. Compile and execute your class using the `VacationScaleTwoTest` class, which is provided.

Note – Modify Step 4 to use the enhanced `for` loop to accomplish this task and then verify the output.

Exercise 3: Parsing the args[] Array

The objective of this exercise is to use the `args[]` array within the `main` method to create a number guessing game.

In this exercise, you create a class, which accepts a runtime argument between 1 to 5. You also randomly generate a number between 1 to 5 in the class and compare the value of the argument with the randomly generated number.

Task – Creating a Game Using the args [ ] Array

In this task, you write a game that accepts an argument and displays an associated message. Follow these steps to write your game:

1. Go to the `SL110/exercises/mod10_arrays/exercise3` directory.
2. Write a `GuessingGame` class that contains a `main` method that accepts one argument: any number in the range of 1 to 5 or the word "help."
3. If the `GuessingGame` class receives the word "`help`" as an argument, the `GuessingGame` class displays the usage of the program.
4. If a 1, 2, 3, 4, or 5 is entered:
   a. Generate a random number 1 to 5 by using the following code snippet:

      ```
      randomNum =((int)( Math.random()*5)+1);
      ```

   b. Compare the argument to this random number. If the value of the argument and the random number are equal, display a message to the user that indicates that the user has guessed the correct number. If the value of the argument and the random number are not equal, display the random number, and ask the user to try again.
5. If an invalid number is entered (less than 1 or greater than 5), display a message to the user that indicates that the user entered an invalid number.
6. Save the file with the name `GuessingGame.java`.
7. Compile your class. Execute your class several times and examine the output on providing a different argument each time.

Exercise 4: Creating and Using Two-Dimensional Arrays

The objective of this exercise is to create and access a two-dimensional array.

This exercise is based on the scenario of a classroom. A classroom has 12 desks arranged in a rectangular grid comprising of 3 rows and 4 columns. Students are allocated a desk at the position found vacant first, by traversing each row.

Figure 3-1 shows the class map as a grid. Each cell represents a desk. Each cell contains the co-ordinates of the desk position in the class map.

XXXXX	Col 1	Col 2	Col 3	Col 4
Row 1	0,0	0,1	0,2	0,3
Row 2	1,0	1,1	1,2	1,3
Row 3	2,0	2,1	2,2	2,3

Figure 3-1 Rectangular Grid Showing the Class Map

In this exercise, you create a Java technology class to represent a classroom map, allocate desk positions to the students of a class, display the desk map of the class, and search for the allocated desk position of a student.

Preparation

Ensure that the `ClassMapTest.java` file exists in the `SL110/exercises/10_arrays/exercise4` directory. This is your working directory.

This exercise consists of three tasks:

- "Task 1 – Creating a Class With a Two-Dimensional Array"
- "Task 2 – Traversing a Two-Dimensional Array"
- "Task 3 – Testing a Class Having a Two-Dimensional Array"

Task 1 – Creating a Class With a Two-Dimensional Array

In this task, you create a class that has a two-dimensional array as its member variable. It is a `String` type array and the size of the array is [3][4].

Each element of the array represents a desk at a position matching its array index. For example, desk[2][1] represents a desk at row 3 and column 2. Refer to Figure 3-1 on page L3-9 for more clarity. Each element in the array stores the name of a student who is allocated that desk in the classroom.

Complete the following steps:

1. Make the `SL110/exercises/10_arrays/exercise4` directory your working directory.
2. Create a class called `ClassMap` and save it as `ClassMap.java`.
3. In the class, declare two member variables, `roomNo` of type `int` and `deskArray` of type `String[][]`.
4. Create an argument constructor that accepts an `id` of type `int`. In the constructor:
   a. Assign the value of the argument to the member variable `RoomNo`.
   b. Initialize the size of the `deskArray` to [3][4].
5. Save the `ClassMap` class.

Task 2 – Traversing a Two-Dimensional Array

In this task, you write certain methods in the `ClassMap` class that traverse the `deskArray` to:

- Allocate students to desks
- Display the classroom layout
- Search the desk position of a student in the class

Complete the following steps:

1. Create a method by the name `setDesk` that accepts a `String` argument and returns no value. This argument will contain the name of the student
2. In the `setDesk` method, write code to perform the following tasks:
   a. Traverse the `deskArray` array to identify the first vacant element in it. You can use a nested `for` loop for this purpose. For example:
   ```
   for ( int row=0;row<3;row++){
   for (int col=0;col<4;col++){
   if (deskArray[row][col]==null){
   ```
   b. Assign the student's name to the vacant element.
   c. Print the position of the desk for the student and exit out of the loops.

Note – You can use `break` statement to branch out of a running loop.

3. Create a `public` method by the name `displayDeskMap` with no argument and no return type. In the method, write code to traverse through the `deskArray` and print the names in each element of the array, such that the names are displayed in grid form.
4. Create a method by the name `searchDesk` that accepts a `String` argument and returns no value. The argument represents the name of the student whose desk position has to be searched from the `deskArray`.
5. In the `searchDesk` method, write code to do the following:
   a. Create a nested `for` loop to traverse through the `deskArray` array.
   b. If the array element is not null, compare the value of the method argument with the element. For example:
   ```
   if (deskArray[row][col] !=null &&
   deskArray[row][col].equals(name)==true){
   ```
   c. Print the position of the desk if the names are equal.
   d. Print an error message if the name is not found in the `deskArray` array.
   e. Use the `break` statement to branch or exit out of the loops wherever required.
6. Save and close the `ClassMap.java` file.

Task 3 – Testing a Class Having a Two-Dimensional Array

Complete the following steps:

1. Open the `ClassMapTest.java` file provided to you in the working directory.
2. Study the `main` method and check if all the methods of `ClassMap` class have been invoked.
3. Compile `ClassMapTest.java`, execute it, and verify the output.

Exercise Summary

Take a few minutes to identify what experiences, issues, or discoveries you had during the lab exercises.

- Experiences
- Interpretations
- Conclusions
- Applications