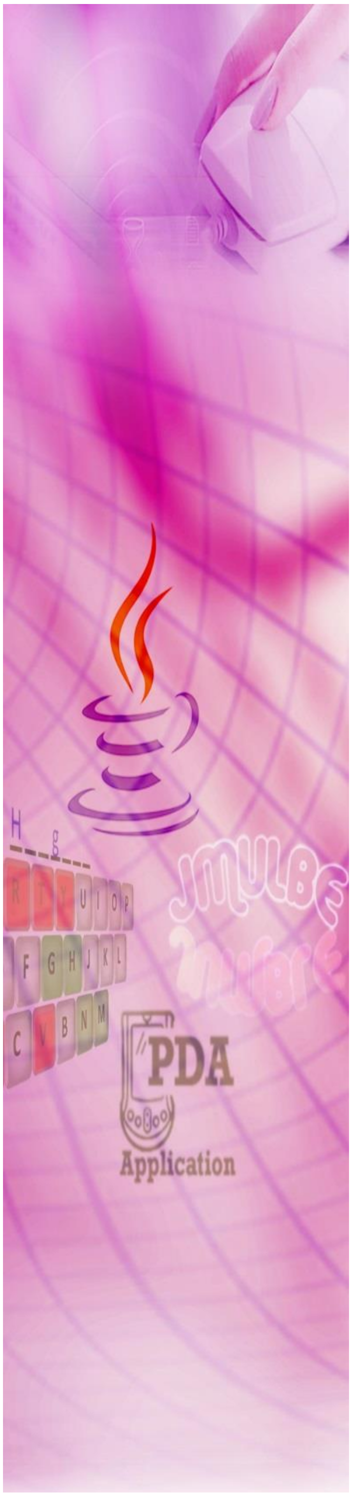

Instructor Inputs

Session 5

Session Overview

This session covers Chapter 4 of the book, Introduction to Java – SG. This session will help the students to become familiar with the arrays, enums, and strings. Moreover, they will learn to create and access arrays and enums.

Manipulating Arrays

Handling Tips

Start the session by discussing the scenario of the Classic Jumble Word game, where you need to store 100 different words. Then, ask the students what they will do to store 100 different words. Elicit the responses, and then tell them that you can declare 100 variables to store 100 different words. However, it is difficult to keep track of 100 variables in a program and it makes the program code long and complex. Therefore, in such a situation, you need to declare a variable that can store 100 words. This can be achieved by declaring an array. Java supports the following types of arrays:

- One-dimensional array
- Multidimensional array

In addition, while discussing the topic, Creating Arrays, explain the three-dimensional array with the help of the information given in the Additional Inputs topic.

Additional Inputs

Three-dimensional Array

The creation of a three-dimensional array involves the following three steps:

1. Declare a three-dimensional array.
2. Assign values to the three-dimensional array.
3. Access a three-dimensional array.

Declaring a Three-dimensional Array

The following syntax is used to declare a three-dimensional array:

```
arraytype arrayname[][][]=new arraytype[tables][rowsize][columnsize];
```

In the preceding syntax, `tables` specifies the number of tables, `rowsize` specifies the number of rows, and `columnsize` specifies the number of columns.

The following code snippet declares a three-dimensional array:

```
String[][][] words=new String[2][2][2];
```

The preceding code snippet creates an array of string named `words`, which contains two tables, two rows, and two columns.

Assigning Values to the Three-dimensional Array

You can assign values to each element of the array by using the index number of the element. For example, to assign the value, `alpep`, to 0th table, 0th row, and 0th column and to assign the value, `apple`, to the 0th table, 0th row, and 1st column, you can use the following code snippet:

```
words[0][0][0]= "alpep";
words[0][0][1]= "apple";
```

You can also assign values to an array at the time of its declaration, as shown in the following code snippet:

```
String[][][] words = new String[][][] {
    {
       {"fruit","elapp", "apple"},{"country","acihn", "china"}

    }
};
```

Accessing a Three-dimensional Array

To access a three-dimensional array, the following syntax is used:

```
arrayname[tables][rowsize][columnsize]
```

In the preceding syntax, `arrayname` specifies the name of the array and `tables`, `rowsize`, and `columnsize` specify the location of the array element.

Consider the following code snippet:

```
String[][][] words = new String[][][] {
    {
       {"fruit","elapp", "apple"},{"country","acihn", "china"}

    }
};
System.out.println(words[0][0][1]);
```

In the preceding code snippet, `System.out.println(words[0][0][1]);` prints the element stored at the 0th table, 0th row, and 1th column. However, if you want to display all the elements, you can use the `for` loop.

Manipulating Enums

Handling Tips

Discuss the scenario where you want to restrict a person to select `Carrie`, `Fairchild`, or `Haden` for the mango type. If the person enters the mango type as `Kent`, then it should cause an error. Then, ask the students what they will do to accomplish the preceding task. Elicit the responses, and then tell them that this task can be accomplished by using the `if` construct. However, if you need to add few more types of mangoes, then the number of `if` constructs increases, and it gradually becomes difficult to work with them. For this, Java provides enum, which is a special data type that allows a variable to be a set of predefined constants. In order to use an enum, you need to know how to declare and access it. Explain the concept of accessing enums with the help of the example given in the Additional Examples topic.

Additional Examples

Accessing Enums

The following code demonstrates the usage of enums with the `switch` construct:

```
enum Mango {
Carrie, Fairchild, Haden
}
class EnumDemo
{
   public static void main(String args[])
   {
         Mango ap;
         ap = Mango.Fairchild;
         if(ap == Mango.Fairchild)
         System.out.println("Value is: " + ap);
         switch(ap)
         {
               case Carrie:
               System.out.println("Carrie is green.");
               break;
               case Fairchild:
               System.out.println("Fairchild is red.");
               break;
               case Haden:
               System.out.println("Hadden is yellow.");
               break;
         }
   }
}
```

The output of the preceding code is:

```
Value is: Fairchild
Fairchild is red.
```

Manipulating Strings

Handling Tips

Discuss a scenario of the Classic Jumble Word game, where Sam wants a user to enter the name before playing the game. Thereafter, the name should appear with the message, “Welcome [user name]”. Then, ask the students what they will do to display the preceding message before playing the game. Elicit the responses, and then tell them to implement this functionality, there is a need to store the name of the user and then append the name with the welcome message. Here, the name and the welcome message are string values. To manipulate strings, Sam can use the classes, such as `String`, `StringBuilder`, and `StringBuffer`, provided by Java. These classes will enable a user to store string literals and provide various methods for string manipulations.

In addition, explain the concept of the `StringBuilder` and the `StringBuffer` classes with the help of the example given in the Additional Examples topic.

Further, you can tell the students about the various types of the `StringBuilder` class constructor and the methods with the help of the information given in the Additional Inputs topic.

Additional Inputs

The following table describes the commonly used class constructors of the `StringBuilder` class.

Constructor	***Description***
`StringBuilder()`	*Creates a string builder with no character in it and an initial capacity of 16 characters.*
`StringBuilder(CharSequence seq)`	*Creates a string builder that contains the same character as that of CharSequence.*
`StringBuilder(int capacity)`	*Creates a string builder with no character in it and an initial capacity specified by the capacity argument.*
`StringBuilder(String str)`	*Creates a string builder initialized to the contents of the specified string.*

The Constructors of the StringBuilder Class

The following table describes the commonly used methods of the `StringBuilder` class.

Method	***Description***
`StringBuilder append(boolean b)`	*Appends the boolean string argument to the sequence.*
`int capacity()`	*Returns the current capacity.*
`int codePointBefore(int index)`	*Returns the character (Unicode code point) before the specified index.*

The Methods of the StringBuilder Class

Additional Examples

Using the StringBuilder Class

The following code demonstrates the usage of the `StringBuilder` class:

```
public class StringBuilderDemo
{
    public static void main(String[] args)
    {
    StringBuilder str = new StringBuilder("Fruits are good. True or false: ");
```

```
        str.append(true);
        System.out.println( "After append = " + str);
        System.out.println("capacity = " + str.capacity());
        int retval = str.codePointBefore(3);
        System.out.println("Character(unicode point) = " + retval);
        }
    }
```

Activity 4.1: Manipulating Arrays and Strings

Handling Tips

Discuss the problem statement with the students.

To perform the activity, 4.1, you need to use the **Activity3.2.txt** file, which is provided at the following location in the TIRM CD:

- **Datafiles For Faculty\Activities\Chapter 04\Activity 4.1\Input Files**

The solution file, **Hangman.java**, for this activity is provided at the following location in the TIRM CD:

- **Datafiles For Faculty\Activities\Chapter 04\Activity 4.1\Solution**

FAQs

- *How can the elements of an array be copied to another array?*

  Ans: The elements of an array can be copied by using the `System.arraycopy()` method. This method is used to copy any number of elements from a source array to a destination array. The syntax of the method is:

  ```
  System.arraycopy(arr_source, start_ind_src, arr_dest, start_ind_dest, n);
  ```

  In the preceding syntax, `arr_source` specifies the name of the source array. `start_ind_src` specifies the position of the source array, from which the copy operation should start. `arr_dest` specifies the name of the destination array. `start_ind_dest` specifies the starting position in the destination array, where the copied element will be inserted, and `n` specifies the number of array elements to be copied.

  Consider the following example:

  ```
  int[] array1 = {1,2,3,4,5,6};
  int[] array2 = new int[6];
  System.arraycopy(array1, 0, array2, 0, 6);
  ```

  In the preceding example, the `System.arraycopy()` method copies all the elements of `array1` to `array2`.

- *How is an array of objects initialized?*

  Ans: When you create an array of objects, each element of the array is initialized to null. You need to initialize the individual elements by invoking a constructor of the relevant class.

Consider the following code snippet, where a constructor is used to initialize each element of an array whose elements are objects of the `Integer` class:

```
Integer[] values = new Integer[3];
values[0] = new Integer(“1”);
values[1] = new Integer(“2”);
values[2] = new Integer(“3”);
```

- *Is string a data type in Java?*

  Ans: No, string is not a data type in Java.