Objectives

- In this session, you will learn to:
  - Manipulate arrays
  - Manipulate enums
  - Manipulate strings

Manipulating Arrays

- Scenario:

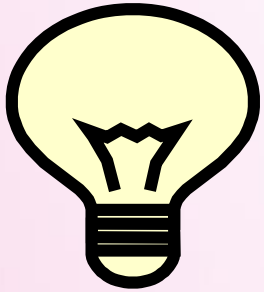
A Programmer

Needs to store 100 different words that will be used in the game.

Therefore, to store these values, a programmer needs to declare 100 variables.

Manipulating Arrays (Contd.)

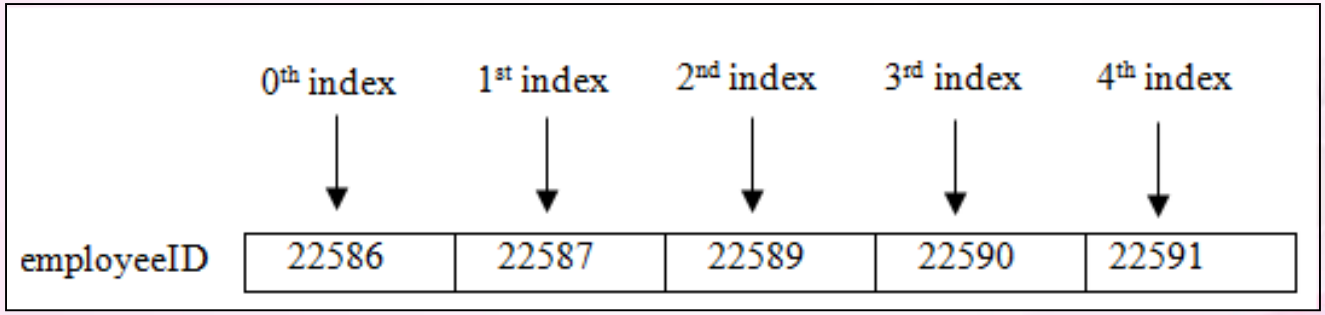
- Scenario (Contd.):



However, it is difficult to keep track of 100 variables in a program which makes the program code long and complex. Therefore, in such a situation, a programmer needs to declare a variable that can store 100 words. This can be achieved by declaring an array variable.

Manipulating Arrays (Contd.)

- An array is a collection of elements of a single data type stored in adjacent memory locations.
- An array element can be accessed by specifying the name and the subscript number of the array.
- The subscript number:
  - Specifies the position of an element within the array.
  - Is also called the index of the element.
- The following figure shows the array of employeeID.



Creating Arrays

- You can create the following types of arrays:
  - One-dimensional array
  - Multidimensional array

- One-dimensional array:
  - Is a collection of elements with a single index value.
  - Can have multiple columns but only one row.

Creating Arrays (Contd.)

- The creation of a one-dimensional array involves two steps:
  1. Declare an array.
  2. Assign values to the array.
- One-dimensional array is declared by using the following syntax:

```
arraytype arrayname[] = new arraytype[size] ;
```

- The following code snippet declares an array to store three string values:

```
String jumbledWords[] = new String[3];
```

- You can assign values to each element of the array by using the index number of the element.

Creating Arrays (Contd.)

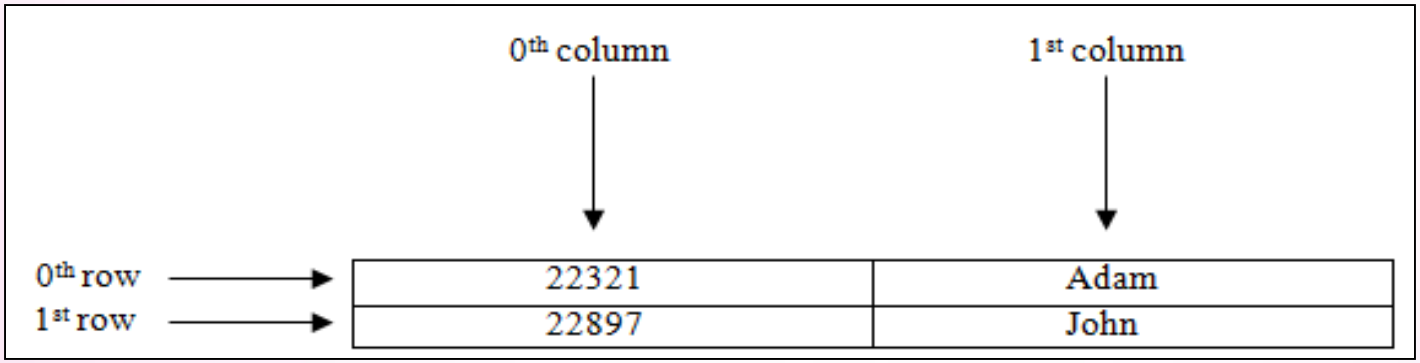
- You can also assign values to the array at the time of declaration.
- To assign values at the time of declaration, you are not required to specify the size of the array, as shown in the following code snippet:

```
String jumbledWords[] ={"alpep","argneo","rgaeps"};
```

- Multidimensional arrays are arrays of arrays.
- The commonly used multidimensional array is a two-dimensional array where you can have multiple rows and columns.

Creating Arrays (Contd.)

- The following figure shows a two-dimensional array.



- The creation of a two-dimensional array involves two steps:
  1. Declare an array.
  2. Assign values to the array.

Creating Arrays (Contd.)

- You can declare a two-dimensional array by using the following syntax:

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

- The following code snippet declares a two-dimensional array:

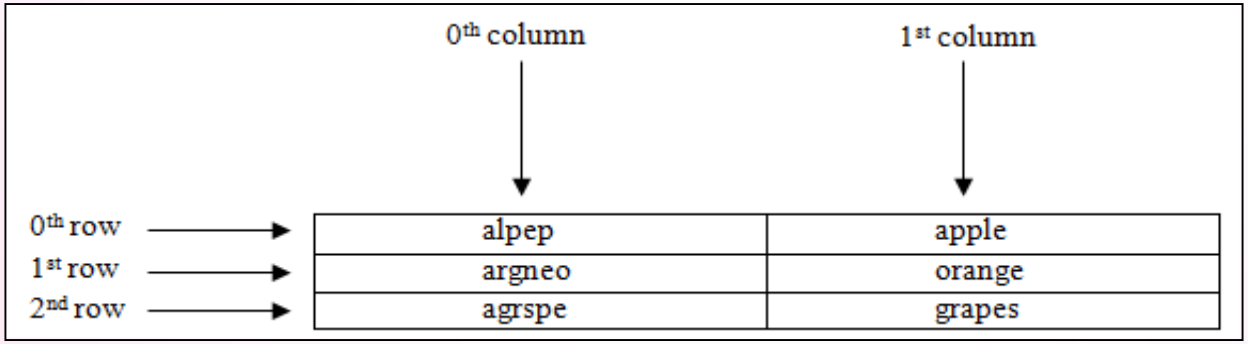
```
String[][] words = new String[4][2];
```

- You can assign values to each element of the array by using the index number of the element.
- You can also assign values to the array at the time of declaration, as shown in the following code snippet:

```
String[][] jumbledWords = new String[][]
{{"elapp","apple"},{"argneo","orange"},
{"agrspe","grapes"}};
```

Creating Arrays (Contd.)

- The following figure shows a two-dimensional array, `JumbledWords`.



Just a minute

- Identify the total number of elements, if an array is declared as:

```
int [] arr = new int [5];
```

  - 3
  - 4
  - 5
  - 6

Just a minute (Contd.)

- Solution:
  - 5

Accessing Arrays

- To perform various manipulations on the array, you need to access the following types of arrays:
  - One-dimensional array
  - Two-dimensional array
- To access a one-dimensional array, the following syntax is used:

```
arrayname[index];
```

- To display all the elements stored in the array, you can use the `for` loop, as shown in the following code snippet:

```
String jumbledWords[] =
{"alpep","argneo","rgaeps"};
for(int i=0;i<3;i++)
System.out.println(jumbledWords[i]);
```

Accessing Arrays (Contd.)

- However, if you do not know the total number of elements in the array, then traversing through the entire array will be difficult. This can be simplified by using the `length` property of an array.
- The following code snippet is used to traverse through the array using the `for` loop and the `length` property:

```
String jumbledWords[] =
{"alpep","argneo","rgaeps"};
for(int i=0;i<jumbledWords.length;i++)
System.out.println(jumbledWords[i]);
```

- Java provides the `for-each` loop to iterate through an array. This loop increases the readability and simplifies the code.
- The syntax of the `for-each` loop to use in an array is:

```
for(type var: arrayobject)
```

Accessing Arrays (Contd.)

- The following code snippet is used to display all the elements stored in the array using the `for-each` loop:

```
String[] jumbledWords =
{"alpep","argneo","rgaeps"};
        System.out.println("Elements stored in
        array are: ");
        for (String i : jumbledWords)
        {
        System.out.println(i);
}
```

- Two-dimensional array is accessed by using the following syntax:

```
arrayname[row][column]]
```

Accessing Arrays (Contd.)

- However, if you want to display all the elements, you can use the `for` loop, as shown in the following code snippet:

```
String[][] jumbledWords = new String[][]{

{"elapp","apple"},{"argneo","orange"},{"agrspe","gr
apes"}};
System.out.println("Elements stored in array are:
");
for (int i=0; i<2; i++)
{
for (int j=0; j<2; j++)
{
System.out.print(jumbledWords[i][j]);
}
```

Accessing Arrays (Contd.)

- You can use the `length` property in the `for` loop, as shown in the following code snippet:

```
int a[][] = {{1,2},{4,3}};
for(int i=0; i<a.length; i++)
{
for(int j=0; j<a[i].length; j++)
System.out.println(a[i][j]);
}
```

Accessing Arrays (Contd.)

- Further, you can use the following code snippet to display all the elements stored in the two-dimensional array using the `for-each` loop:

```
String[][] jumbledWords = new
String{"elapp","apple"},{ "argneo","orange"},{
"agrspe","grapes"}};;

System.out.println("Fruits are: ");
for (String[] i : jumbledWords)
{
    for (String j : i)
    {
        System.out.println(j);
    }
}
```