Learning Goals

Click to Continue

This session will help you to understand

- What is an array ?
- Types of array.
- How to implement arrays?

Array Analogy

Click to Continue

A basket of balls. Basket is the container for balls.

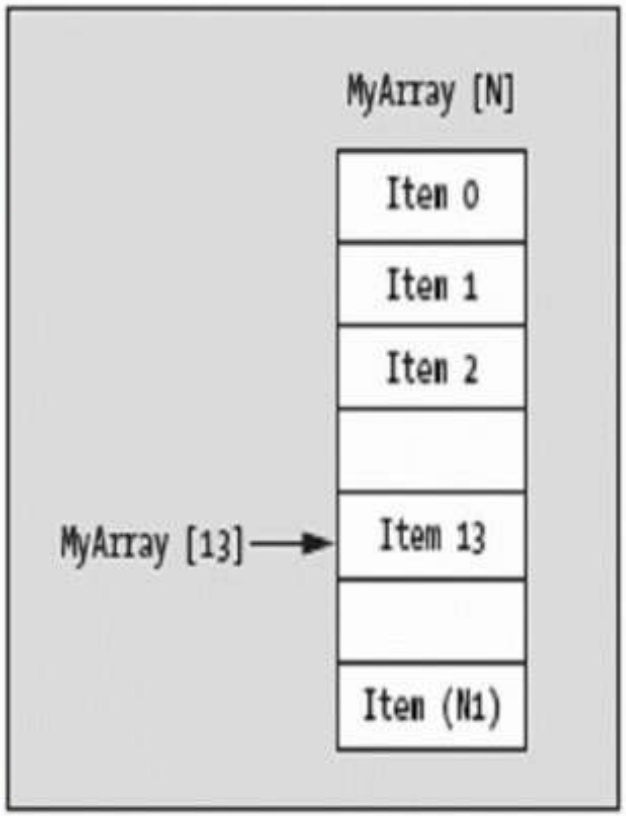
Arrays are like baskets they hold data of similar data type (like the balls).

Array

Click to Continue

An array is data type that can hold a collection of values of ***same type*** which can be referred with a common name.

Arrays can hold primitives or wrapper objects or user defined objects.



An array with name ***DemoArray*** storing a collection of values.

Each item can be accessed using the name ***DemoArray*** and index.

Array Types

Click to Continue

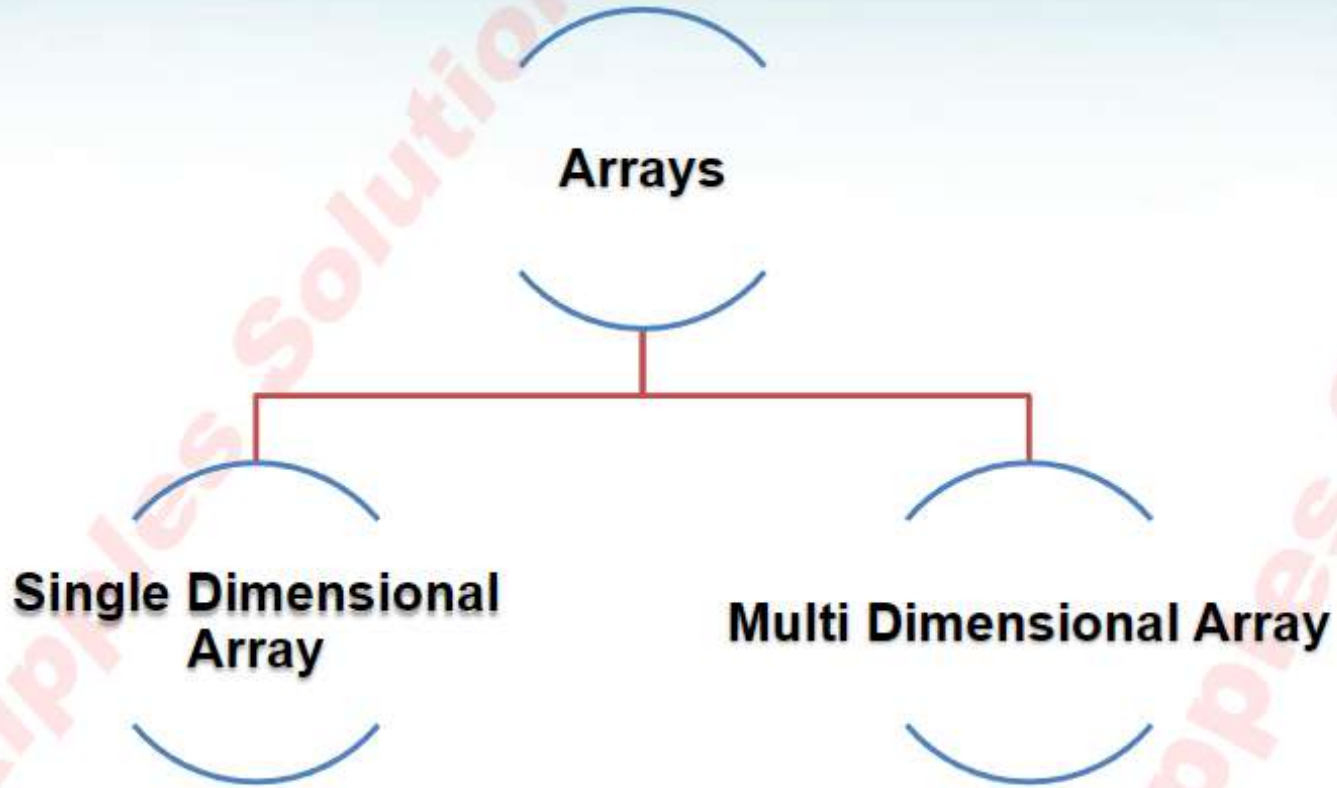


Illustration: `float[] studentId ;`
//Float array named studentId to store student id.

Illustration: `double square[ ][ ];`
// Array which stores a double value in 2*2 matrix.

Implementing Arrays

Click to Continue

Step 1: Array declaration:

In this step, array is declared associating to a data type and a identifier.

Syntax : <type> [ ] <array-name>;

Illustration : long [ ] studentId; } Declares an array of type long

Student[ ] student; } Declares an array of Student Object

Step 2: Array initialization:

New keyword is used to initialize an array.

Syntax: <array-name>= new <type>(<size>);

Illustration: studentId=new long[5]; } Defines array with long type which stores 5 values.

student=new Student[3]; } Creates an array that can hold 3 student objects.

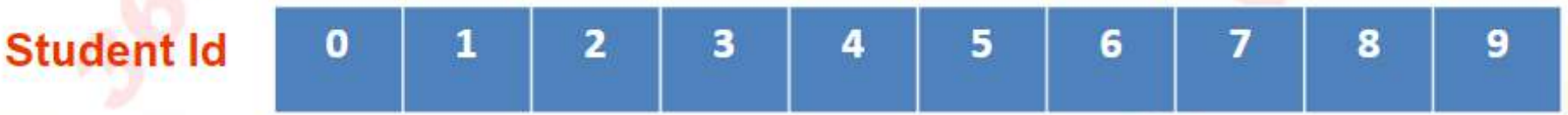
Single Dimensional Arrays

Click to Continue

Few salient points :

- The array length is determined when the array is created.
- The array length is ***fixed*** at the time of its creation.
- Each item of an array is called as ***element***, and each element is accessed using the ***index***.

The below diagram depicts an array of ten elements



Adding Elements

Adding values to a single dimensional array:

Data can be stored in an array based on the data type of the array.

Illustration:

```
student[0]=17;
student[1]=22;
student[2]=33;
```

This adds the student ids in the array

This sets the student name and the student id value in student objects.

```
student[0]=new Student("Jack",3000);
Student[1]=new Student("Ramesh",4000);
student[2]=new Student("Rajesh",10);
```

This adds student object

Alternate Way – Adding Elements **Click to Continue**

Alternate Way to initialize and assign values in one step,

```
long[] studentId = { 6, 3, 5, 8};
```

This declares an array of type long with values 6,3,5 & 8.

```
Student[] student={new Student("Ramesh",20),new
Student("Jack",40)};
```

This declares an student object array, stores two student objects

Length of the array is determined by the number of values stored in the array.

Accessing elements

Click to Continue

Retrieving values from an array?

Array elements are accessed using the index.The index position of the first element is 0, last element is position is array length -1.

Illustration:

long studentId=studentId[0];

Retrieves the first element in student array

Student student=student[3];

Retrieves the fourth element in student array

Finding Length:
Use **student.length;** to get the total number of items in the array.

Click to Continue

Program to print odd numbers between 0 & 1000.

1. Create a program "ArrayExample" add two methods,
    1. saveNumbers – Creates and stores a array with values from 0~1000
    2. printOddNumber – traverse through the array stored and print all the odd numbers.

    From the main method invoke both the methods.

Lend a Hand–Solution

Click to Continue

```
package com.wrapper.demo;

public class ArrayExample {

    int numbers[] = new int[1001];

    public void saveNumbers() {
        for (int i = 0; i <= 1000; i++) {
            numbers[i] = i;
        }
    }

    public void printOddNumbers() {
        for (int i = 0; i <= 1000; i = i + 1) {
            if (numbers[i] % 2!=0) {
                System.out.println("Odd number=" + numbers[i]);
            }
        }
    }

    public static void main(String[] args) {
        ArrayExample ex = new ArrayExample();
        ex.saveNumbers();
        ex.printOddNumbers();

    }
}
```

This method stores 1000 numbers in an array.

This method prints the odd numbers between 0 and 1000.

Invokes the methods.

Multi Dimensional Arrays Click to Continue

Multi Dimensional Arrays are ***array of arrays***.

The two dimensional array is a 2*2 matrix with rows and columns, each row labeled with an index of 0 to its maximum bound.

Syntax:

```
type  array-name = new type[rows][cols];
```

Illustration:

```
int studentMarks[ ][ ] = new int[2][3 ]; // Represents the marks of the student in row 2 and column3
```

A two dimensional array illustration

	0	1	2	3
0	22	24	26	23
1	34	54	38	43
2	22	33	35	36

Multi Dimensional Arrays Click to Continue

Multi Dimensional Arrays are ***array of arrays***.

The two dimensional array is a 2*2 matrix with rows and columns, each row labeled with an index of 0 to its maximum bou

Disadvantages of arrays:

- Fixed size.
- One type of data.

Illustration:

```
int studentMarks[ ][ ] = new int[2][3 ]; // Represents the marks of the student in row 2 and column3
```

A two dimensional array illustration

	0	1	2	3
0	22	24	26	23
1	34	54	38	43
2	22	33	35	36

Try It Out – 2D & 3D array

Click to Continue

1. Write a program which will initialize a 3X3 matrix and store the sum of the column index and row index in each matrix cell. The program then should iterate and display the values stored in a matrix format.

So for example 2X2 matrix output will be

0 1

1 2

```
public static void main(String[] args) {
    int number[][] = new int[3][3];
    // store the sum of i and j index
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            number[i][j] = i + j;
        }
    }
    // for loop which displays the values stored
    System.out.println("Matrix Display");
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            System.out.print(number[i][j]);
        }
        System.out.println("");
    }
    System.out.println("Matrix End");

}
```