CLASS: BCA V Sem

JAVA THEORY

Notes as per IKGPTU Syllabus

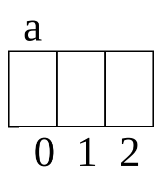
Name of Faculty: Mr. Satyam
Faculty of Computer Applications, SBS College. Ludhiana

Array: Array is the collection of similar data type. Array in java is a group of like-typed variables referred to by a common name.

In Java, an array is a data structure that allows you to store multiple values of the same data type under a single variable name.

- Arrays are stored in contiguous memory [consecutive memory locations].
- First element of array is numbered as zero, so that last element is one less than size of array.

Syntax of array
Datatype_arrayname[size of array],
Int a [3],



An array has two components: the type and the name. *Type* declares the element type of the array. The element type determines the data type of each element that comprises the array.

Name represents the name of array.

Processing Array Contents: Processing the contents of an array involves performing various operations on the elements stored in the array. These operations can include tasks like accessing specific elements, modifying elements, searching for values, sorting, filtering, and performing calculations.

Accessing Elements: In a Java, You can access individual elements of array by using their index.

int[] myArray = {1, 2, 3, 4, 5};

int firstElement = myArray[0]; // Access the first element (1)

Modifying Elements: To modify elements in a Java array, you assign new values to specific indices.

myArray[2] = 10; // Change the third element to 10

Searching for Values: To search for a specific value in an array, you can use a loop and compare elements to the target value.

```
int targetValue = 3;
boolean found = false;

for (int element : myArray) {
    if (element == targetValue) {
        found = true;
        break;
    }
}

if (found) {
    System.out.println("Value found");
}
```

Advantages of using arrays:

- Arrays allow random access to elements. This makes accessing elements by position faster.
- Arrays represent multiple data items of the same type using a single name.
- Simple and Easy to Use.
- Support for Multi-dimensional Arrays.

Disadvantages of using arrays:

- Fixed size.
- Difficulty with multidimensional array.
- Homogeneous data.

TYPES OF ARRAY IN JAVA

In Java, arrays are used to store collections of data of the same type. Java supports several types of arrays, including:

Single-Dimensional Array: A single-dimensional array is a collection of elements of the same data type arranged in a linear sequence. It is also known as a one-dimensional array. An Array having only one dimension is called one dimensional array.

Datatype_arrayname[size of array],
Int a [3],

Multidimensional Arrays: Java supports multidimensional arrays, which are essentially arrays of arrays. The most common multidimensional array is the two-dimensional array.

Two – Dimensional Array: An array having two dimensions is called two – dimensional array.
Data_typearryaname[dim1]]dim2],
Data_typearryaname[row]]col],

Three – Dimensional Array: An array having three dimensions is called three dimensional array.
Data_typearryaname[dim1]]dim2][dim3],

Q. Why array is useful in java?

Ans. Array is use full because of the following features:
Supports Multi- Dimensional Array.
Sorting and Searching.
Supports zero-Based Indexing.
Collection of homogeneous Elements.

String in Java

Strings are the type of objects that can store the character. A string acts the same as an array of characters in Java.
In Java, strings are represented using the String class, which is part of the Java Standard Library. In Java, the String class is a fundamental class that represents a sequence of characters. It's part of the Java Standard Library and provides a wide range of methods for working with strings. The String class is used extensively in Java for text manipulation and processing.
String firstName = "John";
String name = "Geeks";

Features of strings in java

Immutable: Strings in Java are immutable, which means once a string is created, its value cannot be changed.
String Concatenation: You can concatenate strings using the + operator or the concat() method.
String Length: You can determine the length (number of characters) of a string using the length() method.
Case Conversion: Strings can be converted to uppercase and lowercase using the toUpperCase() and toLowerCase() methods.
Comparison: Strings can be compared using the equals() method for content equality.

Substring: A substring is a contiguous sequence of characters within a larger string. In Java, you can extract a substring from a string using the substring method.

String concatenation: String concatenation is the process of combining (joining) two or more strings to create a single string. In Java, you can concatenate strings using various methods and operators. The most common approaches for string concatenation in Java include:

Using the + Operator: You can use the + operator to concatenate strings. When you use the + operator with strings, Java will create a new string by joining the contents of the operands. Here's an example:

```
String str1 = "Hello";
String str2 = "World";
String combined = str1 + " " + str2; // "Hello World"
```

Using the concat Method: The concat() method is available on the String class and can be used to concatenate two strings.

```
String str1 = "Hello";
String str2 = " World!";
String result = str1.concat(str2);
System.out.println(result); // "Hello World!"
```

String buffer: String Buffer is a class in Java that is used for creating and manipulating mutable (modifiable) strings.

Key characteristics

Mutable: You can change the content of a String Buffer.

Performance: It is efficient for string concatenation operations because it avoids creating new strings and copying content frequently, as is the case with regular string concatenation.

Interfaces in Java

An **Interface in Java** programming language is defined as an abstract type used to specify the behavior of a class. An interface in Java is a blueprint of a behavior. A Java interface contains static constants and abstract methods.

The interface in Java is *a* mechanism to achieve abstraction.

Uses of Interfaces in Java

- Since java does not support multiple inheritances in the case of class, by using an interface it can achieve multiple inheritances.

- Interfaces are used to implement abstraction.

Relationship between Class and Interface
A class can extend another class similar to this an interface can extend another interface. But only a class can extend to another interface, and vice-versa is not allowed.

Class	Interface
In class, you can represent variables and create an object.	In an interface, you can't represent variables and create an object.
A class can contain concrete(with implementation) methods	The interface cannot contain concrete(with implementation) methods
The access specifiers used with classes are private, protected, and public.	In Interface only one specifier is used- Public.

To implement an interface we use the keyword **implements.**

Class Bicycle implements Vehicle

Multiple Inheritance Using Interface
In Java, multiple inheritance is not supported for classes (i.e., a class cannot inherit from more than one class). However, you can achieve a form of multiple inheritance by using interfaces.

This is a way to share behavior from multiple sources without the complexities associated with multiple inheritance of classes.

How it works:

1. **Define Interfaces:** First, define one or more interfaces, each specifying a set of methods that participating classes must implement. Interfaces declare what methods must be present without providing their implementations. Here's an example with two interfaces:

```
    interface Interface1 {
     void method1();
    }


interface Interface2 {
    void method2();
}
```

2. **Implement the interfaces in the class:**

```
    public class MyClass implements Interface1, Interface2 {
        public void method1() {
            // implementation of method1
        }


    public void method2() {
        // implementation of method2
    }
}
```

3. Create an object of the class and call the interface methods

```
MyClass obj = new MyClass();
```

```
    obj.method1();
    obj.method2();
```

Packages in java

In Java, packages are a way to organize and structure your code into meaningful groups. Packages provide a hierarchical organization of classes and interfaces, making it easier to manage and maintain your codebase. Here's an overview of packages in Java:

1. **What is a Package?** A package in Java is a namespace that contains a group of related classes and interfaces. It helps avoid naming conflicts by providing a way to uniquely identify classes.
2. **Package Declaration**: At the beginning of a Java source file, you can declare the package to which the class belongs using the package keyword.

There are two types of packages in java:

1. User-defined Package (Create Your Own Package's)
2. Built-in packages are packages from the java application programming interface that are the packages from Java API for example such as, AWT, etc.

Creating a Package:

Choose a Package Name: Package names are typically in reverse domain name format to ensure uniqueness, such as com.example.myapp.

Organize Your Code: Place the Java classes you want to include in the package within a directory structure that reflects the package name.

Declare the Package: In each Java class file that you want to include in the package, you should declare the package at the top of the file using the package keyword.

```
package com.example.myapp;

public class MyClass1 {
   // Class code here
}
```

Accessing a Package:

Import Statement: You can use the import statement to include classes from other packages in your Java class.

Use the Fully Qualified Name: If you don't want to use the import statement, you can use the fully qualified name of the class, including the package name.

Advantages of using package in java

Organization and Structure: Packages provide a way to organize and structure your code. You can group related classes and interfaces together within a package, making it easier to manage and maintain your code.

Reusability: When you organize classes into packages, it becomes easier to reuse code in other projects.

Modularity: Packages encourage modularity, allowing you to break down your application into smaller, manageable pieces.

Static import in Java

Static import in Java is a feature that allows you to access static members (fields and methods) of a class without specifying the class name.

Advantage of static import:
If user wants to access any static member of class then less coding is required.