

Agenda

- Arrays class
- pass by value and reference
- Object/Field Initializer
- final
- static
- Singleton Design Pattern
- Hierarchy

Arrays

- it is a class in java.util package
- consists of helper methods for working on Array.
- eg
 - search
 - sort
 - equals
 - toString
- The array must be sorted (as by the sort() method) prior to making the search call.
- If it is not sorted, the results are undefined.

Method Arguments

- In Java, primitive values are passed by value and objects are passed by reference.
- Pass by reference stores address of the object. Changes done in called method are available in calling method.
- Pass by value -- Creates copy of the variable. Changes done in called method are not available in calling method.
- Pass by reference for value/primitive types can be simulated using array.

Object/Field Initializer

- In C++/Java Fields of the class are initialized using constructor
- In java, field can also be initialized using
 - 1. field initializer
 - 2. object initializer
 - 3. Constructor

final

- In Java, const is reserved word, but not used.
- Java has final keyword instead.
- It can be used for
 - variables

- fields
 - methods
 - class
- if variables and fields are made final, they cannot be modified after initialization.
- final fields of the class must be initialized using any of the following below
 - field initializer
 - object initializer
 - constructor
- final methods cannot be overridden, final class cannot be extended (we will see at the time of inheritance)

static Keyword

- In OOP, static means "shared" i.e. static members belong to the class (not object) and shared by all objects of the class.
- Static members are called as "class members"; whereas non-static members are called as "instance members".
- In Java, static keyword is used for
 - 1. static fields
 - 2. static methods
 - 3. static block
 - 4. static import
- Note that, static local variables cannot be created in Java.

1. static Fields

- Copies of non-static/instance fields are created one for each object.
- Single copy of the static/class field is created (in method area) and is shared by all objects of the class.
- Can be initialized by static field initializer or static block.
- Accessible in static as well as non-static methods of the class.
- Can be accessed by class name or object name outside the class (if not private). However, accessing via object name is misleading (avoid it).
- eg :
 - Integer.SIZE
- Similar to field initializer, static fields can be initialized at declaration.

2. Static methods

- These Methods can be called from outside the class (if not private) using class name or object name. However, accessing via object name is misleading (avoid it).
- When we need to call a method without creating object, then make such methods as static.
- Since static methods are designed to be called on class name, they do not have "this" reference. Hence, they cannot access non-static members in the static method (directly), However, we can access them on an object reference if created inside them.

- eg:
 - `Integer.valueOf(10);`
 - Factory Methods -> to create object of the class

static Field Initializer

- Similar to field initializer, static fields can be initialized at declaration.

```
static double roi = 5000.0;  
// static final field -- constant  
static final double PI = 3.142;
```

static Initializer Block

- Like Object/Instance initializer block, a class can have any number of static initialization blocks, and they can appear anywhere in the class body.
- Static initialization blocks are executed in the order their declaration in the class.
- A static block is executed only once when a class is loaded in JVM.

static import

- To access static members of a class in the same class, the "ClassName." is optional.
- To access static members of another class, the "ClassName." is mandatory.
- If need to access static members of other class frequently, use "import static" so that we can access static members of other class directly (without ClassName.).

Singleton Design Pattern

- Singleton is a design pattern.
- Singleton class is a class whose single object is created throughout the application.
- To make a singleton class in Java
- step 1: Write a class with desired fields and methods.
- step 2: Make constructor(s) private.
- step 3: Add a private static field to hold instance of the class.
- step 4: Initialize the field to single object using static field initializer or static block.
- step 5: Add a public static method to return the object.