Agenda

- Array Continued
- Variable Arity/Argument Method
- Initializers
- final Keyword
- static Keyword
- BuzzWords
- ~~Singleton Design Pattern~~

Variable Arity/Argument Method

- It is a method which can take variable no of arguments.
- We can also pass array to this method.
- If we want to pass different types of variables to this arity method then we can use the object as the type.

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 overriden, 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 memebers".
- 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 cretae 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 mandetory.
- If need to access static members of other class frequently, use "import static" so that we can access static members of other class direcly (without ClassName.).

Java BuzzWords

- 1. Simple
    - Simple for Professional Programmers if aware about OOP.
    - It removed the complicated fetaures like pointers and rarely used features like operator overloading from c++
    - It was simple till java 1.4
    - the new features added made it powerful (but also complex)
- 2. Object Oriented
    - Java is a object-oriented programming language.
    - It supports all the pillars of OOP
- 3. Distributed
    - Java is designed to create distributed applications on networks.
    - Java applications can access remote objects on the Internet as easily as they can do in the local system.
    - Java enables multiple programmers at multiple remote locations to collaborate and work together on a single project.
- 4. Compiled and Interpreted
    - Usually, a computer language is either compiled or Interpreted.
    - Java combines both this approach and makes it a two-stage system.
    - Compiled: Java enables the creation of cross-platform programs by compiling them into an intermediate representation called Java Bytecode.
    - Interpreted: Bytecode is then interpreted, which generates machine code that can be directly executed by the machine/CPU.
- 5. Robust
    - It provides many features that make the program execute reliably in a variety of environments.
    - Java is a strictly typed language. It checks code both at compile time and runtime.
    - Java takes care of all memory management problems with garbage collection.
- 6. Secure
    - Java achieves this protection by confining a Java program to the Java execution environment and not allowing it to access other parts of the computer
- 7. Architecture Neutral
    - Java language and Java Virtual Machine helped in achieving the goal of WORA - Write Once Run Anywhere.
    - Java byte code is interpreted by JIT and convert into CPU machine code/native code.
    - So Java byte code can execute on any CPU architecture (on which JVM is available)
- 8. Portable

- As java is Architecture Neutral it is portable.
- Java is portable because of the Java Virtual Machine (JVM).

- 9. High Performance
  - Java performance is high because of the use of bytecode.
  - The bytecode was used so that it can be efficiently translated into native machine code by JIT compiler (in JVM).
- 10. Multithreaded
  - Multithreaded Programs handled multiple tasks simultaneously (within a process)
  - Java supports multi-process/thread communication and synchronization.
  - When Java application executes 2 threads are started
    - 1. main thread
    - 2. garbage collector thread.
- 11. Dynamic
  - Java is capable of linking in new class libraries, methods, and objects.
  - Java classes has run-time type information that is used to verify and resolve accesses to objects/members at runtime.