Java Basics

Objective

- Uniqueness of Java
- Evolution of Java
- Features of Java
- JDK,JRE,JVM
- Stack & Heap
- Garbage collection
- Sample Application

2

Session Objective

Java Basics

- Comments, Variables & Package
- Data types – Primitive type
- Access specifier and modifiers
- Control structure
- Abstract Data type
    - Array
    - Enum

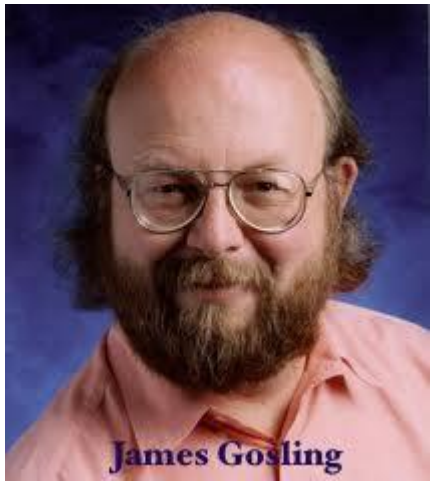
Java Introduction

What is Java?

- Java is a high-level programming language developed by Sun Microsystems and released in 1995.
- Java is a high level, robust, object-oriented and secure programming language.

History of Java

- The initial name was Oak but it was renamed to Java in 1995 as OAK was a registered trademark of another Tech company.
- James Gosling, Mike Sheridan, and Patrick Naughton initiated the Java language project in June 1991.
- The small team of sun engineers called Green Team.



Evolution of Java

Contd…

JAVA 5	JAVA 7
• Generics • Annotations • Autoboxing and autounboxing • Enumerations • For-each Loop • Varargs • Static Import • Formatted I/O • Concurrency utilities	• Now **String** can be used to control Switch statement. • Multi Catch Exception • *try-with-resource* statement • Binary Integer Literals • *Underscore* in numeric literals, etc.

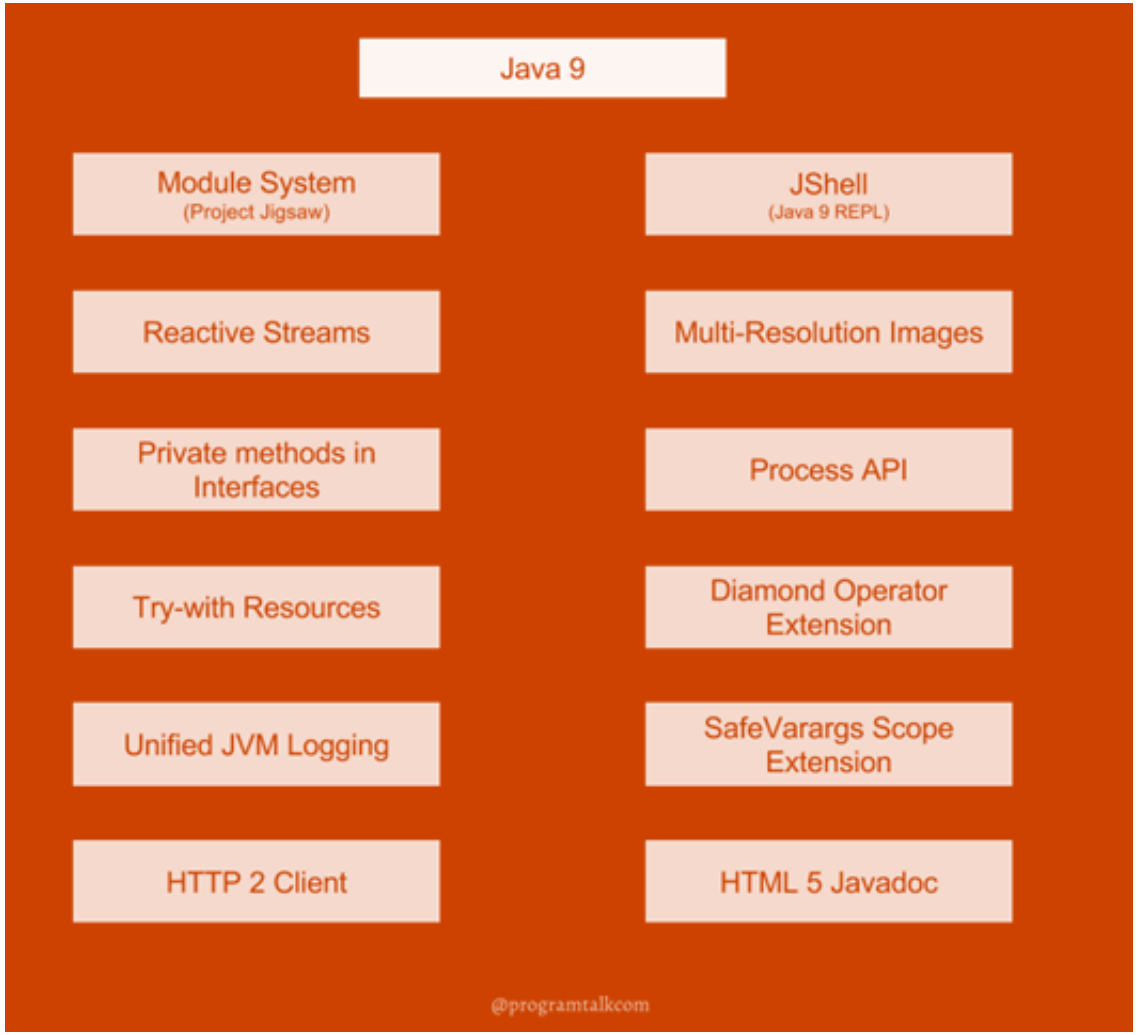
Evolution of Java

Contd…

JAVA 8

- Lambda Expressions
- New Collection Package java.util.stream to provide Stream API.
- Enhanced Security
- Nashorn Javascript Engine included
- Parallel Array Sorting
- The JDBC-ODBC Bridge has been removed etc.

Evolution of Java



Java Platforms

Oracle provides various Java editions to develop and run Java programs:

- **Java Standard Edition.** It is used for developing desktop and console based application
- **Java Enterprise Edition.** It is a platform primarily used for building server-side applications
- **Java Micro Edition.** It enables to build Java application for micro-devices, which include handheld devices, such as cell phones and PDAs

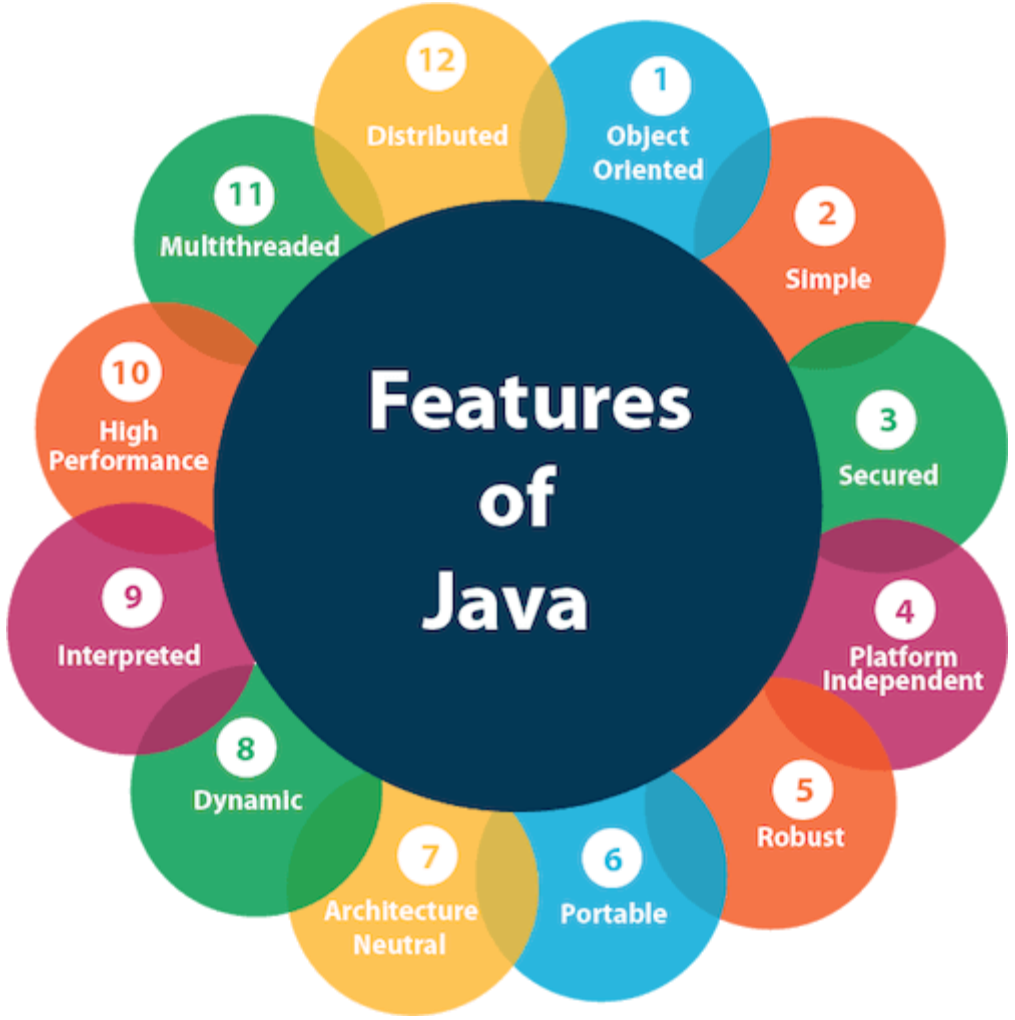
Application of Java

Following are the applications in Java

1. Desktop Applications eg: **Anti-Virus** etc..
2. Web Applications eg: **Linkedin.com, Snapdeal.com** etc..
3. Mobile Operating System eg: **Android**
4. Robotics and games etc.

Java Features

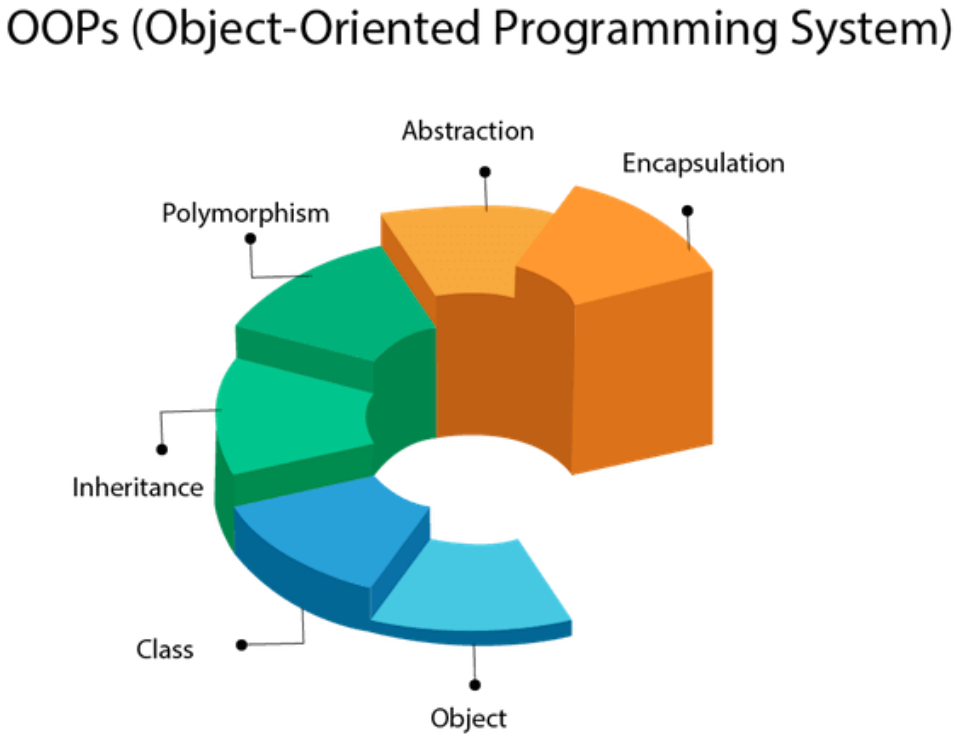
Features of Java Contd…



Features of Java

1. Object-oriented

- Java is an object-oriented programming language.
- Everything in Java is an object.
- Object-oriented programming (OOPs) is a methodology that simplifies software development and maintenance by providing some rules.



Features of Java

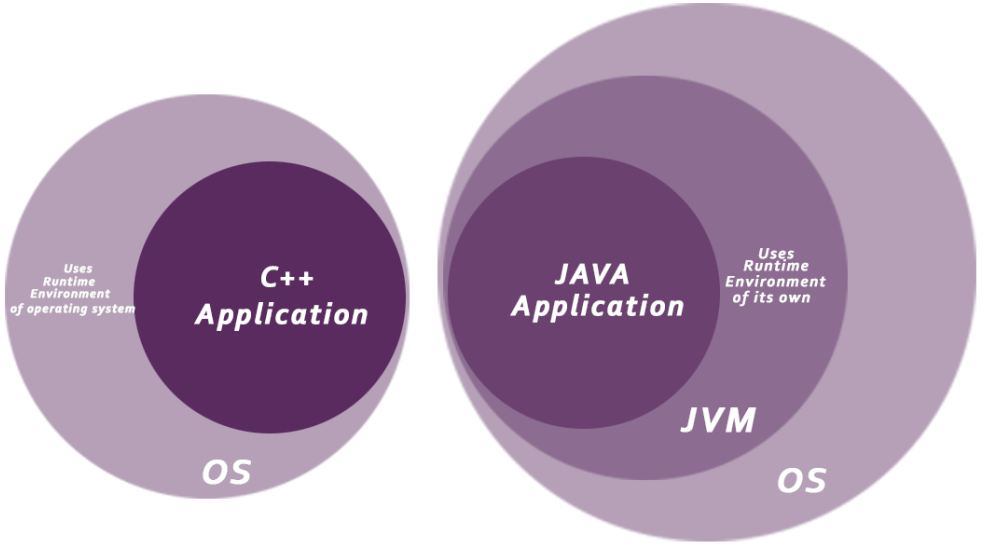
2. Simple

Java is very easy to learn, and its syntax is simple, clean and easy to understand.

3. Secured

Java is best known for its security. With Java, we can develop virus-free systems. Java is secured because:

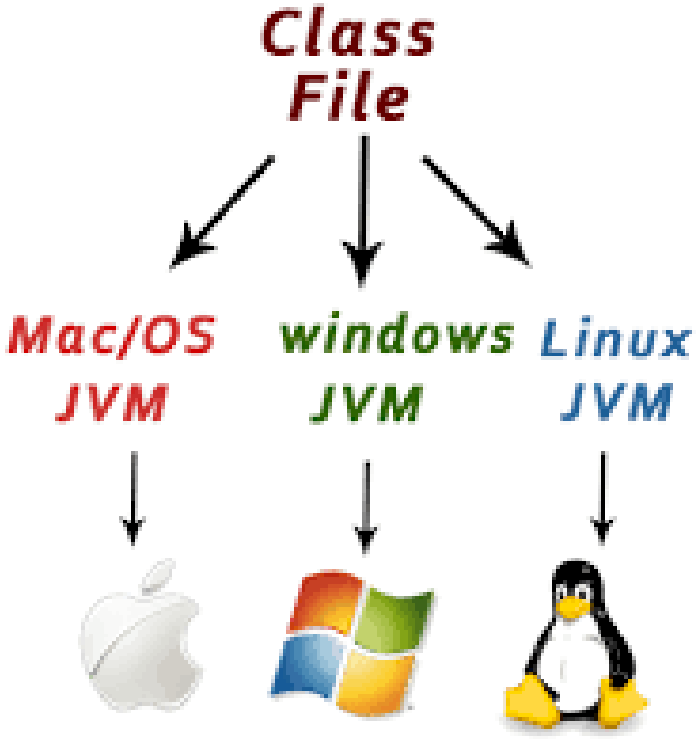
- No explicit pointer
- Java Programs run inside a virtual machine sandbox



Features of Java

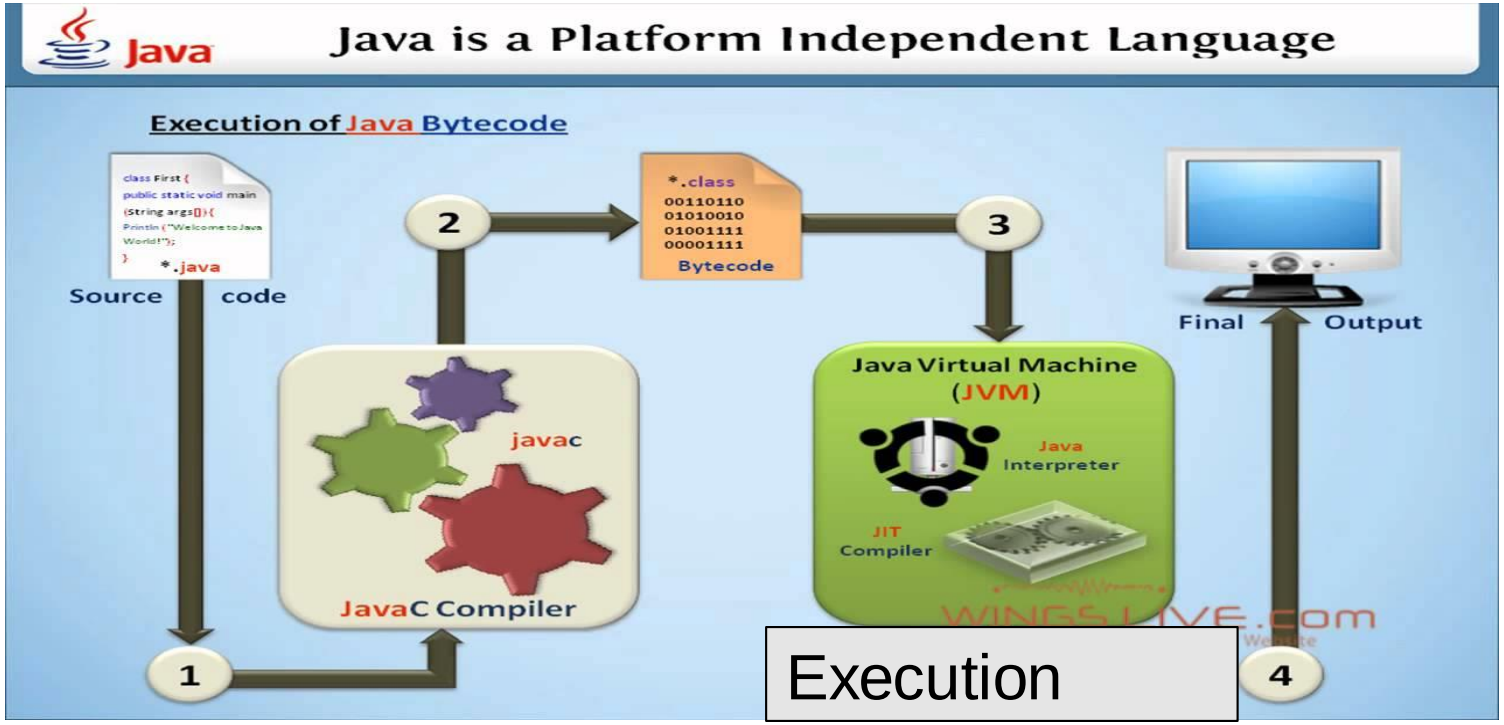
4. Platform Independent

Java is platform independent because it is different from other languages like C, C++, etc. which are compiled into platform specific machines while Java is a write once, run anywhere language



Features of Java

Java code can be run on multiple platforms, for example, Windows, Linux, Sun Solaris, Mac/OS, etc. Java code is compiled by the compiler and converted into bytecode. This bytecode is a platform-independent code because it can be run on multiple platforms



Features of Java

5. Robust

- Robust simply means strong.
- It uses strong memory management.
- There is a lack of pointers that avoids security problems.

6. Portable

Java is portable because it facilitates to carry the Java bytecode to any platform. It doesn't require any implementation.

Features of Java

7. Architecture-neutral

Java is architecture neutral because there are no implementation dependent features, for example, the size of primitive types is fixed.

8. Dynamic

Java is a dynamic language. It supports dynamic loading of classes which means classes are loaded on demand. It also supports functions from its native languages, i.e., C and C++.

Features of Java

9. Interpreted

Java is a compiled programming language, but rather than compile straight to executable machine code, it compiles to an intermediate binary form called JVM byte code. The byte code is then compiled and/or interpreted to run the program.

10. High-performance

Java is faster than other traditional interpreted programming languages because Java bytecode is "close" to native code.

Features of Java

11. Distributed

Java is distributed because it facilitates users to create distributed applications in Java. RMI and EJB are used for creating distributed applications

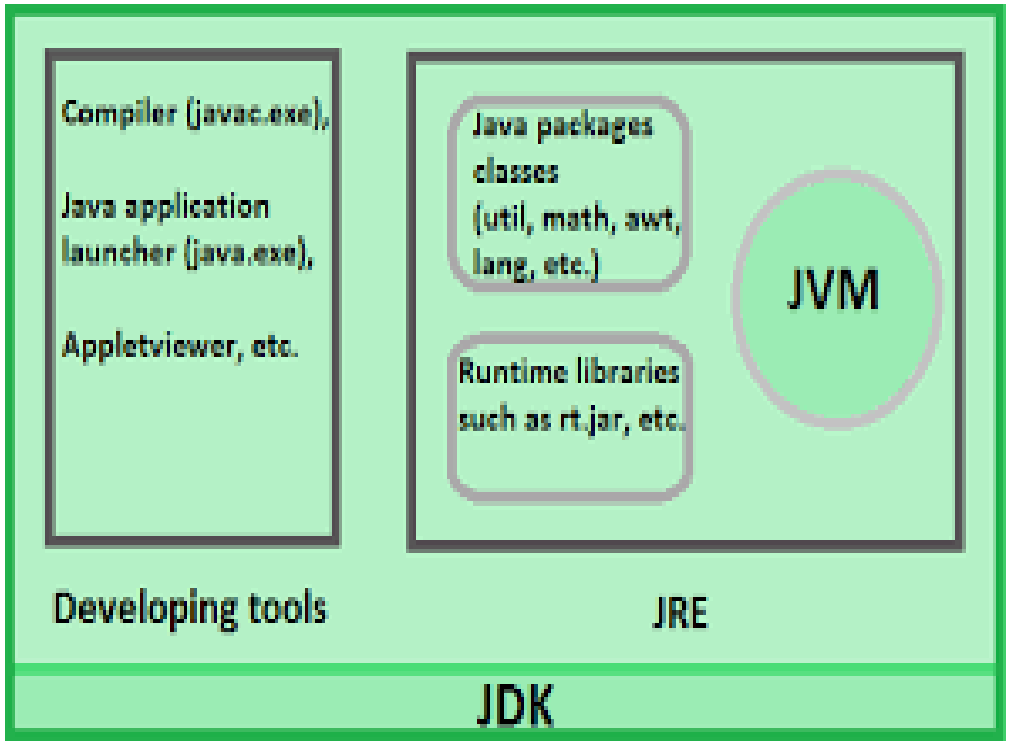
12. Multi-threaded

- ✓A thread is like a separate program, executing concurrently.
- ✓The main advantage of multi-threading is that it doesn't occupy memory for each thread.
- ✓It shares a common memory area.
- ✓Threads are important for multi-media, Web applications, etc.

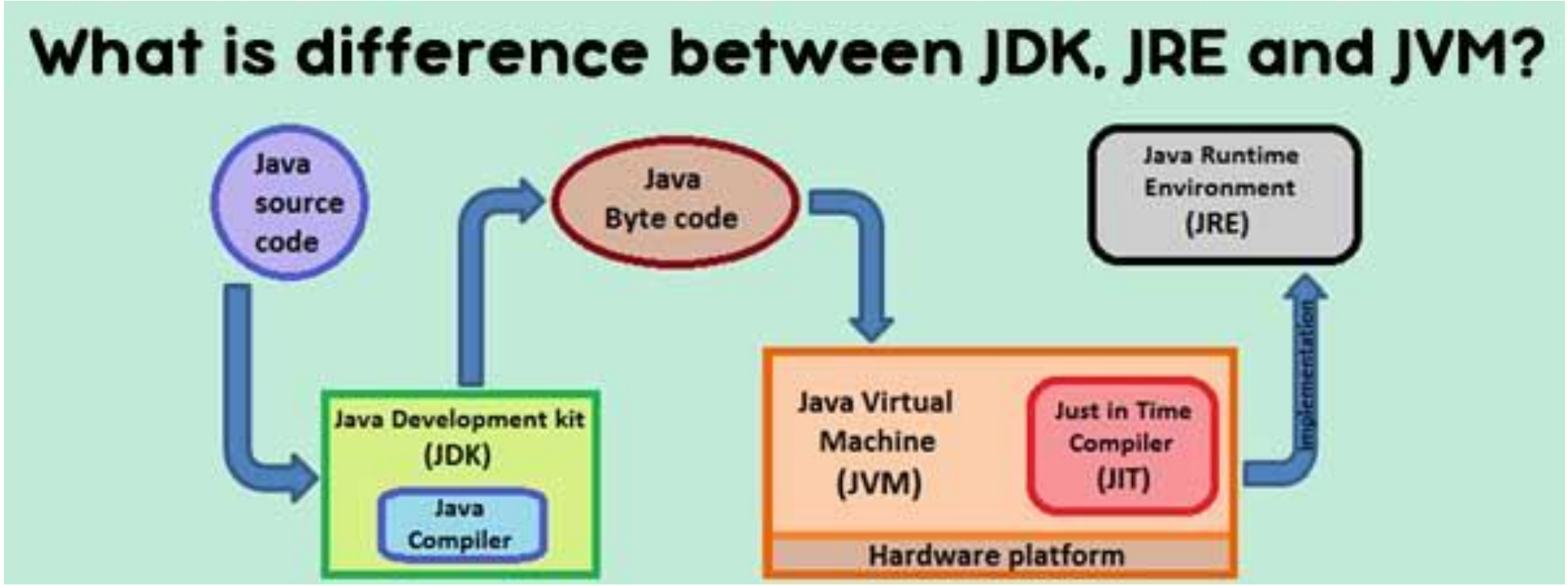
JRE, JDK, JVM

JDK, JRE, JVM

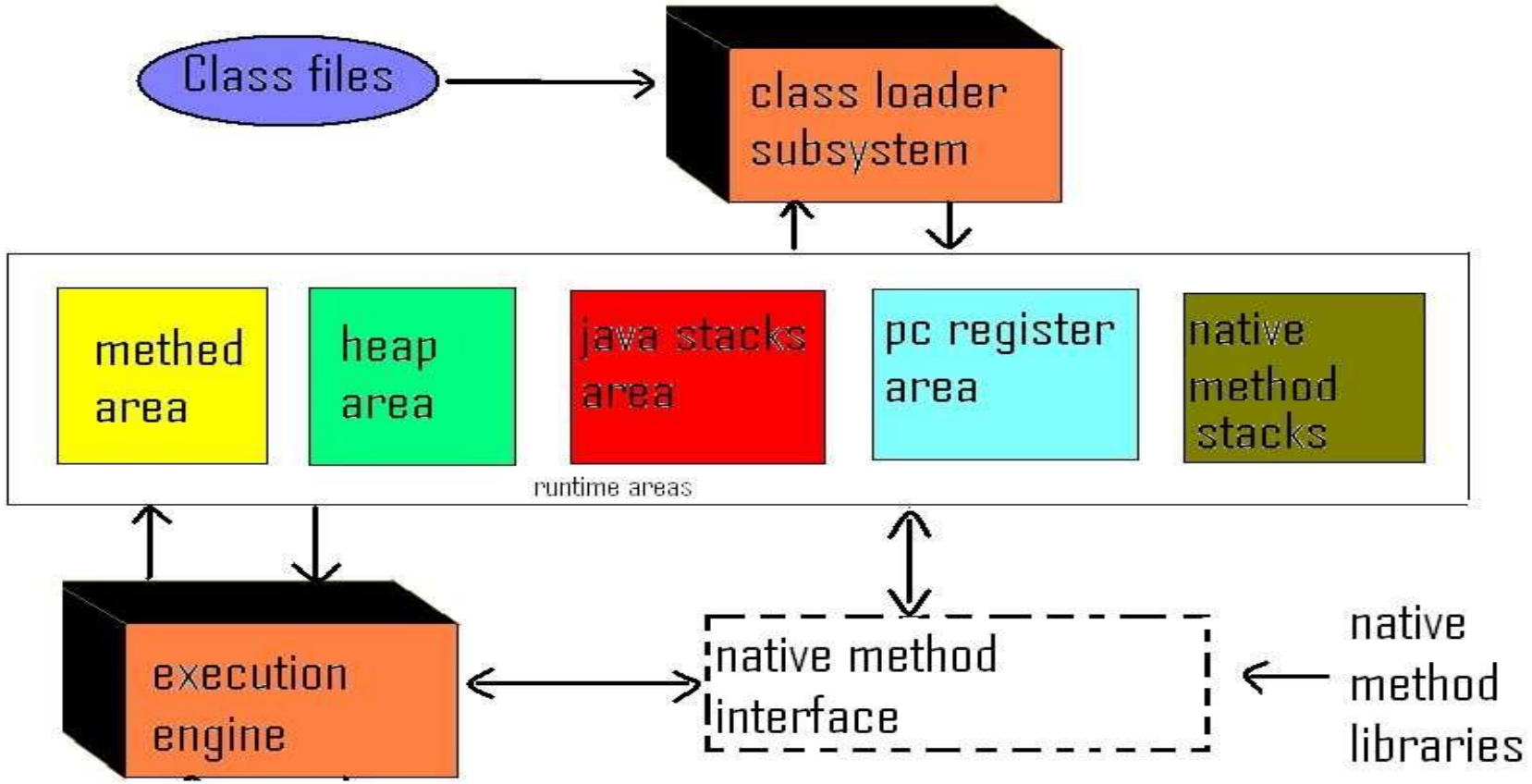
JDK	JRE
1)JDK is a bundle of software that is used to develop Java based applications . 2)Java Development Kit is needed for developing java applications. 3)JDK needs more disk space as it contains the JRE along with various development tools.	1)JRE is an implementation of virtual Machine which actually executes java programs. 2)JAVA Runtime Environment is a plug-in needed for running Java 3)JRE is smaller then JDK so it needs less space.



JDK, JRE, JVM



JVM



JVM ARCHITECTURE

JVM

- A Java virtual machine (JVM), an implementation of the Java Virtual Machine Specification, interprets compiled Java binary code (called bytecode) for a computer's processor
- A Virtual Machine is a software implementation of a physical machine. Java was developed with the concept of WORA (Write Once Run Anywhere), which runs on a VM.
- The compiler compiles the Java file into a Java .class file, then that .class file is input into the JVM, which loads and executes the class file.

JVM

- The JVM performs the following main tasks:
  - Loads code
  - Verifies code
  - Executes code

- **Loading** : The Class loader reads the .class file, generate the corresponding binary data and save it in method area.

- **Linking** : It ensures the correctness of .class file i.e. it check whether this file is properly formatted and generated by valid compiler or not.

JVM

- The **bytecode**, which is assigned to the Runtime Data Area, will be executed by the Execution Engine. The Execution Engine reads the bytecode and executes it piece by piece.

- **Interpreter** – The interpreter interprets the bytecode faster but executes slowly. The disadvantage of the interpreter is that when one method is called multiple times, every time a new interpretation is required.

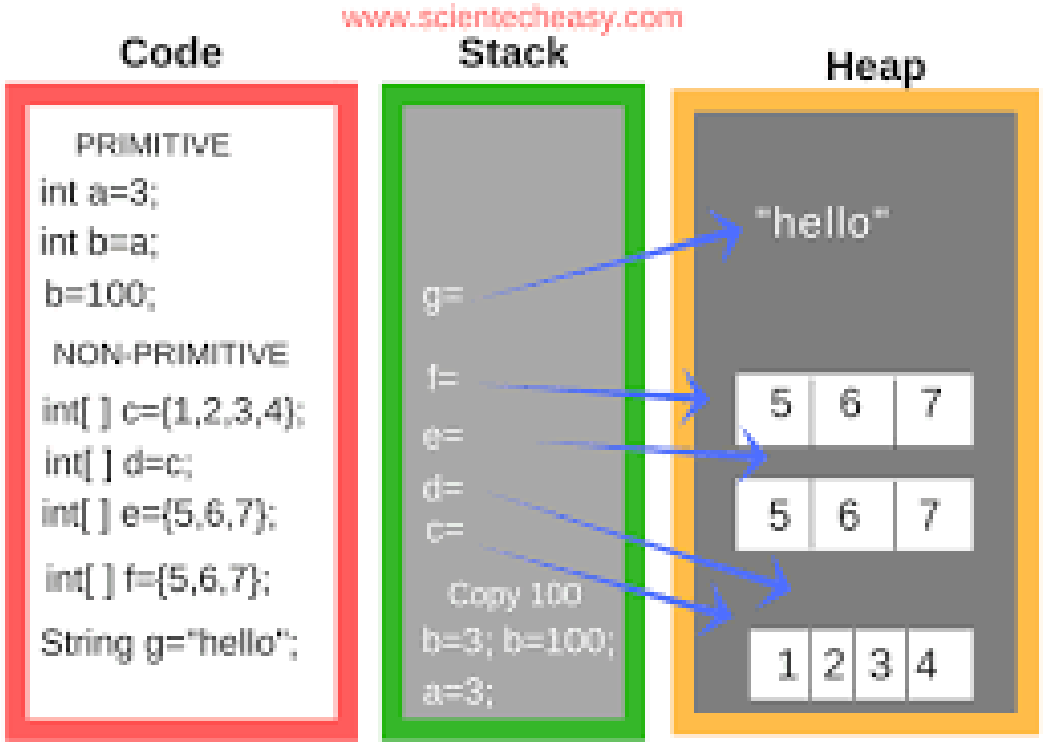
- **JIT Compiler** – The JIT Compiler neutralizes the disadvantage of the interpreter. The Execution Engine will be using the help of the interpreter in converting byte code, but when it finds repeated code it uses the JIT compiler, which compiles the entire bytecode and changes it to native code. This native code will be used directly for repeated method calls, which improve the performance of the system.

Stack and Heap

Stack Memory

- Stack in java is a section of memory which contains methods, local variables, and reference variables.
- Size of stack memory is lesser than the size of heap memory in Java.
- Stack memory is always referenced in LIFO (Last-In-First-Out) order. Whenever a method is invoked, a new block is created in the stack memory for the method to hold local primitive values and reference to other objects in the method.

Stack and Heap



Stack and Heap

Heap Memory

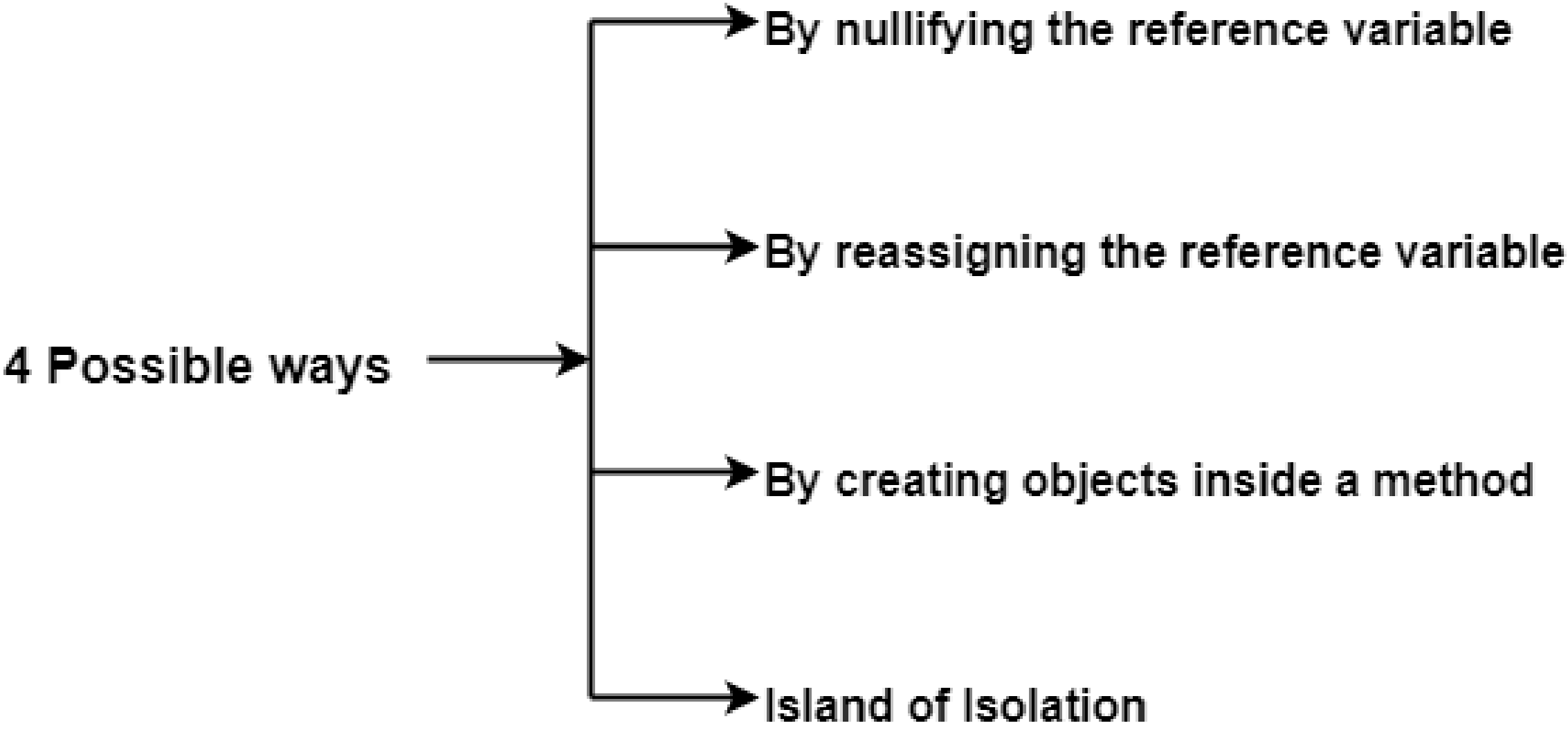
- Heap is a section of memory which contains Objects and may also contain reference variables.
- Instance variables are created in the heap.
- Garbage Collection runs on the heap memory to free the memory used by objects that doesn't have any reference.
- Object created in the heap space has global access and can be referenced from anywhere of the application.

Garbage Collector

- Collects and removes unreferenced objects.
- Garbage Collection can be triggered by calling System.gc(), but the execution is not guaranteed.
- Garbage collector is best example of Daemon thread as it is always running in background.
- Main objective of Garbage Collector is to free heap memory by destroying unreachable objects.

  Island of Isolation

Garbage Collector



Garbage Collector

Finalization

- Just before destroying an object, Garbage Collector calls finalize() method on the object to perform cleanup activities.
- Once finalize() method completes, Garbage Collector destroys that object.
- Based on our requirement, we can override finalize() method for perform our cleanup activities like closing connection from database.

Creating an Application

- Create a file named App.java

```
public class App{
public static void main(String[] args) {
System.out.println("Java World");
}
}
```


Creating an Application Contd…

- Line 1:
  - public class App
  - Create a new Java class named App
  - The public access specifier indicates that this class is available anywhere in a program that make use of it.

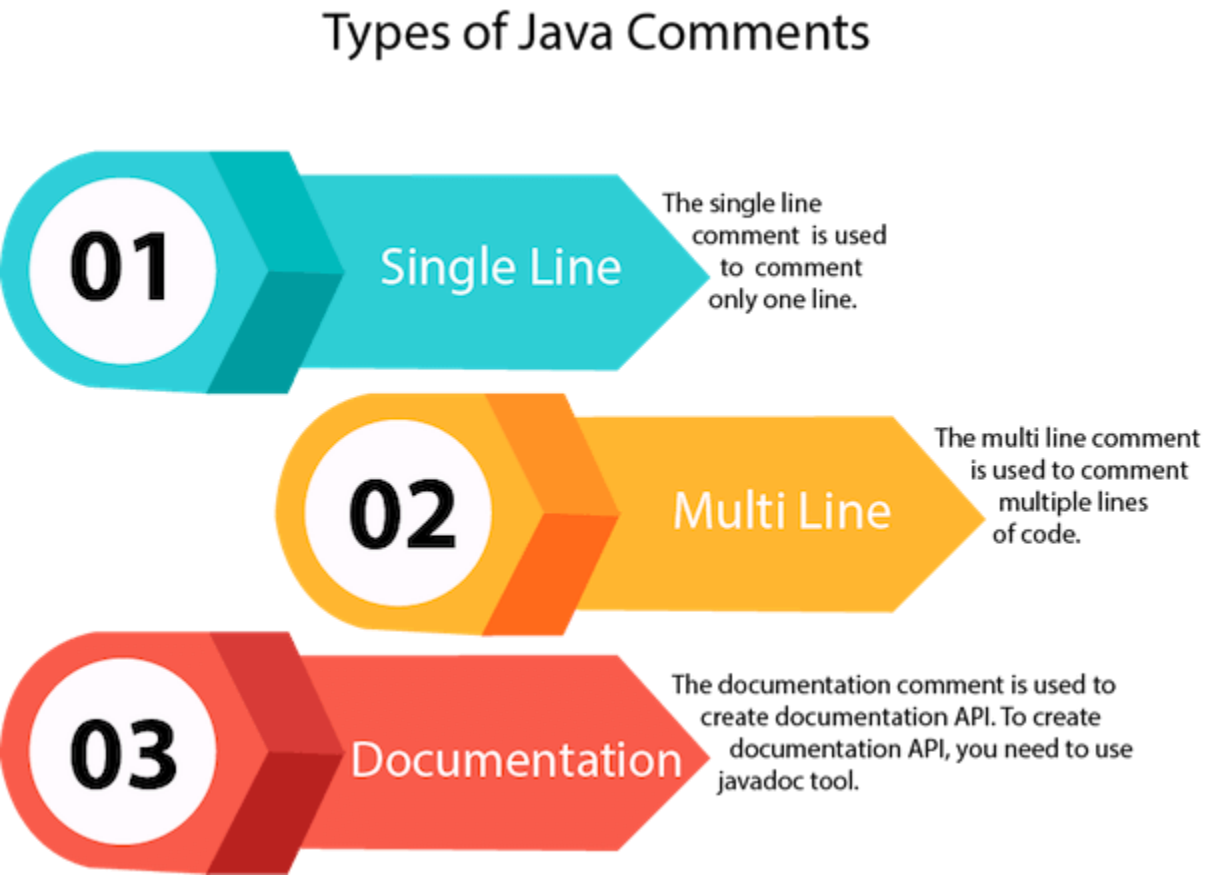
- Line 2:
  - public static void main(String[] args)
  - The main method must be declared with public access specifier
  - It is declared as static, which means main is a class method and not object method
  - It should not return any value on execution, so it is declared as void
  - String[] args is used to place an argument list in the parentheses of the main method

Creating an Application

- Line 3:
  - System.out.println("Java World");
  - The java.lang package's System class includes a field called out, and this field in turn has a method named println, which does the actual displaying of text.

Java Basics

Comments



Single line:

```
//Single line comment
```

Multi line comment:

```
/*The multi line comment is used to comment multiple lines of code.
*/
```

Documentation:

```
/**used when writing code for a project/software package, since it helps to generate a documentation page for reference
/
```

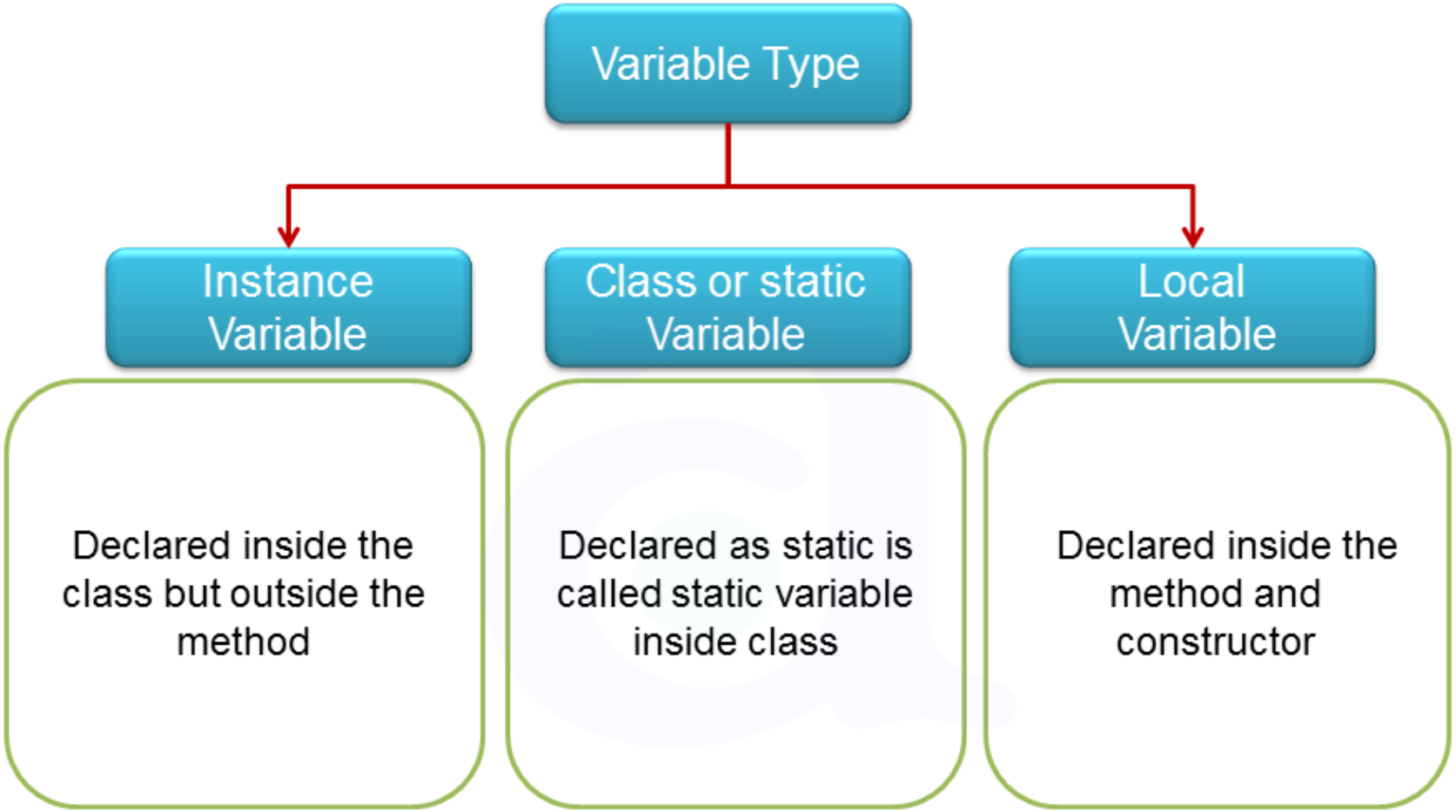

Package

- A Package is a collection of related classes. It helps to organize the classes into a folder structure and make it easy to locate and use them.

Syntax:

- Package MyNewPackage;
- Default package in Java is Lang
- To call a class from the package
- Import java.util.ArrayList;

Variables -Contd…



Variables -Contd…

```
public class Employee{

  //class or static variable
  private static int empId=31410;

  //instance variable
  private String empName="Vimala";
  private float empSalary;

 public float empDetails(float bonus){
  //local variable
  float totalAmt;
  totalAmt=empSalary+bonus;
  return totalAmt;
 }
```


Coding Standards

Identifier Naming and Capitalization

- Use descriptive names for all variables, function names, constants, and other identifiers.
- Use single letter identifiers only for the counter in loops.
- Variable names start with a lower case letter.
- Multi-word identifiers are internally capitalized.

Coding Standards

- Do not use hyphens or underscores to separate multi-word identifiers
- Example

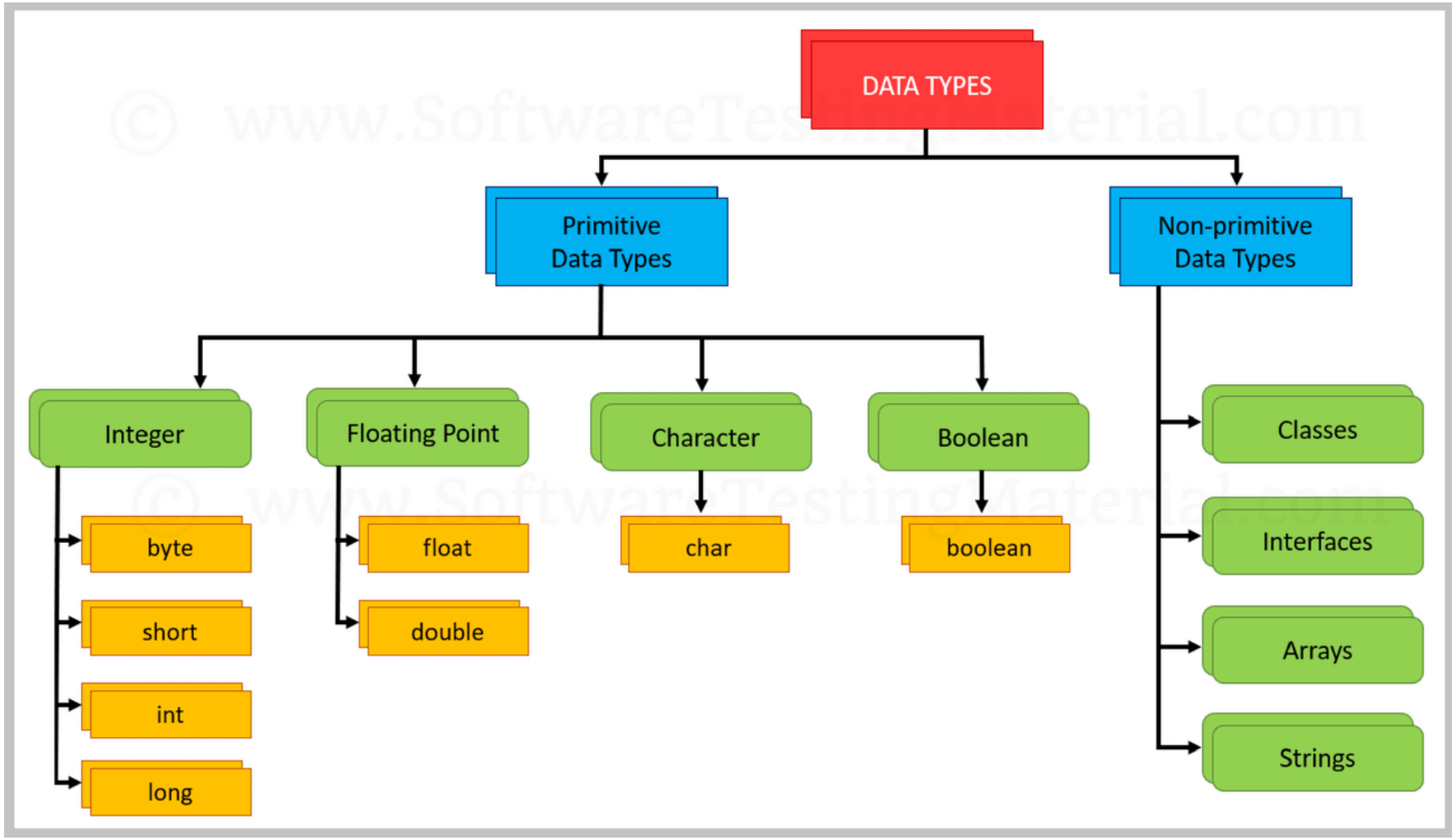
```
float sumOfSquares = 0;
//Use UPPER_CASE for constants (final variables)
final int DAYS_IN_YEAR = 365;
```

Note:

✓ Variables include parameters, local variables, and data fields.

✓ Use UPPER_CASE for constants - final variables.

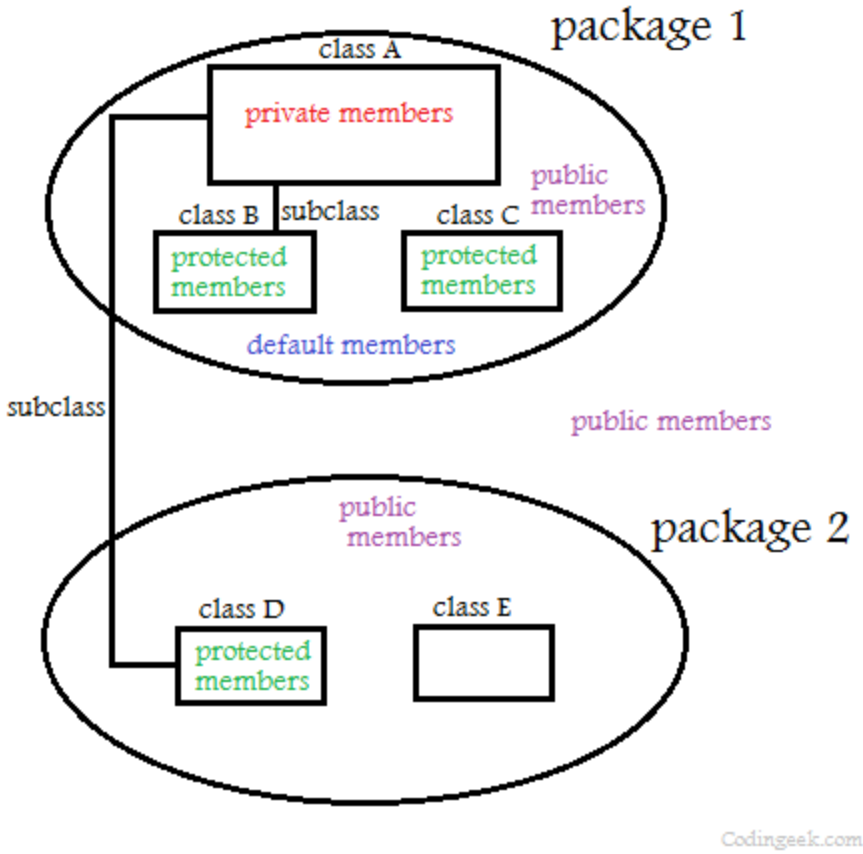
Data Type


DATA TYPES
Primitive Data Types
Non-primitive Data Types
Integer
Floating Point
Character
Boolean
byte
short
int
long
float
double
char
boolean
Classes
Interfaces
Arrays
Strings

Primitive Data Type

	Size	Default Value	Type of Value Stored
byte	1 byte	0	Integral
short	2 byte	0	Integral
int	4 byte	0	Integral
long	8 byte	0L	Integral
char	2 byte	'\u0000'	Character
float	4 byte	0.0f	Decimal
double	8 byte	0.0d	Decimal
boolean	1 bit (till JDK 1.3 it uses 1 byte)	false	True or False

Access Specifier

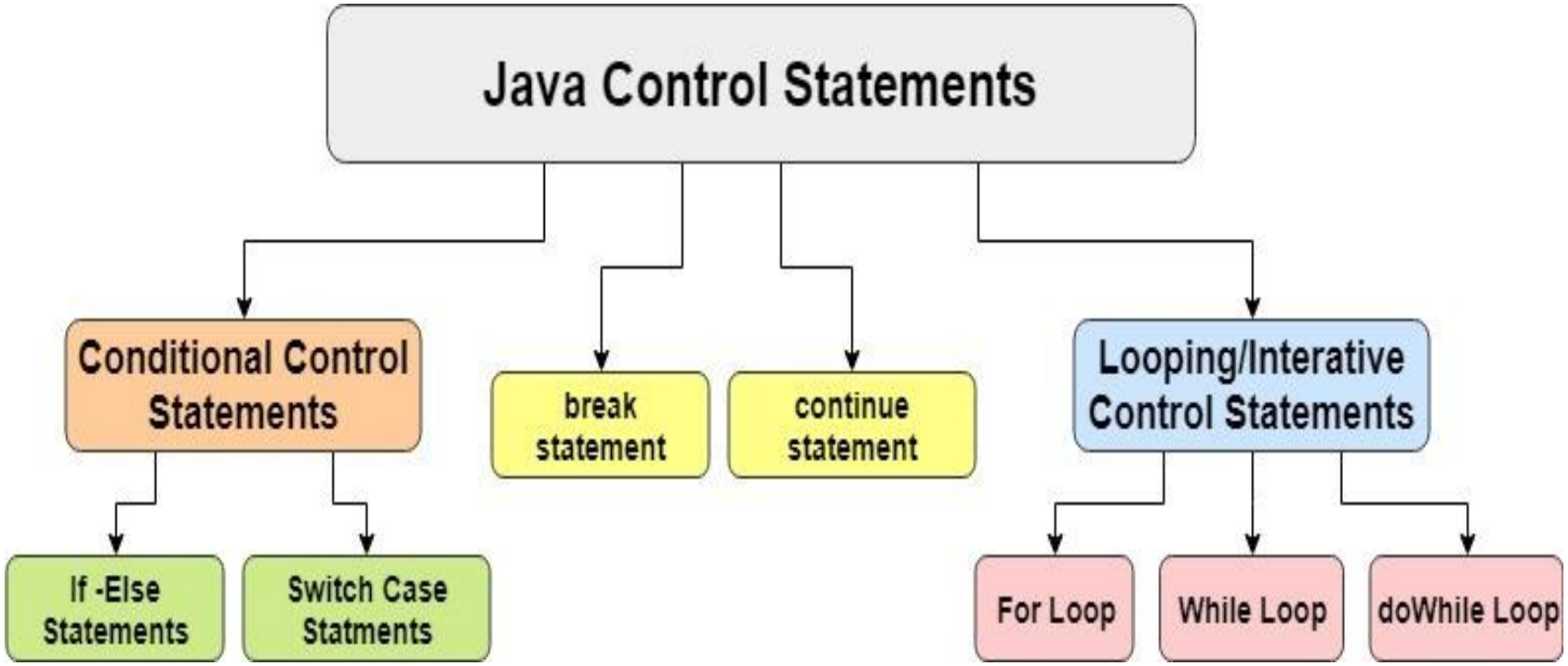


Access Specifier

Access Modifiers

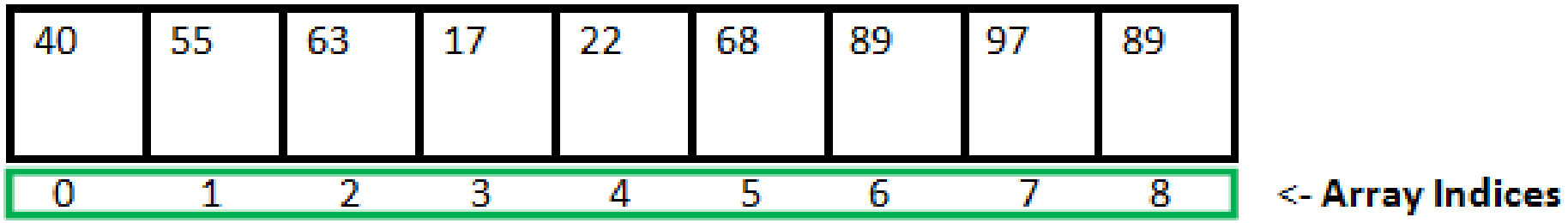
Modifier	Class	Package	Subclass	Global
Public	✓	✓	✓	✓
Protected	✓	✓	✓	✗
Default	✓	✓	✗	✗
Private	✓	✗	✗	✗

Control Statements



Abstract Data type - Array

Array can contains primitives data types as well as objects of a class depending on the definition of array. In case of primitives data types, the actual values are stored in contiguous memory locations.



Array Length = 9
First Index = 0
Last Index = 8

Array Contd… -

Instantiating an Array in Java

- When an array is declared, only a reference of array is created.

- The general form of one-dimensional arrays appears as follows:

var-name = new type [size];

```
int intArray[];    //declaring array
intArray = new int[20];  // allocating memory to array
```


Array

- When the size of the array and variables of array are already known, array literals can be used.
- int[] intArray = new int[]{ 1,2,3,4,5,6,7,8,9,10 };

Array

Advantages

- **Code Optimization**: It makes the code optimized, we can retrieve or sort the data efficiently.
- **Random access**: We can get any data located at an index position.

Disadvantages

- **Size Limit:** We can store only the fixed size of elements in the array. It doesn't grow its size at runtime. To solve this problem, collection framework is used in Java which grows automatically.

Array

For-each Loop

- The Java for-each loop prints the array elements one by one.
- It holds an array element in a variable, then executes the body of the loop.
- The syntax of the for-each loop is given below:

```
for(data_type variable:array){
//body of the loop
}
```


Enum

- A Java Enum is a special Java type used to define collections of constants.

```
public enum Level {
    HIGH,
    MEDIUM,
    LOW
}
Level level = Level.HIGH;
```


Enum

- **Enum toString()**

  String levelText = Level.HIGH.toString();

- **Enum valueOf()**

  Level level = Level.valueOf("HIGH");

- **Enum Printing**

  System.out.println(Level.HIGH);

Array

Example for Array:

Function returning array,

```
class ArrayDemo{
    public int[] fun(){
        int pid[]=new int[]{11,22,33,44};
        return pid;
    }
    public static void main(String[] args) {
        ArrayDemo dem=new ArrayDemo();
        int id[]=dem.fun();
        for(int e:id){
            System.out.println(e);
        }
        }
}
```


Effective Coding Standards

Coding standards related to formatting

- Not more than one statement per line.
- Line length should not exceed 80 or 100 characters.
- One character variable names should only be used in loops or for temporary variables.
- Avoid the use of the default package.

Innovative Services

Passionate Employees

Delighted Customers