# CST281
# Object Oriented Programming

## MODULE 1-PART 2-JAVA

## Syllabus

- Introduction:
  - Approaches to Software Design, Functional Oriented Design, Object Oriented Design, Case Study of Automated Fire Alarm System.
- Object Modeling Using UML
  - Basic Object Oriented concepts, UML (Unified Modeling Language) diagrams, Use case model, Class diagram, Interaction diagram, Activity diagram, State chart diagram.
- Introduction to Java -
  - Java programming Environment and Runtime Environment, Development Platforms Standard, Enterprise Java Virtual Machine (JVM), Java compiler, Byte code, Java applet, Java Buzzwords, Java program structure, Comments, Garbage Collection, Lexical Issues.


# INTRODUCTION TO JAVA



## JAVA

- Java is an Object Oriented programming language which gives a clear structure to programs and allows code to be reused, lowering development costs.
- Developed by Sun Micro System of USA in 1991.
- Development team members are James Gosling, Patrick Naughton, Chris Warth, Ed Frank, and Mike Sheridan
- First name of Java is “Oak,” but was renamed “Java” in 1995.
- Java derives much of its character from C and C++.
- Java Changed the Internet by simplifying web programming.
- Java innovated a new type of networked program called the applet.

- It is used for :
  - Mobile applications (specially Android apps)
  - Desktop applications
  - Web applications
  - Web servers and application servers
  - Gaming applications
  - Cloud based applications
  - Database connection

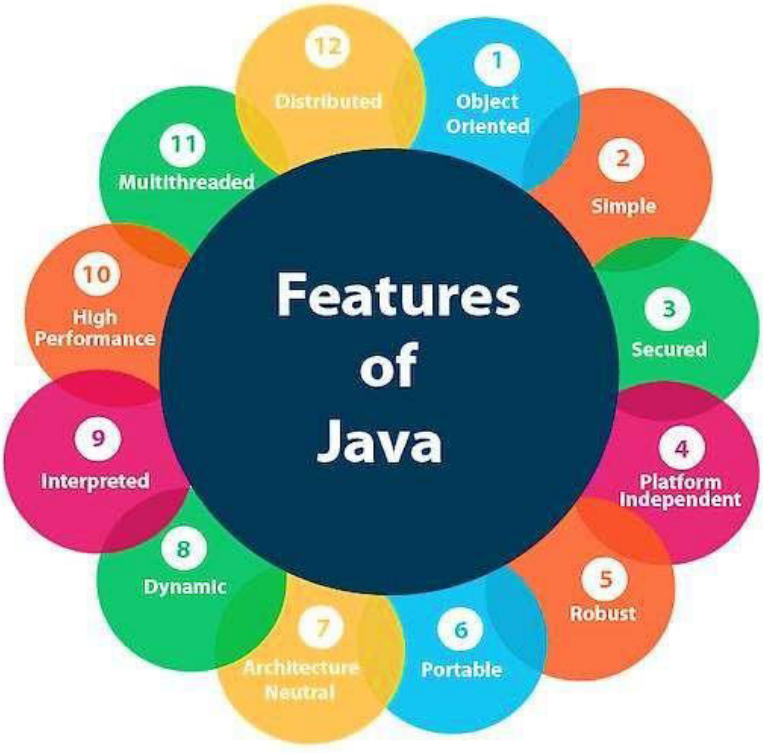


## Why use Java ?

- Most popular programming language.
- Easy to learn and simple to use.
- Open source and free.
- Secure, fast and powerful.
- Huge community support (tens of millions of developers)
- Works on different platforms (Windows, Mac, Linux, etc.)

The target of Java is to write a program once and then run this program on multiple operating systems.






JAVA BUZZWORDS

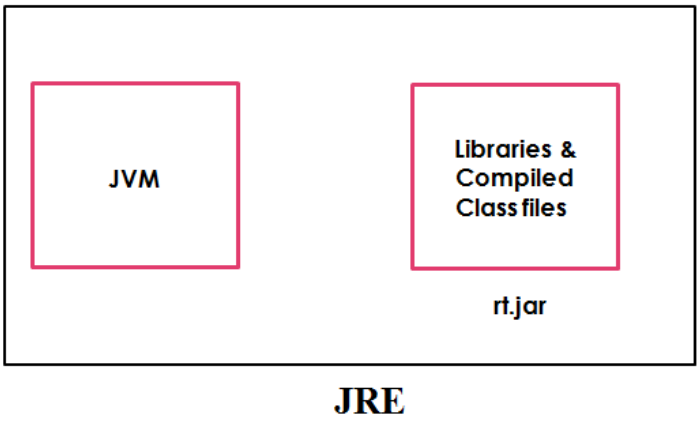
## Java Programming Environment

- Java is a concurrent, class-based, object-oriented programming and runtime environment, consisting of
  - A programming language
  - An API specification
  - A virtual machine specification
- Three powerful components of Java platform used for developing and running Java applications :
  1. Java Runtime Environment (JRE)
  2. Java Development Kit (JDK)
  3. Java Virtual Machine (JVM)



## 1. JAVA RUNTIME ENVIRONMENT (JRE)

- A software program needs an environment to run.
- The runtime environment loads class files and ensures there is access to memory and other system resources to run them.
- Java Runtime Environment provides the minimum requirements for executing a Java application programs.
- JRE is an installation package which provides environment to only run (not develop) the java program(or application)onto your machine.
- JRE is only used by them who only wants to run the Java Programs. i.e. end users of your system.
- JRE can be view as a subset of JDK.



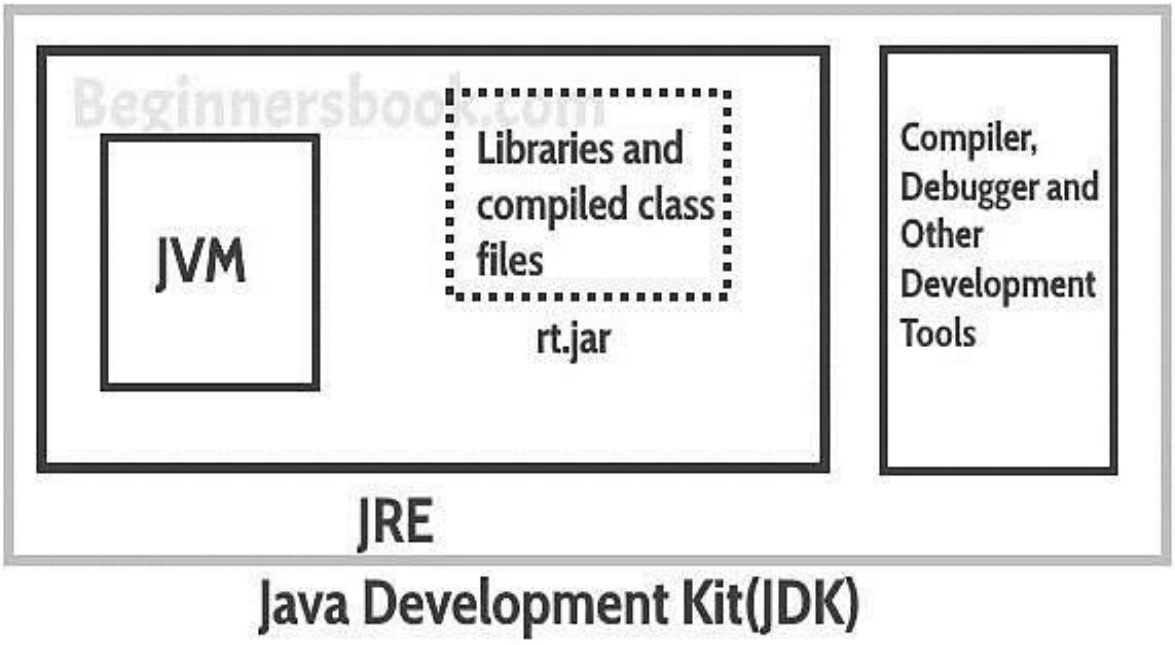



## 2. JAVA DEVELOPMENT KIT (JDK)

- The Java Development Kit (JDK) is a software development environment used for developing and executing Java applications and applets.
- It includes
  - The Java Runtime Environment (JRE)
  - An interpreter/loader (Java)
  - A compiler (javac)
  - An archiver (jar)
  - A documentation generator (Javadoc)
  - Other tools needed in Java development.

- **JDK provides the environment to develop and execute(run) the Java program.**
- **JDK is a kit(or package) which includes two things :**
  - Development Tools (to provide an environment to develop your java programs)
  - JRE (to execute your java program).
- JDK is only used by Java Developers.



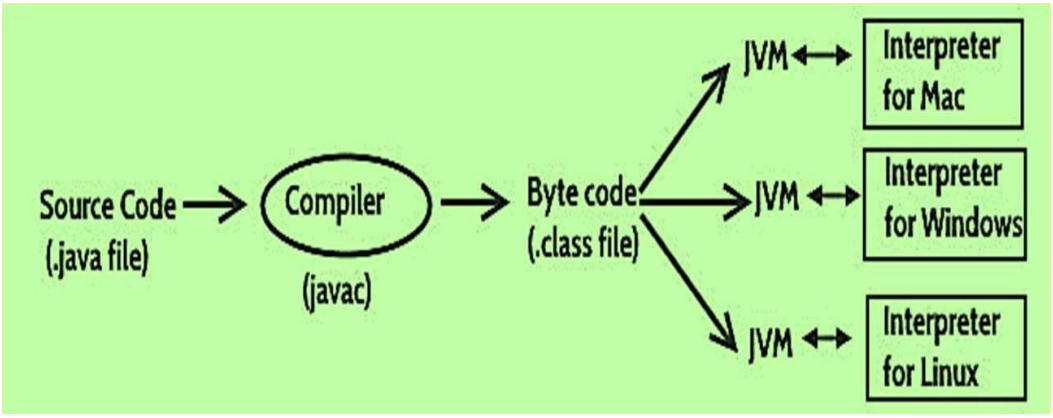


## 3. JAVA VIRTUAL MACHINE (JVM)

- JVM is a program which provides the runtime environment to execute Java programs. Java programs cannot run if a supporting JVM is not available.
- JVM is a virtual machine that resides in the real machine (your computer) and the machine language for JVM is byte code.
- The Java compiler generate byte code for JVM rather than different machine code for each type of machine.
- JVM executes the byte code generated by compiler and produce output.
- JVM is the one that makes java platform independent.



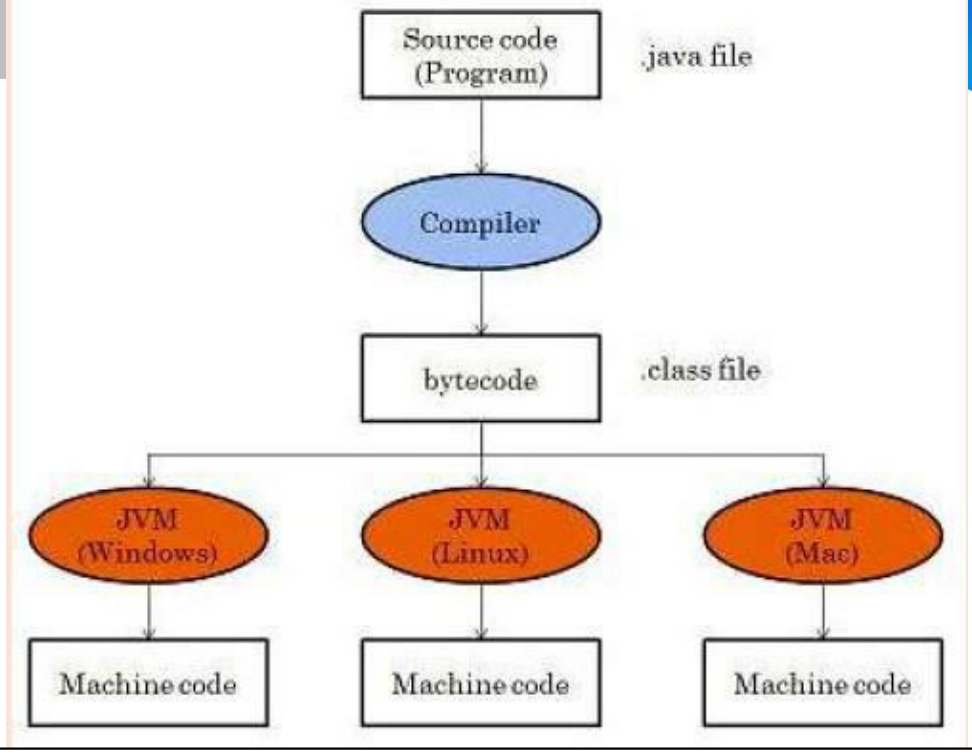
- The primary function of JVM is to execute the byte code produced by compiler
- The JVM doesn't understand Java source code, that's why we need to have javac compiler
- Java compiler (javac) compiles *.java files to obtain *.class files that contain the byte codes understood by the JVM.
- JVM makes java portable (write once, run anywhere).
- Each operating system has different JVM, however the output they produce after execution of byte code is same across all operating systems.






## BYTE CODE

- Java byte code is the instruction set for the Java Virtual Machine
- It is the machine code in the form of a .class file.
- Byte code is a machine independent code
- It is not completely a compiled code but it is an intermediate code somewhere in the middle which is later interpreted and executed by JVM.
- Byte code is a machine code for JVM.
- Byte code implementation makes Java a platform-Independent language.






## JAVA COMPILER

- Java is compiled language. But it is very different from traditional compiling in the way that after compilation source code is converted to byte code.
- Javac is the most popular Java compiler
- Java has a virtual machine called JVM which then converts byte code to target code of machine on which it is run.
- JVM performs like an interpreter. It doesn't do it alone, though. It has its own compiler to convert the byte code to machine code.
- This compiler is called Just In Time or JIT compiler.

## JAVA APPLET

- An applet is a special kind of Java program that is designed to be transmitted over the Internet and automatically executed by a Java-compatible web browser
- It runs inside the web browser and works at client side
- Applets are used to make the web site more dynamic and entertaining
- Applets are not stand-alone programs. Instead, they run within either a web browser or an applet viewer. JDK provides a standard applet viewer tool called applet viewer.



## Java Development Platforms

- Java Platform, Standard Edition (Java SE)
- Java Platform, Enterprise Edition (Java EE)
- Java Platform, Micro Edition (Java ME)
- Java FX

- A Java platform is a particular environment in which Java programming language applications run.
- All Java platforms consist of a Java Virtual Machine (JVM) and an application programming interface (API).
- The Java Virtual Machine is a program, for a particular hardware and software platform, that runs Java technology applications.
- An API is a collection of software components that you can use to create other software components or applications.
- Each Java platform provides a virtual machine and an API, and this allows applications written for that platform to run on any compatible system with all the advantages of the Java programming language: platform-independence, power, stability, ease-of-development and security.



## Development Platforms – Standard Edition

- Java platform, Standard Edition (Java SE) is a computing platform for development and deployment of portable code for desktop and server environments.
- Java SE's API provides the core functionality of the Java programming language.
- It defines everything from the basic types and objects of the Java programming language to high-level classes that are used for networking, security, database access, graphical user interface (GUI) development, and XML parsing.
- In addition to the core API, the Java SE platform consists of a virtual machine, development tools, deployment technologies, and other class libraries and toolkits commonly used in Java technology applications.

## Development Platforms – Enterprise Edition

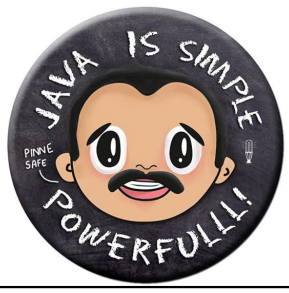
- The Java EE platform is built on top of the Java SE platform.
- The Java EE platform provides an API and runtime environment for developing and running large-scale, multi-tiered, scalable, reliable, and secure network applications.

Java™ Enterprise Edition



## JAVA BUZZWORDS

1. Simple
   - It's simple and easy to learn if you already know the basic concepts of Object Oriented Programming.
   - C++ programmer can move to JAVA with very little effort to learn.
   - Java syntax is based on C++
   - Java has removed many complicated and rarely-used features, for example, explicit pointers, operator overloading, etc.



## 2. Object oriented

- Java is true object oriented language. Everything in Java is an object.
- All program code and data reside within objects and classes.
- Java comes with an extensive set of classes, arranged in packages that can be used in our programs through inheritance.



## 3. Distributed

- Java is designed for distributed environment of the Internet. Its used for creating applications on networks
- 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 creation of a cross platform programs by compiling into an intermediate representation called Java Byte code.
- Interpreted : Byte code is then interpreted, which generates machine code that can be directly executed by the machine that provides a Java Virtual machine



## 5. Robust

- It provides many features that make the program execute reliably in 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.
- Java, with the help of exception handling captures all types of serious errors and eliminates any risk of crashing the system.

## 6. Secure

- Java provides a “firewall” between a networked application and your computer.
- When a Java Compatible Web browser is used, downloading can be done safely without fear of viral infection or malicious intent.
- 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.

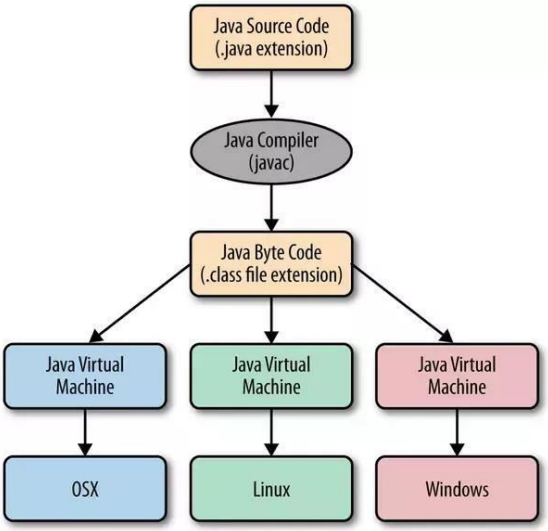


## 7. Architecture Neutral

- Java language and Java Virtual Machine helped in achieving the goal of “write once; run anywhere, any time, forever.”
- Changes and upgrades in operating systems, processors and system resources will not force any changes in Java Programs.

## 8. Portable

- Java is portable because it facilitates you to carry the Java byte code to any platform. It doesn't require any implementation.
- Java Provides a way to download programs dynamically to all the various types of platforms connected to the Internet.






## 9. High Performance

- Java performance is high because of the use of byte code.
- The byte code can be easily translated into native machine code

## 11.Multithreaded

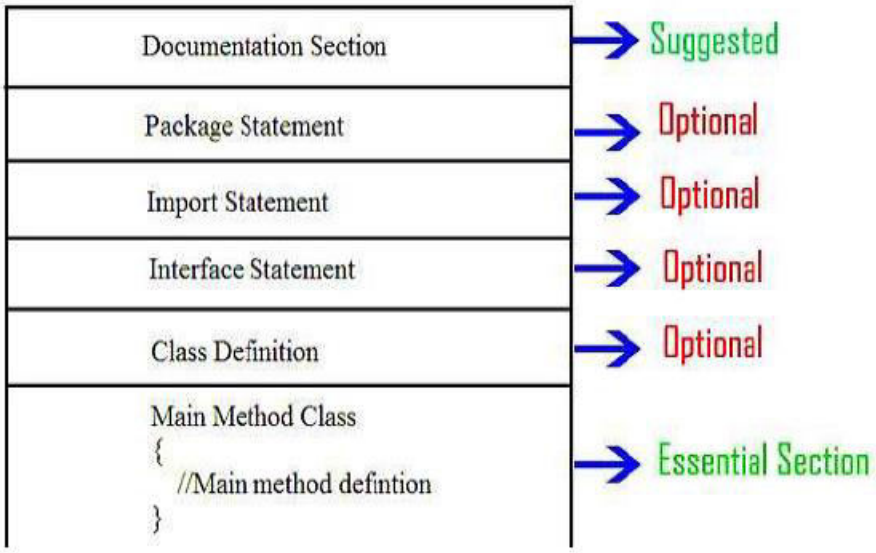
- Multithreaded Programs handled multiple tasks simultaneously, which was helpful in creating interactive, networked programs.
- Java run-time system comes with tools that support multiprocess synchronization used to construct smoothly interactive system



## 12. Dynamic

- Java is capable of linking in new class libraries, methods, and objects.
- It supports functions from native languages (the functions written in other languages such as C and C++).
- It supports dynamic loading of classes. It means classes are loaded on demand

## Java Program Structure






## A simple java program to print hello world

```
/*This is a simple Java Program
Call this file "Hello.java*/
public class Hello
{
    //main method declaration
    public static void main(String[] args)
    {
      System.out.println("hello world");
    }
}
```

## Documentation Section

- You can write a comment in this section. It helps to understand the code. These are optional
- It is used to improve the readability of the program.
- The compiler ignores these comments during the time of execution
- There are three types of comments that Java supports
  - Single line Comment //This is single line comment
  - Multi-line Comment /* this is multiline comment. and support multiple lines*/
  - Documentation Comment /** this is documentation cmnt*/

39

## Package Statement

- We can create a package with any name. A package is a group of classes that are defined by a name.
- That is, if you want to declare many classes within one element, then you can declare it within a package
- It is an optional part of the program, i.e., if you do not want to declare any package, then there will be no problem with it, and you will not get any errors.
- Package is declared as: package package_name;
- Eg: package mypackage;

40

## Import Statement

- If you want to use a class of another package, then you can do this by importing it directly into your program.
- Many predefined classes are stored in packages in Java
- We can import a specific class or classes in an import statement.
- Examples:

```
1 import java.util.Date; //imports the date class
2 import java.applet.*;  //imports all the classes from the java applet package
```



## Interface Statement

- This section is used to specify an interface in Java
- Interfaces are like a class that includes a group of method declarations
- It's an optional section and can be used when programmers want to implement multiple inheritances within a program.

```
1 interface stack{
2 void push(int item);
3 void pop();
4 }
```

## Class Definition

- A Java program may contain several class definitions.
- Classes are the main and essential elements of any Java program.
- It defines the information about the user defined classes in a program.
- A class is a collection of variables and methods that operate on the fields.
- Every program in Java will have at least one class with the main method.



## Main Method Class

- The main method is from where the execution actually starts and follows the order specified for the following statements
- Every Java stand-alone program requires the main method as the starting point of the program.
- This is an essential part of a Java program.
- There may be many classes in a Java program, and only one class defines the main method
- Methods contain data type declaration and executable statements.

## Class Definition

```
1  public class Example{
2  //main method declaration
3  public static void main(String[] args){
4  System.out.println("hello world");
5  }
6  }
```

- This creates a class called Example.
- You should make sure that the class name starts with a capital letter, and the word public means it is accessible from any other classes.



- public class Example
  - This creates a class called Example. You should make sure that the class name starts with a capital letter, and the public word means it is accessible from any other classes.
- Comments
  - To improve the readability, we can use comments to define specific note of functionality of methods, etc for the programmer.
- Braces
  - The curly brackets are used to group all the commands together.
  - To make sure that the commands belong to a class or a method.

- public static void main
  - When the main method is declared public, it means that it can be used outside of this class as well.
  - The word static means that we want to access a method without making its objects.
  - The word void indicates that it does not return any value. The main is declared as void because it does not return any value.
  - main is a method ; this is a starting point of a Java program.

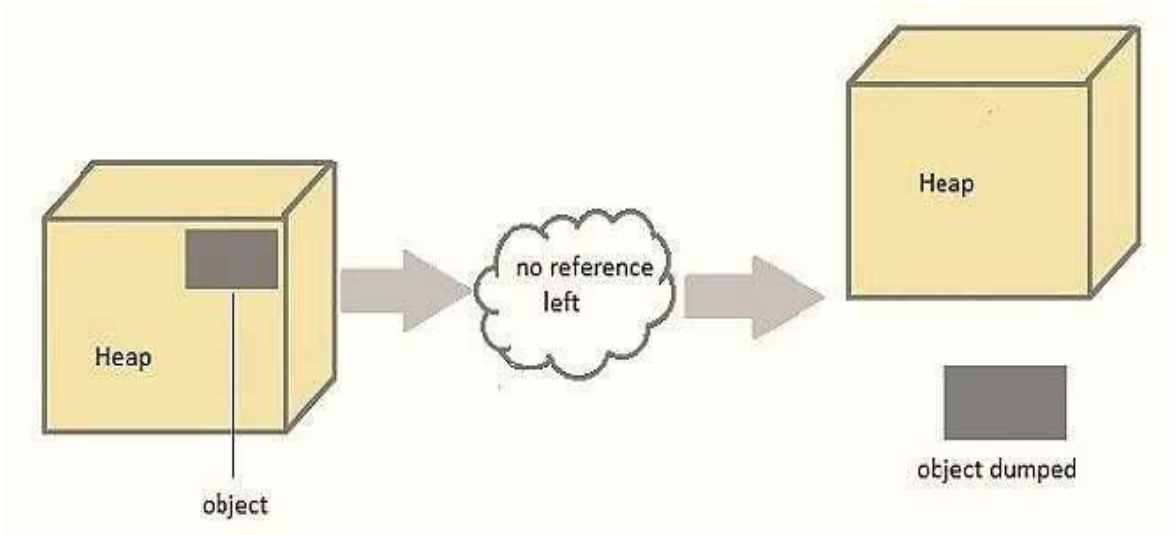


- String [ ] args
  - It is an array where each element is a string, which is named as args.
  - Used to store command line arguments.
  - If you run the Java code through a console, you can pass the input parameter.
  - The main() takes it as an input.
- System.out.println();
  - This statement is used to print the output on the screen.
  - System is a predefined class, and out is an object of the PrintWriter class.
  - The method println prints the text on the screen with a new line.
  - We can also use print() method instead of println() method.
  - All Java statement ends with a semicolon.

## Garbage Collection in Java (A process of releasing unused memory)

- When JVM starts up, it creates a heap area which is known as runtime data area. This is where all the objects (instances of class) are stored
- Since this area is limited, it is required to manage this area efficiently by removing the objects that are no longer in use.
- The process of removing unused objects from heap memory is known as Garbage collection and this is a part of memory management in Java.
- Languages like C/C++ don't support automatic garbage collection, however in java, the garbage collection is automatic.





- In java, garbage means unreferenced objects.
- Main objective of Garbage Collector is to free heap memory by destroying unreachable objects.
- Unreachable objects : An object is said to be unreachable iff it doesn't contain any reference to it.
- Eligibility for garbage collection : An object is said to be eligible for GC(garbage collection) iff it is unreachable.
- finalize() method – This method is invoked each time before the object is garbage collected and it perform cleanup processing.
- The Garbage collector of JVM collects only those objects that are created by new keyword. So if we have created any object without new, we can use finalize method to perform cleanup processing

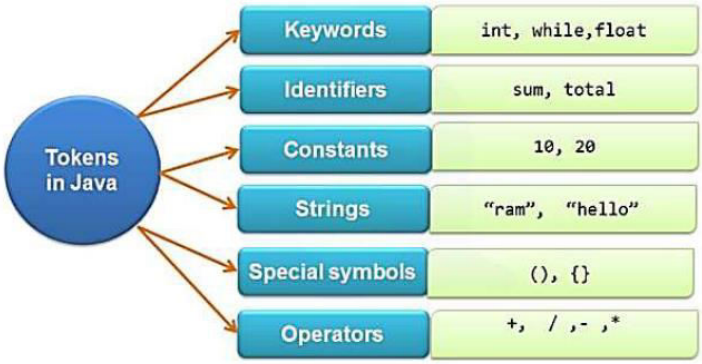


## Request for Garbage Collection

- We can request to JVM for garbage collection however, it is upto the JVM when to start the garbage collector.
- Java gc() method is used to call garbage collector explicitly.
- However gc() method does not guarantee that JVM will perform the garbage collection.
- It only request the JVM for garbage collection. This method is present in System and Runtime class.

## Java Lexical Issues (Java Tokens)

- TOKENS
  - Java Tokens are the smallest individual building block or smallest unit of a Java program
  - Java program is a collection of different types of tokens, comments, and white spaces.






## Keywords

- A keyword is a reserved word. You cannot use it as a variable name, constant name etc.
- The meaning of the keywords has already been described to the java compiler. These meaning cannot be changed.
- Thus, the keywords cannot be used as variable names because that would try to change the existing meaning of the keyword, which is not allowed.
- Java language has reserved 50 words as keywords

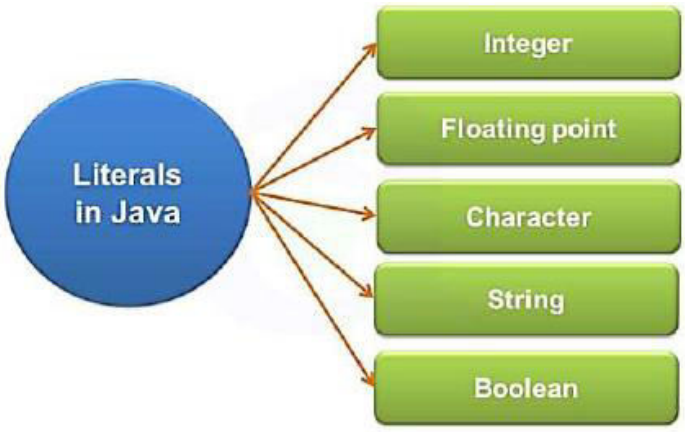
| Keywords in Java | | | | |
|---|---|---|---|---|
| abstract | default | if | private | this |
| assert | do | implements | protected | throw |
| boolean | double | import | public | throws |
| break | else | instanceof | return | transient |
| byte | enum | int | short | try |
| case | extends | interface | static | void |
| catch | final | long | strictfp | volatile |
| char | finally | native | super | while |
| class | float | new | switch | |
| continue | for | package | synchronized | |

## Identifiers

- Identifiers are the names of variables, methods, classes, packages and interfaces
- Identifier must follow some rules.
- All identifiers must start with either a letter( a to z or A to Z ) or currency character($) or an underscore.
- They must not begin with a digit
- After the first character, an identifier can have any combination of characters.
- A Java keywords cannot be used as an identifier.
- Identifiers in Java are case sensitive, foo and Foo are two different identifiers.
- They can be any length Eg: int a; char name;

## Constants or Literals

- Constants are fixed values of a particular type of data,which cannot be modified in a program.
- Java language specifies five major type of literals.
- Eg:
  - Integer literal : 100
  - Floating-point literal : 98.6
  - Character literal : 's'
  - String literal : "sample"






## Comments

- Three types of comments defined by Java.
  - Single line comment
  - Multiline comment
  - Documentation comment

| Comment type | Meaning |
|---|---|
| // comment | Single-line comments |
| /* comment */ | Multi-line comments |
| /** documentation */ | Documentation comments |

## String

- In java, string is basically an object that represents sequence of char values.
- An array of characters works same as java string.
- Eg:

  char[] ch = {'a',‘b',‘c',‘d'};

  String s = “abcd";
- Java String class provides a lot of methods to perform operations on string such as compare(), concat(), equals(), split(), length(), replace(), compareTo(), intern(), substring() etc.



## Separators

- In Java, there are a few characters that are used as separators.
- The most commonly used separator in Java is the semicolon.

| Symbol | Name | Purpose |
|---|---|---|
| ( ) | Parentheses | Used to contain lists of parameters in method definition and invocation. Also used for defining precedence in expressions, containing expressions in control statements, and surrounding cast types. |
| { } | Braces | Used to contain the values of automatically initialized arrays. Also used to define a block of code, for classes, methods, and local scopes. |
| [ ] | Brackets | Used to declare array types. Also used when dereferencing array values. |
| ; | Semicolon | Terminates statements. |
| , | Comma | Separates consecutive identifiers in a variable declaration. Also used to chain statements together inside a **for** statement. |
| . | Period | Used to separate package names from subpackages and classes. Also used to separate a variable or method from a reference variable. |
| :: | Colons | Used to create a method or constructor reference. (Added by JDK 8.) |

## Operators

- An operator is a symbol that takes one or more arguments and operates on them to produce a result.
- Unary Operator
- Arithmetic Operator
- shift Operator
- Relational Operator
- Bitwise Operator
- Logical Operator
- Ternary Operator
- Assignment Operator



## Whitespace

- Java is a free-form language. This means that you do not need to follow any special indentation rules
- White space in Java is used to separate tokens in the source file. It is also used to improve readability of the source code.
- Eg: int i = 0;
- White spaces are required in some places. For example between the int keyword and the variable name.
- In java whitespace is a space, tab, or newline Prepared

# END OF MODULE 1