SES'S L.S.RAHEJA COLLEGE OF ARTS AND COMMERCE

Course: Core Java Unit: I Prepared by: Ms. Prajakta Joshi/ Ms. Srushty Padate

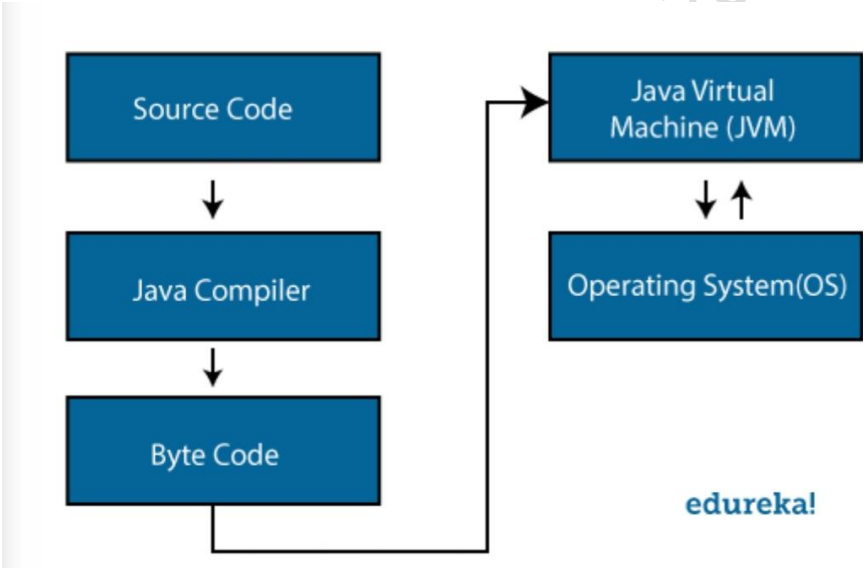
JAVA ARCHITECTURE AND COMPONENTS:

Java Architecture combines the process of compilation and interpretation.

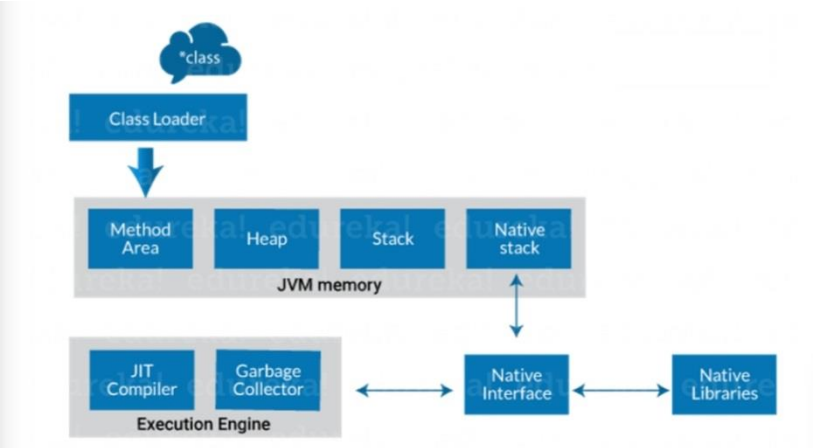
- The code written in Java, is converted into byte codes which is done by the Java Compiler.
- The byte codes, then are converted into machine code by the JVM.
- The Machine code is executed directly by the machine.

Components of Java Architecture
There are three main components of Java language: ***JVM, JRE, and JDK***.

Java Virtual Machine, Java Runtime Environment and Java Development Kit respectively.



Java Virtual Machine:



Explanation:

Class Loader: Class loader is a subsystem of JVM. It is used to load class files. Whenever we run the java program, class loader loads it first.

Class method area: It is one of the Data Area in JVM, in which Class data will be stored. Static Variables, Static Blocks, Static Methods, Instance Methods are stored in this area.

Heap: A heap is created when the JVM starts up. It may increase or decrease in size while the application runs.

Stack: JVM stack is known as a thread stack. It is a data area in the JVM memory which is created for a single execution thread. The JVM stack of a thread is used by the thread to store various elements i.e.; local variables, partial results, and data for calling method and returns.

Native stack: It subsumes all the native methods used in your application.

Execution Engine:

- JIT compiler
- Garbage collector

JIT compiler: The Just-In-Time (JIT) compiler is a part of the runtime environment. It helps in improving the performance of Java applications by compiling bytecodes to machine code at run time. The JIT compiler is enabled by default. When a method is compiled, the JVM calls the compiled code of that method directly. The JIT compiler compiles the bytecode of that method into machine code, compiling it "just in time" to run.

Garbage collector: As the name explains that Garbage Collector means to collect the unused material. Well, in JVM this work is done by Garbage collection. It tracks each and every object available in the JVM heap space and removes unwanted ones.
Garbage collector works in two simple steps known as Mark and Sweep:

- Mark – it is where the garbage collector identifies which piece of memory is in use and which are not
- Sweep – it removes objects identified during the "mark" phase.

Java Runtime Environment:

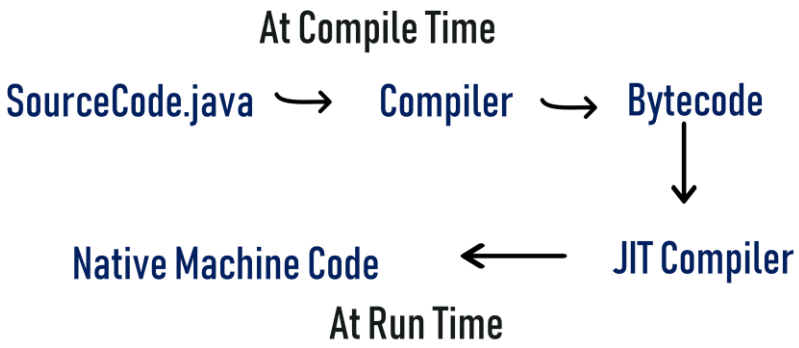
The JRE software builds a runtime environment in which Java programs can be executed. The JRE is the on-disk system that takes your Java code, combines it with the needed libraries, and starts the JVM to execute it. The JRE contains libraries and software needed by your Java programs to run. JRE is a part of JDK (which we will study later) but can be downloaded separately.

Java Development Kit:

The Java Development Kit (JDK) is a software development environment used to develop Java applications and applets. It contains JRE and several development tools, an interpreter/loader (java), a compiler (javac), an archiver (jar), a documentation generator (javadoc) accompanied with another tool.

How is Java platform independent?

When is any programming language called as platform-independent? Well, if and only if it can run on all available operating systems with respect to its development and compilation. Now, Java is platform-independent just because of the bytecode. Let me tell you what exactly is a bytecode? In simple terms, Bytecode is a code of the JVM which is machine-understandable. Bytecode execution in Java proves it is a platform-independent language. Here, I will show you the steps involved in the process of java bytecode execution.



Below is the explanation of the steps involved:

sample.java → javac (sample. class) → JVM(sample.obj) → final output

First source code is used by java compiler and is converted in .class file. The class file code is in byte code form and that class file is used by JVM to convert into an object file. After that, you can see the final output on your screen.

Java Identifiers

In programming languages, identifiers are used for identification purpose. In Java, an identifier can be a class name, method name, variable name or a label.

Example:

```
public class Test
{
   public static void main(String[] args)
   {
      int a = 20;
   }
}
```

Rules for defining Java Identifiers

There are certain rules for defining a valid java identifier. These rules must be followed, otherwise we get compile-time error. These rules are also valid for other languages like C,C++.

- The only allowed characters for identifiers are all alphanumeric characters([**A-Z**],[**a-z**],[**0-9**]), '**$**'(dollar sign) and '**_**' (underscore).For example "geek@" is not a valid java identifier as it contain '@' special character.
- Identifiers should **not** start with digits(**[0-9]**). For example "123geeks" is a not a valid java identifier.
- Java identifiers are **case-sensitive**.
- There is no limit on the length of the identifier but it is advisable to use an optimum length of 4 – 15 letters only.
- **Reserved Words** can't be used as an identifier. For example "int while = 20;" is an invalid statement as while is a reserved word. There are **53** reserved words in Java.

Compiler vs Interpreter

Compiler and Interpreter are two different ways to execute a program written in a programming or scripting language.

A **compiler** takes entire program and converts it into object code which is typically stored in a file. The object code is also refereed as binary code and can be directly executed by the machine after linking. Examples of compiled programming languages are C and C++.

An **Interpreter** directly executes instructions written in a programming or scripting language without previously converting them to an object code or machine code. Examples of interpreted languages are Perl, Python and Matlab.

Following are some interesting facts about interpreters and compilers.

1) Both compilers and interpreters convert source code (text files) into tokens, both may generate a parse tree, and both may generate immediate instructions. The basic difference is that a compiler system, including a (built in or separate) linker, generates a stand alone machine code program, while an interpreter system instead performs the actions described by the high level program.

2) Once a program is compiled, its source code is not useful for running the code. For interpreted programs, the source code is needed to run the program every time.

3) In general, interpreted programs run slower than the compiled programs.

4) Java programs are first compiled to an intermediate form, then interpreted by the interpreter.

Operators in Java

Operator in Java is a symbol which is used to perform operations. For example: +, -, *, / etc.

There are many types of operators in Java which are given below:

- Unary Operator,
- Arithmetic Operator,
- Shift Operator,
- Relational Operator,
- Bitwise Operator,

- Logical Operator,
- Ternary Operator and
- Assignment Operator.

NOTE: ALL THE OPERATOR EXPLAINATION SHOULD BE GIVEN WITH AN EXAMPLE(PROGRAM CODE).

Data Types in Java

Data types specify the different sizes and values that can be stored in the variable. There are two types of data types in Java:

1. **Primitive data types:** The primitive data types include boolean, char, byte, short, int, long, float and double.
2. **Non-primitive data types:** The non-primitive data types include Classes, Interfaces, and Arrays.

NOTE: ALL THE DATA TYPES SHOULD BE EXPLAINED WITH THEIR SIZE, RANGE.

Data Type	Default Value	Default size
boolean	false	1 bit
char	'\u0000'	2 byte
byte	0	1 byte
short	0	2 byte
int	0	4 byte
long	0L	8 byte
float	0.0f	4 byte
double	0.0d	8 byte

Write answers for following

1 Explain the following
- Logical operators
- Conditional operators

2 Write a note on identifiers.

3 What is JVM? Explain JVM components.

4 Write a main() method of java written. Explain it in details.

5 Explain the rules for identifiers in Java.

6 Write a note on:
i) Autoboxing and unboxing
ii) ii) Java Development Kit(JDK).

7 Write a Java code to
i) check whether the string “madam” is starting and ending with a same letter.
ii) countall vowels in a string “Morning”.
iii) replace ‘e’ with ‘E’ in a string “evening”.
iv) append “Welcome” and “MADAM”

8 Java is called as platform independent and strongly typed language. Justify your answer.

9 Write a short note on architecture of java.

10 What is java runtime environment? Explain.

11 Explain keywords, white spaces, braces, comments and code blocks.

12 Differentiate between compiler and interpreter.

13 What is java API?

14 List and explain the silent features of java.

15 What are different operators in java? Explain with example.

16 What are primitive data types? Explain their size and range.

17 List different conditional operators with example for each.

18 What are different components of java architecture?

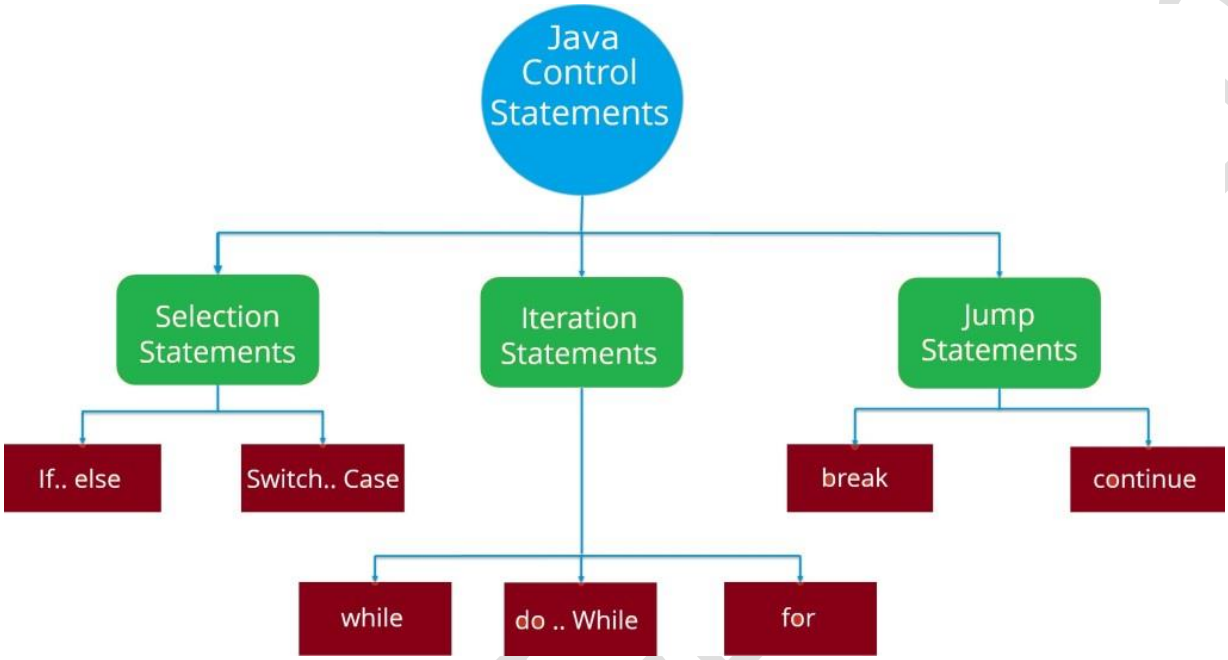
19 What are properties of operators.

20 Write a java code to illustrate working of different relational operators.

SES'S L.S.RAHEJA COLLEGE OF ARTS AND COMMERCE

Course: Core Java Unit: II Prepared by: Ms. Prajakta Joshi/ Ms. Srushty Padate

CONTROL FLOW STATEMENTS IN JAVA



Classes:

What are the different types of Classes in Java?

- POJO Class.
- Static Class.
- Concrete Class.
- Abstract Class.
- Final Class.
- Inner Class. Nested Inner class. Method Local inner classes. Anonymous inner classes. Static nested classes.

NOTE: ALL TYPES SHOULD BE EXPLAINED WITH AN EXAMPLE.

3 Ways to initialize object

There are 3 ways to initialize object in Java.

1. By reference variable
2. By method
3. By constructor

1) Object and Class Example: Initialization through reference

Initializing an object means storing data into the object. Let's see a simple example where we are going to initialize the object through a reference variable.

2) Object and Class Example: Initialization through method

In this example, we are creating the two objects of Student class and initializing the value to these objects by invoking the insertRecord method. Here, we are displaying the state (data) of the objects by invoking the displayInformation() method.

3) Object and Class Example: Initialization through a constructor

NOTE: ALL TYPES EXPLAINATION WITH EXAMPLE.

Method Overloading in Java with examples

Method Overloading is a feature that allows a class to have more than one method having the same name, if their argument lists are different. It is similar to constructor overloading in Java, that allows a class to have more than one constructor having different argument lists.

let's get back to the point, when I say argument list it means the parameters that a method has: For example the argument list of a method add(int a, int b) having two parameters is different from the argument list of the method add(int a, int b, int c) having three parameters.

Three ways to overload a method

In order to overload a method, the argument lists of the methods must differ in either of these:

1. Number of parameters.

Example: This is a valid case of overloading

add(int, int)
add(int, int, int)

2. Data type of parameters.

Example:

add(int, int)
add(int, float)

3. Sequence of Data type of parameters.

Example:

add(int, float)
add(float, int)

EXAMPLE:

```
class DisplayOverloading
{
   public void disp(char c)
   {
       System.out.println(c);
```

```
    }
        public void disp(char c, int num)
    {
        System.out.println(c + " "+num);
    }
}
class Sample
{
  public static void main(String args[])
  {
     DisplayOverloading obj = new DisplayOverloading();
     obj.disp('a');
     obj.disp('a',10);
  }
}
```

Java Constructors

A constructor in Java is a **special method** that is used to initialize objects. The constructor is called when an object of a class is created. It can be used to set initial values for object attributes:

Example

Create a constructor:

```
// Create a MyClass class
public class MyClass {
  int x;  // Create a class attribute

  // Create a class constructor for the MyClass class
  public MyClass() {
    x = 5;  // Set the initial value for the class attribute x
  }

  public static void main(String[] args) {
    MyClass myObj = new MyClass(); // Create an object of class MyClass (This will call the constructor)
    System.out.println(myObj.x); // Print the value of x
  }
}
```

Java Garbage Collection

In java, garbage means unreferenced objects.

Garbage Collection is process of reclaiming the runtime unused memory automatically. In other words, it is a way to destroy the unused objects.

To do so, we were using free() function in C language and delete() in C++. But, in java it is performed automatically. So, java provides better memory management.

Advantage of Garbage Collection

- It makes java **memory efficient** because garbage collector removes the unreferenced objects from heap memory.
- It is **automatically done** by the garbage collector(a part of JVM) so we don't need to make extra efforts.

finalize() method

The finalize() method is invoked each time before the object is garbage collected. This method can be used to perform cleanup processing. This method is defined in Object class as:

```
1. protected void finalize(){}
```

gc() method

The gc() method is used to invoke the garbage collector to perform cleanup processing. The gc() is found in System and Runtime classes.

```
1. public static void gc(){}
```

NOTE: THESE POINTS ARE INDICATIVE AND NOT EXHAUSTIVE. PLEASE ELLABORATE THE ANSWERS WITH PROPER EXAMPLES WHEREVER APPLICABLE.

Write answers for following

1 Write a java code to display grade of students based on basic grading system using switch case statement.
2 Differentiate between while loop and do while loop.
3 What are methods and what is method overloading?
4 Write a short note on access specifies in java.
5 Explain the functionality of different iterative statements with suitable examples.
6 What is a constructor and characteristics of a constructor.?
7 Write a code to illustrate the working of method overloading.
8 List and explain types of classes in java.
9 What is garbage collection in java and how is it helpful?
10 When do you use keywords final and static? Explain.
11 What is a class object and what are its attributes?
12 What are characteristics of a member of a class?
13 Explain if else with an example.
14 Write a code in java to illustrate if statement and nested if statement.
15 Write a code to print pattern using for loop.

```
@ @ @ @ @
@ @ @ @
@ @ @
@ @
@
```

16 Write a code to illustrate do while loop.
17 What are different types of classes in java.?
18 Explain break and continue statements .
19 Explain how to create an object of a class with example.
20 Write a code to create methods with same name for adding, subtracting and multiplying two numbers.