JAVA Programming

**Essentials**

# 

**Suman Shekhar**

**Day 1**

**Types of computing device-**

* Mechanical Computer
* Digital Watch, Computer

**Common Computing-**

* 0 or 1 (Bits)
* 99.999999 % computing machines uses binary digits(0,1) for computing.

**Quantum Computing-**

* 0,1,(0|1) - (QBits)

Smartphone, PC/laptop, MainFrame(Big Iron of IT Industry), Super Computer, Quantum Computer(D-Wave)

**Basics of a common computing device-**

* i/p device - o/p device
* CPU

**CPU**

* Processor

(RAM interacts with the processor)

* RAM (primary Storage Unit)*-volatile*
  + (NFS) Loading is the process of transferring/copying a file/program from HDD to RAM.
* HDD/SSD (Secondary Storage unit)-*permanent*
  + SAVING(Contents are getting permanently stored here)

(NFS) game files are saved in HDD

**Memory Units-**

* Bits (the most fundamental unit of memory storage)
* 1 bit = (0/1) uses transistors for implementation (2 transistor)
* 1 Byte = 2 nibble =8 bits
  + 8 GB = 8 \* 10^9\*2\*8
  + 128\*10^9 transistors

^ (carat symbol)

**JAVA**

1. ***James Gosling,*** He called the father of JAVA.
2. Was the property of **SUN microsystem** but now belongs to **Oracle**.
3. **RANKINGS:**
   1. <https://www.tiobe.com/tiobe-index/>
4. OPEN SOURCE:
   1. (VLC) (FireFOX)
   2. Python
   3. Signal
5. CLOSED SOURCE:
   1. JAVA
6. Java was initially open source. ORACLE owned it and made it closed source.
7. Salient features of java
   1. **Object-Oriented\***

In Java, everything is an Object. Java can be easily extended since it is based on the Object model.

* 1. **Platform Independent\***

Unlike many other programming languages including C and C++, when Java is compiled, it is not compiled into the platform-specific machine-level code, rather into platform-independent bytecode. This byte code is distributed over the web and interpreted by the Virtual Machine (JVM) on whichever platform it is being run on.

* 1. **Simple\***

Java is designed to be easy to learn. If you understand the basic concept of OOP Java, it would be easy to master.

* 1. **Secure**

With Java's secure features it enables to develop virus-free, tamper-free systems. Authentication techniques are based on public-key encryption.

* 1. **Architecture-neutral**

Java compiler generates an architecture-neutral object file format, which makes the compiled code executable on many processors, with the presence of a Java runtime system.

* 1. **Portable\***

Being architecture-neutral and having no implementation dependent aspects of the specification makes Java portable. The compiler in Java is written in ANSI C with a clean portability boundary, which is a POSIX subset.

* 1. **Robust\***

Java makes an effort to eliminate error-prone situations by emphasizing mainly on compile-time error checking and runtime checking.

* 1. **Multithreaded\***

With Java's multithreaded feature it is possible to write programs that can perform many tasks simultaneously. This design feature allows the developers to construct interactive applications that can run smoothly.

* 1. **Interpreted**

Java byte code is translated on the fly to native machine instructions and is not stored anywhere. The development process is more rapid and analytical since linking is an incremental and light-weight process.

* 1. **High Performance\***

With the use of Just-In-Time compilers, Java enables high performance.

* 1. **Distributed**

Java is designed for the distributed environment of the internet.

* 1. **Dynamic**

Java is considered to be more dynamic than C or C++ since it is designed to adapt to an evolving environment. Java programs can carry an extensive amount of run-time information that can be used to verify and resolve accesses to objects at run-time. object Oriented

In Java, everything is an Object. Java can be easily extended since it is based on the Object model.

<https://www.tutorialspoint.com/What-are-the-major-features-of-Java-programming>

**INSTALLATION:**

1. Online go to java compiler

<https://www.onlinegdb.com/online_java_compiler>

1. For running offline in our PC we need JDK (Java Development Kit)

The JDK includes tools useful for developing and testing programs written in the Java programming language and running on the Java platform.

* OPEN JDK (managed by REDHAT)
* **JAVA SE 15.0.2**

<https://www.oracle.com/in/java/technologies/javase-jdk15-downloads.html>

First.java

public class First

{

public static void main(String[] args) {

System.out.println("Hello World");

}

}

👁️

1. Its a rule that The file name and the class name enclosing the

public static void main()

1. It’s a convention that Class names should begin with a capital letter and follow PASCAL CONVENTION ie *ZooAnimals*

Should be exactly the same (even the cases).

1. Avoid using **reserved words** for a class name or variable names.

**(Compilation)**

**The conversion of High-level JAVA source code to native machine-level code (**Machine understands 0,1**)**

So compilation is a process in which a compiler converts the source code into machine-readable code

In JAVA this happens in a 2 step process:

**STEP 1- *compilation***

*First.java (Source code)*

We will invoke JAVA compiler

*> javac First.java*

* javac invokes java compiler
* A new file *First.class* (class file or bytecode) is generated after compilation

*This will produce an intermediate level code called bytecode (First.class)*

**STEP 2-** ***conversion and execution***

*> java First*

o/p > hello world

*In the second step, First.class file will get converted to native machine-level code and then executed. This two-step process gives JAVA many benefits among other programming languages.*

**Day 2**

**Software dev tools:**

1. **Code Editors (Compatible with almost all the programming languages)**
   1. Sublime
   2. VS code
   3. Notepad ++
   4. Edit +
   5. Bracketts
2. **IDE [Integrated Development Environment] (Generally Language Specific and best suited for PROJECT works)**
   1. ECLIPSE
   2. IntelliJ
3. **Architecture of JAVA**

**DATA TYPES IN JAVA:**

1. Numeric data
2. Character data
3. Strings
4. Arrays

**NUMERIC DATA**

1. **Integers**
2. byte (8 bit) -128 to 127
3. short (16 bit)
4. int (32 bit)
5. long (64 bit) or 8 bytes

***RANGE FORMULA*** **-(2 (n-1)) to + (2(n-1) -1)**

***Internally utilises base 2 format to store***

**b) decimal (real number)**

1. float (4 Bytes) = 32 bit (have to mention f or F explicitly after float value)
2. double (8 Bytes)

***Internally utilises IEEE format to store***

**[IEEE Arithmetic (oracle.com)](https://docs.oracle.com/cd/E19957-01/806-3568/ncg_math.html" \l ":~:text=The IEEE single format consists,shown in FIGURE 2-1.)**

[**(1240) Floating Point Number Representation in IEEE 754 Format - YouTube**](https://www.youtube.com/watch?v=qBHUGy1xteg)

[**(1240) Floating Point Number Representation - YouTube**](https://www.youtube.com/watch?v=XOMTNy2qiZ0)

**Character DATA**

**Uses 2 Bytes**

***Internally utilises UTF character set to store***

***UTF 16***

**Boolean DATA**

**boolean b= false;**

**Day 3**

**ARRAYS**

**Types of Encoding:**

1. ANSI
2. UNICODE
3. Unicode big-endian
4. UTF-8

## List of Reserved Java Keywords

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| abstract | assert | [boolean](https://www.thoughtco.com/definition-of-bool-958287) | break | byte | case |
| catch | char | class | const | continue | default |
| double | do | else | enum | extends | false |
| final | finally | float | for | goto | if |
| implements | import | instanceof | int | interface | long |
| native | new | null | package | private | protected |
| public | return | short | static | strictfp | super |
| switch | synchronized | this | throw | throws | transient |
| true | try | void | volatile | while |  |

\*The *strictfp* keyword was added to this list in Java Standard Edition version 1.2, *assert* in version 1.4, and [*enum*](https://www.thoughtco.com/what-is-an-enum-958326) in version 5.0.

55 restricted / reserved words

**Day 5**

During the execution

JRE (Java Runtime Environment) gets loaded onto RAM

JRM-memory location

JRE , JVM ( Java Virtual Machine)

|  |
| --- |
| CODE (Actual Code) |
| Stack (All the method calls are stored here, local var, ref var) |
| Static (All the static element) |
| Heap (All the instance(OBJECT gets located here) |

**STACK**

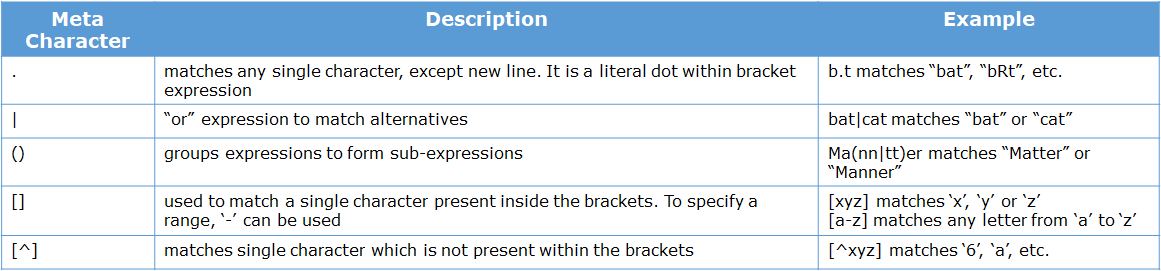
**Its a Data Structure following LIFO / FILO order**

|  |  |
| --- | --- |
|  |  |
|  |
|  |
| **check()** |
| **p s v main(**  **int a=400;**  **Fruit (** [5253ab])  **Vegetable (** [5253ab])  **)** | **Activation RECORD** |

**HEAP (Objects are stored here)**

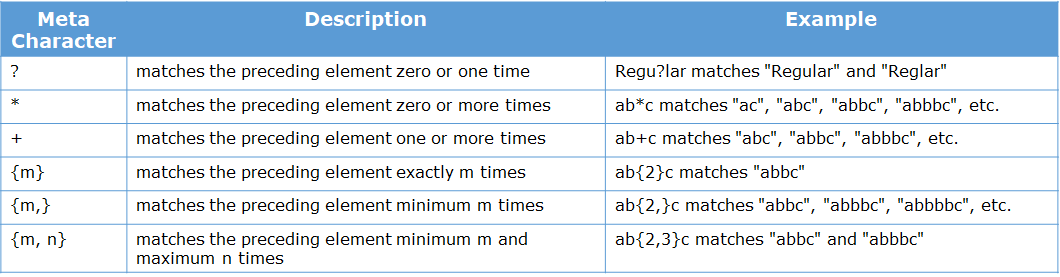
|  |
| --- |
| **CP (Constant Pool)** Duplicates are not allowed  “Tomato” [5253ab]  “Tomato are RED” [4634] |
| **NCP (Non Constant Pool)** Duplicates are allowed |
|  |

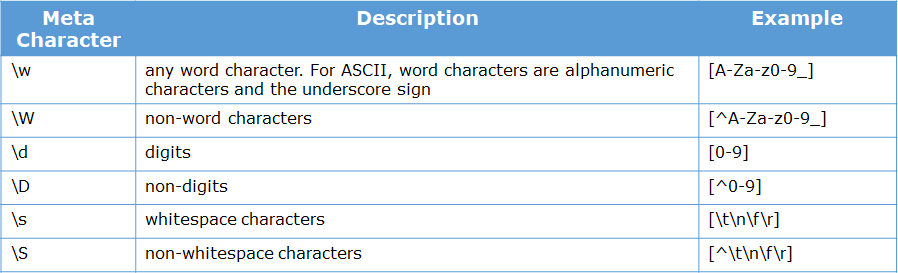
1. JVM creates OBJECT and converts bytecode to MACHINE level code
2. *.class* file is a platform independent code.
3. JAVA is platform independent language but JVM is platform dependent.

**REGEX**

***Meta Characters***

***Meta Quantifiers***

****

****

Note: "\w" is a meta character. But in Java, '\' is an escape character. Hence, we have to escape it using another backslash.

E.g. - "[\\w]+"

class Tester {

public static void main(String args[]) {

String regex1 = "Welcome.\*";

String str1 = "Welcome to India";

String regex2 = "Welcome to (India|Mysore)";

String str2 = "Welcome to Mysore";

String regex3 = "[^Welcome]";

String str3 = "1";

System.out.println(str1.matches(regex1));

System.out.println(str2.matches(regex2));

System.out.println(str3.matches(regex3));

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

String regex1 = "[Employee].?";

String str1 = "Employeee";

String regex2 = "[Employee].\*";

String str2 = "Employeee";

String regex3 = "[P-Z]{8,10}";

String str3 = "QRSTUVWX";

String regex4 = "[Employee]+";

String str4 = "Employeeeeee";

System.out.println(str1.matches(regex1));

System.out.println(str2.matches(regex2));

System.out.println(str3.matches(regex3));

System.out.println(str4.matches(regex4));

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

String regex1 = "[\\w]+ [\\w]+ [\\d]+";

String str1 = "Java Training 123";

String regex2 = "[\\w]+ [\\t] [\\w]+";

String str2 = "Java Training";

String regex3 = "[\\w]+ [^\\w]+ [\\d]+";

String str3 = "Java Training 123";

String regex4 = "[\\D]+ [\\W]+";

String str4 = "Java \*";

System.out.println(str1.matches(regex1));

System.out.println(str2.matches(regex2));

System.out.println(str3.matches(regex3));

System.out.println(str4.matches(regex4));

}

}

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

class Tester {

public static boolean isValidPassword(String password) {

boolean flag = false;

String regex = "[A-Za-z]{8,10}[0-9]{4}";

if (password.matches(regex)) {

flag = true;

}

return flag;

}

public static void main(String[] args) {

String password = "gftdrstr8645";

System.out.println("The password of the customer is " + password);

if (isValidPassword(password))

System.out.println("The password is valid!");

else

System.out.println("The password is not valid!");

}

}

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

class Tester {

public static boolean isValidEmail(String email) {

boolean flag = false;

String regex = "[a-zA-Z0-9]+@[a-zA-Z]{3,}+\\.(com|in)";

if (email.matches(regex)) {

flag = true;

}

return flag;

}

public static void main(String[] args) {

String email = "steve123@gmail.com";

System.out.println("The Email Id of the customer is " + email);

if (isValidEmail(email))

System.out.println("The Email Id is valid!");

else

System.out.println("The Email Id is invalid!");

}

}

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**OOPS PILLARS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ABSTRRACTION | INHERITANCE | ENCAPSULATION | POLYMORPHISM | *MULTI-*  *THREADING* |

***Q What’s the difference between Scripting and Programming Languages?***

Basically, all scripting languages are programming languages. The theoretical difference between the two is that scripting languages do not require the compilation step and are rather interpreted. For example, normally, a C program needs to be compiled before running whereas normally, a scripting language like JavaScript or PHP need not be compiled.

Generally, compiled programs run faster than interpreted programs because they are first converted native machine code. Also, compilers read and analyze the code only once, and report the errors collectively that the code might have, but the interpreter will read and analyze the code statements each time it meets them and halts at that very instance if there is some error. In practice, the distinction between the two is getting blurred owing to improved computation capabilities of the modern hardware and advanced coding practices.

Another point to be noted is that while classifying a language as a scripting language or programming language, the environment on which it would execute must be taken into consideration. The reason why this is important is that we can design an interpreter for C language and use it as a scripting language, and at the same time, we can design a compiler for JavaScript and use it as a non-scripting(compiled language). A live example of this is V8, the JavaScript engine of Google Chrome, which compiles the JavaScript code into machine code, rather than interpreting it.

Some scripting languages traditionally used without an explicit compilation step are JavaScript, PHP, Python, VBScript.

Some programming languages traditionally used with an explicit compilation step are C, C++.

### **Applications of Scripting Languages :**

1. To automate certain tasks in a program

2. Extracting information from a data set

3. Less code-intensive as compared to traditional programming languages

### **Applications of Programming Languages :**

1. They typically run inside a parent program like scripts

2. More compatible while integrating code with mathematical models

3. Languages like JAVA can be compiled and then used on any platform

<https://www.geeksforgeeks.org/whats-the-difference-between-scripting-and-programming-languages/>

MEMORY VIEW

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  | **GC** |
|  | **STACK** |  | **STATIC** |  | **HEAP** |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  | Zomato -> |  |  |  |  |
|  |  |  | Counter=1018 |  | gf5454 |  |  |
|  |  |  |  |  | res | Shimla |  |
|  |  |  |  |  | food | salad |  |
|  |  |  |  |  | bill | 1018 |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  | df452 |  |  |
|  |  |  |  |  | res | Dominos |  |
|  | main() |  |  |  | food | Pizza |  |
|  |  |  |  |  | bill | 1017 |  |
|  | Motu -> ab125 |  |  |  |  |  |  |
|  | Kartikey > df452 |  |  |  |  |  |  |
|  | Sumu -> gf5454 |  | main() |  |  |  |  |
|  |  |  |  |  | ab125 | Zomato |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  | res | santa |  |
|  |  |  |  |  | food | cbm |  |
|  |  |  |  |  | bill | 1016 |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

UML Diagram

IPLTeams \_\_\_\_\_\_\_> Basic \_\_\_\_\_\_\_\_> Dev \_\_\_\_\_\_\_> Object class

*Singleton Class and Private Constructor*

https://www.geeksforgeeks.org/private-constructors-and-singleton-classes-in-java/

Some handy sites for programmers

***Important References for later use***

1. Interface:

[https://www.studytonight.com/java/java-interface.php#](https://www.studytonight.com/java/java-interface.php)

1. Basic Programs

[https://www.javatpoint.com/programs-list#matrix](https://www.javatpoint.com/programs-list" \l "matrix)

1. Operators in JAVA

<https://www.geeksforgeeks.org/operators-in-java/>

#### JAVA Reflection API

Normaly accoding to JAVA convention, we cannot call a private method from outside the class. But there is a differnt way for this. We have reflection API for this.

Reflection API is a part of Java in which we can know about

* the behavior of a class
* fields of a class
* if you want to know the class file is an interface or a class

Its especially used for testing.

**Student.java**

**package** com.advanced.reflection;

**class** Student

{

**public** String name;

**public** **int** age;

**private** **void** display() {

System.*out*.println("Inside Display Method --> name: "+**this**.name+"\t AGE: "+**this**.age);

}

}

**ReflectionDemo.java**

**package** com.advanced.reflection;

**import** java.lang.reflect.Field;

**import** java.lang.reflect.InvocationTargetException;

**import** java.lang.reflect.Method;

/\*\*

\*

\* Getting and Setting Field Values.

\*

\*/

**public** **class** ReflectionDemo {

**public** **static** **void** main(String[] args) **throws** ClassNotFoundException {

Class student = Class.*forName*("com.advanced.reflection.Student");

**try** {

Student s1 = (Student) student.~~newInstance~~();

/\*You could use getConstrouctor() , although note it will only return public

constructors. Also, note that newInstance() isn't deprecated in Java 8

(although using it still has all the problems that made the JDK developers

deprecate it in 9\*/

Method declaredDisplayMethod = student.getDeclaredMethod("display", **null**);// fetching the specific method

// Still the method is unavailable as its private

declaredDisplayMethod.setAccessible(**true**);// to make it accessible

declaredDisplayMethod.invoke(s1, **null**);// invokes the method p1:object ref ; p2:method parameters

} **catch** (InstantiationException | IllegalAccessException | NoSuchMethodException | SecurityException| IllegalArgumentException | InvocationTargetException e1) {

// **TODO** Auto-generated catch block

e1.printStackTrace();

}

**try** {

Student s1 = (Student) student.getDeclaredConstructor().newInstance();

Method declaredDisplayMethod = student.getDeclaredMethod("display", **null**);// fetching the specific method

// Still the method is unavailable as its private

declaredDisplayMethod.setAccessible(**true**);// to make it accessible

declaredDisplayMethod.invoke(s1, **null**);// invokes the method p1:object ref ; p2:method parameters

} **catch** (InstantiationException | IllegalAccessException | IllegalArgumentException | InvocationTargetException| NoSuchMethodException | SecurityException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

**void** extra() {

{

**try** {

Class<Student> classObj = Student.**class**;

/\*Returns:the Field object of this class specified byname\*/

Field ageField = classObj.getField("age");

System.*out*.println("Field Name = " + ageField.getName());// age

System.*out*.println("Field Type = " + ageField.getType());// int

Student student = classObj.~~newInstance~~();

/\* Sets the field represented by this Field object on the specified object

argument to the specified new value. \*/

ageField.set(student, 23);

Object ageValue = ageField.get(student);

System.*out*.println("age = " + ageValue);

}

**catch** (NoSuchFieldException | SecurityException | IllegalArgumentException | IllegalAccessException| InstantiationException e) {

e.printStackTrace();

}

}

}

}