Part-1 : Introduction to JAVA

* Introduction and History of Java
* Java characteristics / Features / Buzzwords
* Java language program development and Execution model.
* JDK installation and Eclipse IDE setup
* Environments Path setup
* Directory structure of JDK/JRE
* Writting, compilation and Execution of a Java Program
* Comments
* Java Coding Std.

Part - 1

Introduction to JAVA

Introduction and History

- Java is a simple, High level, Platform independent, Architecture-Neutral, Secure, Robust, Multithreading, distributed and object-oriented programming language.
- developed by Sun microsystems and it was released in 1995.
- James Gosling initially developed Java in Sun Microsystems (which was later merged with Oracle corporation).
- Initial version of Java is called as OAK
- OAK was renamed as Java in the year 1996
- Java is a WORA (write once, run anywhere) programming language.

↓ Stable release is Java SE 16.0.2 (20 July 2021)

Types of Java Application

(1) web application : serverside web applications [JSE]
((Need Internet)) Ex: servelet, JSP, Stouts, JSF ... etc.

(2) Standalone Application : desktop application or windows based application. [JSE] Java std. Edition
Ex: media player, antivirus...

(3) Enterprise Application : distributed applications
Ex: Banking application, CRM, ERP
JEE : Java Enterprise Edition

(4) Mobile Application: Software based on Java Technologies on mobile device.
(JAVA ME) : Java Micro Edition.
- Google Android application ⇒ Java Programming.

Java characteristics / Features / Buzzwords

(1) Simple (2) Secure (3) Platform Independent
(4) Architecture-Neutral (5) Robust
(6) Multi Threading (7) Distributed (8) Object-oriented.

* Simple :⇒ Java is easy to learn because most of the complex or confusing features of C & C++ like Pointers, operator overloading etc are not provided in java. It has most Built in methods to perform common operations (destructor). data structure, sort ... search.

* Secure :⇒
- Java program run within the JVM, which protects from unauthorized or illegal access to system resource
- Provides wide range of protection from viruses and malicous programs. [File download]
- Access specifics, through which we can control data access outside the restricted Area.

* Platform Independent :⇒ [[WORA]]
Java program can be executed (Byte code) on any kind of machine or operating system.
i.e when we say Java is platform independent it means that the Byte code is independent of Any platforms.

④ Architecture-Neutral :→
Java application can run on any kind of CPU, so it's architecture-neutral.

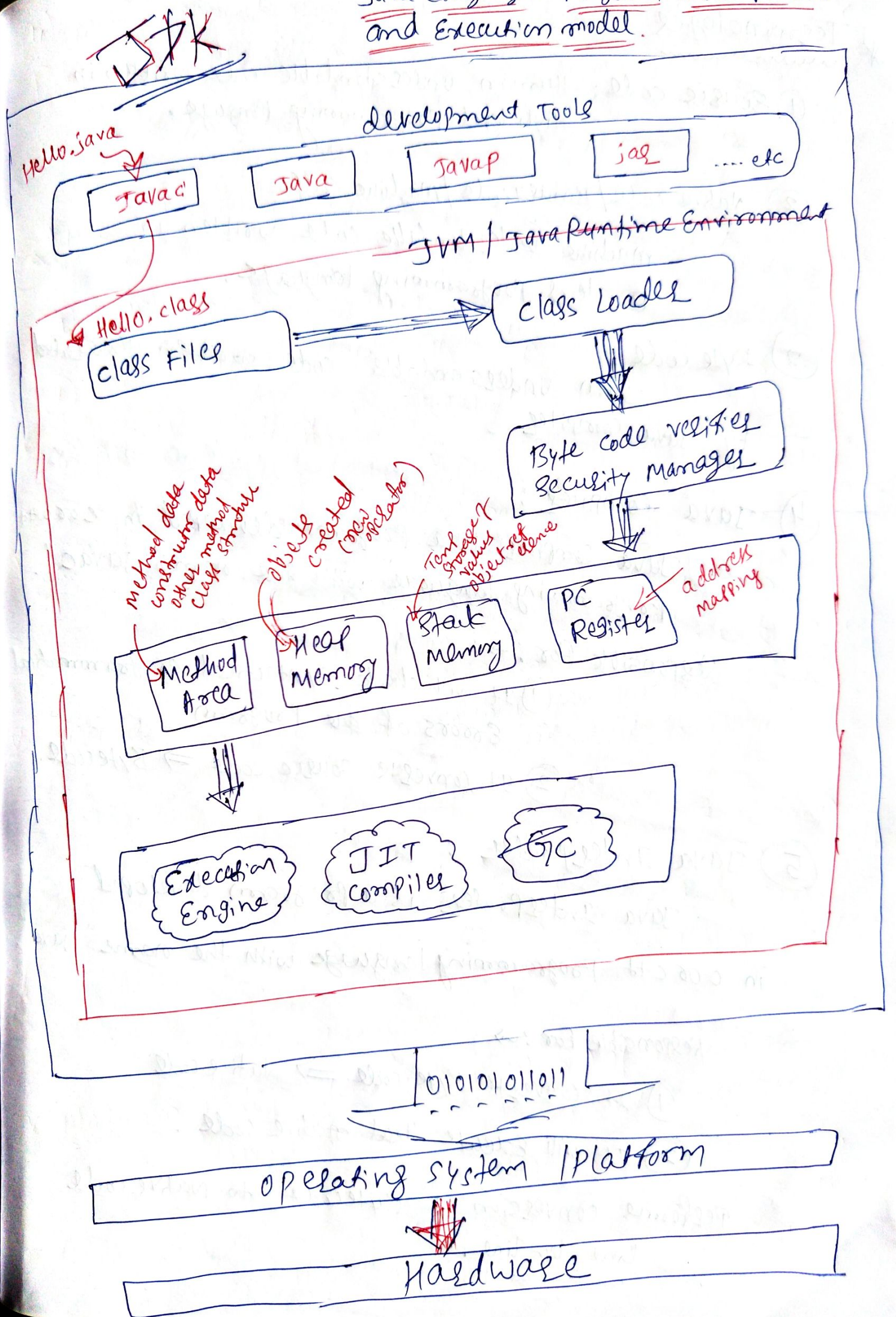
⑤ Robust :→
Java is Robust Because,
- Strong memory management
- No pointers.
- Exception Handling
- Type checking mechanism
- platform Independent.
- during development of Program mistakes can be identified.

⑥ Multi Threading :→
Multi Threading allows us to write a Program that do many things simultaneously.
Ex: VLC player while playing video you can change volume, settings .... etc.

⑦ Distributed :→ supports TCP/IP
Using RMI and EJB we can create distributed applications in Java (Banking application).

⑧ Object-Oriented :→
Java supports features of object-oriented Programming. It's object model is simple and easy to expand.

Java Language Program development and Execution model.



* Terminologies

(1) Source code: Human understandable code written in High level programming language.

(2) Native code / Binary code / Machine code:
machine understandable code written in low level programming language.

(3) Byte code:
JVM understandable code ~~written~~ generated by Java compiler

(4) Java compiler:
Java compiler is a program developed in C or C++ programming language with the name "Javac".

Responsible for :→
(1) It will check syntactical or grammatical Errors of the Program
(2) It converts source code ⇒ Bytecode.

(5) Java Interpreter :
Java Interpreter is a program developed in C or C++ Programming language with the name "java".

Responsible for :→
(1) It converts Byte code ⇒ Native code
(2) It will Execute that native code.

Performs conversion of Bytecode to Native code Line-by-line.

(6) JIT (Just-In-Time) compiler:

it's a component of JVM compiles or translates or converts the neccessary part of the Bytecode into machine code instead of converting line by line.

(7) JDK (Java development kit) and JRE (Java Runtime Environment)

JDK Is a set of various utility programs which are required for developing and executing the java programs.

- platform dependent
- JDK's are for different OS

utility programs provided under JDK:

1. Java development tools
   Ex: javac, java, javap, jar ... etc
2. source files
3. JRE
4. Java doc ... etc

(8) JRE:

- JRE is a implementation of JVM, it contains
  1. Libraries
  2. Interpreter
  3. JIT compiler
- only JRE is enough to run the Java program But to develop and run for both JDK need.

⑨ JVM (Java virtual machine)

- JVM is a specifications provided by Sun whose implementation provides the Environment to run our java application.
- JVM implementations are called JRE

QA Practice:

① A java (.class File) Program is platform Independent
Answer ⇒ True

② Java compiler is platform independent ⇒ False

③ Java Interpreter is platform dependent ⇒ True

④ JVM is platform dependent ⇒ True

⑤ JRE is platform dependent ⇒ True

⑥ class loader ?

⑦ C, C++, Java which is faster ?

* JDK installation and Eclipse IDE setup.

Note: Refer the screenshot provided in PDF.
Eclipse Shortcut PDF.

* Environment Path setup

Note: Refer the screenshot provided in PDF.

* Directory structure of JDK/JRE

Note: Explore the installed Folder on Hard disk for JDK/JRE.

Writting, compilation and Execution of a Java program.

1. Command Prompt
2. Eclipse IDE

HelloWoord.java ↵

```
class HelloWoord {
    public static void main(String[] args) {
        System.out.println("Hello, world!");
    }
}
```

Save as HelloWorld.java

Output: ↵

Hello, world!

* Command Prompt

1. change the dir to cursed file location (HelloWoord.Java)

Ex: `cd D:\myworkspace` ~~HelloWorld~~

2. compilation

javac <sourceFilename.java>

Ex: `javac HelloWorld.java` ↵

3. run / Execute

`java HelloWorld` ↵

* Comments

(1) single line comments (//)

(2) Multi line comments (/* */)

- single line:

```
// The hole line will be ignored by the compiler
```

- Multiline comments:

```
/* you can define
   as many line as
   you want to
   Ignore form
   compilation.
*/
```

* Java coding Standards

Whenever we are writing the code we should follow some coding conversions and are called coding std. in java.

(1) coding standards for classes

Usually class names are nouns, should start with uppercase & if it contains multiple words every inner word should starts with uppercase letter.

Ex: Student, Employee, SbiLoan, InvestmentBanking, String, CurrentAccount, StringBuffer.

(2) coding standards for Interfaces:

usually interface names are Adjectives, should starts with uppercase & if it contains multiple words every inner word should starts with uppercase letter.

Ex: Runnable, Serializable, Cloneable
WebDriver, WebElement

(3) coding standards for Methods:

- verbs or verb noun combination
- starts with lowercase and if contains multiple words every inner words should starts with uppercase letter (camelcase).

Ex: run(); sleep(); ← verbs
getName(); setSalary(); ← verb + noun

(4) coding standards for ~~variables~~ Methods:
Variables

- variable names are Nouns
- starts with lowercase char & if it ~~is~~ contains multiple words, every inner word should starts with uppercase character. (camelcase)

Ex: name; mobileNumber;
rollNumber; accountNumber;

firstName;
LastName;

(5) Coding Standards for Constants

- Constants are noun
- should only uppercase char if they contain multipal words are seprated with "_" underscore Symbol.

Ex:
```
MAX_VALUE
MIN_VALUE
MAX_PRIORITY
MIN_PRIORITY
```

END OF PART → (1)