Agenda

- Module Introduction
- Java history
- Java Versions
- Java Platforms
- Java Installation
- Object Oriented
    - class and object in java
- Hello World
- C/C++ vs Java Compilation and Execution
- JDK vs JRE vs JVM
- Hello World on STS
- Java Buzzwords

Module Introduction

- Total -> 100 marks
    - 40 Marks for CCEE
    - 40 Marks for Lab Exam
    - 20 Marks for Internals
- 18 Days
- Java 8 features, Java 11

Java history

- In 1991 , the team from Sun Microsystems led by James Gosling and patrick Naughtan decided to cretate thier own language which will be capable of executing on smaller electronic devices with very low memory and low power
- they called this project as green
- while developing they have to think about that the language should be capable of executing on multiple hardware/architectures
- this team came from a background of UNIX, so they decided to keep their base language as C++
- James initially called this language as OAK, however the language with OAK already existed hence it was further changed to JAVA
- The team designed their first product called as *7 (Smart remote)
- Nor the Sun and the cunsumer electronics company were intrested in using this.
- so they decided to come up with better solution for marketing as well as a better product
- They spent almost next 2 years on reworking on that language.
- Meanwhile in WWW (World wide web) was getting popular
- the key for the www was the browser that was capable of executing hypertext pages to the screen
- the team introduced their brower called as HotJava Browser
- this browser was capabale of running Java code inside the browser which was called as Appletes
- It helped to aceive dynamic behaviour for the pages.

Java Versions

- Java Beta - 1995
- JDK 1.0 - 1996
- JDK 1.1 - 1997
- J2SE 1.2 - 1998
    - Collections
- J2SE 5.0 - 2004
    - enums
    - Generics
    - Annotations
- Java SE 8 (LTS) - 2014
    - Functional Progamming
    - Lambda Expressions
- Java SE 11 (LTS) - 2018
- Java SE 17 (LTS) - 2021

Java Platforms

- Java works on multiple devices
- 1. Java Card
    - We can use the java for very small devices with low memory and low power, for eg smart cards also called as java cards
- 2. Java ME (Micro Edition)
    - It used for developing java appplications for smaller devices with low power and with small screens.
- 3. Java SE (Standard Edition)
    - It is a platform that is used for developing applications for desktop machines
- 4. Java EE (Enterprise Edition)
    - It is used to develop web Applications

Java Installation

- JDK-11
    - sudo apt install openjdk-11-jdk
- STS - 3.9.18 (Download STS 3 for Linux from below link ->)
    - https://download.springsource.com/release/STS/3.9.18.RELEASE/dist/e4.21/spring-tool-suite-3.9.18.RELEASE-e4.21.0-linux-gtk-x86_64.tar.gz
- JDK

Documentation

- Java 8 docs
    - https://docs.oracle.com/javase/8/docs/api/index.html
- Java 11 docs

    - https://docs.oracle.com/en/java/javase/11/docs/api/index.html

JDK

- path -> /usr/lib/jvm/java-11-openjdk-amd64
    - bin -> Tools for java developement (javac, java, jar,jdb,etc)
    - include -> It consists of libraries for native methods
    - docs -> Documentation in HTML format for libraries
    - lib -> Java Core libraries (rt.jar)
    - src -> source code for java

class

- class is logical enity
- class is also called as blueprint of an object
- class consists of
    - 1. Fields (Data Members)
    - 2. Methods (Member Functions)
        - static methods
            - these methods are designed to call on classname directly without creating the object
        - non static methods
            - these are the methods that are designed to call on object

object

- It is a physical entity
- It is also called as instance of a class
- We can cretae multiple objects of a single class

Hello World (Program.java)

- Create a file called as Program.java
- Inside this write a class called as Program
- define main method inside the class

```
class Program {

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

- to compile and execute use below commands

```
// for comppilation
javac Program.java

// for execution
java Program
```

Explaination - main()

- public static void main(String args[]){}
- JVM invokes the main method
- JVM calls the main method without cretaing the object of the class, hence it is made as static
- main method does not returnn any thing to the JVM and hence its return type is void
- main methods takes command line arguments and so the array of string is passed as a parameter
- main method should be accessiabble outside the class and hence public

Explaination - System.out.println(" ");

- System
    - It is a is final class declared inside java.lang package
- out
    - It is an static field declared inside the System class
    - It is an object of PrintStream class
    - Printstream is a class declared inside java.io
- println()
    - It is a method declared inside the PrintStream class
    - it is a nonstatic method

Compiled and executed the code from src and bin (RectangleArea)

- create a directory RectangleArea
- inside it create two sub directories src and bin
- inside src cretae a file called as Rectangngle.java
- Inside the .java file define a class Rectangle with a main method
- use the below steps for compilation and execution

```
// open the terminal from the src directory and give the below command
javac -d ../bin Rectangle.java

// To execute we have to set the classpath. use the below command for the
same
export CLASSPATH=../bin

// To execute
java Rectangle
```

CLASSPATH

- CLASSPATH is a java environment variable that stores multiple path of the directories in which .class files are present seperated by :
- to set CLASSPATH
    - export CLASSPATH=../bin

C/C++ vs Java Compilation and Execution

- C/CPP
    - Main.cpp -> compiled -> Main.obj -> Linker -> a.out
- JAVA
    - Main.java -> compiled -> Main.class -> JVM -> execute

JDK vs JRE vs JVM

- Software Development
    - SDK - Software Development Kit
    - SDK = tools + docs + libraries + runtime environment + ide
- Java Application Development
    - JDK - Java Development Kit
    - JDK = tools + docs + libraries + JRE (Java RunTime Environment)
        - JRE = rt.jar + JVM (Java Virtual Machine)

Hello World on STS (HelloWorld)

- Select the proper workspace
- Every day create a new folder Day01, Day02,... and select it as workspace
- Select the workspace and first we have to switch the perspective to java
- click on file -> new -> Java Project
- Give a proper name to the Project (HelloWorld)
- Select the java version as 8 (1.8) , click on finish
- right click on src -> New ->java class
- give proper name to the class(Program), select the checkbox for main -> ok
- inside Program.java in the main method write the code and execute.

Java Buzzwords

- 1. Simple
    - java is simple till java 1.2 then as the library expanded it is simple for the professional Developers to use.
    - as multiple libraries are introduced it made the development simpel however it is bit complex to understand.

    - java is simple because it removed the complex pointer from cpp, it also removed th unnecessary and rarely used concepts of operator overloading from cpp
- 2. Distributed
    - Java can work on distributed networks over the internet.
    - It can work with the java objects over the internet in the same way as that on local machine
    - that is why multiple developers over the distributed network can work on the single project
- 3. Compiled and Interpreted
    - java is compiled as well as interpreted language because first compilation is done to create .class file and then the JVM interpretes this .class file which has bytecodes and convert it to machine understandable code.
- 4. Architecture Neutral
    - Java can run on multiple architectures seemlesly.
- 5. Portable
    - WORA -> Write Once Run Anywhere
    - Java follows this WORA seriously
    - It is possible because of JVM
- 6. Robust
    - Java is robust because of automatic memory management which is possible witn the help of Garbage Collector.
- 7. Dynamic
    - Java can handle dynamic inclusions of libararies and objects into the code.
    - It supports rtti.
- 8. Secure
    - It is secure as the objects that are accessed are from the JVM memory and not from the actual memories from the computer.
- 9. MultiThreaded
    - It supports multiple process/task creatin through one application
    - Multithreading is supported in java
- 10. High Performance
    - Java performance is high because of the use of bytecode.
    - The bytecode was used so that it can be efficiently translated into native machine code by JIT compiler (in JVM).
- 11. OOP
    - Java is an OOP langauge, as it follows all the major pillars of OOP

LabWork

- Complete the installation and Hello world program through cmd line and STS
- OOPS
- static functions
- class and objects