Agenda

- OOP Concepts
- Introduction
- Installation
- Hello world Program
- Execution Flow
- main() variations
- ~~Console input and output~~

Java History

- In 1991, group of sun engineers led by James Gosling and Patrick Naughton decided to design a language that could run on small devices like remote controls, cable tv boxes.
- Since these devices have very small power and memory the language needs to be small.
- Also differnt manufactures can choose different CPU's the language cannot be bound to single architecture
- this project was named as green.
- these engineers came from UNIX background, so they used c++ as their base.
- James decided to call this language as OAK, however the language with this name was already existing, Hence it was later renamed by James to Java.
- In 1992 they delivered their first product called as "*7"(a smart remote control)
- Unfortunately Sun Miccrosystem was not intrested in producing this, also nor the consumer electronic companies were intrested in it.
- The team then deciced to market their technology in some other way where they worked for next 1 and half year on it.
- Meanwhile world wide web (www) was growing bigger.
- the key to it was browser transalating hyper text pages to the screen.
- the java developers developed a browser called as HotJava browser which was based on client server architeture and was working in real time.
- the developers made the browser capable of executing java code inside the web pages called as Applets.

Java Versions

- JDK Beta - 1995
- JDK 1.0 - January 23, 1996
- JDK 1.1 - February 19, 1997
- J2SE 1.2 - December 8, 1998
    - Java collections
- J2SE 1.3 - May 8, 2000
- J2SE 1.4 - February 6, 2002
- J2SE 5.0 - September 30, 2004
    - enum
    - Generics
    - Annotations
- Java SE 6 - December 11, 2006

- Java SE 7 - July 28, 2011
- Java SE 8 (LTS) - March 18, 2014
    - Functional programming: Streams, Lambda expressions
- Java SE 9 - September 21, 2017
- Java SE 10 - March 20, 2018
- Java SE 11 (LTS) - September 25, 2018
- Java SE 12 - March 19, 2019
- Java SE 13 - September 17, 2019
- Java SE 14 - March 17, 2020
- Java SE 15 - September 15, 2020
- Java SE 16 - March 16, 2021
- Java SE 17 (LTS) - September 14, 2021
- Java SE 18 - March 22, 2022
- Java SE 19 - September 20, 2022
- Java SE 20 - March 21, 2023

Java Platforms

- Java is not specific to any processor or operating system as it is implemented for wide variety of hardware and operating system
- 1. Java Card
    - used to run java based applications on small devices with small memory devices like smart cards
- 2. Java ME(Micro Edition)
    - used to develop applications for small devices with less memory, display and power capacities like mobiles, printers
- 3. Java SE(Standard Edition)
    - It is widely used for development of portable code for desktop environment
- 4. Java EE(Enterprise Edition)
    - It is widely used in development of enterprise applications/softwares. -also used for web application development

Java Installation

- Windows and Mac:
- Download .msi/.dmg file and follow installation steps.

```
https://adoptium.net/temurin/releases?version=17&os=any&arch=any
```

JDK vs JRE vs JVM

- SDK -> Software Development Kit
- SDK = Software Development Tools + Libraries + Runtime environment + Documentation + IDE
    - Software Development Tools = Compiler, Debugger, etc.
    - Libraries = Set of functions/classes.
- JDK -> Java Development kit
    - used for developing Java applications.

- JDK = Java Development tools + Java docs + JRE
- JRE = Java API(Java class libraries) (rt.jar replaced by jmods in java9) + Java Virtual Machine
    - till java 8
        - JRE = rt.jar + JVM
    - from java 9
        - JRE = jmods + libraries + JVM

IntelliJ

```
https://www.jetbrains.com/idea/#
```

Documentation and tutorial Link

```
- Java SE 8 Document Link
https://docs.oracle.com/javase/8/docs/api/index.html

- Java SE 11 Document Link
https://docs.oracle.com/javase/11/docs/api/index.html

- Oracle Java Tutorial
https://docs.oracle.com/javase/tutorial/
```

HelloWorld

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

main()

- In java every variable/function should be class (Encapsulation)
- JVM calls main method without creating object of the class, so main method should be static
- main does not return anythinig to JVM so it is void.
- main takes command line arguments and hence String args[]
- main should be accessiable outside the class directly and hence public.
- JVM invokes main method.
- Can be overloaded.
- Can write one entry-point in each Java class.

System.out.println()

- System is predefined Java class (java.lang.System).
- out is public static field of the System class --> System.out.
- out is object of PrintStream class (java.io.PrintStream).
- println() is public non-static method of PrintStream class

Compilation & Execution

In same directory

```
javac Program.java
java Program
```

In src and bin directory

- create a directory called as RectangleArea
- add two subdirectories src and bin inside it.
- in src add Rectangle.java file
- from the src directory open the terminal.

```
javac -d ../bin Rectangle.java

//For Windows
set CLASSPATH=..\bin

// For Linux
export CLASSPATH=../bin

java Rectangle
```

PATH vs CLASSPATH

- Enviorment variables: Contains important information about the system e.g. OS, CPU, PATH, USER, etc.

PATH:

- Contains set of directories separated by ; (Windows) or : (Linux).
- When any program (executable file) is executed without its full path (on terminal/Run), then OS search it in all directories given in PATH variable.

```
terminal> mspaint.exe
terminal> notepad.exe
terminal> taskmgr.exe
terminal> java.exe -version
terminal> javac.exe -version
```

- To display PATH variable

```
Windows cmd> set PATH
Linux terminal> echo $PATH
```

- PATH variable can be modified using "set" command (Windows) or "export" command (Linux).
- PATH variable can be modified permanently in Windows System settings or Linux ~/.bashrc.

CLASSPATH:

- Contains set of directories separated by ; (Windows) or : (Linux).
- Java's environment variable by which one can inform Java compiler, application launcher, JVM and other Java tools about the directories in which Java classes/packages are kept.
- CLASSPATH variable can be modified using "set" command (Windows) or "export" command (Linux).

```
Windows cmd> set CLASSPATH=\path\to\set;%CLASSPATH%
Linux terminal> export CLASSPATH=/path/to/set:$CLASSPATH
```

- It is a JAVA environment variable which holds all directories seprated by ;(Windows) :(Linux)
- It informs java compiler, application launcher, JVM, and other java tools about the directories in which classes/packages are kept(location of the class files)
- To display CLASSPATH variable

```
Windows cmd> set CLASSPATH
Linux terminal> echo $CLASSPATH
```

Bytecode

- Bytecode is an intermediate representation of a program that is generated by a compiler and typically executed by a virtual machine.
- In the context of Java programming, bytecode refers specifically to the binary format that Java source code is compiled into.
- It enables platform independence, portability, security, and potential performance optimizations in Java programming.
- It forms a crucial part of the Java platform's architecture, allowing Java programs to run on a wide range of devices and operating systems.

.class File

- A .class file in Java contains the bytecode instructions that are executed by the Java Virtual Machine (JVM).
- When you compile Java source code using a Java compiler like javac, it translates the source code into bytecode, which is then stored in a .class file.

main() Variations

- In STS .class files are placed under bin directory after auto compilation
- one java project can have multiple .java files.
- each java file can have its own main method which can be executed seperately
- the main() must be public static void main otherwise we get an error.
- the entry point method must be be main(String args[]) otherwise error main not found
- The main() method can be overloaded i.e. method with same name but different parameters (in same class).
- If a .java file contains multiple classes, for each class a separate .class file is created
- Name of (non-public) Java class may be different than the file name.
- The name of generated .class file is same as class name.