Week1

Day1 – Intro to Team, Zoom, Installing Software & Tools

Day 2 – Unix OS, Commands, Git

Day 3 – Adv Unix Commands, File Processing, SDLC, Waterfall VS Agile, JDK, JRE, JVM, Java Intro

Day 4 – Installing, Setting the Path, Adding to Environment Variables, Setting up IDE & Workspace

Day 5 – Core Java Concepts

OS Fundamentals – Operating System (It’s a Type of System Software)

I Gen (1940-56) [Vacuum tubes]

II Gen (1956-63) [Transistors, Capacitors, Resisters]

III Gen (1964-71) [IC – Integrated Circuits] [Reduced the size ]

IV Gen (1976-99) VLSI – Very Large scale integration – Processors (Micro-processors)

V Gen (2000-till date) – Multi core Processor

OS – DOS, Unix (command based OS) – CUI [Character User Interface]

GUI OS – Windows, Linux, Mac [ Graphical User Interface]

Mobile base OS – [iOS, Android, Mobile OS etc.,]

Types of Memory

1. WRT material [ a) Magnetic b) optical c) semi-conductor based ]
   1. Magnetic [Hard disk, floppy disk etc.,
   2. Optical [CD, DVD, BRD]
   3. Semi-conductor based [Cache Memory/Internal Memory]
2. WRT usage Type
   1. RAM [Random Access Memory / Volatile]
   2. ROM [Read Only Memory / Non-Volatile] [ EROM, PROM, EPROM, EEPROM]

EROM – Erasable ROM

PROM – Programmable ROM

EPROM – Erasable & Programmable ROM

EEPROM – Electrically Erasable & Programmable ROM

Unix Commands

Mkdir, cd, rm, cp, ls, echo, date, time, cal, users, whoami, touch, cat

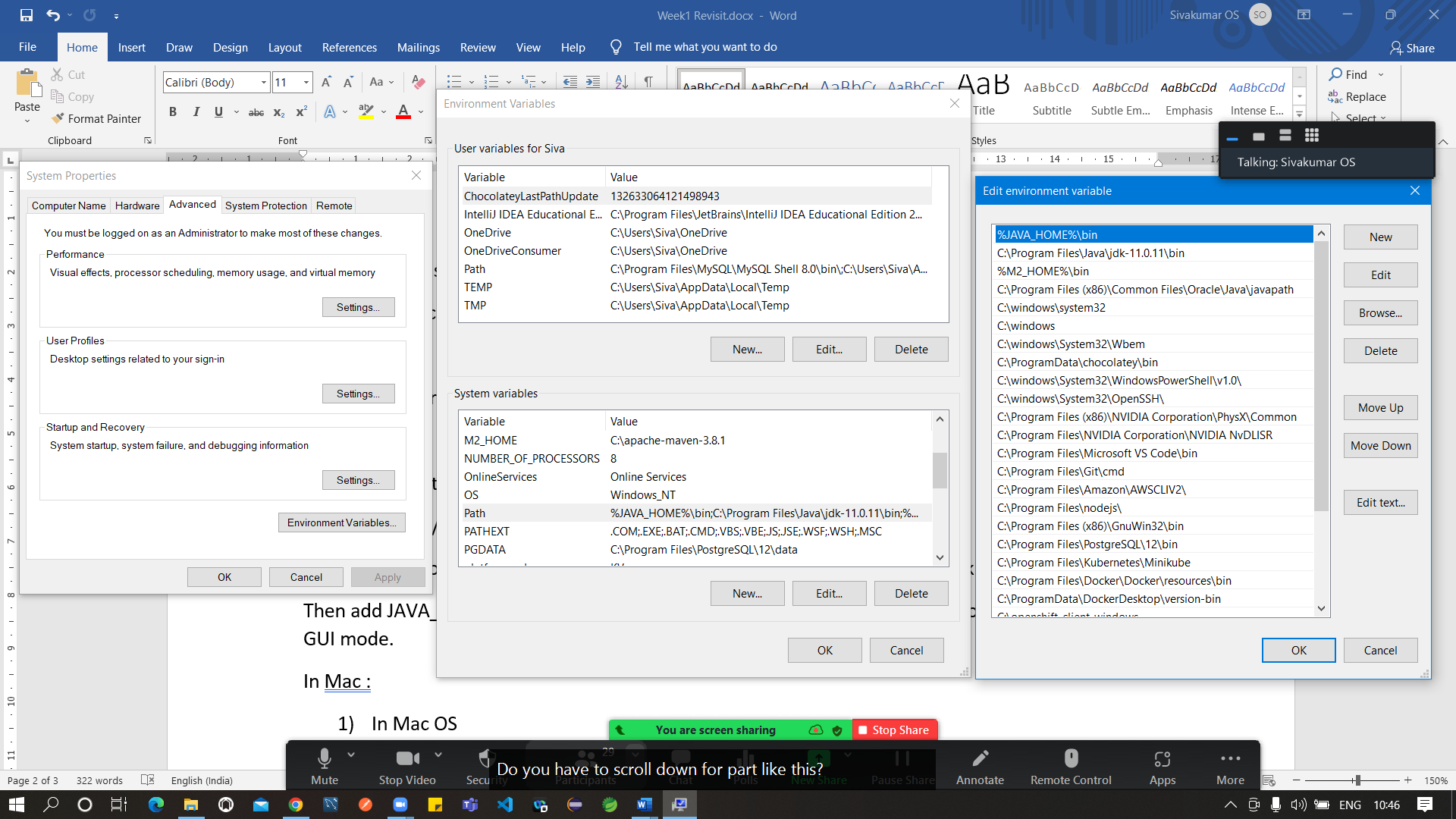
Setting up Environment Variables [ User Level & System Level Env Var]

Adding Java to the Environment Variable.

In windows : C:/Program Files or Program Files (x86)/ Java/Jdkx.x.xxx

Add New Environment Variable called JAVA\_HOME = C:\Program Files\Java\jdk1.8.0\_281

Then add JAVA\_HOME to Path path=%path%;JAVA\_HOME\bin; (CUI Method) or you can edit it in GUI mode.



In Mac :

1. In Mac OS
2. Execute the following command on Mac Terminal:

/usr/libexec/java\_home -V

1. It will show you all the versions of JDK installed in your Mac.
2. Now, let's say it shows you paths like as given below:

/Library/Java/JavaVirtualMachines/jdk-14.0.1.jdk/Contents/Home

1. Execute the following command:

cd /Library/Java/JavaVirtualMachines/

ls

Q: Can we install more than one JDK in our Laptop/Desktop?

Ans : Yes, We can.

Q: How to manage different version of Java installed in a system

Ans : Using JAVA\_HOME env variable & path

Package Managers [rpm – Red Hat Package Manager, apt – Adv Packing Tool , yum – Yellow dog Updater & Modifier]

* They manage all the dependencies
* NPM – Node Package Manager
* Used to manage the dependencies while working on particular technology

For Ex: All JS based frameworks NPM is the package manager.

For JAVA : Maven is the package manager

For .Net: NuGet

For JS/Node : NPM

Changing File Permissions in Linux : chmod (r -read (4), w -write (2), x – execute (1) , u – user/owner, g-group, o -other )

Chmod +r+w-x+r-w-x+r-w+x file1.txt

Chmod 745 file1.txt

Ls -al

Ls -l

Ls -ah

Day 3:

JAVA is High Level, Object Oriented, Platform independent, Multi- threaded Programming Lang.

* Highly Secured
* C impressed syntax
* Removed complex challenges of C (pointer, multiple inheritance, memory mgmt.)
* Platform independent [ WORA – Write Once & Run Anywhere]
* All Purpose /General Purpose Programming Lang
* Solaris OS is based on Java
* H2 (in-memory DB – Just 2 mb in size) completely written in Java
* Big – Data, Android, Pega

Object Oriented Programming [Importance will be given to Objects]

Object – Instance of a Class which has unique properties

Object = Property + Methods (State + Behaviour)

Class = It’s a Blue print for Objects .

In Java/ OOP, everything is Object.

JAVA is not a pure oop lang, bcos it contains primitive data type (int, float, char, boolean, byte, double, long)

All the objects will have same property but different values for property

public class Employee {

private int id;

private String name;

private String email;

}

//starter class is a Java class with main method

//Only starter class can be executed

public class Starter {

public static void main (String[] args) {

Employee emp1 = new Employee( 100, “abc”, “[abc@gmail.com](mailto:abc@gmail.com)”);

Employee emp2 = new Employee(101, “xyz”, “[xyz@gmail.com](mailto:xyz@gmail.com)”);

}

}

Note : In Java, the name of the public class should be similar to the name of the file (filename is also case sensitive in Java)

Java is a case sensitive Lang. (Hello!=hello)

JDK – Java Development Kit

JRE – Java RunTime Environment

JVM – Java Virtual Machine

Package naming convention (Reverse of the company URL)

If the URL (Uniform Resource Locator) google.com then the package name should be com.google

Constructor is a special method used for initializing the member variable.

Constructor will have the same name as the Class.

Types of Constructors

1. No Argument / Default constructor (It will be automatically added by JVM if no other constructor is specified)
2. Parameterized Constructor ( Fully [sAll Arguments Constructor]/Partially)

“this” keyword is used to represent the current object

“super” keyword is used to represent the parent object.

By default, the parent class for all the user-defined class is Object.

In Java, A class can extend only one class but it can implement n no of interfaces.

In other words, Java class can have only one parent class but it can be part of multiple interfaces.

If you want to execute the java code, then add main method to that class. Other wise “main” method is not needed.

If a class contains main method, then it falls under starter class category.

Types of variables

1. Instance variables ( Variables declared inside the class and outside any method)
2. Local variables (Variables defined/declared inside any method or passed as an argument to the method)
3. Class variables / Static variable (Variables declared inside the class & outside any method with static keyword) – This variable will be shared by all objects of the class [ Only one instance of the variable will be available]

While creating object with new keyword, calling the constructor is compulsory ( Default/ parameterized constructor)

Java won’t support “Global Variables” & unconditional “goto” statement.

JAVA is platform independent, but JDK, JRE & JVM are not platform independent (They are platform dependent)

Scope of the variable

|  |  |
| --- | --- |
| Local variable | Within the method or block |
| Instance variable | Depends on the access modifier. Accessed using the object name. Multiple instances/ copies of same variable will be created by each object. |
| Class/static variable | Depends on the access modifier. Accessed using class name. Only one instance/copy of that variable will be shared by all the objects. Only one copy/ instance (Singleton variable) |

Employee emp = new Employee();

“emp” is not object. It’s a object reference. Or reference variable.

“emp” is holding the address of Object.

Object will be created only when using new keyword along with it’s constructor.

Objects will be created in heap area. [It’s a memory location in the JVM]

Stack, heap, program counter, string pool – different memory areas in JVM.

Pillers of OOP [A PIE]

1. Abstraction [Maps, Car dashboard] – Hiding the implementation [Protecting the implementation] Showing relevant data and hiding un-wanted data
2. Polymorphism [ Human being]
3. Inheritance [Human Relationships]
4. Encapsulation [ Securing Data – Hiding the Data] – Security Feature

In Java abstraction is achieved with the help of abstract keyword & interfaces

With respect to Java Abstract means partial or in-complete.

Abstract class = In-complete class [ A class with one or abstract method]

Abstract method = A method which is in-complete. Method with no body. Method contains only declaration not the definition.

**package** com.revature;

**public** **abstract** **class** AbstractDemo {

//concrete or complete method or non-abstract

//This method contains declaration & definition [ Signature (head) & body ]

//this is an example of empty method [No statements inside body of the method]

**public** **void** display()

{ // body starts here

} // method body ends here

**public** **abstract** **void** show();

}

Creating Object of abstract class directly is not possible

Polymorphism -- (poly – Many ) Morphing – Videography technique

Trainer Siva =

* For the batch4 associates = Siva is a Trainer
* For my kids = Siva is a father
* For my wife = Siva is a Husband
* For my parents = Siva is a Kid
* For my Friends = Siva is a Friend

In Java, two types of polymorphism

1. Method Over loading ( Static polymorphism / compile time polymorphism)
2. Method Over riding (Dynamic /run-time polymorphism)

Inheritance ( Father – Child, Husband – wife)

In java inheritance is achieved with the help of “extends” & “implements” keyword

“extends” keyword will be used with class

“implements” keyword will be used with interface

Inheritance in Java – Getting properties & behaviours of one class by another class

Types of Inheritance in Java

1. Simple inheritance [Parent – child]
2. Multi -level Inheritance [Grand-parent -> Parent -> Child or Parent ->Child -> Grand Child]

Base class / Parent class / Super Class

Sub Class / Child Class / Derived Class

A extend B ( Class A is getting properties from Class B) A -> B

Public class A extends B {

}

Encapsulation – Hiding the Data [ Class members will be having access modifiers to define the accessibility]

Four access Modifiers in JAVA

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Access modifier name | Within the class | Within the package | Sub-class in other package | Anywhere |
| private | Yes | No | No | No |
| default/ package | Yes | Yes | No | No |
| protected | Yes | Yes | Yes | No |
| public | Yes | Yes | Yes | Yes |

While creating classes, all the properties will be defined with private access modifier and all the methods will be defined with public access modifier.

|  |  |  |
| --- | --- | --- |
| Sl No | Method Overloading | Method Overriding |
| 1 | Within the same class, methods with different signature but same name | Method redefined in the derived class without changing the signature |
| 2 | Static polymorphism | Dynamic polymorphism |
| 3 | Method to execute will be decided during the compilation time | Method to execute will be decided during the runtime only |
| 4 | Signature should be changed | Signature should remain same |

Control Statements

1. Normal statement
2. Conditional Statements (if, else if, nested if, switch case, ternary operator)
3. Looping or Repetitive Statements (To perform same operation multiple times)
   1. Entry Control Loop (for, while) [ 0 or more time]
   2. Exit Control Loop (do while) [1 or more time]

Syntax :

If (Condition) {

// true statements;

} else {

//false statements;

}

Types of Operator : LTR – Left to Right, RTL – Right to Left

1. Arithmetic Operator [ Depending upon no of operands] LTR
   1. Unary Operator [++, --]
   2. Binary Operator [+.-. \*. /, %]
   3. Ternary Operator [?:] [One line if else stmt] (condition)?true stmt:false stmt;
2. Assignment operator (=) RTL
3. Equality Operator (==, !=)
4. Shorthand Assignment (+=, -=, \*=, /=, %=)
5. Conditional Operator (&&, ||, !)
6. Relational Operators (<, >, <=, >=)
7. Bitwise Operators (<<, >>, <<<, >>>, ^, |, &)

Editions of JAVA

1. Standard Edition
2. Enterprise Edition
3. Micro/Mobile Edition

JAVA 8 Features

1. Lambda
2. Functional Interface
3. New IO
4. Concurrency improvements
5. New Date & Time processing classes
6. Streams

Casting – Type Casting

* Is a process of converting one form of primitive data type to another form

Types of Casting

1. Implicit Type Casting (Narrowing)
2. Explicit Type Casting (Widening)

boolean, byte, char, short, int, float, double, long (Left to Right implicit conversion / casting)

(Right to Left – Explicit type casting)

GIT – Is a distributed version control system.

Git can be accessed using CUI (Character User Interface) & GUI (Graphical user Interface)

Git CUI = Git Bash (unix & linux inspired syntax)

GIT GUI = Graphical representation of Git Bash

Important Git Commands

1. Git init -- To initialize empty repo locally
2. Git add <file\_name or .> - Add files/folders for tracking purpose
3. Git commit; -- Move the content from staging to local repo.
4. Git push – To sync with remote repo (Github)
5. Git branch – To change or create a branch

Non-Access Modifiers (Behaviour Modifier)

|  |  |  |  |
| --- | --- | --- | --- |
| Sl No |  |  |  |
| Abstract | Incomplete / Non-concrete | Applied to Class, methods only |  |
| Static | Once per class | Applied to Block, methods, variables |  |
| final | To Create ALL CAPS Constant | Applied to Class, methods, variables | Class will be become non-extendable  Method overriding is not possible |
| transient | To prevent serialize a data | Used when serializing an object |  |