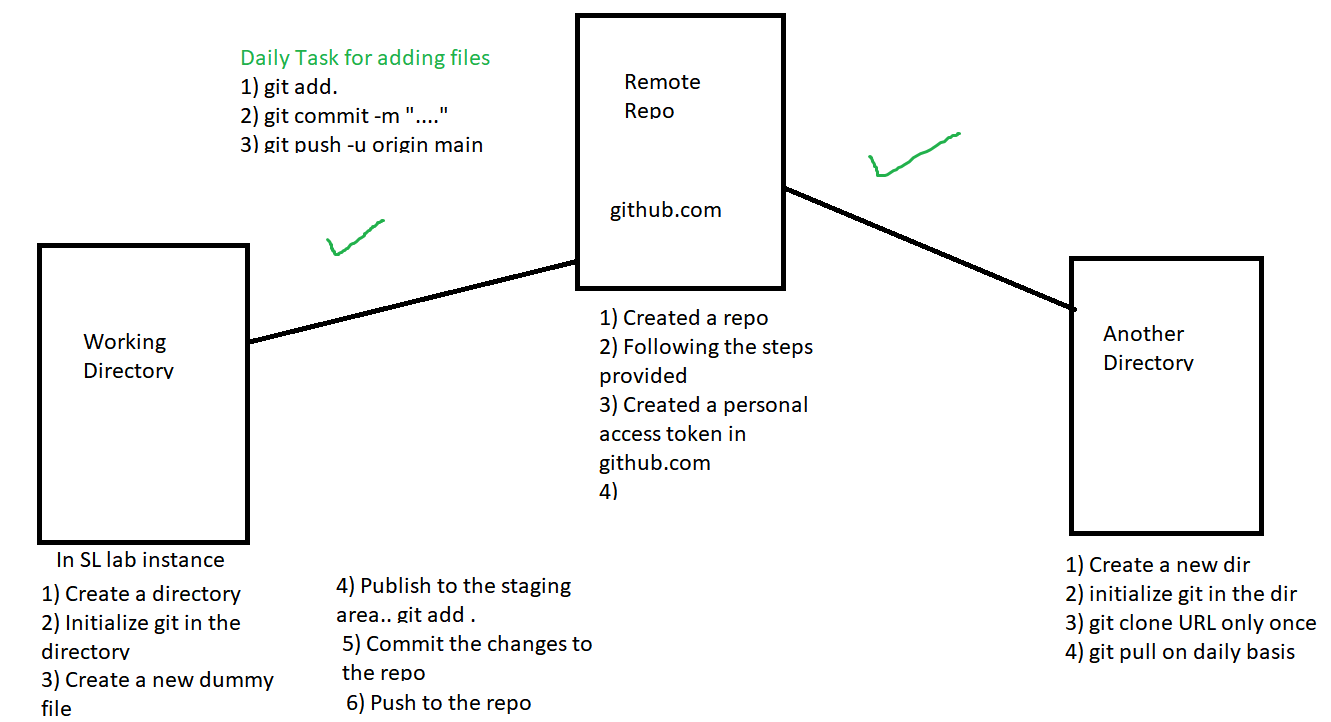
**JAVA**

**Day 1 (24 Oct 2022)**



1) Check if Java is installed (java -version)

2) If not then install using java installer

3) Set up enviromental variables for the java path

4) Write your first java program and compile and run it

/\* Welcome everyone

\* This is my first program in Java

\*/

public class HelloWorld

{

//This is the main method

public static void main(String args[])

{

//This method will print the message

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

}

}

5) Compile it using **javac Filename.java**

6) After compilation you will get a .class file

7) You can run the file using **java Filename**

**Class Syntax**

**public (access modifer) class ClassName**

**{**

**Write methods/functions**

**Create variables**

**}**

**Comments in Java**

Lines which are not compiled or executed by the compiler

1. Single line comment //
2. Multi-line comment /\*\*/

**Variable:** Is a placeholder

**Syntax**

**Declare and initialize the variable**

int a = 10;

DataType VariableName = Value

**Declare a variable**

int a;

**Data Types**

Indicates what kind of data you can store in the variable

2 kind of data types

1. ***Primitive data type***

Store only value

1. Byte 1 byte
2. Short 2 byte
3. Int 4 byte
4. Long 8 byte
5. Float 4 byte
6. Double 8 byte
7. Char 2 byte
8. Boolean 1 byte
9. ***Non-primitive***

Store the value and reference of another data type also

Arrays, Class, Interfaces

**Conditional Statements: used to check conditions**

1. If
2. If else
3. If else if
4. Switch statement (multiple cases - suitable for creating a menu in application)

int a = 10;

If (a==10)

System.out.println(“a is 10”);

Else

System.out.println(“a is not equal to 10)

Switch(variable){

Case a: S.o.p (“You’re in 1st option);

}

**Loop**

To execute some line of code in iteration/multiple times.

3 parts of loop

1. Initialize the looping variable
2. Put condition for the looping variable
3. Increment or decrement looping variable
4. **For loop**

Syntax:

for(int a=0; a<10; a++) //a++ means a = a + 1 //increment

//if decrement I will write a--

for(int a=10; a>=0; a--)

{

S.o.p(“Value of a is”+a);

}

1. **While Loop**

Initialize the value

While (condition)

{

Increment/decrement

}

1. **Do while loop**

do{

Increment/decrement

} while(condition)

**Day 2 (26 oct 2022)**

1. **Methods**

**Method with no return type**

Void is used to return nothing

AccessModifier ReturnType MethodName()

{

//Method body

}

public void MyMessage(){

}

**Method with 1 parameter and call by value**

package com;

public class CallbyValueDemo {

int data = 10; //Instance Variable

//Parameterized Method - it accepts one int value as a parameter

void changevalue(int data)

{

data = data + 10; //Local variable

System.out.println("Value of local data" + data);

}

public static void main(String[] args) {

CallbyValueDemo demo = new CallbyValueDemo(); //Creating object

System.out.println("Before changing" + demo.data); //10

demo.changevalue(50); //Call by value

System.out.println("After changing" + demo.data); //10

}

}

**Method Overloading**

package com;

public class MethodOverloading {

//Method with same name and return type

void add(int a, int b)

{

int sum = a +b;

System.out.println("The value of sum is" + sum);

}

//Method with same name and return type

void add(int a, int b, int c)

{

int sum = a +b+c;

System.out.println("The value of sum is" + sum);

}

public static void main(String[] args) {

MethodOverloading obj1 = new MethodOverloading();

obj1.add(5, 5);

obj1.add(2,6,2);

}

}

**Classes and Objects**

Objects: any real world entity which has some properties and behaviour

Classes: Blue print or template of object

Example ( car class and 2 objects of class)

Car (Properties and methods/behaviors)

( color, company, engine no, wheels….)

(start() stop(), gear()…..)

Car2 Toyota

Company = Toyota

Color= Blue

Start()

Stop()

Car1 BMW

Company = BMW

Color= Blue

Start()

Stop()

package com;

public class CarApp {

//Properties of car - Instance variable

int carId;

String color;

int wheels;

static String CR = "12738791283";

//Methods of car

void start()

{

System.out.println("Car Starting");

}

void stop()

{

System.out.println("Car Stopping");

}

void display()

{

String company ="ABC Carshowroom"; //local Variable

System.out.println("Car Id is " + carId);

System.out.println("Color of the car is" + color);

System.out.println("Number of wheels" + wheels);

System.out.println("Cars owned by " + company);

}

public static void main(String[] args) {

//Creating 1st object

CarApp car1 = new CarApp();

car1.carId = 001;

car1.color ="Red";

car1.wheels = 4;

System.out.println("Car 1 object is created");

car1.display();

car1.start();

car1.stop();

}

}

**Types of variables**

1. Local Variable: defined inside the method body
2. Instance Variable: declared inside the class but outside the methods
3. Static variable: To set a value throughout the program static is used

Class A{

int a = 10; //Instance Variable

**static** String name = ”Java” //Static Variable

void method()

{

int b =5; //Local Variable

System.out.println(“Value of b is” + b);

}

int b = 10; //Instance Variable

**Constructor**

1. A special method used to initialized the object
2. Invoked at the time of object creation
3. Name of the constructor must be the same as of class name
4. It provides data for the object that’s why its know as constructor
5. It must have no explicit return type

**Syntax**

<class\_name>(){}

**Types of constructor**

1. Default or no-arg
2. Parameterized

package com;

public class CarApp {

//Properties of car - Instance variable

int carId;

String color;

int wheels;

static String CR = "12738791283";

//No-arg Constructor

CarApp(){

System.out.println("Object Created");

}

//Parameterized Constructor using this keyword

CarApp(int carId, String color,int wheels)

{

//calling the default constructor using this keyword

this();

this.carId= carId;

this.color = color;

this.wheels = wheels;

}

CarApp(int car, String col)

{

carId = car;

color = col;

}

//Methods of car

void start()

{

System.out.println("Car Starting");

}

void stop()

{

System.out.println("Car Stopping");

}

void display()

{

String company ="ABC Carshowroom"; //local Variable

System.out.println("Car Id is " + carId);

System.out.println("Color of the car is" + color);

System.out.println("Number of wheels" + wheels);

System.out.println("Cars owned by " + company);

}

public static void main(String[] args) {

//Creating 1st object

//new keyword allocates memory to the object

//Calling default constructor

CarApp car1 = new CarApp();

car1.carId = 1;

car1.color ="Red";

car1.wheels = 4;

System.out.println("Car 1 object is created");

car1.display();

car1.start();

car1.stop();

//Calling 3 parameter constructor

CarApp car2 = new CarApp(2,"Blue",4);

car2.display();

}

}

**This keyword**

*If instance variable and local variable have same name then local variable hide the visibility of instance variable. To refer to instance variable we have to use this keyword. This keyword is use to refer the current object.*

*this.instancevariableName*

**Day 3 (27 oct 2022)**

**Pillars of OOP/Java**

**4 Pillars**

**P**olymorphism

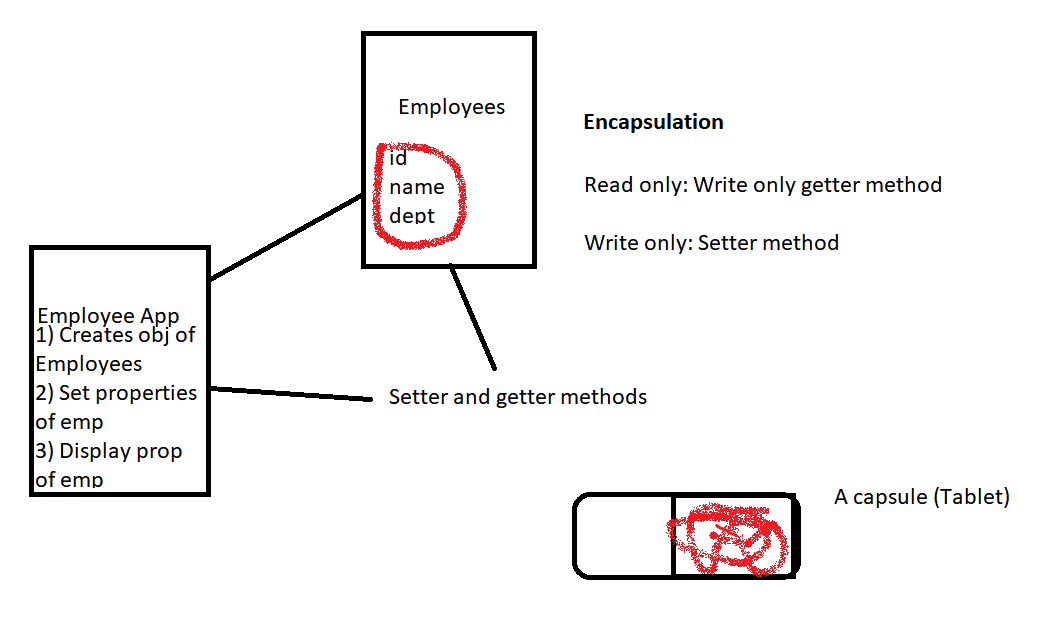
**A**bstraction

**I**nheritance

**E**ncapsulation

**Encapsulation**

Binding or wraping data(properties) and methods in a single unit



package com;

class Employees{

private int empId;

private String name;

private String dept;

private int Salary;

public void setEmpName(String name)

{

this.name = name;

}

//public void getEmpName()

//{

// System.out.println("Employee name is " + name);

// }

public void setSalary(int Salary)

{

if(Salary < 1000)

{

System.out.println("Sorry! the salary has been automatically set to a default value");

this.Salary = 1000;

}

else

{

this.Salary = Salary;

}

}

public void display()

{

System.out.println("Salary is:" + Salary);

System.out.println("Name of emp is " + name);

}

}

public class EmployeeApp {

public static void main(String[] args) {

// TODO Auto-generated method stub

Employees emp1 = new Employees();

//emp1.empId = 1;

//emp1.dept = "Sales";

//emp1.name = "Alice";

emp1.setEmpName("Alice");

emp1.setSalary(200);

emp1.display();

}

}