**Day 1 : 17-01-2022**

**C**

**Procedure language**

**Oops**

**object : object is a any real world entity**

**properties or state --🡪 have 🡪 fields / variables**

**person**

**behaviour --🡪do/does -🡪 methods / functions**

**object is a concept.**

**bank**

**animal**

**car**

**mobile**

**etc**

**int a;**

**a int;**

**a number;**

**class : blue print or template of object. class is also known as user-defined data type which help to create the object. with the help of class we can define the object. class is a implementation of object.**

**class Car {**

**int wheel = 4;**

**String color;**

**float price;**

**void start() {**

**coding ….**

**}**

**void appliedGear() {**

**}**

**void moving() {**

**}**

**void stop() {**

**}**

**}**

**Car innova = new Car(); memory created (heap memory)**

**innova.start();**

**Car santro = new Car(); another car memory created..**

**Car bmw = new Car();**

**C+ OOPs = C++ (C with classes )**

**Java: Java is a platform independent and pure object oriented programming language.**

**Java Version 1.0 , 1.1,1.2 1.8, Java 9, 10,11,12, Java 17**

**Open the cloud lab**

**Open the terminal**

**java –version**

**syntax of class**

**class ClassName {**

**variable declaration**

**method declaration**

**}**

class Demo {

public static void main(String args[]){

System.out.println(“Welcome to Java…”);

}

}

Save program with extension .java

**Demo.java**

**Open the terminal**

**javac Demo.java (To compile the program)**

**java Demo (To run the program)**

**Day 2 : 19-01-2022**

**Eclipse IDE**

**Package is a collection classes and interfaces.**

**Data Types : Data type is a type of data which tells which type of data it can hold.**

**2 types**

1. **Primitive data types : it is use to store only values.**
2. **Non primitive data types or reference data types : it use to store value as well as reference of another data types.**

**Primitive : 8 types**

1. **byte 1 byte**
2. **short 2 byte**
3. **int 4 byte**
4. **long 8 byte : it is use to store value without decimal**
5. **float 4 byte**
6. **double 8 byte : it is use to store the value with decimal**
7. **char 2 byte : any single character**
8. **boolean 1 bit : true or false.**

**Syntax to declare the variable**

**datatype variableName;**

**datatype variableName=value;**

**int a=10;**

**type casting : converting from one data type to another data type is known as type casting.**

1. **Implicit type casting**
2. **Explicit type casting**

**int family**

**---------------------------🡪Implicit type casting ---------------🡪**

**byte short int long**

**🡨-----------Explicit type casting --------------------------**

**Explicit type casting**

**datatype variablename = (type)variabeName;**

**int to float**

**implicit --🡪**

**int float**

**🡨-- explicit**

**In Java Every decimal number by default double consider.**

**Operator :**

**Arithmetic Operator : +, -, \*, /, %**

**Conditional operator : > , >=, <, <=, ==, !=**

**Logical operator : &&, ||, !**

**Assignment operator : =**

**Increment and decrement : ++ , --**

**Conditional statement**

**If statement**

**If else**

**Nested if**

**If else if**

**Switch :switch statement help to execute the set of statement base upon the user requirement.s**

**Syntax**

**switch(variableName) {**

**case label1: block1;**

**break;**

**case label2: block2;**

**break;**

**case label3: block3;**

**break;**

**default : wrong block**

**break;**

**}**

**switch, case, default and break are keywords.**

**Variable can be type of int, char or string.**

**Looping : looping is use to execute the set of statement again and again till the condition becomes false.**

**Initialization : start and end position i=1,n=10**

**Condition true i<=n**

**Do the task hello**

**Increment or decrement. I++ or I = i+1**

**While loop**

**Do while loop**

**For loop**

**For each loop or enhanced loop : This type of loop we can use with array or collection of classes.**

**Taking the value through keyboards.**

**Scanner : Scanner is a pre-defined class which help to take the value through keyboards.**

**Syntax to create the Scanner class object.**

**Scanner obj = new Scanner(System.in);**

**Scanner pre-defined class part of util package.**

**import java.util.Scanner;**

**reference data type or non primitive data type**

1. **array**
2. **class (it may be user-defined or pre-defined) : String is a pre-defined class. it is also known as reference data type.**
3. **interface (it may be user-defined or pre-defined)**
4. **enum**

**by default every java program imported lang package.**

**Day 3 :**

**24-01-2022**

**int a;**

**Reference data types**

**array : array is type of reference data type which is use to store more than one value same type.**

**Syntax**

**datatype arrayName[];**

**declaration of array**

**int abc[];**

**initialization of array**

**int abc[]={10,20,30,40,50,60};**

**size of array is 6. To access the array values we can use index position. Index position start from 0.**

**System.out.println(abc[0]);**

**System.out.println(abc[1]);**

**System.out.println(abc[5]);**

**To find the size of the array we can use pre-defined fields**

**arrayname.length;**

**syntax to declare for each loop or enhanced loop**

**for(datatype variableName : arrayName) {**

**}**

**Creating the memory size for array**

**Syntax**

**datatype arrayName[]=new datatype[size];**

**int abc[]=new int[10]; java.**

**Int leng = abc.length; size 10, memory size 40 byte.**

**System.out.println(abc[0]);**

**System.out.println(abc[9]); value is zero.**

**Int family -🡪0**

**Float family 🡪 0.0**

**Char 🡪 space (white space)**

**Char name =’ ’**

**Boolean 🡪 false**

**String 🡪 null**

**referenceName = null;**

**System.gc();**

**int abc[10]; in C or C++**

**printf(“%d”,abc[0]);**

**java provide auto garbage collector.**

**object and class**

**object : object is any real world entity or instance of a class.**

**Properties or state have variables / fields.**

**Person**

**Behaviour do/does functions / methods**

**Bank**

**Animal**

**Car**

**Mobile**

**Class : class is a blue print of object or template of object**

**Or user-defined data types or reference data type which help to create the object.**

**Syntax to create the memory**

**className referenceName = new ClassName();**

**refereceName.methodName();**

**refereceneName.variableName = value;**

**Car innova = new Car();**

**Types of variables or fields.**

**In Java variable are divided into 3 types**

1. **instance variable**
   1. **The variable which declare inside a class but outside method including main method is known as instance variable.**
   2. **Instance variable hold default value with respective their data types. int family 0, float family 0.0, char space, String null, boolean false.**
   3. **If variable is type of instance we can access those variable directly with a same class but method must be part of same class and it must be non static.**
2. **local variable** 
   1. **The variable which declare inside a method including main method is known as local variable.**
   2. **Local variable doesn’t hold default value. we have to initialize.**
   3. **Scope of the variable within that method where it declared.**
3. **static variable**

**Day 4:**

**25-01-2022**

**Constructor : constructor is a type of special method which help to create the object.**

**Points**

1. **Constructor have the same name as the class itself.**
2. **Don’t provide return type for constructor not even void also.**
3. **Constructor no need to call explicitly it will call automatically when we create the object.**

**If we want to perform any task in the life of the object may be fixed task or dynamic task then we have to write the code inside a constructor.**

**But in the life of the object if we want to perform any task more than one time that type of code we have to write inside a method.**

**If we not write any constructor by default jvm provide default constructor it empty constructor.**

**But if we write explicitly empty or parameter then jvm doesn’t provide default constructor.**

**cid, cname,age : instance variable**

**default constructor**

**cid 123**

**cname unknown**

**age 21**

**parameter constructor**

**cid, cname, age, address, phonumber.**

**Encapsulation : Binding or wrapping data and code in a single unit is known as Encapsulation.**

**By default with the help of class concept we can achieve encapsulation.**

**Example : class**

**If instance variable is private we can’t access or change the value of variable directly as well as through object from outside class but within class we can access.**

**When instance variable and local variable or parameter variable have same name then local variable hide the visibility of instance variable.**

**this keyword : if we want to refer to instance variable when instance variable and local variable have same that type we can use this keyword to refer to instance variable.**

**Inheritance : Inheritance is use to inherits the properties and behaviour of old class to new class.**

**class OldClass { // super class or base class or parent class.**

**properties**

**behaviour**

**}**

**class NewClass extends OldClass{ // sub class or derived class or child class**

**properties**

**behaviour**

**}**

**With the help of sub class object we can access it own properties and behaviour as well as super class properties and behaviour.**

**With the help of super class object we can access only its property and behaviour.**

**Types of inheritance**

1. **Single inheritance : One super class and one sub class**

**class A { }**

**class B extends A{ }**

1. **Multilevel inheritance : One super class and n number of sub classes connected one by one**

**class A{ }**

**class B extends A { }**

**class C extends B { }**

**class D extends C { }**

1. **Hierarchical inheritance : One super class and n number of sub classes connected directly to super class.**

**class A { }**

**class B extends A{ }**

**class C extends A{ }**

**class D extends A{ }**

1. **Multiple inheritance : more than one super class and one sub class**

**class A { }**

**class B { }**

**class C extends A,B {} Java doesn’t support this type of inheritance.**

**This type of inheritance in java we can achieve using interface.**

**Super class must be general and sub class must be specific.**

**class Employee {**

**id,name,salary; primitive property**

**}**

**class Manager extends Employee {**

**numberOfEmp;**

**Address add = new Address(); complex property**

**}**

**class Developer extends Employee {**

**tech**

**}**

**class ProjectManager extends Manager{**

**numberOfProject**

**}**

**class Address {**

**city, state**

**}**

**OOPs relationship**

**Is a relationship**

**Has a relationship**

**Manager Is a Employee**

**Employee Has a Address**

**Day 5:**

**07-02-2022**

**Polymorphism : One name many forms or many implementation**

**2 types**

1. **Compile time polymorphism or static binding or early binding**

**Example**

**Method Overloading : The method have same name but different parameter list. Ie type of parameter list or number of parameter list must be different.**

1. **Run time polymorphism or dynamic binding or late binding**

**Example**

**Method Overriding : The method have same name as well as same method signature.**

**To achieve method override we have to use inheritance concept.**

**abstract keyword**

**abstract is a keyword we can use with method and class but not with variable.**

1. **abstract method: The method without body or without curly braces or incomplete method is known as abstract method.**

**syntax**

**abstract returnType methodName(parameterList);**

1. **abstract class : if class contains abstract method we have to declare the that class as abstract class**

**abstract class className {**

**abstract method as well normal method.**

**}**

1. **Which ever class extends abstract class that class must be provide the body for all abstract method mandatory. That class can ignore only if that class itself is a abstract class.**
2. **We can’t create the object of abstract class.**
3. **Abstract class can contains normal as well as abstract method.**
4. **Abstract class can contains zero or 1 or many abstract methods.**