**Day 1**

**Java Notes**

Java is pure object oriented and platform independent programming language.

Features

Class Syntax

class className { pascal naming rules

fields

methods;

}

Data Types :

Primitive data types

Non primitive data types or reference data types.

Operator

If statement

Looping

For each loop or enhanced loop

Array :

int abc[10]; //C or C++

int abc[];

For loop example with array

class Demo {

public static void main(String args[]) {

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

System.out.println("Welcome to java.....");

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

System.out.println("using for loop");

for(int i=0;i<abc.length;i++) {

System.out.println(abc[i]);

}

System.out.println("for each loop");

for(int a:abc) {

System.out.println(a);

}

}

}

Memory creation for array

int abc[]=new int[10];

int abc[][]=int [4][4];

int xyz[][]={{1,2,3},{4,5,6},{7,8,9}}

abc[0][0]=100;

OOPs

object : any real world entity

properties or state have variables /fields

Person

Behaviour do/does functions / methods

Bank

Animal

Car

Manager

Laptop

class : blue print of object or template of object or user defined data types.

Constructor : it is a type of special methods which help to create the memory.

Points

1. Same names as class itself
2. NO return type
3. No need to call it will call automatically when we create the object.

Encapsulation : Binding or wrapping data and code in a single unit

class :

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

class OldClass { super

property

behaviour

}

class NewClass extends OldClass{ sub

property

behaviour

}

Types of Inheritance

1. Single :

class A {}

class B extends A {}

1. Multilevel :

class A {}

class B extends A {}

class C extends B {}

class D extends C {}

1. Hierarchical Inheritance

class A {}

class B extends A {}

class C extends A {}

class D extends A {}

1. Multiple Inheritance

class A {}

class B {}

class C extends A , B {} Java doesn’t support

**Polymorphism :** one name many forms

Compile time

: Method Overloading : The method have same name but different parameter list (type of parameter list and number of parameter list)

Run time

Method overriding : The method have same name and same method signature.

getConnection(“url”);

getConnection(“url”,userName);

getConnection(“url”,username,password);

static keyword we can use with variable and methods but not with class.

**Day 2**

**Final keyword**

Final is a keyword we can use with variable, method and class

1. Final variable : can’t change value(constant)

static final int A=10;

final int B=20;

1. Final method : if method is final we can’t override that methods.
2. Final class : we can’t extends that class.

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

1. Incomplete method or without body.
2. If class contains abstract method then we have to declare the class as abstract.

**Interface : it is known as 100% pure abstract class till java 7**

Interface Abc {

int A=10; // by default public static and final

void dis1(); //by default all methods are public and abstract.

}

interface Xyz {

Int B=20;

void dis2();

}

interface Mno extends Abc,Xyz{

int C=30;

void dis3();

}

class Demo implements Abc,Xyz {

}

class extends class : only one

interface extends interface : more than one

class implements interface : more than one

interface extends/implements class: not possible

Java 8 onward it can contains method with body but it must be default or static keyword.

JavaBean class

private : we can use instance variable, static variable, non static method, static method, constructor but not with class and local variable.

Scope : with a same class.

default(nothing) : we can use with all

scope with in same package

protected : we can use instance variable, static variable, non static method, static method, constructor but not with class and local variable.

Scope : within same package other package if it is sub class

public : we can use instance variable, static variable, non static method, static method, constructor, class but not with

local variable.

Scope : everywhere

OOPs relationship

Manager is a Employee

has a

class Employee {

id,name,salary

}

class Manager extends Employee {

Address add = new Address();

}

class Programmer extends Employee {

}

class Address {

city,name

}

class Student {

StudentHistory sh = new StudentHistory();

}

class StudentHistory {

}

class A {

B obj1 = new B();

}

class B {

//A obj2 = new A();

}

Association

Aggregation

Composition

**Exception handling**

Exception is a object which occurs when unexpected or abnormal condition occurs during the execution of program

Java

javac java

compile time error run time error

syntax error

Error Exception

JVM crash, Out of memory , software or hardware issue : Error

It is a type of run time error which we can handle it. Divided by zero,

Object

Throwable

Exception

Checked exception unchecked exception

RuntimeException

IOException

SQLException ArithmeticException

ArrayIndexOutOfBoundsException

NumberFormatException

5

try

catch

finally

throw

throws

throw : it is use to throw pre-defined or user-defined exception depending upon the conditions.

throw new Exception();

or

throw new ExceptionSubClass()

throws : it is use to throw the exception to caller methods.

**Day 3**

**Multithreading :**

Program : set of instruction to perform a specific task.

Processor : Processor is responsible to execute the code.

Process : program in execution or time taken to execute the code.

Thread : It is a small execution of a code within a process. Thread also know as light weighted process. Takes less resources or less memory of your machine.

By default Java is thread base programming language.

Inside main method always one default thread execute.

Thread t = Thread.currentThread();

t

Thread[main,5,main];

Main – name of thread

5 –priority of thread

Main –group of the thread

t.setName(“Demo Thread”);

t.setPriority(1)

min 🡪1

max 🡪 10

Thread.MAX\_PRIORITY

MIN\_PRIORITY

NORM\_PRIORITY

Multi tasking :

Process Base

Thread Base Java

Task1 task2 task3

2 ways

1. Extends Thread
2. Implements Runnable interface

Life cycle of thread

isAlive()

join()

sleep()

wait()

notify()

notifyAll()

I=10,j=10

Create--------------- ----🡪 Runnable ------------------🡪 Running ----------🡪 destroy

obj1 obj.start()

t1 t1.start()

**synchronization**

it is use to block or lock or allow all resource for any one thread at time.

**synchronized**

for method or block

wait() ,notify() and notifyAll()

These method are part of object class.

Wait() method is use to make the thread to wait() (suspend) and notify() method is use to call back waited thread or resume the waited thread.

Points

1. More than one thread must be part of same memory.
2. The method must be synchronized.

IO Stream : Flow of data

Byte wise

Char wise

Source 🡪 Keyboards, File, Database, networking etc

Destination 🡪 Monitor or console, file, database and network etc.

**Day 4**

**Object Serialization :**

object 🡪 property (variables)

* Behaviour (methods/functions)
* Identity (reference Name)

Storing object itself in external file.

We are storing only property not behaviour and identity.

Storing object property in external file.

Converting objects into stream/ byte format.

Is Known as Object Serialization.

Which class object we want to do serialization that class must be implements Serializable interface.

This interface doesn’t contains any methods or zero methods.

Marker interface : The interface contains zero or no method is known as marker interface.

**Collection Framework and Generics**

Collection Framework is a like a Data Structure In C/C++

Int a=10;

a=20;

int abc[];

class Employee {

int id;

String name;

float salary;

}

Employee emp = new Employee();

emp.id=100;

emp.name=”Ravi”;

emp.salary = 12000;

array object

int abc[]=new int[10];

Employee employees[]=new Employee[100];

employees[0]=new Employee();

employees[0].id=100;

employees[0].name=”Ramesh”;

employees[0].display();

Collection Framework contains set of classes and interfaces which help to store collection of object or elements of any types(int, float, char, double, Abc, Employee etc).

As well as It provide set of methods with the help of those methods we can add, remove, search, iterate very easily.

Util package

Collection Framework Hierarchy

Collection -🡪 interface

Set List Queue Map

Interface interface interface interface

All are interfaces where Set, List, Queue extends Collection but Map doesn’t extends Collection.

Set : It doesn’t allow duplicate. Under set some classes maintains order, unorder and sorted.

HashSet, LinkedHashSet, TreeSet etc

The above classes directly and indirectly implements Set Interface.

List : It allow duplicate and Maintain the order.

Stack, ArrayList, LinkedList, Vector etc

Queue : First In First Out

PriorityQueue : type of Queue class

Map : Key – value pairs. Where key is unique and value may be duplicate.

HashMap

LinkedHashMap

TreeMap

Hashtable : type of map classes.