**Day 4: 20-10-2025: OOPs concept**

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

2 types

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

Method overloading: Method have same name and different parameter list (number of parameter list or type of parameter list must be different).

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

Method Overriding: Method have same name and same method signature (number of parameter list, type of parameter list, and number of parameter list must be same with different logic inside a method).

**Annotation:** annotation is known as meta-data. Meta-data means data about data. Java provided lot of pre defined annotation. Few annotation we can use on class level or method level annotation or property/variable annotation.

Core java provided one the basic annotation ie @Override. This annotation we can use on method level. If method is overriding we doesn’t get any error else we get the error.

**abstract keyword**

1. abstract keyword we can use with method and class not with variable.
2. abstract method : the method without body or without burly braces or incomplete method is known as abstract method.

Syntax

abstract returnType methodName(parameterList);

abstract void speed();

1. abstract class : if class contains abstract method that class must be declare as abstract class.

abstract class Bike {

}

1. abstract class can contains normal as well as abstract method. means it can contain 1 or many or zero abstract method.
2. abstract class we can’t create the object. but we can create the reference. (memory can’t create using new keyboard).
3. class can be abstract but not mandatory it contains abstract method. use of this class(we can’t create the object of that class).
4. abstract class can contains default constructor. Even we can write empty as well as parameterized constructor.
5. Whichever class extends abstract class that class must be provide all abstract method belong to that class. that class can ignore only if that class itself is abstract class.

static keyword

1. static keyword we can use with variable and method but not with class. (if class is inner class means class within another class that class can be static but not outer class).
2. static variable : if variable is static we can call or access those variable with help of class name object not required.
3. Static method : if method is static we can call or access that method with help of static keyword. object not required.
4. Static method or variable we can access through object also.
5. Inside static method we can access only static variable but inside non static method we can access both static as well as non static variable.

Heap memory Vs Static memory

1. Number of classes equal to number of static memory. Static every always belong to class. if class contains static variable those variable is part of static memory. Static memory only one copy for each class.
2. Heap memory, number of object of any particular class equal to number heap or instance memory. Instance variable always belong to heap memory. Numbe of object equals to number of copy of instance memory.

final keyword :

final keyword we can use with variable, method and class.

1. final variable : if variable is final we can’t change the value of that variable. to declare constant variable in java we use final keyword.

final int A=10;

1. final method : if method is final we can’t override that method. but we can use it.
2. final class : if class final we can’t inherits or extends final class.

**interface :** interface is type of reference data type which also known as 100% pure abstract class.

**syntax**

interface interfaceName {

property

behaviour

}

By default all variable part of interface are public, static and final

By default all methods part of interface are public and abstract

interface Abc {

public static final int A=10;

static final int B=20;

final int C=30;

int D=40;

public abstract void dis1() ;

public void dis2();

abstract void dis3();

void dis4();

}

interface Abc { super interface

int A=10;

void dis1();

}

interface Xyz { super interface

int B=20;

void dis2();

}

interface Mno extends Abc,Xyz{ sub interface

Int C=30;

void dis3();

}

class Test implements Abc,Xyz { // achieve multiple inheritance

must be provide the body for dis1() and dis2() methods.

}

Like a class one interface can extends another interface. Even interface can extends more than one interface. Using above example we are achieving multiple inheritance. Class always implements interface. Class can implements more than one interface. Which ever class implements one or many interface that class must be provide the body for all those methods belong to interface. Test class can ignore if Test class is abstract class.

class Abc {

void show() {

System.out.println(“Abc class show method”)

}

}

class Xyz {

void show() {

System.out.println(“Xyz class show method”)

}

}

class Mno extends Abc,Xyz{ error in this line

2 show method copied present with different logic

}

Mno obj = new Mno();

obj.show();

interface Abc {

int id=100;

void show();

}

interface Xyz {

int id=200;

void show();

}

class Mno implements Abc,Xyz{

public void show() {

System.out.println(Abc.id+” ”+Xyz.id);

}

}

Mno obj = new Mno();

obj.show();