**Day 2: 30 – July 2024 – Core Java**

Polymorphism : One name many forms or many implementation

2 types

1. Compile time or static binding or early binding

Example : Method overloading

The method have same name different parameter list ie type of parameter list or number of parameter list must be different.

1. Run time or dynamic binding or late binding

Example : Method overriding

The method have same name and same method signature ( number of parameter list, type of parameter list and return type must be same).

To achieve method overriding we need mandatory inheritance.

Annotation : annotation is known as meta-data. Data about data. Java provided lot of pre defined annotation as well as we can create custom annotation depending upon our requirement. All annotation start with pre-fix @ followed by annotation name.

We can use annotation on class level or method level or property level or constructor level.

@Override annotation : this annotation we use on method level. If method is overriding compile time we doesn’t get any error. If method is not overriding we can compile time.

abstract keyword

1. abstract keyword we can use with method and class but not with variable.
2. abstract method incomplete method or method without body

abstract void speed();

1. abstract class : if class contains abstract method then we need to declare the class as abstract class.

abstract class className {

}

1. abstract class can contains normal as well as abstract method. it contains 0 or 1 or many.
2. abstract class we can’t create the object.
3. abstract class can contains constructor. We can write parameterized constructor.

final keyword

1. final keyword we can use with variable, method and class.
2. final variable to declare constant value in java we use final keyword

final int A=10;

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

static keyword

1. static keyword we can use with variable and method but not with class. (nested class we can use static keyword but not with outer class).
2. static variable : if variable is static we can assign the value for that variable using class name as well as object.
3. static method : if method is static we can call that method with help of class name as well as object.
4. inside static method we can access only static variable non static not possible directly

every class we get only one static memory.

Interface : interface also known as reference data types.

Syntax

interface interfaceName {

Fields;

Methods;

}

By default all variable inside interface are public static and final

By default all methods are public and abstract.

interface Abc {

public static final int A=10;

public abstract void dis1();

}

interface Xyz {

int B=10;

void dis2();

}

interface Mno extends Abc,Xyz {

int C=10;

void dis3();

}

class Demo implements Abc ,Xyz{

provide the body for dis1 and dis2

}

Packages and access specifiers

Using package/access specifiers we can expose visibility of class, variable, method and constructor within a same package as well as other package.

Package is a collection of classes and interfaces.

Education

school college

Attendance Attendance

Private : scope : within a same class

: we can use with : with all expect class and local variable.

Default (nothing): scope : within a same package

: we can use with : all

Protected :scope : within a same package other package if sub class.

: we can use with : with all expect class and local variable

Public : scope within a same package as well as other package.

: we can use with : with all expect local variable.