**Day 2**

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

To achieve inheritance in java we need to use extends.

class OldClass { super class or base class or parent class

property

behaviour

}

class newClass extends OldClass{ sub class or child class or derived class.

property

behaviour

}

Type 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{}

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{}

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. Which we can achieve using interface.

Oops relationship

Manager/ Developer is a Employee

Employee has a Address

class Employee {

id,name,salary

Scanner obj = new Scanner();

Address add = new Address();

readEmp()

disEmp();

}

class Manager extends Employee{

numberOfEmp;

readMgr()

disMgr()

}

class Developer extends Employee{

projectName;

readDev()

disDev()

}

class ProjectManager extends Manager{

clientId

readPMgr();

disPmgr();

}

class Address {

city,state

Scanner obj = new Scanner();

readAdd()

disAdd();

}

**Polymorphism :**

One name and many forms or many implementation

2 types

1. Compile time or static binding or early binding

Method overloading:

Method have same name but different parameter list ie type of parameter list or number of parameter list must be different.

1. Run time or late binding or dynamic binding

Method overriding: method have same name and same method signature (number of parameter list, type of parameter list and return type must be same) . we can achieve method overriding using inheritance.

Annotation : meta-data. Data about data. Java provided lot of pre defined annotation. All annotation start with pre-fix @ followed by annotation name. few annotation we can use on class level, method level, property level etc.

Non access specifier keywords

1. abstract keyword
   1. abstract keyword we can use with method and class but not with variable.
   2. abstract method : the method without body or incomplete method or without curly braces.
   3. If class contains one or many abstract method. that class we need to declare as abstract class.
   4. Whichever class extends abstract class that class must be provide body for all abstract methods mandatory. That class can ignore if that class itself is abstract class.
   5. Abstract class can contains normal as well as abstract method.
   6. Abstract class can contains 0 or 1 or many abstract method.
   7. Abstract class we can’t create the object.
   8. Abstract class can container default constructor as well as we can write parameterized constructor.
2. static
   1. static keyword we can use with variable and method but not with class (inner class can be static but not outer class).
   2. static variable : if variable is static we can access that variable with help of class name.
   3. static method : if method is static we can call that method with help of class name.
   4. static variable and static method we can access through object also.
   5. Inside a static method we can’t access non static (instance variable) directly.
3. final
   1. final keyword we can use with variable, method and class.
   2. final variable : to declare a constant variable in java we use final keyword.
   3. final int A=100;
   4. final method : if method is final we can’t override that method.
   5. final class : if class if final we can’t inherits that class.

interface: interface is a type of reference data types also known as 100% pure abstract class till java 7.

Syntax

interface interfaceName {

Fields

Methods;

}

By default all fields in interface are public static and final

By default method in interface are public and abstract

interface Abc {

public static final int A=10;

public abstract void dis1();

}

Interface Mno {

int B=20;

void dis2();

}

Interface Xyz extends Abc,Mno {

int C=30;

void dis3();

}

class Test implements Abc,Mno {

void dis1() {}

void dis2() {}

}