Encapsulation : binding or wrapping data(variables ) and code (methods/functions) in a single unit.

Class :

Java Bean:

1. All variable must be private
2. For each variable setter and getter method.
3. Setter method is use to set the value. The method must be start with pre-fix set followed by variable name;
4. Getter method is use to get the value. Getter method start with pre-fix get followed by variable name.

Java Bean class is known as pure encapsulation class.

Inheritance : Inheritance is use to inherits property and behaviour of old class to new class.

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

}

Types of inheritance

1. Single inheritance

class A {}

class B extends A {}

1. Multilevel inheritance

class A {}

class B extends A{}

class C extends B{}

1. Multiple inheritance

class A {}

class B {}

class C extends A,B{} java doesn’t support this type of inheritance. This type of inheritance we can achieve using interface.

1. Hierarchical inheritance

Class A {}

Class B extends A{}

Class C extends A{}

Oops relationship

Manage/Developer Is a Employee

Employee/Manager/Developer Has a Address

class Employee {

id,name,salary

readEmp()

disEmp();

Scanner object

}

class Manager extends Employee{

numberOfEmp

Address add = new Addres();

readManager()

disManager()

}

class Developer extends Employee{

projectName

readDev()

disDev()

}

class ProjectManager extends Manager{

clientInfo

readPManager()

disPManager();

}

class Address {

city and state

Scanner

readAdd()

disAdd()

}

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

1. Compile time or static binding or early binding

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 late binding or dynamic binding

Method overriding: the method have same name and same method signature. To achieve method overriding we need inheritance.

Non access specifier keyword

1. abstract
   1. this keyword we can use with method and class but not with variable.
   2. Abstract method : the method without body is known as abstract methods.
   3. If class contains one or more abstract method that class we need to declare as abstract class.
   4. Abstract class can contain normal as well as abstract method.
   5. Abstract class can contain 0 or 1 or many abstract method.
   6. Abstract class can contain default as well as we can write parameter constructor.
2. final
   1. final keyword we can use with variable, method and class.
   2. final variable: to declare a constant variable we use final keyword with variable. final int A=10;
   3. final method : if method is final we can’t override that method.
   4. final class : we can’t extends final class.
3. static
   1. static keyword we can use with variable and method but not with class. (if class is nested we can use static keyword but not for outer class).
   2. if variable is static we can assign or access that variable with help of class as well as object.
   3. if static method : method can call using class name as well as object.
   4. inside static method we can’t access not static variable.

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

Syntax

interface InterfaceName {

fields

methods;

}

All variable in interface by default public static and final

All methods in interface by default public and abstract

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 Demo implements Abc,Xyz {

demo need to provide the body for dis1 and dis2 mandatory. This class can ignore if class is abstract class.

}

Interface to interface : it can extends one or many.

Class to class : it can extends only one

Class to interface : it can implements one or many

Interface to class : no extends no implements

Abstract class Vs interface

1. abstract class can contain normal as well as final variable. but interface contains only final variable.
2. abstract class can contain normal as well as abstract method. but interface contain only abstract method.
3. abstract class can contain default as well as parameter constructor but interfaces doesn’t contain.
4. Normal or abstract class extends only one abstract class. normal or abstract class can implements more than one interface.
5. Using abstract class we can achieve partial abstraction but using interface we can achieve 100% abstraction.