1. a) Inheritance

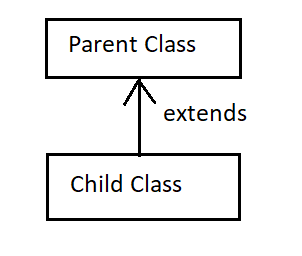
Accquring the properties and behavior of their parent class is called

Inheritance. It is possible by using extends keyword in child class. Inheritance represents the Is-a relationship. Only child class can acess the properties of the parent class, Parent class cannot acess the child properties.

There are 5 types of Inheritance

1. Single Inheritance
2. Multilevel Inheritance
3. Hirarchicle Inheritance
4. Multiple Inheritance
5. Hybrid Inheritance
6. Single Inheritance

Single class extends to single parent class those are called Single inheritance



Example:

Class A{

void call(){

Syso(“Hello”);

Class B extends A {

Void msg(){

Syso(“Hi”);

Main(String[] args){

B b = new B();

Syso(b.call());

Syso(b.msg());

}

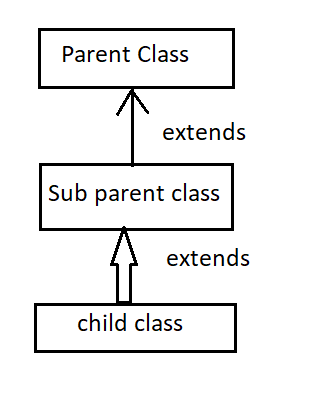
1. MuiltiLevel Inheritance

Multilevel Inheritance means Chain of Extending, which means one

Class is extending to one parent class and that child class is become parent when the other class is extending to that class , this is called MultiLevel Inheritance

We can directly acess the supper parent class by chain extending.

This is the major advantage of this Inheritance



Example:

Class A1{

void call(){

Syso(“Hello”);}

}

Class A extends A1{

}

Class B extends A {

Void msg(){

Syso(“Hi”);

Main(String[] args){

B b = new B();

Syso(b.call());

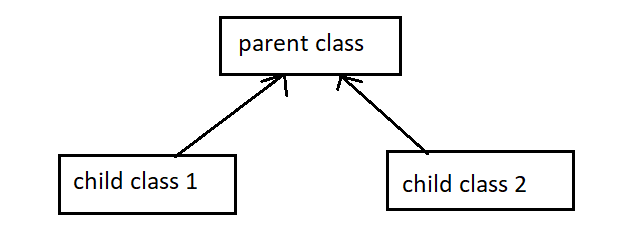
Syso(b.msg());

}

1. Hirarchicle Inheritance

In this Inheritance there is only one parent class and having multiple

Child class which extend to only one parent Class. All child class can acess the same method and properties that are present in parent class.



Example:

Class A1{

void call(){

Syso(“Hello”);}

}

Class A extends A1{

Void msg(){

Syso(“hi”);

}

Class B extends A1 {

Main(String[] args){

B b = new B();

Syso(b.call());

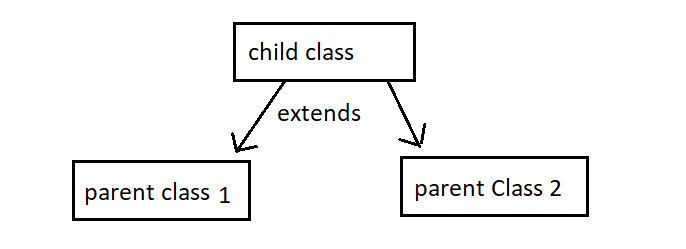
Syso(b.msg());

}

Other 2 Inheritance are not supported by java in terms of Java class, It can be achived by Interfaces

1. Multiple Inheritance

It has only one child class and having to parent class, it was not supported in java class because there is a high chance of getting the ambiguity by using this Inheritance. If two parent class having same method name then it confuse to go for which class. So this Inheritance is not suppoted in java class.



Example:

Class A1{

void call(){

Syso(“Hello”);}

}

Class A{

Void call(){

Syso(“hi”);

}

Class B extends A1,A {

Main(String[] args){

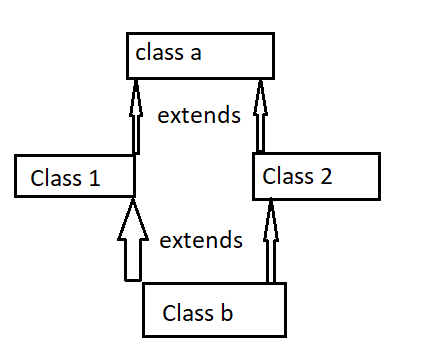
B b = new B();

Syso(b.call()); //Ambigity occur

}

1. Hybrid Inheritance

It is Combination of Multiple and Hirarchicle Inheritance, where there is a Multiple inheritance Inheritance is not supported by class.



b) Polymorphism

One action or a task can be be done in many forms or many ways this is called Polymorphism. The world poly means many morphism means forms.

There are 2 types of Poly Morphism

1. Complile time poly Morphism
2. Runtime Polymorphism
3. Complitime poly Mohrphism

Method over loading is the best example for complitime Polymorphism,

In this method overloading method should have same name but different parameters

And it is not possible by changing the return type, we should increase the parameter or decrease the paramenter.

Example:

class B{

int sum(int a, int b){

return a+b;

}

int sum(int a, int b, int c){

return c-b-a;

}

Main(){

B b = new B();

Syso(b.sum(10,20));

Syso(b.sum(10,20,30);

}}

Output:

30

0

1. Runtime Polymorphism

Methor OverRidding is the best example for the Runtime Polymorphism.

In this Method Overrriding Method should contain same name and same parameter, and should be in inheritance(is-a relation).

Example:

Class A1{

void call(){

Syso(“Hello”);}

}

Class A extends A1{

Void call(){

Syso(“hi……”);

}

Class B extends A1 {

Main(String[] args){

B b = new B();

Syso(b.call());

Syso(b.msg());

}

Output: hi……

c) Abstraction

**Abstraction** is a process of hiding the implementation details and showing only

functionality to user,  it shows only essential things to user and hide internal details. We can

achive 0-100% abstraction in java class , in inter face we can achive 100% abstraction. Because

java class contain both Abstract method and concrete methods. We use Abstract key word to

abstract methods and classes. The class which are extends the abstract class should implement

the unimplemented method in their class or else the extended class also an abstract class.

Example, sending SMS : don't know the internal processing of message delivery.

Example:

abstract class A1{

void call(){

Syso(“Hello”);}

abstract void msg();

}

Class A extends A1{

Void msg(){

Syso(“hi……”);

}

Class B extends A1 {

Main(String[] args){

B b = new B();

Syso(b.call());

Syso(b.msg());

}

Output:

Hello

hi……

d) Encapsulation

Warpingup of Data or code in a Single Unit is called Encapsulation. Java Bean class is fully encapsulated class, because it contains private properties, all argu constructor, default constructor, public getter and setter, the main advantage of Encapsulation is we can make our project only readable or writable setter and getters, we can Control our data.

Example;

class A{

private int a;

public A(){}

public A(int a){

this.a=a;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

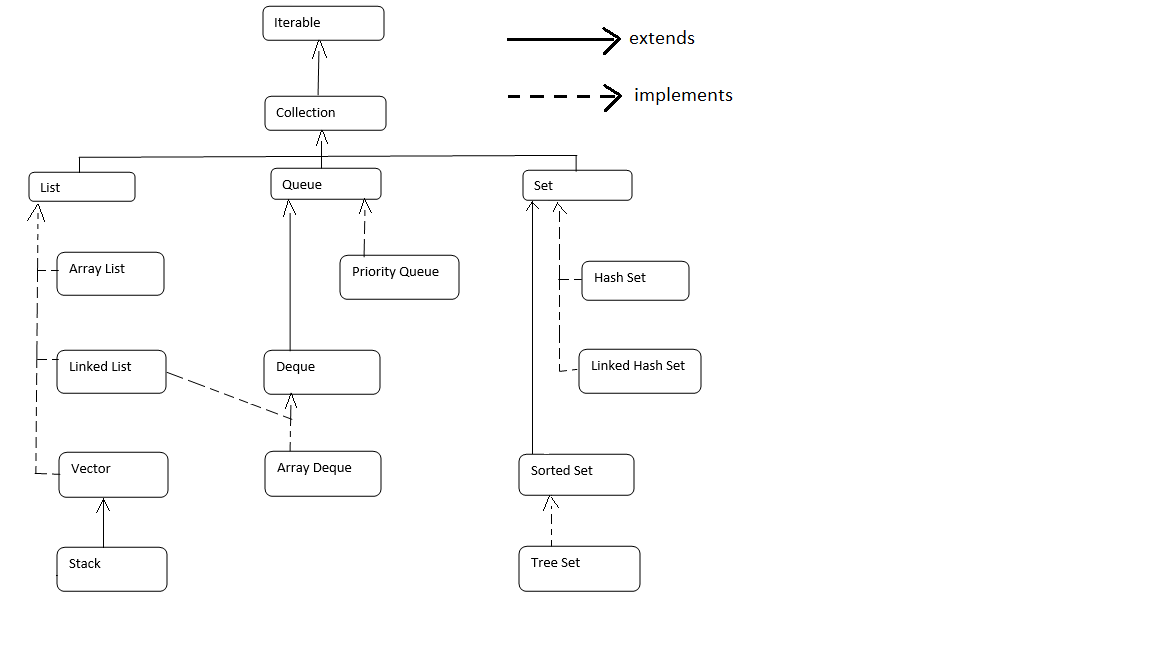
2. Summary about JDBC

JDBC stands for Java Database Connection, Jdbc is an API (Application programming interface) which is used to connect and execute the query to the database. JDBC API uses JDBC drivers to connect to the database JDBC API can be used to access tabular data stored into any relational database. JDBC Driver is a software component that enables Java application to interact with the database.

Befor Jdbc invention the developers struggled lot of to study different different Databse to connect with their java code. Not only developers, also company also face some serious issue to hire the knowdegeble candidate for connecting their databases. Then JDBC Came into picture all difficulties were gone for both developers and companies.

JDBC also has some major disadvantages of Manual Maping, this is the only problem occur in JDBC

3. Hirarchy of Collection



1. Linked List

LinkedList class implements List and Deque interfaces.

* Java LinkedList class uses a doubly linked list to store the elements
* Java LinkedList class can contain duplicate elements
* Java LinkedList class maintains insertion order
* Java LinkedList class is non synchronized
* LinkedList manipulation is fast because no shifting occur
* Java LinkedList class can be used as a list, stack or queue
* The location for the elements of a linked list is not contagious
* There is no case of default capacity in a LinkedList
* It has two Constructor, default constructor and parameterized Constructor with Collection, and have some methods which are available from List Interface.

1. Hash Map

**HashMap** class implements the Map interface which allows usto store key and value pair, where keys should be unique.

* Java HashMap contains only unique keys
* Java HashMap may have one null key and multiple null values
* Java HashMap is non synchronized
* Java HashMap maintains no order
* The initial default capacity of Java HashMap class is 16 with a load factor of 0.75
* Java HashMap contains values based on the ke

1. Tree Set

TreeSet class implements the Set interface The objects of the TreeSet class are

stored in ascending order

* TreeSet class contains unique elements only like HashSet
* TreeSet class access and retrieval times are quiet fast
* It  doesn't allow null element
* It is non synchronized
* It maintains ascending order
* It contains unique elements only like HashSet
* TreeSet can only allow those generic types that are comparable

4 . Servlet

Servlet is a server component that is developed on the server to create a dynamic web page by using web Container, only one object is created at the time of first request by servlet. There are methods called get and post. Get method can send limited amount of data where in post method can send large amount of data

Request Dispatcher

RequestDispatcher interface provides dispatching the request to one servlet to another servlet or it may be html, (servlet or jsp). This interface can also be used to include the content of another resource also using same servlet

5 . Difference B/w JPA and Hibernate

|  |  |
| --- | --- |
| JPA | Hibernate |
| JPA is described in **javax.persistence** package | Hibernate is described in **org.hibernate** package. |
| It describes the handling of relational data in Java applications. | Hibernate is an Object-Relational Mapping (ORM) tool |
| it uses **Java Persistence Query Language (JPQL)** to execute database operations. | it uses **Hibernate Query Language (HQL)** to execute database operations. |
| It is not an implementation | Hibernate is an implementation of JPA |
| It uses **EntityManagerFactory** interface | it uses **SessionFactory** interface. |
| it uses **EntityManager** interface | it uses **Session**interface |