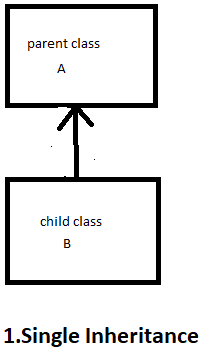
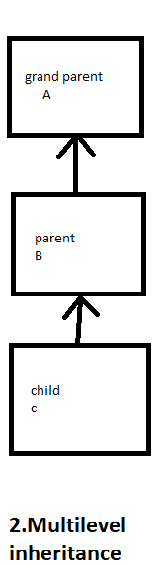
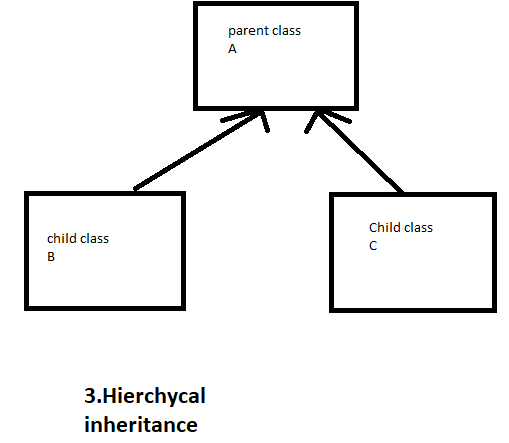
**Inheritance**

* It is the process that one object is to acquired the all properties of the parent class is called inheritance.
* It is code reuseability.
* We can call it Is-A relationship or parent child relation.
* It is 4types of inheritances.
* **Single inheritance :** It is one parent class and one child class or one class inherit from another class.
* Example: Vehicle and bike. 
* Class Vehicle {
* void run(){
* System.out.println(“Vehicle can run”);
* }
* Class Bike extends Vehicle{
* Void drift(){
* System.out.println(“Bike can drift”);
* }
* Public Static void main(String[] args){
* Bike b = new Bike();
* b.run();
* b.drift();
* **Multilevel inheritance:** It is the process of chaining of inheritance or there is one grand parent and one parent and child.
* Class Animal{ 
* void eat(){
* System.out.println(“Eating..”);
* }
* Class Dog extends Animal{
* Void bark(){
* System.out.println(“barking..”);
* }
* Class puppy extends dog{
* Void pop(){
* System.out.println(“poping…”)
* }
* Public Static void main(String[] args){
* Puppy p = new Puppy ();
* p.eat();
* P.bark();
* P.pop();
* **Hierchycal inheritance:** It is the process of one parent class and multiple child class.
* Class Vehicle { 
* void run(){
* System.out.println(“Vehicle can run”);
* }
* Class Car extends Vehicle{
* Void drift(){
* System.out.println(“Car can drift”);
* }
* Class Bike extends Vehicle{
* Void drag(){
* System.out.println(“Bike in Drag race”)
* Public Static void main(String[] args){
* Bike b = new Bike();
* b.run();
* b.drift();
* b.drag();
* **Multiple inheritance :** It is not support is java because it one child and multiple parent so java confuse to detect the which parent class but we can multiple inheritance throw interface.

**Polymorphism**

* It is the process one action doing many ways. Here polymorphism means poly means ‘many’ and morphism means ‘forms’ So one persone can posess many behavior.
* Polymorphism is used to 2types
* **Compiletime polymorphism:** It is archive throw method overloading.
* It is one or more methods but different parameters.
* **Runtime polymorphism:** It is archive throw method overriding.
* Runtime polymorphism is a process thatresolves a call to an overridden method at runtime. The process involves the use of the reference variable of a superclass to call for an overridde method.
* Example :
* class Bike{

  void run(){System.out.println("running");}

}

class Splendor extends Bike{

 void run(){

System.out.println("running safely with 60km");}

   public static void main(String args[]){

     Bike b = new Splendor();

     b.run();

  }

}

**Abstraction**

* It is the process hiding the implementation and showing only functionality to the user is called abstraction.
* It is a class that is declared with an [abstract keyword.](https://www.geeksforgeeks.org/abstract-keyword-in-java/)
* Abstract method is a method that is declared without implementation.
* Abstract class may or may not have all abstract methods..
* Any class that contains one or more abstract methods must also be declared with an abstract keyword.
* There is 2 ways to archive abstraction

1.Abstract class(0 to 100%)

2.Interface(100%)

1.**Abstract class:** It is the class declare as abstact is called abstract class.

* Example:

abstract class Bike{

   abstract void run();

}

class Hero extends Bike{

void run(){

System.out.println("running…");}

public static void main(String args[]){

 Bike obj = new Hero();

 obj.run();

}

}

2.**Interface:** An interface is a blueprint of a class .

* Throw Interface we can achive abstraction or multiple inheritance.
* It is also represet IS-A relationship.
* It can archive loose coupling.
* It can be declare by interface keyword.
* A class extends another class or a interface extends another interface but a class implements interface.
* Example:
* interface A{
* void print();
* }
* class B implements A {
* public void print(){
* System.out.println("Hello");}
* public static void main(String args[]){
* B obj = new B();
* obj.print();
* }
* }

**Encapsulation**

* It is wrappin or binding data or code in single unit is called encapsulating.
* We can create a fully encapsulated class in Java by making all the data members of the class private. Now we can use setter and getter methods to set and get the data in it.
* It is used getter and setter method the class is read only and write only.
  1. Example: capsul inside this there is many mix medicine.
* public  class  Student {
* private  String  name;
* public  String  getName(){
* return name;
* }
* public  void  setName(String name){
* this.name=name
* }
* }

**2.JDBC Summery**

* Jdbc stands for Java database connectivity. Jdbc is a Java Api to connect and execute the query with the database.
* Jdbc Api uses Jdbc drivers to connect with the database.
* Mainly we used to create jdbc is
  + 1. Driver class
    2. Create connection
    3. Create statements
    4. Execute the queries
    5. Close connection
* Load the driver using forname()method.

Class.forname() to load the mysql driver .

* We use drivermanager.getconnection() and put the parameters there

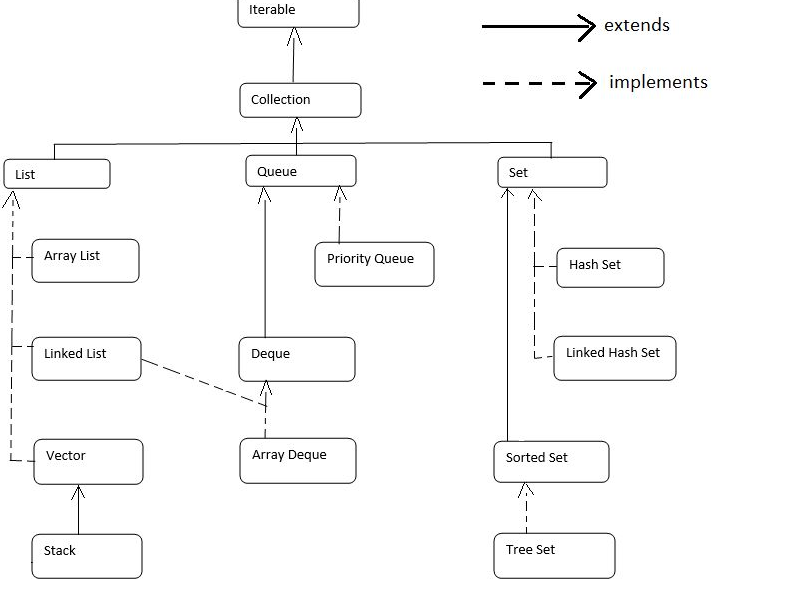
username from sql accessed,password and url.

* Create a statement the PreparedStatement interface define the method and send to sql commands and receive data from your databace.
* ExecuteQuery is used to execute the sql query is for find and read means to reretiving value.
* It is methode returns resultSet.
* And their id executeUpdate() method of statement interface execute the queries to

Update and delete the data.

* This way we can know the jdbc program.

**3.Hierachy of Collection:**

****

* + 1. **Linkedlist:**
* It is implements list and deque interface.
* It uses a doubly linked list internally to store the elements.
* It can allow the duplicate value. It maintains the insertion order and is not synchronized.
* It is a class can be used as list,stack and queue.
* We can use both constructor .default and parameter.

ii)**Hash Map:**

* HashMap class implements the Map interface which allows us to 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

iii)**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

**4.Servlet:**

* **Servlet** technology is used to create a web application.
* There is many interfaces and classes in servlet Api.
* Servlet is an interface that must be implemented for creating any Servlet.
* Servlet is a class that extends the capabilities of the servers and responds to the incoming requests. It can respond to any requests.
* Servlet is a web component that is deployed on the server to create a dynamic web page.

**Request dispatch:**

* The RequestDispatcher interface is used to dispatching the request to another resource it may be html, servlet or jsp.
* This interface can also be used to include the content of another resource also. It is one of the way of servlet collaboration.
* There are two methods defined in the RequestDispatcher interface.

1.forword():Forwards a request from a servlet to another resource on the server.

2.include():Includes method is send first servlet then goes to 2nd servlet again came to 1st servlet then it is give response.

**5.Hibernate and jpa difference**

* Hibernate is a Java framework that simplifies the development of Java application to interact with the database.
* Hibernet is used in org.hibernet.
* It is an open source ORM (Object Relational Mapping) tool. Hibernate implements the specifications of JPA (Java Persistence API) for data persistence.
* The Orm tool internally uses the Jdbc API to interact with the database.

**Jpa:**

* It stand for java persistence Api.
* Jpa is used in javax.persistence package.
* It describes the handling of relational data in Java applications.
* It is not an implementation. It is only a Java specification.