**Java**

------

1)It is a programming language.It is a high level,robust,object-oriented programming language.

2)In java we can develop web application,mobile application ,standlone application.

3)Java is simpler than c or c++ as we don't have pointer concept,operator overloading .

4)It is portable and light weight.

5)In java we have constructor but no destructor as there is automatic garbage clearance.

6)Java development kit (JDK) are platform dependent as they differ from each OS.

it performs task like loads code,verifies code,exceute code and provide run time environment.

7)we have different versions of java avialable.currently we are using JDK 1.14

-------------------------------

Data types in java

i)Primative data type

byte --1 byte

short--2 bytes

int---4 bytes

long--8 bytes

float--4 bytes

double--8 bytes

char--2 bytes

boolean --1 bit

ii)Non-primative

String

Array

------------------------------------------------------------------

operator

------------

1)Unary

i)i++,++i or i--,--i

2)Binary

i)Arithmatic operator:- +,-,\*,/,%(modulus)

ii)comparative operator :-<,>,<=,>=,=

iii)logical operator:-&&,||

iv)Bitwise operator:-<<,>>

v) assignment operator :-=,!=

3)Ternary

i)?,:

--------------------------------------------------------------------

OOPS (object oriented programming system)

1)Object:- It is a reference pointer to access the variables and methods of a class.

2)class:-It is blue print of an object or it is also called as object factory.

3)Inheritance:-It is when one object acquires all the properties and behaviors of a parent object.

4)Polymorphism:-If one task is performed in different ways it is known as polymorphism.

in java we use method overloading and method overriding.

5)Abstraction:-Hiding internal details and showing functionality is known as abstraction.(we switch on the fan the internal part is hidden and it rotate)

6)Encapsulation:-Binding code and data together into a single unit are known as encapsulation.(capsule wrapped with different medicines)

----------------------------------------------------------------------------

Access specifier

-------------------------

1)private:- if we declare variables and methods as private we can only access it within the class.

2)public:-if we declare variables and methods as public we can it within the class,outside the class and package.

3)protected:-if we declare variables and methods as protected we can only access it within the class and in the child class

4)default:-if we declare variables and methods as default or no access specifier we can only access it within the class and also outside the class but not outside the package.

----------------------------------------------------------------------------------------------------

util package --->Scanner --->methods

lang package --->String (non-primative /class)---->methods

-------------------------------------------------------------------------------

All datatypes in java is also class known as wrapper class.

\*It belong to the lang package.it is also known as default package.

\* all classes in java are in caps

\*all packages are in small

\*methods will be in camel case.

datatype Wrapper class(methods).

-----------------------------------

byte Byte

short Short

int Integer

long Long

float Float

double Double

char Character

String String

--------------------------------------

String :-

-----------

1)String is a class in java.lang package

2)In java string is also considered as datatype

3)String object is Immutable(cannot be changed)

4)When we assign a value it create a new object and the value remains in memory.

Example :- String s1=“hello”;

String s2=new String(“hello”);

Method :- length(),isEmpty(),charAt(),equals(),startWith(),endsWith(),indexOf(),lastIndexOf(),toUpperCase(),toLowerCase()

-----------------------------------------------------------------------------------------------------------------------

String Buffer:-

---------------------

1)It is a peer class of string and provides all functions of string.

2)It is mutable (can be modified) and expanded dynamically.

3)It is synchronized

Ex:-1) StringBuffer b=new StringBuffer(“hello”);

We pass “hello” to StringBuffer object “b”

2)StringBuffer b=new StringBuffer(50);

We can store 50 or more character .It is mutable so it expands dynamically

3)StringBuffer b=new StringBuffer()

StringBuffer object b is created with default capacity 16 character.

4)Methods :- append(),insert(),delete(),reverse()

--------------------------------------------------------------------------------------------

String Builder :-

--------------------

1)It is same as String Buffer

2)String Builder is not synchronized

3)String Buffer class takes more time to execute than StringBuilder.

-----------------------------------------------------------------------------------------------

String s1=”sandip”;

String s1=”kumar”;

Now the reference s1 contain new data the old object data is lost.

It is known as unreferenced object and garbage collector will remove it from memory.

---------------------------------------------------------

package First.Friday;

class Firstprg

{ public static void main(String args[])

{

String s1="Mphasis and Global soft";

System.out.println(s1);

System.out.println(s1.toUpperCase());

System.out.println(s1.toLowerCase());

System.out.println(s1.indexOf('a'));

System.out.println(s1.lastIndexOf('a'));

System.out.println(s1.substring(5));

System.out.println(s1.substring(5,10));

System.out.println(s1.startsWith("Mp"));

System.out.println(s1.endsWith("ft"));

System.out.println(s1.replace("soft","software"));

String ss="Mphasis and Global soft,mumbai";

System.out.println(ss);

System.out.println("the length of string="+ss.length());

System.out.println(ss.replace("mumbai","Bangalore"));

char name[]=s1.toCharArray();//convert string to char array

int leng=name.length;

System.out.println("length of an array="+leng);

for(int i=0;i<leng;i++)

{

System.out.println(name[i]);

}}}

----------------------------------------------------------------------

package First.Friday;

public class Secondprg

{

public static void main(String args[])

{ int age=25;

StringBuffer s=new StringBuffer(" She is ");

StringBuffer s1=new StringBuffer("Hello to HP ");

System.out.println("string buffer="+s);

System.out.println("s append="+s.append(age));

System.out.println("s append="+s.append(" years old."));

System.out.println("entire string="+s.toString());//convert object to string

System.out.println("length="+s.length());

System.out.println("capacity="+s.capacity());

System.out.println("s1="+s1);

System.out.println("charAt="+s1.charAt(1));

s1.setCharAt(1,'i');

s1.setLength(5);

System.out.println("s1="+s1);

System.out.println("set insert="+s1.insert(5,"welcome"));

System.out.println("to delete="+s1.delete(5,6));

System.out.println("to reverse="+s1.reverse());

System.out.println(s1);

}}

--------------------------------------------------

toString

---------------

class Employee

{

private int empno;

private String name;

private Address address; //has a relationship

public int getEmpno() {

return empno;

}

public void setEmpno(int empno) {

this.empno = empno;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public Address getAddress() {

return address;

}

public void setAddress(Address address) {

this.address = address;

}

}

----------------------------------------------------------------

public class Address

{

private int houseno;

private String city;

private String state;

public int getHouseno() {

return houseno;

}

public void setHouseno(int houseno) {

this.houseno = houseno;

}

public String getCity() {

return city;

}

public void setCity(String city) {

this.city = city;

}

public String getState() {

return state;

}

public void setState(String state) {

this.state = state;

}

@Override

public String toString() {

return "Address [houseno=" + houseno + ", city=" + city + ", state=" + state + "]";

}

}

------------------------------------------------------------------------

public class Mainclass

{

public static void main(String[] args)

{

Address ad=new Address();

ad.setHouseno(101);

ad.setCity("bangalore");

ad.setState("karnataka");

Employee em=new Employee();

em.setEmpno(1001);

em.setName("Farhan");

em.setAddress(ad);//we are setting the object of Address here

System.out.println("the empno is "+em.getEmpno());

System.out.println("the name is "+em.getName());

System.out.println("the address is "+em.getAddress());

}

}

-----------------------------------------------------------------------------------

constructor():-

1) A constructor is a function which has the same name as that of the class name.

2)A constructor doesnot return any values.

3)A constructor is mainly use to assign values .because a constructor excute first when a object is created for a class.

4)There are 2 types of constructor .

i)default constructor:- A constructor without parameter.

ii)parameterized constructor :- A constructor with parameter.

5)A constructor excute auotmatically when an object is created for a class.

--------------------------------------------------------------------------------------------------

example:-

class Firstprg

{

Firstprg()

{

System.out.println("This is default constructor");

}

Firstprg(int a,int b)

{

System.out.println("the sum of 2 nos are"+(a+b));

}

int sum(int a,int b)//it returns a value

{ return a+b;}

public static void main(String[] args) {

Firstprg ob=new Firstprg();

Firstprg ob1=new Firstprg(5,6);

Firstprg ob2=new Firstprg();

Firstprg ob3=new Firstprg(15,16);

System.out.println("the sum is "+ob.sum(7,8));

System.out.println("the sum is "+ob.sum(17,18));

}}

-------------------------------------------------------------

package First.Friday;

class Firstprg

{

int empno;//instance variables

String name;

Firstprg(int empno,String name)//local variables

{

this.empno=empno;

this.name=name;

System.out.println("the empno "+empno);

System.out.println("the name is "+name);

}

public static void main(String[] args)

{

Firstprg ob=new Firstprg(101,"sandip");

}}

---------------------------------------------------

Object Array

---------------------

package First.Friday;

import java.util.\*;

class Firstprg

{

int accno,bal;//instance variables

String name,address;

Firstprg(int accno,String name,String address,int bal)//local variables

{

this.accno=accno;

this.name=name;

this.address=address;

this.bal=bal;

}

void display()

{

System.out.println("Accno :"+accno);

System.out.println("Name :"+name);

System.out.println("Address :"+address);

System.out.println("Balance :"+bal);

}

public static void main(String[] args)

{

Scanner obj=new Scanner(System.in);

System.out.println("How many customers want to enter");

int x=obj.nextInt();

//Firstprg ob2=new Firstprg(101,"prasanna","pune",8000); //creating single object

Firstprg[] ob=new Firstprg[x];//to create object array. <classname>[] object=new <classname>[5];

for(int i=0;i<x;i++)

{

System.out.println("enter accno,name,address,balance");

int accno=obj.nextInt();

String name=obj.next();

String address=obj.next();

int bal=obj.nextInt();

ob[i]=new Firstprg(accno,name,address,bal); //to enter data into each array object

}

//ob2.display();

for(int i=0;i<x;i++)

ob[i].display();

System.out.println("enter the accno to search");

int temp=obj.nextInt();

for(int i=0;i<x;i++)

{

if(ob[i].accno==temp)

{

System.out.println("the name is "+ob[i].name);

System.out.println("the address is "+ob[i].address);

System.out.println("the balance is "+ob[i].bal);

}

}

}}

-----------------------------------------------------------------------------------------

package First.Friday;

import java.util.\*;

class Firstprg

{

int accno,bal;//instance variables

String name,address;

Scanner obj=new Scanner(System.in);

void input()

{

//this.accno=accno; //local variable is assigned to instance variable

System.out.println("enter accno,name,address,balance");

accno=obj.nextInt();

name=obj.next();

address=obj.next();

bal=obj.nextInt();

}

void display()

{

System.out.println("Accno :"+accno);

System.out.println("Name :"+name);

System.out.println("Address :"+address);

System.out.println("Balance :"+bal);

}

public static void main(String[] args)

{

Scanner obj=new Scanner(System.in);

System.out.println("How many customers want to enter");

int x=obj.nextInt();

Firstprg[] ob=new Firstprg[x];//declaring of array object

for(int i=0;i<x;i++)

{

ob[i]=new Firstprg(); //object created

ob[i].input();

//ob[i].display();

}

System.out.println("enter the accno to search");

int temp=obj.nextInt();

for(int i=0;i<x;i++)

{

if(ob[i].accno==temp)

{

System.out.println("the name is "+ob[i].name);

System.out.println("the address is "+ob[i].address);

System.out.println("the balance is "+ob[i].bal);

}

}

}}

--------------------------------------------------------------------------

Inheritance

---------------------

It is a mechanism in which one object acquires all the properties and behaviors of a parent object.

we can tell code reusability.

In this we will be having a parent class and a child class.

we use to keyword in inheritance.1)extend key to inherite the parent class into the child class.

2)super key word to inherite the variables and methods of parent class into the child class.

Java support single inheritance.(single,multilevel,hierarchical)

Java doesnot support multiple inheritance. we use interface for multiple inheritance.

-----------------------------------------------------------------------------------------------

Employee class(Parent)

package second.Wednesday;

import java.util.\*;

public class Employee

{

int empno;

float salary;

String name;

void input()

{

Scanner ob=new Scanner(System.in);

System.out.println("enter empno,name,salary");

empno=ob.nextInt();

name=ob.next();

salary=ob.nextFloat();

}

void display()

{

System.out.println("the empno is "+empno+ "name is "+name+ "salary is "+salary);

}

}

----------------------------------------------------------------------

package second.Wednesday;

import java.util.\*;

public class Developer extends Employee //inherite the parent class into the child class

{

String project,duration;

void input() //method overridding

{

super.input(); // to access the parent method into the child class.

Scanner ob=new Scanner(System.in);

System.out.println("enter project name,duration of project");

project=ob.next();

duration=ob.next();

}

void display()

{

super.display();

System.out.println("the project name is "+project+ " project duration is "+duration);

}

public static void main(String[] args)

{

Developer dev=new Developer();

dev.input();

dev.display();

}

}

----------------------------------------------------------------

Using Constructor

---------------------------

package second.Wednesday;

import java.util.\*;

public class Employee //parent class

{

int empno;

float salary;

String name;

public Employee(int empno, float salary, String name)

{

this.empno = empno;

this.salary = salary;

this.name = name;

}

void display()

{

System.out.println("the empno is "+empno+" salary is "+salary+"name is "+name );

}

}

-----------------------------------------------------

package second.Wednesday;

import java.util.\*;

public class Developer extends Employee //inherite the parent class into the child class

{

String project,duration;

public Developer(int empno, float salary, String name, String project, String duration)

{

super(empno, salary, name);//first the parent class constructor will execute then other will execute

this.project = project;

this.duration = duration;

}

void display()

{

super.display();

System.out.println("project name is "+project+" duration is "+duration);

}

public static void main(String[] args)

{

Developer dev=new Developer(101,6700.50f,"Nitish","AirLine Reservation system","2 months");

dev.display();

}

}

--------------------------------------------

static :- it is a keyword.we can declare a variable as static ,method as static ,block as static and class as static when we take a inner class.

A single copy is created and shared to the JVM.So if we declare variable as static ,method as static we can access it without creating an object.

we cannot use a non-static variable inside a static method.

-----------------------------------------------

package second.Wednesday;

public class Staticexample

{

static int a=10;

int b=20;

static void display()

{

//int c=a+b;//we cannot call a non-static variable inside a static method

System.out.println("the value of a is "+a);

}

static

{

System.out.println("this is a static block");

System.out.println("This block will exceute before the main method");

}

public static void main(String[] args)

{

System.out.println(a);

display();

}

}

----------------------------------------------------------

final keyword

-----------------------------

we can declare a class as final,method as final and variable as final.

if we declare class as final we cannot inherite it.

if we declare variable as final we have to assign it and it cannot be changed.

if we declare method as final we cannot override it.

-------------------------------------------------------------------------------

pre-defined final class?we cannot change it (immutable).example String class.

StringBuffer,Integer,

----------------------------------

package second.Wednesday;

public final class Finalexample

{

final int a = 10;

final void display()

{

System.out.println("this is a final method");

}

public static void main(String[] args) {

Finalexample ob=new Finalexample();

//ob.a=20; this is not allowed

System.out.println(ob.a);

ob.display();

}

}

---------------------------------------------------------------------------------------

Abstract class and Interface

--------------------------------------------

abstract class is a class which contains abstract methods as well as concrete methods.

abstract methods are those which donot have method body.

concrete methods are those which has method body.

we cannot create object of a abstract class.we have to inherite a abstract class into a subclass.override the abstract methods and create object of the child class.

--------------------------------------------

package second.Wednesday;

abstract class Abstractexample1

{

abstract void display();//method without body.

void display1() //concrete method

{

System.out.println("this is concrete method");

}

}

class BankDetails extends Abstractexample1

{

@Override //we are overriding a abstract method

void display()

{

System.out.println("this is a override method");

}

}

class Abstractexample

{

public static void main(String[] args)

{

BankDetails obj=new BankDetails();//we cannot instanciate a abstract class.

obj.display();

}

}

--------------------------------------------------------------------------

purpose of abstract class:-

---------------------------------------

abstract is a Story or brief about your project.

abstract class and abstract methods are incomplete class or method.

example:-

Bank project---client wants some methods for sure.withdraw(),deposite(),loan(),balancecheck()

so in the base class/parent class the team lead will declare these methods .

so that the developers can override and define these methods in the sub-class class.

--------------------------------------------------------------------------------------------------------------------------

Interface

--------------------

1)it is similar abstract class but we donot have concrete methods.(only abstract methods in it)

2)By interface we can do multiple inheritance.

3)we donot use the keyword abstract .

4)we cannot create object of a interface.we have to inherite a interface into a subclass.override the abstract methods and create object of the child class.

------------------------------------------------------------------------

package second.Wednesday;

interface SBIbank

{

int bal=1000;//if you declare a variable within a interface it is bydefault final and static.

void displaybal();

}

interface ICICIBank

{

void displaybalcheck();

}

interface IDBIBank extends ICICIBank //an interface can extend another interface.

{

void dispBal();

}

public class InterfaceExample implements SBIbank,IDBIBank

{

@Override

public void displaybal() {

System.out.println("SBIBank balance check");

}

@Override

public void displaybalcheck() {

System.out.println("ICICIBank balance check");

}

public static void main(String[] args) {

InterfaceExample ob=new InterfaceExample();

System.out.println(bal);

ob.displaybal();

ob.displaybalcheck();

ob.dispBal();

}

@Override

public void dispBal() {

System.out.println("this is IDBI Bank");

}

}

----------------------------------------------------------------------

Functional Interface :- A functional interface is a interface which has only one abstract method .

The Lambda expression is used to provide the implementation of an interface which has functional interface. It saves a lot of code

--------------------------------

package Third.wednesday;

interface bank

{

public void withdraw();//abstract method

}

class InterfaceExample

{

public static void main(String[] args)

{

int x=10;

bank ob=new bank() //functional Interface

{

@Override

public void withdraw()

{

System.out.println("this is override method");

System.out.println("the value of x is "+x);

}

};

ob.withdraw();

}

}

-----------------------------------------------------------------------------------------------

Functional interface using Lambda expression

-------------------------

package Third.wednesday;

@FunctionalInterface

interface bank

{

public void withdraw();

}

class InterfaceExample

{

public static void main(String[] args)

{

int x=10;

bank ob=()->

{//no need to override method

System.out.println("this is override method");

System.out.println("the value of x is "+x);

};

ob.withdraw();

}}

--------------------------------------------------

Lambda function with parameter

------------------------------------------

package Third.wednesday;

//with parameter

@FunctionalInterface

interface bank

{

public int withdraw(int amount);

}

class InterfaceExample

{

public static void main(String[] args)

{

bank ob=(amount)->

{

return amount;

};

System.out.println("the withdrawal amount is "+ob.withdraw(1000));

}}

--------------------------------------------------------------------------

package Third.wednesday;

//with multiple parameter

@FunctionalInterface

interface bank

{

public int withdraw(int amount1,int amount2);

}

class InterfaceExample

{

public static void main(String[] args)

{

bank ob=(amount1,amount2)->(amount1+amount2);

{

System.out.println("the withdrawal amount is "+ob.withdraw(1000,2000));

};

}}

-------------------------------------------------------------------------------------------

package Third.wednesday;

//with multiple parameter with return

@FunctionalInterface

interface bank

{

public int withdraw(int amount1,int amount2);

}

class InterfaceExample

{

public static void main(String[] args)

{

bank ob=(amount1,amount2)->

{

return (amount1+amount2);

};

System.out.println("the withdrawal amount is "+ob.withdraw(1000,2000));

}}

---------------------------------------------------------------------------------------------

Exception Handling/Error Handling

----------------------------------------------------

There are three types of errors

1)logical error/syntax error

2)Giving wrong input during runtime(wrong datatype)

3)Spelling mistake or missing of semi-colon etc.

-----------------------------------------------------------------------------------

Exception handling deals with runtime errors.

example:-

1)divide a number by zero

2)in place of integer passing String datatype

3)crossing the array limit.

4)reading/writing data into a file which is not present.

etc.

------------------------------------------------------------------

The main purpose of exception handling is to handle the error and stopping the program from terminating in between.

So that the program should excecute till the end .

The super class of exception is Throwable.

Throwable

i)Exception

ii)error

---------------------------------------------

we have 5 keywords used to handle the runtime error.

1)try

2)catch

3)finally

4)throw

5)throws

----------------------------------------------------------------------

Example 1:

package First.exception;

import java.util.\*;

public class ExceptionExample1

{

public static void main(String[] args)

{ try {

Scanner ob=new Scanner(System.in);

System.out.println("enter 2 nos");

int a=ob.nextInt();

int b=ob.nextInt();

int c=a/b;

System.out.println("the result is "+c);

}

catch(Exception ae)

{

System.out.println("This Exception is :"+ae);

}

System.out.println("This is divide by zero error");

}}

---------------------------------------------------------------

try-catch :-

In this if try is success then the catch is also success.Means there is error in try then the catch will display error message.

If try is failure means noo error then catch is also failure.The control will not flow inside the catch block.

--------------------------------------------------------------

try with multiple catch

--------------------------

package First.exception;

import java.util.\*;

public class ExceptionExample1

{

public static void main(String[] args)

{ try {

Scanner ob=new Scanner(System.in);

System.out.println("enter 2 nos");

int a=ob.nextInt();

int b=ob.nextInt();

int c=a/b;

System.out.println("the result is "+c);

}

catch(ArithmeticException ae) //handle specific exception

{

System.out.println("This Exception is :"+ae);

}

catch(Exception ae)//handle all types of exception

{

System.out.println("Exception is :"+ae);

}

System.out.println("This is divide by zero error");

}

}

----------------------------------------------------------------------------------------------

ArrayIndexOutOfBoundsException (crossing the array limit)

-----------------------------------------------------------------------------------------------

package First.exception;

import java.util.\*;

public class ExceptionExample1

{

public static void main(String[] args)

{ try{

int a[]=new int[5];

Scanner ob=new Scanner(System.in);

System.out.println("enter 5 nos");

for(int i=0;i<5;i++)

a[i]=ob.nextInt();

System.out.println("5 nos are");

for(int i=0;i<=5;i++)

System.out.println(a[i]);

}

catch(Exception ae)

{ System.out.println("Exception is :"+ae);

} }}

----------------------------------------------------------------------------------------------

try-catch-finally

-----------------------------------------

package First.exception;

import java.util.\*;

public class ExceptionExample1

{

public static void main(String[] args)

{ try{

int a[]=new int[5];

Scanner ob=new Scanner(System.in);

System.out.println("enter 5 nos");

for(int i=0;i<5;i++)

a[i]=ob.nextInt();

System.out.println("5 nos are");

for(int i=0;i<5;i++)

System.out.println(a[i]);

}

catch(ArrayIndexOutOfBoundsException ae)

{

System.out.println("Exception is :"+ae);

}

finally

{

System.out.println("we are inside finally block");

}

}}

---------------------------------------------------------

try-finally :-

In this the try is success or not it does not matter the finally block will definately exceute.

So Closing of connection,closing of class etc are written inside finally block.

package First.exception;

import java.util.\*;

public class ExceptionExample1

{

public static void main(String[] args)

{ try{

int a[]=new int[5];

Scanner ob=new Scanner(System.in);

System.out.println("enter 5 nos");

for(int i=0;i<5;i++)

a[i]=ob.nextInt();

System.out.println("5 nos are");

for(int i=0;i<=5;i++)

System.out.println(a[i]);

}

finally

{

System.out.println("we are inside finally block");

}

}}

------------------------------------------------------------------------------

throws exception :- this is used to handle mostly checked exception.

it does have body so the lines of code will be reduced.

-------------------------

package First.exception;

import java.util.\*;

public class ExceptionExample1

{

public static void main(String[] args) throws InterruptedException

{

for(int i=1;i<=10;i++)

{

Thread.sleep(1000);

System.out.println("This is delay of control");

}

System.out.println("This is the End");

}

}

------------------------------------------------------------------------

Unchecked Exceptions are runtion time exceptions.

example :- ArithmaticException,ArrayIndexOutOfBound,InputMisMatch

Checked Exceptions are compile time exception.

example:-IOException,SQLException,InterruptedException.

----------------------------------------------------------------------------------------------

throw exception:-

------------------------------

This is used for user defined exception.

example:- marks should be greater than zero.

salary should be greater than 15k.

etc.

-------------------------------------------------------------------------------------------------

example:-

package First.exception;

import java.util.\*;

public class ExceptionExample1

{

public static void main(String[] args) throws Exception

{

Scanner ob=new Scanner(System.in);

System.out.println("enter your salary");

int salary=ob.nextInt();

if(salary>15000)

System.out.println("your salary processed is ok");

else

throw new Exception("salary should be greater than 15000");

}

}

---------------------------------------------------------------------------------------------------

Collection framework

-----------------------------------------

Collections in java is a framework that provides an architecture to store and manipulate the group of objects.

All the operations that you perform on a data such as searching, sorting, insertion, manipulation, deletion etc. can be performed by Java Collections.

Java Collection simply means a single unit of objects. Java Collection framework provides many interfaces (Set, List, Queue, Deque) and classes (ArrayList, Vector, LinkedList,HashSet,LinkedHashSet, TreeSet etc).

---------------------------------------------------

array :- int a[]=new int[5];//array size is assigned

but in collection the size is dynamic.it expands as we enter the elements....

1)set(interface):-(classes)TreeSet,HashSet,LinkedHashSet

2)list (interface):-(classes)LinkedList,ArrayList,Vector,Stack

3)map(interface):-(classes)HashTable,HashMap

4)Queue (interface) maintains the first-in-first-out order.(classes) PriorityQueue, Deque, and ArrayDeque.

1.set(interface)->

Does not allow duplicate value.

HashSet,LinkedHashSet,TreeSet(classes).

a.Hashset->Display elements in random order.

b.LinkedHashSet ->it displays in the same order.

c.TreeSet ->It display in the sorted order.

------------------------------------------------------------------

2.List->

It allows duplicate value.

Stack,ArrayList,Linkedlist,Vector

Stack:FILO .the last element will be arr[0].

Linked List:it displays in same order.

-----------------------------

list iterator is used to view data forward and backward

-----------------------------------------------------------------------------------------

3.Map->

HashMap:-we enter Key and value pair

HashTable:-we enter Key and value pair

\*The HashMap class is roughly equivalent to Hashtable, except that it is unsynchronized and permits nulls.

(HashMap allows null values as key and value whereas Hashtable doesnt allow).

HashMap does not guarantee that the order of the map will remain constant every time.

HashMap is unsynchronized and Hashtable is synchronized.

Key-value

key should be unique- but value can be duplicate

-----------------------------------------------------------------------------------------

note:Iterator,ListIterator and enumeration are interfaces which contains the set of methods. with the help of these methods we can retrieve objects from the collection of classes.

-------------------------------------

Iterator:three methods:hasNext();next();remove();

----------------------------------------------------------------------------------------

Vector is synchronized whereas arraylist is not.

---------------------------------------------------------------------------------------

Autoboxing and Unboxing

autoboxing :- converting of primative datatype into its equivalent wrapper type is known as boxing.(int to Integer)

unboxing:-converting wrapper type into primative datatype is known as unboxing.(Integer to int)

------------------------------------------

example of autoboxing

-----------------------------------

class autobox

{

public static void main(String h[])

{

int a=50;

Integer b=new Integer(a);//autoboxing

Integer c=5;//autoboxing

System.out.println(b);

System.out.println(c);

}}

-----------------------------------------------------------

example of unboxing

-----------------------------------

class unbox

{

public static void main(String h[])

{

Integer a=new Integer(50);

int x=a;//unboxing

System.out.println(x);

}}

----------------------------------------------------

TypeCasting

--------------------------

---------------------------------------------------------

example-1(TreeSet -display in shorted order)

-----------------

package org.Collection;

import java.util.TreeSet;

public class Treeexample

{

public static void main(String[] args)

{

TreeSet ts=new TreeSet();

Integer x=new Integer(3);//x is an object which store int datatype value.

ts.add(10);//wraps the int to an object and store.

ts.add(14);

ts.add(56);

ts.add(67);

ts.add(8);

ts.add(x);

System.out.println(ts);

}

}

---------------------------------------------------------------------------------------

example-2

--------------------

package org.Collection;

public class employee

{

int empno;

String name,address;

public employee(int empno, String name, String address) {

super();

this.empno = empno;

this.name = name;

this.address = address;

}

@Override

public String toString() {

return "employee [empno=" + empno + ", name=" + name + ", address=" + address + "]";

}

}

---------------------------------------------------------------------

package org.Collection;

import java.util.\*;

public class Collectionexample1

{public static void main(String[] args)

{

LinkedList ts=new LinkedList();

employee e=new employee(101,"delip","hyd");

employee e1=new employee(102,"satya","hyd");

ts.add(e);

ts.add(e1);

System.out.println(ts);

}}

-----------------------------------------------------------------------------------------

example -3(LinkedHashSet :- it will display in the same order)

package org.Collection;

import java.util.\*;

public class Treeexample

{

public static void main(String[] args)

{

LinkedHashSet ts=new LinkedHashSet();

Integer x=new Integer(3);//x is an object which store int datatype value.

ts.add(10);//wraps the int to an object and store.

ts.add(14);

ts.add(56);

ts.add(67);

ts.add(8);

ts.add(x);

System.out.println(ts);

}

}

------------------------------------------------

example-4(HashSet:- This will display data in random order )

package org.Collection;

import java.util.\*;

public class Treeexample

{

public static void main(String[] args)

{

HashSet ts=new HashSet();

Integer x=new Integer(3);//x is an object which store int datatype value.

ts.add(10);//wraps the int to an object and store.

ts.add(14);

ts.add(56);

ts.add(67);

ts.add(8);

ts.add(x);

System.out.println(ts);

}

}

-------------------------------------------------------------------

example-5(Hashset/LiskedHashSet we can enter hetrogenious data)

package org.Collection;

import java.util.\*;

public class Treeexample

{

public static void main(String[] args)

{

HashSet hs=new HashSet();

Integer a=new Integer(10);

hs.add(a);

Float b=new Float(10.35);

hs.add(b);

String name=new String("raj");

hs.add(name);

employee obj=new employee(103,"Nitish","Noida");

hs.add(obj);

System.out.println(hs);

}

}

------------------------------------------------------------------------------------

example-6

package org.Collection;

import java.util.\*;

public class Treeexample

{

public static void main(String[] args)

{

HashSet hs=new HashSet();

System.out.println(hs.isEmpty());//return boolean value(true)

System.out.println(hs.size());

hs.add(10); hs.add(30); hs.add(40); hs.add(80); hs.add(100); hs.add(10);

System.out.println(hs);

System.out.println(hs.isEmpty());

System.out.println(hs.size());

System.out.println(hs.contains(30));

System.out.println(hs.contains(300));

hs.remove(100);

System.out.println(hs); hs.clear();System.out.println(hs); System.out.println(hs.size());

}}

--------------------------------------------------------------------------------------------

example-7(eterating using for each loop)

package org.Collection;

import java.util.\*;

public class Treeexample

{

public static void main(String[] args)

{

HashSet hs=new HashSet();

hs.add(10);

hs.add(30);

hs.add(40);

hs.add(80);

hs.add(100);

System.out.println(hs);

//To retrieve obj by for each

for(Object obj:hs)

{

System.out.println(obj);

} }}

----------------------------------------------------------------------------------

example-8(ArrayList -display in the same order)

package org.Collection;

import java.util.\*;

public class Treeexample

{

public static void main(String[] args)

{

ArrayList al=new ArrayList();

al.add(10);

al.add(50);

al.add(60);

al.add(100);

al.add(90);

al.add(80);

System.out.println(al);

al.add(3,500); //100 will move next position

System.out.println(al);

al.remove(2); //remove 60

System.out.println(al);

al.set(4,1000); //replace 90 with 1000

System.out.println(al);

} }

---------------------------------------------------------------------------------------

example-9(Stack -in this we use push and pop)

package org.Collection;

import java.util.\*;

public class Treeexample

{

public static void main(String[] args)

{

Stack ss=new Stack();//FILO

ss.push(10);

ss.push(20);

ss.push(30);

ss.push(40);

ss.push(50);

ss.push(60);

ss.push(70);

ss.push(10);

System.out.println(ss);

System.out.println(ss.pop());//remove item from stack

System.out.println(ss);

System.out.println(ss.peek());//detect last item in stack but dont delete

System.out.println(ss);

System.out.println(ss.search(20));//display the index start from 1

System.out.println(ss.search(200));//display -1 if not there

} }

------------------------------------------------------------------------------------

example-10(LinkedList :- adding two liked list)

package org.Collection;

import java.util.\*;

public class Treeexample

{

public static void main(String[] args)

{

LinkedList ll=new LinkedList();

LinkedList ll1=new LinkedList();

ll1.add(100); ll1.add(200); ll1.add(300); ll1.add(400);

ll.add(10); ll.add(40); ll.add(20); ll.add(80); ll.add(90); ll.add(50);

System.out.println(ll1); System.out.println(ll);

ll.addFirst(100);

System.out.println(ll);

ll.addLast(200);

System.out.println(ll);

ll.removeFirst();

ll.removeLast();

System.out.println(ll);

System.out.println(ll.getFirst());

System.out.println(ll.getLast());

ll.addAll(ll1);

System.out.println(ll);

} }

------------------------------------------------------------------------------------------------

//w.ap to create a employee class having empno,name,salary,designation.

store the employee object in a ArrayList and display it.

Enter the data at the run time.

---------------------------------------------------------------------------------------------------------

Example-11

package org.iiht.com;

import java.util.\*;

public class ArrayListExample

{

public static void main(String[] args) {

ArrayList<String> list=new ArrayList<String>();

list.add("Mango");

list.add("Apple");

list.add("Banana");

list.add("Grapes");

Collections.sort(list);//collection class has static methods like sort to sort the elements .

Iterator itr=list.iterator();

while(itr.hasNext())//return boolean value

{

System.out.println(itr.next());//returns the next element in the iteration

}

}

}

------------------------------------------------------------------------------------

Example-12

package org.iiht.com;

public class Student

{

int rollno;

String name;

int age;

Student(int rollno,String name,int age)

{

this.rollno=rollno;

this.name=name;

this.age=age;

}

}

---------------------------------------------------

package org.iiht.com;

import java.util.ArrayList;

import java.util.Iterator;

public class ArrayListExample1

{

public static void main(String[] args) {

Student s1=new Student(101,"amit",22);

Student s2=new Student(102,"sandip",23);

Student s3=new Student(103,"kiran",24);

ArrayList<Student> ob1=new ArrayList<Student>();

ob1.add(s1);

ob1.add(s2);

ob1.add(s3);

Iterator itr=ob1.iterator();

while(itr.hasNext())

{

Student st=(Student)itr.next();//typecasting the itr object to student type and stored in st object.

System.out.println(st.rollno+" "+st.name+" "+st.age);

}

}

}

-------------------------------------------------------------------------------------------

Create a class Bank having accno,name,balance,phoneno

insert the data and search the data according to accno.

if you enter a accno it should display all information.

-----------------------------------------------------------------------------------

Example-13

package org.Collection;

import java.util.\*;

class linklist\_listiterator

{

public static void main(String args[])

{

LinkedList ll=new LinkedList();

ll.add(10);ll.add(40);ll.add(20);ll.add(80);ll.add(90);ll.add(50);

ListIterator ii=ll.listIterator();

System.out.println("forward direction");

while(ii.hasNext())

{

System.out.println(ii.next());

}

System.out.println("backward direction");

while(ii.hasPrevious())

{

System.out.println(ii.previous());

}}}

----------------------------------------------------------------------------

example-14(HashMap:- in this we enter key and value pair.Key cannot be duplicate)

package org.Collection;

import java.util.\*;

class hashmap

{

public static void main(String args[])

{

Scanner obj=new Scanner(System.in);

System.out.println("enter the key and name");

int b=obj.nextInt();

String st=obj.next();

HashMap hs=new HashMap();

hs.put(1,"raj");

hs.put(2,"seetha");

hs.put(3,"reeta");

System.out.println(hs);

hs.put(4,"meetha");

hs.put(5,"venu");

hs.put(6,"pankaj");

hs.put(7,"raj");

hs.put(10,"lakshmi");

hs.put(10,"lllll");

hs.put(b,st);

System.out.println(hs);

}}

-----------------------------------------------------------------------

HashMap :- in this we enter key and value pair.

Set set=hs.entrySet(); //HashMap is converted to set we have entrySet() in HashMap .Returns a Set view of the mappings contained in this map

Iterator itr=set.iterator();

while(itr.hasNext())

{

Map.Entry en=(Map.Entry)itr.next();//typecasting the next() element with Map.Entry

System.out.println(en.getKey()+" "+en.getValue());//Map.Entry has 2 methods getKey() and getValue()

----------------------------------------------------------------------------------------

for(Map.Entry m:map.entrySet())

--------------------------------------------------------------------------------------

example-15

----------------------

package org.Collection;

import java.util.\*;

class hashmap

{

public static void main(String args[])

{

Scanner obj=new Scanner(System.in);

System.out.println("enter the key and name");

int b=obj.nextInt();

String st=obj.next();

HashMap hs=new HashMap();

hs.put(1,"raj");

hs.put(2,"seetha");

hs.put(3,"reeta");

System.out.println(hs);

hs.put(4,"meetha");

hs.put(5,"venu");

hs.put(6,"pankaj");

hs.put(7,"raj");

hs.put(10,"lakshmi");

hs.put(10,"lllll");

hs.put(b,st);

System.out.println(hs);

Set set=hs.entrySet(); //

Iterator itr=set.iterator();

while(itr.hasNext())

{

Map.Entry en=(Map.Entry)itr.next();

System.out.println(en.getKey()+" "+en.getValue());

}

}}

--------------------------------------------------------------------------------------------------

example-16

----------------------

package org.Collection;

import java.util.\*;

class MapExample2{

public static void main(String args[]){

Map<Integer,String> map=new HashMap<Integer,String>();

map.put(100,"Amit");

map.put(101,"Vijay");

map.put(102,"Rahul");

//Elements can traverse in any order

for(Map.Entry m:map.entrySet()){

System.out.println(m.getKey()+" "+m.getValue());

} }}

------------------------------------------------------------------------------------------

example-17

--------------------

import java.util.\*;

class Book

{

int id;

String name,author,publisher;

int quantity;

public Book(int id, String name, String author, String publisher, int quantity)

{

this.id = id;

this.name = name;

this.author = author;

this.publisher = publisher;

this.quantity = quantity;

}

}

----------------------------------------------------------------------------

public class MapExample {

public static void main(String[] args) {

//Creating map of Books

Map<Integer,Book> map=new HashMap<Integer,Book>();

//Creating Books

Book b1=new Book(101,"Let us C","Yashwant Kanetkar","BPB",8);

Book b2=new Book(102,"Data Communications & Networking","Forouzan","Mc Graw Hill",4);

Book b3=new Book(103,"Operating System","Galvin","Wiley",6);

//Adding Books to map

map.put(1,b1);

map.put(2,b2);

map.put(3,b3);

//Traversing map

for(Map.Entry<Integer, Book> entry:map.entrySet()){

int key=entry.getKey();

Book b=entry.getValue();

System.out.println(key+" Details:");

System.out.println(b.id+" "+b.name+" "+b.author+" "+b.publisher+" "+b.quantity);

}

}

}

------------------------------------------------------------------------------------------

Assignment

----------------------

In a cruise there are 3 types of tickets.

1)crew member :- free

2)Adult above 10 years:- Rs 500

3)Children below 10 years :-Rs 250

Enter details of all the passengers and count no of crew members ,Adult and children traveling .

Also display the details of all the travellers.Using HashMap.

----------------------------------------------------------------------------------------------------------

example-18

------------------

package org.Collection;

import java.util.\*;

public class HashMap2 {

public static void main(String args[]) {

HashMap<Integer,String> map=new HashMap<Integer,String>();

map.put(100,"Amit");

map.put(101,"Vijay");

map.put(102,"Rahul");

map.put(103, "Gaurav");

System.out.println("Initial list of elements: "+map);

//key-based removal

map.remove(100);

System.out.println("Updated list of elements: "+map);

//key-value pair based removal

map.remove(102, "Rahul");

System.out.println("Updated list of elements: "+map);

}

}

--------------------------------------------------------------------------------------------------

Example-19

---------------------

package org.Collection;

import java.util.\*;

public class CruiseClass {

public static void main(String[] args)

{

MemberType ob;

Scanner sc = new Scanner(System.in);

Map<Integer,MemberType> mapValue = new HashMap<Integer,MemberType>();

System.out.println("Enter the Number of Travelers You want On the Cruise");

int num = sc.nextInt();

for(int i=1; i<=num; i++)

{

mapValue.put(i, new MemberType());

}

//System.out.println("The Values are: "+mapValue.toString());

for(Map.Entry<Integer, MemberType> entry:mapValue.entrySet()){

int key=entry.getKey();

MemberType b=entry.getValue();

System.out.println(key+" Data:");

System.out.println("Name: " + b.name + "\nAge: " + b.age + "\nMember Type: " + b.memberTypes+"\nFees: "+b.fee);

}

System.out.println(MemberType.countMembers());

}

}

-------------------------------------------------------------------

package org.Collection;

import java.util.\*;

public class MemberType {

int age;

String name;

int memberType;

String memberTypes;

static int countCrew,countAdult,countChild;

int fee;

Scanner scan = new Scanner(System.in);

public MemberType()

{

System.out.println("Enter Your Age");

this.age=scan.nextInt();

System.out.println("Enter Your Name");

this.name=scan.next();

if(age>20) {

System.out.println("Type 1 for Crew Member or 2 for Adults");

this.memberType=scan.nextInt();

if(memberType==1)

{

this.memberTypes="Crew";

countCrew++;

fee=0;

}

else if(memberType==2)

{

this.memberTypes="Adult";

countAdult++;

fee=500;

}

}

else

{

this.memberTypes="Children";

countChild++;

fee=250;

}

}

public static String countMembers() {

return "Number of Crew Members: "+countCrew+"\nNumber of Adults: "+countAdult+"\nNumber of Children: "+countChild;

}

}

-------------------------------------------------------------------------------------

Thread :-

--------

The thread is a sort of execution of instruction.

In a program it start from public static void main(String arg[])

Statement start executing one after the other.

MultiThread :-

-------------------

In an application that is able to manage and coordinate multiple tasks simultaneously is called concurrent, multithread application.

Multithreaded applications make use of thread switching and scheduling that allow

multiple threads to make use of system resources.

To create a multithread application we have to implement runnable interface or extends Thread class.

In runnable interface we have one abstract method ->run()

In Thread class we have different methods.

Life cycle of Thread :-

-------------------------

new born thread,

start,

runnable(choose of thread),

running,

block/wait/sleep,

dead

methods of Thread:-

1)getName():-Obtain the thread name.

2)isAlive():- check if a thread is still running.

3)run() :-Entry point for the thread.

4)start():-Start a thread by calling the run method.

5)yield():-this method pauses the currently executing thread temporarily for giving chance to the remaining waiting threads of same priority to execute. if there is no waiting thread or all the waiting thread have a lower

priority then the same thread will continue its execution.

6)join():-The join() method of thread class waits for a thread to die. It is used when you want one thread to wait for completion of another. This process is like a relay race where the second runner waits until the first runner comes and hand over the flag to him.

7)sleep():-Based on our requirement we can make a thread to be in sleeping state

for a specified period of time.

8)setPriority():-To change the priority of the thread.

9)getPriority() :-To get the thread priority.

MIN\_PRIORITY=1 to 4

NORM\_PRIORITY=5

MAX\_PRIORITY=6 to 10

10)Daemon Thread() :-It is a low priority thread which run in the background doing the garbage collection operation .

11)wait() :-Thread will go to wait until some other thread doesnot notify.

12)notify() :- Wakes up a thread that called wait() on some thread.

13)notifyAll() :-wakes up all the thread that called wait()on some object.

14)The suspend() method of thread class puts the thread from running to waiting state. This method is used if you want to stop the thread execution and start it again when a certain event occurs. This method allows a thread to temporarily cease execution. The suspended thread can be resumed using the resume() method.

15)The holdLock() method of thread class returns true if the current thread holds the monitor lock on the specified object.

16)Java Thread interrupt() method

The interrupt() method of thread class is used to interrupt the thread. If any thread is in sleeping or waiting state (i.e. sleep() or wait() is invoked) then using the interrupt() method, we can interrupt the thread execution by throwing InterruptedException.

If the thread is not in the sleeping or waiting state then calling the interrupt() method performs a normal behavior and doesn't interrupt the thread but sets the interrupt flag to true.

Synchronization:-

--------------------

When two or more threads need access to a shared resource, need some way to

ensure that the resource will be used by only one thread at a time. The process by

which it is achieved is called synchronization.

DeadLock:-

-------------

When two threads are waiting each other and cannot procced the program is said to

be deadlock.

------------------------------------------------------------------------------------------------------

Example-1

-----------------

Flow of Program :-

1)we created 3 classes Thread1,Thread2,Thread3 each having run().

2)start() method will call the run().

3)we created objects of each class.

Thead1 obj=new Thead1();

Thead2 obj1=new Thead2();

Thead3 obj2=new Thead3();

4)we called the run methods of each class by using start method.

obj.start();

obj1.start();

obj2.start();

5)all the run methods start exceuting concurrently

6)to see that which thread is excuting we use a method

Thread.currentThread().getName()

7)JVM will allocate the first thread as Thread-0

----------------------------------------------------------------------------

package org.iiht.com;

class Thead1 extends Thread

{

public void run()

{

for(int i=1;i<=10;i++)

{

System.out.println(i);

try {

sleep(1000);

} catch (InterruptedException e)

{

e.printStackTrace();

}

}

}

}

class Thead2 extends Thread

{

public void run()

{

for(int i=1;i<=10;i++)

{

System.out.println(i);

try {

sleep(1000);

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}}

class Thead3 extends Thread

{

public void run()

{

for(int i=1;i<=10;i++)

{

System.out.println(i);

try {

sleep(1000);

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}}

public class Theadprg1

{

public static void main(String[] args)

{

System.out.println(obj.isAlive());

Thead1 obj=new Thead1();

Thead2 obj1=new Thead2();

Thead3 obj2=new Thead3();

obj.start();

obj1.start(); //all the run methods start exceuting concurrently

obj2.start();

System.out.println(obj.isAlive()); }

}

-----------------------------------------------------------

Example-2

-------------------

package org.iiht.com;

class Theadprg1 extends Thread

{

public void run()

{

for(int i=1;i<=10;i++)

{

System.out.println(i);

try {

sleep(1000);

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

public static void main(String[] args)

{

Theadprg1 obj=new Theadprg1();

Theadprg1 obj1=new Theadprg1();

Theadprg1 obj2=new Theadprg1();

System.out.println(obj.isAlive());

obj.start();

obj1.start();

obj2.start();

System.out.println(obj.isAlive());

}

}

-------------------------------------------------------------------------------------------

Example-3

------------------

package org.iiht.com;

public class JavaYieldExp extends Thread

{

public void run()

{

for (int i=0; i<3 ; i++)

System.out.println(Thread.currentThread().getName() + " in control");

}

public static void main(String[]args)

{

JavaYieldExp t1 = new JavaYieldExp();

JavaYieldExp t2 = new JavaYieldExp();

t1.start();

t2.setPriority(10);

t2.start();

System.out.println(t1.getPriority());

System.out.println(t2.getPriority());

for (int i=0; i<3; i++)

{

t1.yield();

System.out.println(Thread.currentThread().getName() + " in control");

}

}

}

-------------------------------------------------------------------------------------------------

Example-4

-----------------------

public class JoinExample1 extends Thread

{

public void run()

{

for(int i=1; i<=4; i++)

{

try

{

Thread.sleep(500);

}catch(Exception e){System.out.println(e);}

System.out.println(i);

}

}

public static void main(String args[])

{

// creating three threads

JoinExample1 t1 = new JoinExample1();

JoinExample1 t2 = new JoinExample1();

JoinExample1 t3 = new JoinExample1();

// thread t1 starts

t1.start();

// starts second thread when first thread t1 is died.

try

{

t1.join();

}catch(Exception e){System.out.println(e);}

// start t2 and t3 thread

t2.start();

t3.start();

}

}

-------------------------------------------------------------------------------------------------------

Example-5

-------------------------

package org.iiht.com;

public class JoinExample1 extends Thread

{

public void run()

{

for(int i=1; i<=4; i++)

{

try

{

Thread.sleep(500);

}catch(Exception e){System.out.println(e);}

System.out.println(Thread.currentThread().getName()+" "+i);

}

}

public static void main(String args[])

{

// creating three threads

JoinExample1 t1 = new JoinExample1();

JoinExample1 t2 = new JoinExample1();

JoinExample1 t3 = new JoinExample1();

// thread t1 starts

t1.start();

// starts second thread when first thread t1 is died.

try

{

t1.join();

}catch(Exception e){System.out.println(e);}

// start t2 and t3 thread

t2.start();

t3.start();

}

}

----------------------------------------------------------------------------

Example-6

--------------------

public class JavaSuspendExp extends Thread

{

public void run() {

for (int i = 1; i < 5; i++) {

try {

// thread to sleep for 500 milliseconds

sleep(500);

System.out.println(Thread.currentThread().getName());

} catch (InterruptedException e) {

System.out.println(e);

}

System.out.println(i);

}

}

public static void main(String args[]) {

// creating three threads

JavaSuspendExp t1 = new JavaSuspendExp();

JavaSuspendExp t2 = new JavaSuspendExp();

JavaSuspendExp t3 = new JavaSuspendExp();

// call run() method

t1.start();

t2.start();

// suspend t2 thread

t2.suspend();

t2.resume();

// call run() method

t3.start();

}

}

----------------------------------------------------------------------------

Example-7

------------------

package org.iiht.com;

class Second implements Runnable

{

synchronized public void run()

{

try

{

Thread t=Thread.currentThread();

String name=t.getName();

for(int i=0;i<10;i++)

{

System.out.println(name+"="+i);

Thread.sleep(500);

if(name.equals("raj") && (i==4))

{

wait();

}

if(name.equals("geeta") && (i==4))

{

wait();

}

if(name.equals("seeta") && (i==6))

{

System.out.println("raj and geeta thread wakes up...");

notifyAll();

}

if(name.equals("raj") && (i==9))

{

notify();

} }

}catch(Exception e){}

}

public static void main(String args[]) throws Exception

{

Second obj=new Second();

Thread t1=new Thread(obj,"raj");

Thread t2=new Thread(obj,"seeta");

Thread t3=new Thread(obj,"geeta");

t1.start();

t2.start();

t3.start();

}}

--------------------------------------------------------------

Function return a row

--------------------------------------------

CREATE TABLE account1 (

account\_id INT,

name VARCHAR2(20),

address varchar2(20)

);

--------------------------------------

account1%rowtype:- Indicate the all datatypes of the row of account1;

------------------------------------------------------------------------------------------

INSERT INTO account1 VALUES ( 1, 'Satya','Hyderabad' );

-----------------------------------------------------------------------------------------

CREATE OR REPLACE FUNCTION get\_accounts(

Acc\_id IN Account1.account\_id%TYPE) RETURN account1%ROWTYPE

AS

x account1%ROWTYPE;

BEGIN

SELECT \* INTO x FROM account1 WHERE account\_id = Acc\_id;

RETURN x;

END;

/

-----------------------------------------------------------------------------------

DECLARE

r\_acct ACCOUNT1%ROWTYPE;

BEGIN

r\_acct := get\_accounts( 1 );

DBMS\_OUTPUT.PUT\_LINE( r\_acct.name ||' ' || r\_acct.address);

END;

/

------------------------------------------------------------------------------------------------------------

Example-8

------------------------

package org.iiht.com;

class First implements Runnable

{

int available=2;

synchronized public void run()

{

try

{

Thread t=Thread.currentThread();

String name=t.getName();

if(available>0)

{

System.out.println(name+" got the ticket");

System.out.println("ticket printing.......");

Thread.sleep(2000);

available=available-1;

}

else

{ System.out.println(name+" sorry no ticket...."); }

}catch(Exception e){}

}}

class Second

{

public static void main(String args[]) throws Exception

{

First obj=new First();

new a();

Thread t1=new Thread(obj,"Surya");

Thread t2=new Thread(obj,"Kiran");

Thread t3=new Thread(obj,"Yash");

t1.start(); t2.start(); t3.start();

}

}

----------------------------------------------------------------------

Example-9

class Thread2 implements Runnable

{

synchronized public void run()

{

try

{

Thread t=Thread.currentThread();

String name=t.getName();

for(int i=0;i<10;i++)

{

System.out.println(name);

Thread.sleep(500);

}

}catch(Exception e){}

}

}

class a

{

public static void main(String args[]) throws Exception

{

Thread2 obj=new Thread2();

Thread t1=new Thread(obj,"1st Bike (Satya)");

Thread t2=new Thread(obj,"2nd Bike(Nitish)");

Thread t3=new Thread(obj,"3rd Bike(Prasanna)");

t1.start();

t2.start();

t3.start();

}

}

----------------------------------------------------------------------------------------------

File

---------

It is used to Store and manage data .

Reading or writing of data in a file can be done in byte or character format.

The process of reading and writing object into file is known as serialization.

The java.io package contain a large number of stream classes that provide capacity

for processing all types of data.

1)Byte stream class provide support for handling I/O operation on byte.

2)Character stream classes provide support for handling I/O operation on character.

Stream :-

---------------

Java uses the concept of streams to represent the ordered sequence of data ,

a common character shared by all the input/output device.

A stream in java is a path along which data flows.

It is a sequence of data or bytes traveling from source to destination.

There are 2 types of stream

1)byte stream

------------------

It has 2 abstract class

1)InputStream-read

2)OutputStream-write

-----------------------------------------------------------------------------------------------------

2)character stream

------------------------

There is has 2 abstract class

1)Reader 2)writer

-----------------------------------------------------------

IO Exception:-

-----------------

1)EOFException

2)FileNotFoundException

3)InterruptedIOException

4)IOException

--------------------

Serialization :- Serialization is the process of writing the data of an object to a

byte stream.

This is useful when we want to save the state of a program into storage area such

as file.

Later we restore these object by using deserialization.

Only an Object that implements the serializable interface can be saved and restored

by the serializable facilities .

The serializable interface have no methods(marker Interface)

The writeObject() method of ObjectOutputStreams used to serialize a object.

The readObject() method of ObjectIntputStreams used to deserialize a object.

Byte Stream:-

1)BufferInputStream/BufferOutputStream

2)FileInputStream/FileOutputStream

3)ObjectInputStream/ObjectOutputStream

4)DataInputStream/DataOutputStream

------------------------------------------------------------------------------------------------------------------------

Character Stream:-

1)BufferReader/BufferWriter

2)FileReader/FileWriter

------------------------------------------------------------

InputStreamReader/OutputStreamReader ->bridge from character stream to byte stream.

File class Methods:-

1)boolean isFile():- This method returns true if the file object contains a filename,

otherwise false.

2)booelan isDirectory():-This method returns true if the file object contains a

directory name.

3)boolean canRead():-This method returns true if the file object contains a file

which is readable.

4)booelan canWrite():-This method return true if the file is writeable.

5)booelan canExcecute:- This method return true if the file is executable.

6)boolean exists():-This method return true when the file object contains a file

or directory exists in the computer.

7)String getParent():- This method return the name of parent directory of a file or

directory.

8)String getAbsolutePath:- this method gives the absolute directory path.

9)long length():- This method returns a nummber that is the file size in bytes.

10)boolean delete():- This method deletes the file .

11)boolean createNewFile():- This method create a new file if file doesnot exists.

12)boolean mkdir():-this method create the directory .

example:-

import java.io.\*;

class FileDemo

{

public static void main(String arg[])

{

String fname=arg[0];

File f=new File(fname);

System.out.println("file name:"+f.getName());

System.out.println("file path:"+f.getPath());

System.out.println("file absolutepath:"+f.getAbsolutePath());

System.out.println("file exists:"+f.exists());

if(f.exists())

{

System.out.println("file canwrite:"+f.canWrite());

System.out.println("file canread:"+f.canRead());

System.out.println("file isdirectory:"+f.isDirectory());

System.out.println("file length:"+f.length());

}}}

----------------------------------------------------------------------------------------------------------------

Example-2

-----------------------

import java.io.\*;

class Second

{

public static void main(String args[]) throws IOException

{

DataInputStream dis=new DataInputStream(System.in);//input from the keyboard or read from the keyboard

FileOutputStream fos=new FileOutputStream("pqr.doc");//output to a file or write into the file.

System.out.println("enter the text");

int ch;

while((ch=dis.read())!='\n')

{

fos.write(ch);

}

fos.close();

}}

------------------------------------------

Example-3

--------------------------

//copy data from one file to file

import java.io.\*;

class d

{

public static void main(String args[]) throws Exception

{

FileInputStream fis=new FileInputStream("First.txt");//read the data from the file.

FileOutputStream fos=new FileOutputStream("demo2.doc");//write the data from the file.

int ch;

while((ch=fis.read())!=-1)//-1 indicates the end of the file.

{

fos.write(ch);

}

fis.close();fos.close();

System.out.println("file copied.....");

}}

--------------------------------------------------------

Example-4

-----------------------

import java.io.\*;

class e

{ //copy file to file

public static void main(String args[]) throws Exception

{

FileInputStream fis=new FileInputStream("test.txt");//open in read mode

FileOutputStream fos=new FileOutputStream("demo1.doc");//open in write mode

BufferedInputStream bis=new BufferedInputStream(fis);

BufferedOutputStream bos=new BufferedOutputStream(fos);

int ch;

while((ch=bis.read())!=-1)

{

bos.write(ch);

}

bos.close();fis.close();fos.close();

System.out.println("file copied.....");

}}

----------------------------------

Example-5

--------------------------

import java.io.\*;

class f

{

public static void main(String args[]) throws Exception

{

BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

System.out.println("enter the empid");

int emp=Integer.parseInt(br.readLine());

System.out.println("enter the emp name");

String name=br.readLine();

System.out.println("enter the emp sal");

float sal=Float.parseFloat(br.readLine());

System.out.println("enter the empid"+emp);

System.out.println("enter the emp name"+name);

System.out.println("enter the emp sal"+sal);

}}

------------------------------------------------------------------

Example-6

----------------------

import java.io.\*;

class g

{

public static void main(String args[]) throws Exception

{

FileReader fr=new FileReader("test.txt");//read data in character format

BufferedReader br=new BufferedReader(fr);

String name;

while((name=br.readLine())!=null)//null is EOF in character stream

{

System.out.print(name);

}

fr.close();

br.close();

System.out.println("file copied.....");

}}

-------------------------------------------------

Serialization :- Serialization is the process of writing the data of an object to a file.

we can do serialization by implementing Serializable inteface.It is a marker interface.(no abstract methods in it)

transient :- if we declare the variable as transient it will not be serializable.

-------------------------------------------------

example-7

----------------

import java.io.\*;

public class employee implements Serializable

{

public String name;

public String address;

public transient int SSN;

public int number;

}

---------------------------------------------------

import java.io.\*;

public class SerializeDemo

{

public static void main(String [] args)

{

employee e = new employee();

e.name = "sandip";

e.address = "jaynagar, 4th Block";

e.SSN = 11123;

e.number = 101;

try

{

FileOutputStream fileOut =new FileOutputStream("abc.txt");//file will be created and open in write mode.

ObjectOutputStream out = new ObjectOutputStream(fileOut);//write the object into the file

out.writeObject(e);

out.close();

fileOut.close();

System.out.printf("Serialized data is saved in abc.txt");

}catch(IOException i)

{

i.printStackTrace();

}

}

}

-------------------------------------------------------------------

import java.io.\*;

public class DeserializeDemo

{

public static void main(String [] args)

{

employee e = null;

try

{

FileInputStream fileIn = new FileInputStream("abc.txt");

ObjectInputStream in = new ObjectInputStream(fileIn);

e = (employee) in.readObject();

in.close();

fileIn.close();

}

catch(IOException i)

{

i.printStackTrace();

System.out.println(i);

}

catch(ClassNotFoundException c)

{

System.out.println("Employee class not found");

c.printStackTrace();

}

System.out.println("Deserialized Employee...");

System.out.println("Name: " + e.name);

System.out.println("Address: " + e.address);

System.out.println("sssn no: " + e.SSN);

System.out.println("Number: " + e.number);

}

}

-----------------------------------------------------------------------------------

JDBC

-------------------------

example-1

--------------------

package org.mphasis.jdbc;

import java.sql.\*;

public class Firstjdbc

{

public static void main(String[] args) throws Exception

{

Class.forName("oracle.jdbc.driver.OracleDriver");//loading of driver class

Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","1234");

Statement st=con.createStatement();

//st.execute("create table farhan(empno number,name varchar2(30),address varchar2(30))");

//System.out.println("Table created");

//st.execute("insert into farhan values(101,'farhan','Calcutta')");

//System.out.println("Row insereted");

//st.execute("update farhan set address='Bangalore' where empno=101");

//System.out.println("Row updated");

st.execute("delete from farhan where empno=101");

System.out.println("Row deleted");

}

}

--------------------------------------------------------------------------

example-2

-------------------------

package org.mphasis.jdbc;

import java.sql.\*;

import java.util.\*;

public class Firstjdbc

{

public static void main(String[] args) throws Exception

{

Scanner ob=new Scanner(System.in);

Class.forName("oracle.jdbc.driver.OracleDriver");//loading of driver class

Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","1234");

PreparedStatement ps=con.prepareStatement("insert into farhan values(?,?,?)");

System.out.println("enter empno,name,address");

int empno=ob.nextInt();

String name=ob.next();

String address=ob.next();

ps.setInt(1, empno);

ps.setString(2,name);

ps.setString(3,address);

ps.execute();

System.out.println("row inserted");

}}

--------------------------------------------------------

example-3

--------------------

package org.mphasis.jdbc;

import java.sql.\*;

import java.util.\*;

public class Firstjdbc

{

public static void main(String[] args) throws Exception

{

Scanner ob=new Scanner(System.in);

Class.forName("oracle.jdbc.driver.OracleDriver");//loading of driver class

Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","1234");

PreparedStatement ps=con.prepareStatement("insert into farhan(empno,name) values(?,?)");

System.out.println("enter empno,name");

int empno=ob.nextInt();

String name=ob.next();

ps.setInt(1, empno);

ps.setString(2,name);

ps.execute();

System.out.println("row inserted");

}}

--------------------------------------------------

example-4

---------------------

package org.mphasis.jdbc;

import java.sql.\*;

import java.util.\*;

public class Firstjdbc

{

public static void main(String[] args) throws Exception

{

Scanner ob=new Scanner(System.in);

Class.forName("oracle.jdbc.driver.OracleDriver");//loading of driver class

Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","1234");

PreparedStatement ps=con.prepareStatement("update farhan set name=?,address=? where empno=?");

System.out.println("enter name,address and empno to update");

String name=ob.next();

String address=ob.next();

int empno=ob.nextInt();

ps.setString(1,name);

ps.setString(2,address);

ps.setInt(3, empno);

ps.execute();

System.out.println("row updated");

}}

----------------------------------------------------------------------

example-5

-------------------------

public static void main(String[] args) throws Exception

{

Scanner ob=new Scanner(System.in);

Class.forName("oracle.jdbc.driver.OracleDriver");//loading of driver class

Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","admin");

PreparedStatement ps=con.prepareStatement("delete from farhan where empno=?");

System.out.println("enter empno");

int empno=ob.nextInt();

ps.setInt(1, empno);

ps.execute();

System.out.println("row Deleted");

}

-------------------------------------------------------------------

Story

-----------

WAP in java using case and switch.

create a bank table having fields accno,name,balance

A person can deposite

A person can withdraw amount.

A person can close his account.

Display the final table in oracle.

---------------------------------------------------------------------------------------------

example-6

----------------------

package org.mphasis.jdbc;

import java.sql.\*;

import java.util.\*;

public class Firstjdbc

{

public static void main(String[] args) throws Exception

{

Scanner ob=new Scanner(System.in);

Class.forName("oracle.jdbc.driver.OracleDriver");//loading of driver class

Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","1234");

Statement st=con.createStatement();

ResultSet rs=st.executeQuery("select \* from farhan");

while(rs.next())

{

System.out.println(rs.getInt(1)+" "+rs.getString(2)+" "+rs.getString(3));

}

}}

-----------------------------------------------------------------------------

JDBC (Java database connectivity)

----------------------------------------------------------------------

java(byte code) ----->driver(odbc)----------->database(oracle/sqlserver/MySql/MongoDb)(ascii code)

A java database connectivity is an application which will behave as an interface between the java program and the database.

Database driver are 2 types:-

-------------------------------------------

1)thick driver(java->driver->odbc->database)

2)thin driver(java->driver->database)

java have 4 driver:-

1)type 1 driver acts as a bridge between jdbc and other database connectivity mechanisms such as odbc.

java prg->jdbc api->jdbc driver->odbc driver->native api ->db

2)type 2 driver converts jdbc call into database vendor specific .

java prg->jdbc api->jdbc driver->native api ->db

3)type 3 driver translates jdbc calls into database server independent and middleware server specific net protocol calls .

java prg->jdbc api->middleware server ->db

4)type 4 driver is a pure java driver which implements the database protocol to interact directly with a database.

java prg->jdbc api->jdbc driver->db

To interact with database the necessary steps are :-

----------------------------------------------------------------

creating connection with jdbc using ODBC

------------------------------------------------------------

1)Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

load the driver class in the memory

2)Connection con=DriverManager.getConnection("jdbc:odbc:mphasis");

get connection from DriverManager

3)Statement st=con.createStatement();

get a statement from connection

4)ResultSet rs=st.executeQuery("select \* from emp");

ResultSet interface refer memory Buffer

using statement executeQuery

Sql package has 2 classes and 8 interfaces:-

--------------------------------------------------------

classes :-DriverManager,Types

interface :-Driver,Connection,Statement,PreparedStatement,CallableStatement,

ResultSet,ResultSetMetaData,DatabaseMetaData

there are 3 methods in statement interface

1)executeQuery()->select

2)executeUpdate()->update(return 0,1)

3)execute()->insert,delete,create,drop(return true,false)

java datatype:-String,int,double,JavaObject

Jdbc datatype:-varchar,char,number,number(7,2),clob,blob

To obtain data:-

getInt(),getString(),getDouble(),getFloat(),getByte(),getShort(),getBoolean(),getLong(),getObject(),getChar()

Statement ->

It can handle one statement at one time.

PreparedStatement->

It can handle multiple sql queries.

It also used for putting the values for the sql queries at the runtime.

It is faster than the statement

Callable Statement->

It is used to call stored functions and procedures of database.

Type 4 driver:-

-------------------

import java.sql.\*;

class SqlTest

{

public static void main(String arg[])throws Exception

{

Class.forName("oracle.jdbc.driver.OracleDriver");

Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","1234");

jdbc->protocal

oracle->database

thin->type4

localhost->local machine

1521->port no

xe->service id of oracle

system->username

1234->password

----------------------------------------------------------------------------------------------

CallableStatement st=con.prepareCall("{call addition(?,?,?)}");

st.setInt(1,10);

st.setInt(2,20);

st.registerOutParameter(3,Types.INTEGER);

st.execute();

int n=st.getInt(3);

System.out.println(n);

}

}

-----------------

create or replace procedure addition(i IN number,j IN number,K OUT number)

is

begin

K:=i+j;

end;

--------------------------

ResultSetMetaData:-

ResultSetMetaData md=rs.getMetaData();

System.out.println(md.getColumnCount());

System.out.println(md.getColumnName());

DatabaseMetaData:-

DatabaseMetaData md=con.getMetaData();

System.out.println(md.getDriverVersion());

System.out.println(md.getDriverName());

----------------------------------------------------------------------

MySql

--------------

package org.mphasis.jdbc;

import java.sql.\*;

class MysqlCon{

public static void main(String args[]){

try{

Class.forName("com.mysql.jdbc.Driver");

Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/sandip","root","1234");

//here sandip is database name, root is username and password

Statement stmt=con.createStatement();

ResultSet rs=stmt.executeQuery("select \* from product");

while(rs.next())

System.out.println(rs.getString(1)+" "+rs.getString(2)+" "+rs.getString(3));

con.close();

}catch(Exception e){ System.out.println(e);}

}

}

-----------------------------------------------------

Using mySql and Statement do

i)insert

ii)update

iii)delete

-----------------------------------------------------

do using PrepareStatement

i)insert

ii)update

iii)delete

-----------------------------------------------------

Real Time :-

1)connection class

2)Exception class

3)Interface

i)addEmployee()

ii)deleteEmployee()

iii)updateEmployee()

iv)viewEmployee()

4)model /entiry class

getter(),setter(),constructor(),toString()

5)implement class

6)Test class

i)insert

ii)update

iii)delete

iv)view

---------------------------------------------------------------

DAO example

-----------------------

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.Statement;

public class DbUtil {

private DbUtil() {}

public static Connection getConnection() throws DaoException {

try {

String driver, url, user, password;

driver = "com.mysql.jdbc.Driver";

//driver = "oracle.jdbc.driver.OracleDriver";

url = "jdbc:mysql://localhost/sandip";

//url="jdbc:oracle:thin:@localhost:1521:xe";

user = "root";

password = "1234";

Class.forName(driver);

return DriverManager.getConnection(url, user, password);

} catch (Exception e) {

throw new DaoException(e);

}

}

public static void releaseResource(Connection conn, Statement stmt, ResultSet rs)

throws DaoException {

try {

if(rs!=null){

rs.close();

}

if(stmt!=null){

stmt.close();

}

if(conn!=null){

conn.close();

}

} catch (Exception e) {

throw new DaoException(e);

}

}

}

------------------------------------------------------

public class DaoException extends Exception

{

private static final long serialVersionUID = 1L;

public DaoException() {

}

public DaoException(String message) {

super(message);

}

public DaoException(Throwable obj) {

super(obj);

}

}

-------------------------------------------------------------------

import java.sql.\*;

import java.util.\*;

public class Firstjdbc

{

public static void main(String[] args) throws Exception

{

Scanner ob=new Scanner(System.in);

Connection con=DbUtil.getConnection();

Statement st=con.createStatement();

ResultSet rs=st.executeQuery("select \* from product");

while(rs.next())

{

System.out.println(rs.getInt(1)+" "+rs.getString(2)+" "+rs.getString(3));

}

}}

--------------------------------------------------

1)create a new project .create a folder structure

2)create an entity class called books ---bookid,name,auther name,year of publish,cost

3)DAO:-addBook(),deleteBook(),updateBook(),viewBookById(),viewAllBooks()

4)DAOImpl

5)test to display the o/p

---------------------------------------------------------

ResultSet object can be moved forward only and it is not updatable.

But we can make this object to move forward and backward direction by passing either TYPE\_SCROLL\_INSENSITIVE or TYPE\_SCROLL\_SENSITIVE in createStatement(int,int) method as well as we can make this object as updatable by:

Statement stmt = con.createStatement(ResultSet.TYPE\_SCROLL\_INSENSITIVE,

ResultSet.CONCUR\_UPDATABLE);

----------------------------------------------------------------------------------

example-1

-------------------

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.SQLException;

public class MyBatchPreparedStmt {

public static void main(String a[]){

Connection con = null;

PreparedStatement pst = null;

try {

Class.forName("oracle.jdbc.driver.OracleDriver");

con = DriverManager.

getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","1234");

con.setAutoCommit(false);

pst = con.prepareStatement("update mphasisemp set salary=? where empid=?");

pst.setInt(1, 30000);

pst.setInt(2, 101);

pst.addBatch();

pst.setInt(1, 40000);

pst.setInt(2, 102);

pst.addBatch();

int count[] = pst.executeBatch();//returns an array of update counts.

for(int i=1;i<count.length;i++){

System.out.println("Query "+i+" has effected "+count[i]+" times");

}

con.commit();

} catch (ClassNotFoundException e) {

try {

con.rollback();

} catch (SQLException e1) {

e1.printStackTrace();

}

e.printStackTrace();

} catch (SQLException e) {

try {

con.rollback();

} catch (SQLException e1) {

e1.printStackTrace();

}

e.printStackTrace();

} finally{

try{

if(pst != null) pst.close();

if(con != null) con.close();

} catch(Exception ex){}

}

}

}

-------------------------------------------------------------------------

example-2

---------------

import java.sql.\*;

class Rsmd{

public static void main(String args[]){

try{

Class.forName("oracle.jdbc.driver.OracleDriver");

Connection con=DriverManager.getConnection( "jdbc:oracle:thin:@localhost:1521:xe","system","1234");

PreparedStatement ps=con.prepareStatement("select \* from mphasisemp");

ResultSet rs=ps.executeQuery();

ResultSetMetaData rsmd=rs.getMetaData();

System.out.println("Total columns: "+rsmd.getColumnCount());

System.out.println("Column Name of 1st column: "+rsmd.getColumnName(1));

System.out.println("Column Type Name of 1st column: "+rsmd.getColumnTypeName(1));

con.close();

}catch(Exception e){ System.out.println(e);}

}

}

----------------------------------------------------------------------------

Example-3

--------------------

import java.sql.\*;

class SqlTest

{

public static void main(String arg[])throws Exception

{

Class.forName("oracle.jdbc.driver.OracleDriver");

Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","1234");

CallableStatement st=con.prepareCall("{call addition(?,?,?)}");

st.setInt(1,10);

st.setInt(2,20);

st.registerOutParameter(3,Types.INTEGER);

st.execute();

int n=st.getInt(3);

System.out.println("The result is "+n);

}

}

/\*

\* create or replace procedure addition(i IN number,j IN number,K OUT number) is

\* begin K:=i+j;

\* end;

\*/

--------------------------------------------------------------------------------

example-4

-------------------

import java.sql.\*;

class SqlTest1

{

public static void main(String arg[])throws Exception

{

Class.forName("oracle.jdbc.driver.OracleDriver");

Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","1234");

CallableStatement cs=con.prepareCall("{?=call add\_pro(?,?)}");

cs.registerOutParameter(1,Types.INTEGER);

cs.setInt(2,11);

cs.setInt(3,22);

cs.execute();

int n=cs.getInt(1);

System.out.println("the result is "+n);

}

}

/\*

create or replace function add\_pro(i IN number,j IN number)return number is

K number;

begin

k:=i+j;

return K;

end;

\*/

-------------------------------------------------------------------------

example-5

----------------------

import java.sql.\*;

class SqlTest2

{

public static void main(String arg[])throws Exception

{

Class.forName("oracle.jdbc.driver.OracleDriver");

Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","1234");

CallableStatement cs=con.prepareCall("{call dept\_pro}");

cs.execute();

}

}

/\*

\* create or replace procedure dept\_pro

\* as

\* begin

\* delete from dept where deptno=50;

\* end;

\*/

-----------------------------------------------------------------------------

example-6

-----------------------------

import java.sql.\*;

class SqlTest3

{

public static void main(String arg[])throws Exception

{

Class.forName("oracle.jdbc.driver.OracleDriver");

Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","1234");

CallableStatement stmt=con.prepareCall("{call insertR(?,?)}");

stmt.setInt(1,101);

stmt.setString(2,"Amit");

stmt.execute();

System.out.println("success");

}

}

/\*

create or replace procedure "INSERTR"

(id IN NUMBER,

name IN VARCHAR2)

is

begin

insert into users values(id,name);

end;

/

-------------

create table users(id number(10), name varchar2(200));

--------------------

\*/

---------------------------------------------------

Example-7

--------------------------

import java.sql.\*;

class FetchRecord{

public static void main(String args[])throws Exception{

Class.forName("oracle.jdbc.driver.OracleDriver");

Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","1234");

Statement stmt=con.createStatement(ResultSet.TYPE\_SCROLL\_SENSITIVE,

ResultSet.CONCUR\_UPDATABLE);

ResultSet rs=stmt.executeQuery("select \* from mphasisemp");

//getting the record of 3rd row

rs.absolute(3);

System.out.println(rs.getString(1)+" "+rs.getString(2)+" "+rs.getString(3));

rs.last();

System.out.println(rs.getString(1)+" "+rs.getString(2)+" "+rs.getString(3));

rs.first();

System.out.println(rs.getString(1)+" "+rs.getString(2)+" "+rs.getString(3));

con.close();

}}

------------------------------------------------------

Assignment

-------------------

entity class

-----------------------

import java.util.Date;

public class Product {

private int pId;

private String productName;

private Date dateOfManuf;

private Date dateOfExp;

private int quantity;

private double price;

}

-------------------------------------------------------

DButil class

------------------

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

public class OracleDBConnection {

private static Connection connection;

public static Connection getConnection() throws ClassNotFoundException, SQLException {

Class.forName("oracle.jdbc.OracleDriver");

connection = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "system", "1234");

return connection;

}

}

-------------------------------------------------------

main class

-------------

System.out.println("Enter the id");

int id = scan.nextInt();

System.out.println("Enter the product name");

String pName = scan.next();

System.out.println("Enter the Manufacture date");

String mafDate = scan.next();

System.out.println("Enter the expiry date");

String expDate = scan.next();

System.out.println("Enter the quantity");

int qty = scan.nextInt();

System.out.println("Enter the price");

double price = scan.nextDouble();

Connection con = OracleDBConnection.getConnection();

System.out.println("1. Insert");

System.out.println("2. Read");

System.out.println("3. Exit");

System.out.println("Please select the option..");

--------------------------------------------------------------------------------------------

--------------------------------------------------------------------------

Design Patterns

----------------------

We must use design pattern during the analysis and requirement phase of SDLC.

It is used to provide solution or design to the application .

It provide transparency to the design of a application.

It is divided into two category.

1)Core java design patterns

2)JEE design pattern

-----------------------------------

creational Design pattern

---------------------------------------

1)Factory

2)Abstract Factory

3)Singleton

4)Prototype

5)Builder

Structural Design pattern

-------------------------------------

1)Adapter

2)Bridge

3)composite

4)decorator

5)Facade

6)proxy

Behavioral Design pattern

--------------------------------------------

1)command

2)Interpreter

3)Iterator

4)Mediator

5)Observer

6)Template

7)State

------------------------------------------------------------------------

singleton design pattern

-------------------------------------------------

It define a class tht has only one instance and provides a global point of access to it.

It saves memory because object is not created again and again.only a single instance is reused again and again.

To create a singleton class we need have static method or variables.

We should have private constructor.so that only within the class object is created not outside.

example:-creating a static method getConnection() and defining connection insde it and using it through out the application.

------------------------------------------------------------------------

Factory Design Pattern

-------------------------------------

It define an interface or abstract class for creating an object but let the subclass decide which class to instantiate.

------------------------------

What is an Algorithm?

An algorithm is a process or a set of rules required to perform calculations or some other problem-solving operations especially by a computer. The formal definition of an algorithm is that it contains the finite set of instructions which are being carried in a specific order to perform the specific task.

--------------------------------------------------------------------------------------------

The data structure name indicates itself that organizing the data in memory.

Types of Data Structures

There are two types of data structures:

-----------------------------------------------------

Primitive data structure

Non-primitive data structure

Primitive Data structure

----------------------------------------------

The primitive data structures are primitive data types. The int, char, float, double, and pointer are the primitive data structures that can hold a single value.

----------------------------------------------------------------

Non-Primitive Data structure

----------------------------------------

The non-primitive data structure is divided into two types:

------------------------------------------------------------------

Linear data structure

Non-linear data structure

----------------------------------------

Linear Data Structure

----------------------------------------------

The arrangement of data in a sequential manner is known as a linear data structure. The data structures used for this purpose are Arrays, Linked list, Stacks, and Queues.

--------------------------------------------------------------------

The major or the common operations that can be performed on the data structures are:

----------------------------------------------------------------------------------------------------------------------

Searching: We can search for any element in a data structure.

Sorting: We can sort the elements of a data structure either in an ascending or descending order.

Insertion: We can also insert the new element in a data structure.

Updation: We can also update the element, i.e., we can replace the element with another element.

Deletion: We can also perform the delete operation to remove the element from the data structure.

--------------------------------------------------------------

public class BubbleSort {

public static void main(String[] args) {

int[] a = {10, 9, 7, 101, 23, 44, 12, 78, 34, 23};

for(int i=0;i<10;i++)

{

for (int j=0;j<10;j++)

{

if(a[i]<a[j])

{

int temp = a[i];

a[i]=a[j];

a[j] = temp;

}

}

}

System.out.println("Printing Sorted List ...");

for(int i=0;i<10;i++)

{

System.out.println(a[i]);

}

}

}

--------------------------------------------------------------------

public class SinglyLinkedList {

//Represent a node of the singly linked list

class Node{

int data;

Node next;

public Node(int data) {

this.data = data;

this.next = null;

}

}

//Represent the head and tail of the singly linked list

public Node head = null;

public Node tail = null;

//addNode() will add a new node to the list

public void addNode(int data) {

//Create a new node

Node newNode = new Node(data);

//Checks if the list is empty

if(head == null) {

//If list is empty, both head and tail will point to new node

head = newNode;

tail = newNode;

}

else {

//newNode will be added after tail such that tail's next will point to newNode

tail.next = newNode;

//newNode will become new tail of the list

tail = newNode;

}

}

//display() will display all the nodes present in the list

public void display() {

//Node current will point to head

Node current = head;

if(head == null) {

System.out.println("List is empty");

return;

}

System.out.println("Nodes of singly linked list: ");

while(current != null) {

//Prints each node by incrementing pointer

System.out.print(current.data + " ");

current = current.next;

}

System.out.println();

}

public static void main(String[] args) {

SinglyLinkedList sList = new SinglyLinkedList();

//Add nodes to the list

sList.addNode(1);

sList.addNode(2);

sList.addNode(3);

sList.addNode(4);

//Displays the nodes present in the list

sList.display();

}

}

---------------------------------------------------------------------------------------

Count no of nodes

-----------------------------

public class CountNodes {

//Represent a node of singly linked list

class Node{

int data;

Node next;

public Node(int data) {

this.data = data;

this.next = null;

}

}

//Represent the head and tail of the singly linked list

public Node head = null;

public Node tail = null;

//addNode() will add a new node to the list

public void addNode(int data) {

//Create a new node

Node newNode = new Node(data);

//Checks if the list is empty

if(head == null) {

//If list is empty, both head and tail will point to new node

head = newNode;

tail = newNode;

}

else {

//newNode will be added after tail such that tail's next will point to newNode

tail.next = newNode;

//newNode will become new tail of the list

tail = newNode;

}

}

//countNodes() will count the nodes present in the list

public int countNodes() {

int count = 0;

//Node current will point to head

Node current = head;

while(current != null) {

//Increment the count by 1 for each node

count++;

current = current.next;

}

return count;

}

//display() will display all the nodes present in the list

public void display() {

//Node current will point to head

Node current = head;

if(head == null) {

System.out.println("List is empty");

return;

}

System.out.println("Nodes of singly linked list: ");

while(current != null) {

//Prints each node by incrementing pointer

System.out.print(current.data + " ");

current = current.next;

}

System.out.println();

}

public static void main(String[] args) {

CountNodes sList = new CountNodes();

//Add nodes to the list

sList.addNode(1);

sList.addNode(2);

sList.addNode(3);

sList.addNode(4);

//Displays the nodes present in the list

sList.display();

//Counts the nodes present in the given list

System.out.println("Count of nodes present in the list: " + sList.countNodes());

}

}

-------------------------------------------------------------------------------------

delete the first node

-----------------------------------

public class DeleteStart {

//Represent a node of the singly linked list

class Node{

int data;

Node next;

public Node(int data) {

this.data = data;

this.next = null;

}

}

//Represent the head and tail of the singly linked list

public Node head = null;

public Node tail = null;

//addNode() will add a new node to the list

public void addNode(int data) {

//Create a new node

Node newNode = new Node(data);

//Checks if the list is empty

if(head == null) {

//If list is empty, both head and tail will point to new node

head = newNode;

tail = newNode;

}

else {

//newNode will be added after tail such that tail's next will point to newNode

tail.next = newNode;

//newNode will become new tail of the list

tail = newNode;

}

}

//deleteFromStart() will delete a node from the beginning of the list

public void deleteFromStart() {

//Checks if the list is empty

if(head == null) {

System.out.println("List is empty");

return;

}

else {

//Checks whether the list contains only one node

//If not, the head will point to next node in the list and tail will point to the new head.

if(head != tail) {

head = head.next;

}

//If the list contains only one node

//then, it will remove it and both head and tail will point to null

else {

head = tail = null;

}

}

}

//display() will display all the nodes present in the list

public void display() {

//Node current will point to head

Node current = head;

if(head == null) {

System.out.println("List is empty");

return;

}

while(current != null) {

//Prints each node by incrementing pointer

System.out.print(current.data + " ");

current = current.next;

}

System.out.println();

}

public static void main(String[] args) {

DeleteStart sList = new DeleteStart();

//Adds data to the list

sList.addNode(1);

sList.addNode(2);

sList.addNode(3);

sList.addNode(4);

//Printing original list

System.out.println("Original List: ");

sList.display();

while(sList.head != null) {

sList.deleteFromStart();

//Printing updated list

System.out.println("Updated List: ");

sList.display();

}

}

}

----------------------------------------------------

delete last node

----------------------------------

public class DeleteStart {

//Represent a node of the singly linked list

class Node{

int data;

Node next;

public Node(int data) {

this.data = data;

this.next = null;

}

}

//Represent the head and tail of the singly linked list

public Node head = null;

public Node tail = null;

//addNode() will add a new node to the list

public void addNode(int data) {

//Create a new node

Node newNode = new Node(data);

//Checks if the list is empty

if(head == null) {

//If list is empty, both head and tail will point to new node

head = newNode;

tail = newNode;

}

else {

//newNode will be added after tail such that tail's next will point to newNode

tail.next = newNode;

//newNode will become new tail of the list

tail = newNode;

}

}

//deleteFromStart() will delete a node from the beginning of the list

public void deleteFromStart() {

//Checks if the list is empty

if(head == null) {

System.out.println("List is empty");

return;

}

else {

//Checks whether the list contains only one node

//If not, the head will point to next node in the list and tail will point to the new head.

if(head != tail) {

head = head.next;

}

//If the list contains only one node

//then, it will remove it and both head and tail will point to null

else {

head = tail = null;

}

}

}

//display() will display all the nodes present in the list

public void display() {

//Node current will point to head

Node current = head;

if(head == null) {

System.out.println("List is empty");

return;

}

while(current != null) {

//Prints each node by incrementing pointer

System.out.print(current.data + " ");

current = current.next;

}

System.out.println();

}

public static void main(String[] args) {

DeleteStart sList = new DeleteStart();

//Adds data to the list

sList.addNode(1);

sList.addNode(2);

sList.addNode(3);

sList.addNode(4);

//Printing original list

System.out.println("Original List: ");

sList.display();

while(sList.head != null) {

sList.deleteFromStart();

//Printing updated list

System.out.println("Updated List: ");

sList.display();

}

}

}

----------------------------------------------------------------------------------------

wap to enter a customer details and product details

product-----productid,name,price,quantity

customer ----customerid,name,phoneno,product.

and display the details.

-------------------------------------------------------------------------------------------

1)to which package wait(),notify(),notifyAll() belongs to -

ans:- java.lang.Object

-------------------------------------------------------------------------------

2)public class test1

{

public static void main(String[] args) {

int b=Integer.parseInt("123a");

System.out.println("the value is "+b);

}

}

ans:- NumberFormatException

-----------------------------------------------------------------------------------

3)

public class foo

{

int a=10;

public static void main(String[] args) {

System.out.println("the value is "+a);

}

}

ans:- compilation error

--------------------------------------------------------------------------------

4)no duplicate entry is allowed.

ans :- set

-----------------------------------------------------------------------

5)sorted order :- TreeMap

------------------------------------------------------------------------

6)user defined Exception :- throw

-----------------------------------------------------------------------------

7)arbitry thread move from wait pool to lock pool

ans :-notify

-----------------------------------------------------------------------------------

8)read(),close() will throw ans :-ans:- IOException

------------------------------------------------------------------------------------

9)create new file method throw which exception :-

ans :- IOException

--------------------------------------------------------------------------------

10)primative data enter in file :- DataInputStream

-----------------------------------------------------------------------------------

11)

int total=0;

for(int i=0, j=10;total > 30;++i,--j)

{

total+=i+j;

System.out.println(i+" "+j);

}

System.out.println(total);

}

ans:- 0

-------------------------------------------------------------------------------------

12)transient :-

int empno;

String name,address;

transient int phoneno;

when you convert to object the by using serialization phone no will ans :- not be serialized.

-----------------------------------------------------------------------------------

13)runnable interface

public class abc implements runnable

{

public void run()

{}

}

------------------------------------------------------------------

14)interface can be nested

----------------------------------------------------------------------------------

15)int a[5] ;

system.in.read(a[1])

ans:- Compilation error

-------------------------------------------------------------------------------------

16)to execute thread one by one we use:-

ans :- synchronized.

------------------------------------------------------------------------------

17)function to copy an array :- ans :- System.arraycopy

----------------------------------------------------------------------------------------

18)stack operation :-ans push and pop

--------------------------------------------------------------------------------

19)which collection is used to shrink the data :-

ans :- Java.Util.ArrayList

------------------------------------------------------------------------------

20)StringBuffer and StringBuilder

ans :-1)String Builder is not synchronized and faster than StringBuffer.

-------------------------------------------------------------------

21)exception with FileDataWriting

ans :-FileNotFoundException

----------------------------------------------------------

22)

public class Test implements runnable

{

public void run()

{

System.out.println("Mphasis call run()");

}

public static void main(String args[])

{

new Thread(new Test()).start();

}}

ans:-Mphasis call run()

--------------------------------------------------------

23)

HashSet ar=new HashSet();

System.out.println(ar.add("abce"));

System.out.println(ar.add("abcf"));

System.out.println(ar.add("abcg"));

System.out.println(ar.add("abce"));

ans:-

true

true

true

false

------------------------------------

24)

setLoc= new HashSet();

setLoc.put("a",Integer(1));

setLoc.put("b",Integer(2));

setLoc.put("c",Integer(3));

System.out.println(b);

ans:- complile time error

-----------------------------------------------

25) interface methods can be declare as final,public and static

--------------------------------------------------------------------------

26)a class can implements more than one interface

-----------------------------------------------------------------------------

27)comparator is present int java.util package

-------------------------------------------------------------------

28)

public class Test implements runnable

{

public void run()

{

sop("foo");

}

public void run(String s)

{

sop("bar");

}}

public static void main(String args())

{

Thread ob=new Thread(new Test())

ob.start();

}

------------------

ans:-foo

----------------------------------------------------------------------------------

29)what does replace do ?

It will replace all the occurace by invoking the string with another character

-----------------------------------------------------------------------------------------

30)

FileInputStream fin;

int c=0;

while(c=fin.eof()!= -1)

{

fin=new FileInputStream(args[0]);

((char)c));

}

ans:- TypeCast Error

--------------------------------------------------------------------------------------------

1)Which of the following is not a component/class of JDBC API?

DriverManager

Driver

Connection

Transaction

choice 4

2)In which of the following type of ResultSet, the cursor can only move forward in the result set?

ResultSet.TYPE\_FORWARD\_ONLY

ResultSet.TYPE\_SCROLL\_INSENSITIVE ResultSet.TYPE\_SCROLL\_SENSITIVE

None of the above.

choice 1

3)Which of the following type of JDBC driver, is also called Type 4 JDBC driver?

JDBC-ODBC Bridge plus ODBC driver

Native-API, partly Java driver

JDBC-Net, pure Java driver

Native-protocol, pure Java driver

choice 4

4)Which of the following is used to call stored procedures on the database?

Statement

PreparedStatement

CallableStatement

None of the above

choice 3

5)Which of the following consists of methods for contacting a database?

DriverManager

JDBC driver

Connection

Statement

choice 3

6)Which of the following is a Metadata interfaces of JDBC?

DatabaseMetaData

ResultSetMetaData

Both of the above.

None of the above

choice 3

7)Which of the following is not interfaces?

DriverManager

Connection

Statement

ResultSet

choice 1

8)Database meta data are retrieved through \_\_\_\_\_\_\_\_\_\_\_\_.

ResultSet Object

PreparedStatement object

Connection object

Statement object

choice 3

9)SQL \_\_\_\_\_\_\_\_ statements may not change the contents of a database

DELETE

UPDATE

SELECT

INSERT

choice 3

10)Result set meta data are retrieved through \_\_\_\_\_\_\_\_\_\_\_\_

PreparedStatement object

ResultSet Object

Connection object

Statement object

choice 2

11)Invoking Class.forName method may throw \_\_\_\_\_\_\_\_\_\_\_

ClassNotFoundException

RuntimeException

SQLException

IOException

choice 1

12)What information may be obtained from a ResultSetMetaData object?

number of columns in the result set

number of rows in the result set database URL and product name

JDBC driver name and version

choice 1

13)RowSet is an extension of \_\_\_\_\_\_\_

Statement

CLOB

ResultSet

Connection

choice 3

14)You can store images in a database using data type \_\_\_\_\_\_\_

BLOB

varchar2

CLOB

varchar

choice 1

15)In a scrollable and updateable resultset, you can use \_\_\_\_\_\_\_\_\_\_\_ methods on a result set

updateRow()

first()

deleteRow()

insertRow()

all of the above

choice 5

16)\_\_\_\_\_\_\_\_ is an attribute or a set of attributes that uniquely identifies the relation.

A key

A superkey

A primary key

A candidate key

choice 2

17)What are the JDBC statements

Statement

PreparedStatement

CallableStatement

All of the above

choice 4

18)What is JDBC Driver

JDBC-ODBC bridge driver

Native-API driver

Network Protocol driver

Thin driver

all of the above

choice 5

19)What are the steps to connect to the database in java

Registering the driver class

Creating connection

Creating statement

Executing queries

all of the above

choice 5

20)What are the JDBC API components

Connection

Statement

PreparedStatement

ResultSet

all of the above

choice 5

21)What are the classes in JDBC API

Blob

Clob

Types

all of the above

choice 3

22)IBM DB2,oracle sybas,informax.....

ans :- native api

23)DDL (alter,create,drop)

ans:- exceuteUpdate(),execute()

24)which driver is not vendor specific

ans:- type 1 and type 3

25)ascii and unicode character

ans:- getCharacterStream()

27)which driver has 3 tier architecture

ans :- type 3

28)seven digit mantissa

ans:- real

29)isSearchable(int column) returns -----------------

ans :- binary values

30)ResultSet object can be moved in forward direction and can be updatable

ans :- false

31)data like table,view and,procedure,function are stored in

ans :- DataBaseMetadata

32)date and time

java.util.date

java.sql.timestamp

33)How many ResultSet avilable in JDBC

ans:- 3

1)ScrollInsensitive

2)Scrollsensitive

3)FORWARD\_only

34)which model do the jdbc api support fro database access.

ans:-2 and 3 tier

35)How can we get which field is primary key .

ans:- database metadata object.

-----------------------------------------------------------------------------------------

1)getPrimarykey() belongs to DataBaseMetaData

2)which type of driver is 3 tier Architecture ---Type 3

3)which type of driver is not vendor specific-----Type 1,3

4)

Connection con;

PreparedStatement ps=con.prepareStatement("create table abc(empno int)");

ps.executeUpdate();

ps.close();

ans:- sqlException

5)

Connection con;

PreparedStatement ps=con.prepareStatement("select \* from abc");

ResultSet rs=ps.executeQuery();

ps.close();

ans:- error in :-ResultSet rs=ps.executeQuery();

6)what happens when setAutocommit(true) ?

ans:- all indivisual sql statements are treated as transactions.

and will be commited after execution.

7)when the resultset cursor goto the last record

ans:- -1

8)executeUpdate(),execute(),exceuteQuery() which is suitable for DDL (create,drop,alter)

ans:- execute(),executeUpdate()

DML (insert,update,delete):-execute(),executeUpdate()

9)for out we will use RegisterOutParameter();

10)we cannot have table with same column names.

-------------------------------------------------------------------------