**Assignment-3**

**1.File Input Stream and File Output Stream in java**

package com.cts.io;

import java.io.File;

// Write the text into the document

public class Testfile {

public static void main(String args[]){

File f=new File("C:\\dontknow\\text.txt");

if(f.exists())

{

System.out.println(f.getName());

System.out.println(f.getParent());

System.out.println(f.canRead());

System.out.println(f.canWrite());

System.out.println("heelo everyone”);

}

}

}

Output;

This is ok bro

**2.Read character by character by character**

package com.cts.io;

import java.io.FileReader;

import java.io.IOException;

//Read File char by char

public class ReadFileCharByChar {

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

//File f=new File("C:\\dontknow\\test.txt");

FileReader reader=new FileReader("C:\\dontknow\\test.txt");

int ch= reader.read();

while(ch!=-1)

{

System.out.println(ch+"===>"+(char)ch);

ch=reader.read();

}

System.out.println("ty");

}

}

Output:

104===>h

101===>e

101===>e

108===>l

111===>o

32===>

101===>e

118===>v

101===>e

114===>r

121===>y

111===>o

110===>n

101===>e

hy

**3.Line by Line**

package com.cts.io;

import java.io.FileReader;

import java.io.IOException;

import java.io.BufferedReader;

//Read line by line

public class Linebyline {

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

FileReader reader=new FileReader("C:\\dontknow\\test.txt");

BufferedReader br=new BufferedReader(reader);

String line=br.readLine();

while(line!=null) {

System.out.println(line);

line=br.readLine();

}

}

}

Output: heelo everyone

**4. This Program for writing something in file**

package com.cts.io;

import java.io.IOException;

import java.io.File;

import java.io.PrintWriter;

import java.io.FileWriter;

//This is program something writing in the file

class Testfile1 {

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

FileWriter file=new FileWriter("C:\\dontknow\\test.txt");

{

PrintWriter pw= new PrintWriter(file);

pw.append("This is program for writing something in file");

pw.close();

file.close();

}

}

}

Output:heelo everyone

Testout.txt: This is program for writing something in file heelo everyone

**5. Coping text**

package com.cts.file;

import java.io.\*;

//copying text from one text document to another text document

public class Copy {

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

FileOutputStream fout1=new FileOutputStream("D:\\f1.txt");

FileOutputStream fout2=new FileOutputStream("D:\\f2.txt");

ByteArrayOutputStream bout=new ByteArrayOutputStream();

bout.write(65);

bout.writeTo(fout1);

bout.writeTo(fout2);

bout.flush();

bout.close();//has no effect

System.out.println("Success...");

}

}

Output:

Success...

f1.txt:

A

f2.txt:

A

**2.MultiThreading**

**1.New born Thread**

package com.cts.ml;

class Thread1 extends Thread{

public void run(){

System.out.println("Hello lekhana!");

}

}

public class Threadeg1 {

public static void main(String[] args) {

//New born thread

Thread1 t1=new Thread1();

t1.start();

System.out.println("hyyy ------->>>");

}

}

Output:

hyyy ------->>>

Hello lekhana!

**2.Using try and catch block**

package com.cts.ml;

import java.io.IOException;

//indiathread extends form the thread

class IndiaThread extends Thread

{

public void run(){

String name[]={"Virat","Dhoni","Sachin"};

for(int i=0;i<name.length;i++){

//try and catch block

try{

Thread.sleep(500);

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

}catch(InterruptedException e){

e.printStackTrace();

}

}

}

}

//pak extends from thread

class PakThread extends Thread

{

public void run(){

String name[]={"saklen","Afridi","Mustapa"};

for(int i=0;i<name.length;i++){

//try and catch block

try{

Thread.sleep(500);

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

}catch(InterruptedException e){

e.printStackTrace();

}

}

}

}

class SrilankaThread extends Thread

{

public void run(){

String name[]={"Malinga","Mulidar","Sangakara"};

for(int i=0;i<name.length;i++){

//try and catch block

try{

Thread.sleep(500);

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

}catch(InterruptedException e){

e.printStackTrace();

}

}

}

}

public class Thread2 {

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

//creating object

IndiaThread india=new IndiaThread();

india.start();

india.join();

PakThread pak=new PakThread();

pak.start();

SrilankaThread sri=new SrilankaThread();

sri.start();

}

}

Output:

Virat

Dhoni

Sachin

saklen

Malinga

Afridi

Mulidar

Sangakara

Mustapa

**3. Using the Thread class :Thread(String Name)**

package com.cts.ml;

import java.io.IOException;

// Using the Thread Class: Thread(String Name)

public class Thread3 {

// Main method

public static void main(String args[])

{

// creating an object of the Thread class using the constructor Thread(String name)

Thread t= new Thread("My first thread");

// the start() method moves the thread to the active state

t.start();

// getting the thread name by invoking the getName() method

String str = t.getName();

System.out.println(str);

}

}

**4. Using the Thread Class: Thread(Runnable r, String name)**

package com.cts.ml;

public class Thread4 implements Runnable{

{

public void run()

{

System.out.println("Now the thread is running ..."); }

// main method

public static void main(String args[])

{

// creating an object of the class MyThread2

Runnable r1 = new MyThread4();

// creating an object of the class Thread using Thread(Runnable r, String name)

Thread3 th3 = new Thread3(r1, "My new thread");

// the start() method moves the thread to the active state

th3.start();

// getting the thread name by invoking the getName() method

String str = th3.getName();

System.out.println(str);

}

}

}

Output: My new threadNow the thread is running ...

My new thread

Now the thread is running ...

**5. the sleep() Method in Java : on the main thread**

package com.cts.ml;

import java.lang.Thread;

import java.io.\*;

//import statements

// Using sleep() method in java

public class Thread5 {

public static void main(String args[])

{

//main method

try {

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

// the main thread sleeps for the 1000 milliseconds,whic is 1sec

//when ever the loop runs

Thread.sleep(1000);

//displaying the value of the variable

System.out.println(i);

}

}catch(Exception e) {

System.out.println(e);

}

}

}

Output: 0

1

2

3

4

5

6

7

8

9

**3.Java8**

**1. Lamda Expressions**

**a) package com.cts.java8;**

//Using Lamdaexpression

interface Drawable{

public void draw();

}

public class Lamda1ex {

public static void main(String args[])

{

int width=15;

//with lamda

Drawable d2=()->{

System.out.println("Drawing"+width);

};

d2.draw();

}

}

Output: Drawing15

**B)**

**package com.cts.java8;**

//interface the shape

interface Shape

{

public void draw();

}

//implements from shape

class Rectangle implements Shape

{

public void draw() {

System.out.println("Shape:Rectangle");

}

}

class Circle implements Shape

{

public void draw()

{

System.out.println("Shape:circle");

}

}

class Square implements Shape{

public void draw()

{

System.out.println("Shape:Square");

}

}

public class Lamda2ex {

//main method

public static void main(String args[])

{

// Shape rectangle=new Rectangle();

//rectangle.draw();

Shape rectangle= ()->{

System.out.println("Shape:Rectangle");};

rectangle.draw();

Shape circle=()->{

System.out.println("Shape:circle");};

circle.draw();

Shape square=()->{

System.out.println("Shape:square");};

square.draw();

}

}

Output: Shape:Rectangle

Shape:circle

Shape:square

**c) calculations**

package com.cts.java8;

interface Calculation{

void cal(int a, int b);

}

class Sum implements Calculation

{

//passing parameters

public void cal(int a,int b) {

System.out.println("c="+(a+b));

}

}

public class Lamda3ex {

//main method

public static void main(String args[]) {

Calculation sum=(a,b)->{

int c=a+b;

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

};

sum.cal(20, 96);

}

}

Output: c==116

2.Function Interface

a)with out parameter and with out return type

package com.cts.java8;

//with out parameter and with out return type

interface Calcu{

void cal();

}

class Multiplication implements Calcu

{

public void cal() {

int a=20, b=89;

int c=a\*b;

System.***out***.println("c"+c);

}

}

public class funI1 {

public static void main(String args[])

{

Calcu mul=()->{

int a=20,b=89;

int c=a\*b;

System.***out***.println("c=="+c);

};

mul.cal();

}

}

Output: c==1780

**b)With parameter and with out return type**

package com.cts.java8;

//with parameter and with out return type

interface Calcu1{

void cal(int a,int b);

}

class Multiplication1 implements Calcu1

{

public void cal(int a, int b) {

int c=a\*b;

System.***out***.println("c"+c);

}

}

public class FunI2 {

//input and not output

//with parameter and with out return type

public static void main(String args[])

{

Calcu1 mul1=(a,b)->{

int c=a\*b;

System.***out***.println("c=="+c);

};

mul1.cal(20,10);

}

}

Output: c==200

**C).With out parameter and with return type**

package com.cts.java8;

interface Calcu2{

int cal();

}

class Multiplication2 implements Calcu2

{

public int cal() {

int a=10,b=10;

int c=a\*b;

return(c);

}

}

public class FunI3 {

//without parameter and with return type

public static void main(String args[])

{

Calcu2 mul2=()-> {

int a=10,b=10;

int c=a\*b;

return(c);

};

System.***out***.println(mul2.cal());

}

}

Output: 100

**D)With parameter and with return type**

package com.cts.java8;

//with parameter and with return type

interface Calcu3{

int cal(int a,int b);

}

class Multiplication4 implements Calcu3

{

public int cal(int a, int b) {

int c=a\*b;

return(c);

}

}

public class FunI4 {

//input and not output

//with parameter and with out return type

public static void main(String args[])

{

Calcu3 mul3=(a,b)->{

int c=a\*b;

return(c);

};

System.***out***.println( mul3.cal(20,10));

}

}

Output: 200

**3.Method Interface**

**a)Static Method Interface**

package com.cts.java8;

interface ShowAble{

public void message();

}

//Static Method reference

class SMS{

public static void ShowSMS()

{

System.***out***.println("Hyyy from SMS");

}

}

public class MethodI1 {

public static void main(String args[])

{

//Static Method reference

ShowAble sms=SMS::*ShowSMS*;

sms.message();

}

}

Output:Hyyy from SMS

**b)Instance method reference**

package com.cts.java8;

interface ShowAble{

public void message();

}

//Instance Method reference

class Latter{

public void ShowLatter()

{

System.***out***.println("Hyyy from Latter");

}

}

public class MethodI1{

public static void main(String args[])

{

//Instance Method reference

Latter latter=new Latter();

ShowAble ShowLatter=latter::ShowLatter;

ShowLatter.message();

}

}

Output:Hyy from Latter

**c)Using constructer method reference**

package com.cts.java8;

interface ShowAble{

public void message();

}

//constructor Method reference

class Mail{

Mail()

{

System.***out***.println("Hyyy from Mail");

}

}

public class MethodI1{

public static void main(String args[])

{

//constructor Method reference

ShowAble mail=Mail::new;

mail.message();

}

}

Output: Hyyy from Mail

**Sql commands**

1.SQL CREATE TABLE

SQL CREATE TABLE statement is used to create table in a database.

If you want to create a table, you should name the table and define its column and each column's data type.

Syntax:

create table "tablename"

("column1" "data type",

"column2" "data type",

"column3" "data type",

...

"columnN" "data type");

2.SQL DROP Database

The SQL Drop Database statement deletes the existing database permanently from the database system. This statement deletes all the views and tables if stored in the database, so be careful while using this query in SQL.

Syntax:

DROP DATABASE Database\_Name;

3.SQL RENAME Database

In some situations, database users and administrators want to change the name of the database for some technical reasons. So, the Rename Database statement in SQL is used to change the name of the existing database.

Syntax:

ALTER DATABASE old\_database\_name MODIFY NAME = new\_database\_name;

4.SQL SELECT Database

Suppose database users and administrators want to perform some operations on tables, views, and indexes on the specific existing database in SQL. Firstly, they have to select the database on which they want to run the database queries.

Syntax;USE database\_name;

5.SQL TRUNCATE TABLE

A truncate SQL statement is used to remove all rows (complete data) from a table. It is similar to the DELETE statement with no WHERE clause.

Syntax:TRUNCATE TABLE table\_name;

6.SQL COPY TABLE

If you want to copy the data of one SQL table into another SQL table in the same SQL server, then it is possible by using the SELECT INTO statement in SQL.

Syntax:SELECT \* INTO New\_table\_name FROM old\_table\_name;

7.SQL TEMP TABLE

Temporary tables can be created at run-time and can do all kinds of operations that a normal table can do. These temporary tables are created inside tempdb database.

There are two types of temp tables based on the behavior and scope.

Local Temp Variable

Global Temp Variable

Local Temp Variable

Local temp tables are only available at current connection time. It is automatically deleted when user disconnects from instances. It is started with hash (#) sign.

CREATE TABLE #local temp table (

User id int,

Username varchar (50),

User address varchar (150)

)

Global Temp Variable

Global temp tables name starts with double hash (##). Once this table is created, it is like a permanent table. It is always ready for all users and not deleted until the total connection is withdrawn.

CREATE TABLE ##new global temp table (

User id int,

User name varchar (50),

User address varchar (150)

)

8.SQL ALTER TABLE Add

The ALTER TABLE statement in Structured Query Language allows you to add, modify, and delete columns of an existing table.

SyntaxALTER TABLE table\_name ADD column\_name column-definition;

9.Alter table modify

Syntax:ALTER TABLE table\_name MODIFY column\_name column-definition;

10.Alter table Rename

Syntax:ALTER TABLE Employee RENAME COLUMN Emp\_City to Emp\_Address;

11.SQL SELECT Statement

The SELECT statement is the most commonly used command in Structured Query Language. It is used to access the records from one or more database tables and views. It also retrieves the selected data that follow the conditions we want.

Syntax:SELECT Column\_Name\_1, Column\_Name\_2, ....., Column\_Name\_N FROM Table\_Name;

12.Select with Where condition

Syntax:SELECT \* FROM Name\_of\_Table WHERE [condition];

13.SQL SELECT UNIQUE

Actually, there is no difference between DISTINCT and UNIQUE.

SELECT UNIQUE is an old syntax which was used in oracle description but later ANSI standard defines DISTINCT as the official keyword.

Syntax;SELECT UNIQUE column\_name

FROM table\_name;

14.SQL SELECT DISTINCT

The SQL DISTINCT command is used with SELECT key word to retrieve only distinct or unique data.

Syntax:SELECT DISTINCT column\_name ,column\_name

FROM table\_name;

15.SQL SELECT COUNT

The SQL COUNT() is a function that returns the number of records of the table in the output.

Syntax:SELECT COUNT(column\_name) FROM table\_name;

16.SQL SELECT TOP

The SELECT TOP statement in SQL shows the limited number of records or rows from the database table. The TOP clause in the statement specifies how many rows are returned.

Syntax:SELECT TOP number | percent column\_Name1, column\_Name2, ....., column\_NameN FROM table\_name WHERE [Condition] ;

17.SQL SELECT FIRST

The SQL first() function is used to return the first value of the selected column.

syntax ;

SELECT FIRST(column\_name) FROM table\_name;

18.SQL ORDER BY CLAUSE WITH ASCENDING ORDER

This statement is used to sort data in ascending order. If you miss the ASC attribute, SQL ORDER BY query takes ascending order by default.

example of supplier:

SELECT supplier\_city

FROM suppliers

WHERE supplier\_name = 'IBM'

ORDER BY supplier\_city;

19.SQL ORDER BY CLAUSE WITH DESCENDING ORDER:

This statement is used to sort data in descending order.

SELECT supplier\_city

FROM suppliers

WHERE supplier\_name = 'IBM'

ORDER BY supplier\_city DESC;

20.SQL JOIN

.As the name shows, JOIN means to combine something. In case of SQL, JOIN means "to combine two or more tables".

The SQL JOIN clause takes records from two or more tables in a database and combines it together.

SQL defines five types of JOIN :

inner join,

left outer join,

right outer join,

full outer join, and

cross join.

21.SQL SELECT DATE

SQL SELECT DATE is used to retrieve a date from a database. If you want to find a particular date from a database, you can use this statement.

Syntax:SELECT \* FROM

table-name WHERE your date-column >= '2013-12-12'

22.SQL SELECT SUM

It is also known as SQL SUM() function. It is used in a SQL query to return summed value of an expression.

Syntax:SELECT SUM (expression)

FROM tables

WHERE conditions;

23.SQL LEFT JOIN

The SQL left join returns all the values from the left table and it also includes matching values from right table, if there are no matching join value it returns NULL.

SYNTAX FOR LEFT JOIN:

SELECT table1.column1, table2.column2....

FROM table1

LEFTJOIN table2

ON table1.column\_field ;

24.SQL FULL JOIN

The SQL full join is the result of combination of both left and right outer join and the join tables have all the records from both tables. It puts NULL on the place of matches not found.

Syntax:SELECT \*

FROM table1

FULL OUTER JOIN table2

ON table1.column\_name = table2.column\_name;

25.SQL PRIMARY KEY

A column or columns is called primary key (PK) that uniquely identifies each row in the table.

If you want to create a primary key, you should define a PRIMARY KEY constraint when you create or modify a table.

CREATE TABLE students

(

S\_Id int NOT NULL PRIMARY KEY,

LastName varchar (255) NOT NULL,

FirstName varchar (255),

Address varchar (255),

City varchar (255),

)