**(EXPERIMENT -1)**

**Method OverLoading**

class OverLoad{

public static int addnum(int a, int b){

return a+b;

}

public static int addnum(int a, int b, int c){

return a+b+c;

}

public static double addnum(double a, double b){

return a+b;

}}

class main{

public static void main (String[] args){

int result1 = OverLoad.addnum(5,4);

int result2 = OverLoad.addnum(3,5,7);

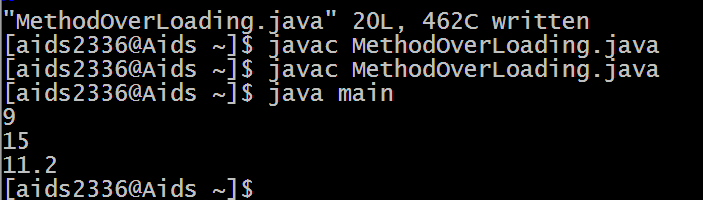
double result3 = OverLoad.addnum(4.5,6.7);

System.out.println(result1);

System.out.println(result2);

System.out.println(result3);

}}



**(EXPERIMENT -2)**

**String Tokenizer**

import java.util.\*;

class STDemo

{

public static void main(String args[])

{

int sum=0;

Scanner sc=new Scanner(System.in);

System.out.println("Enter integers with one space gap");

String str=sc.nextLine();

StringTokenizerst=new StringTokenizer(str," ");

while(st.hasMoreTokens())

{

String token=st.nextToken();

int t=Integer.parseInt(token);

System.out.println(t);

sum+=t;

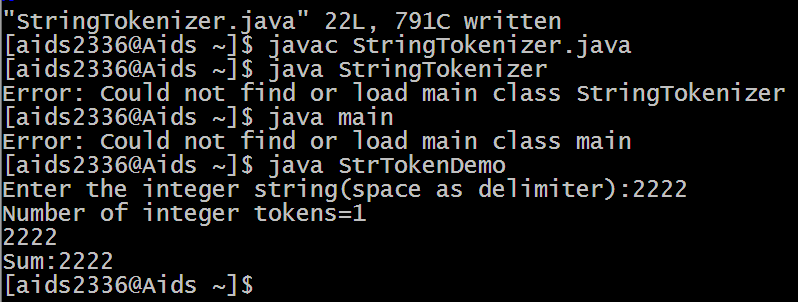
}

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

sc.close();

}

}

****

**(EXPERIMENT-3.1)**

**SingleLevel Inheritance**

class Employee

{

float salary=40000;

}

class Inherited extends Employee

{

int bonus=10000;

public static void main(String args[])

{

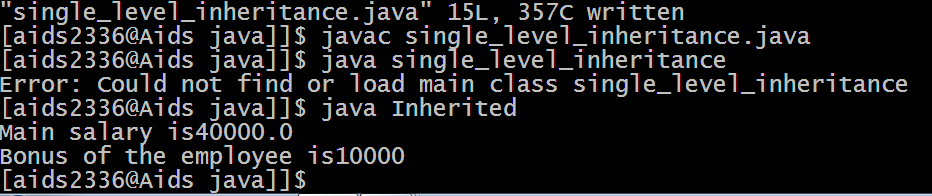
Inherited p=new Inherited();

System.out.println("Main salary is"+p.salary);

System.out.println("Bonus of the employee is"+p.bonus);

}

}



**(EXPERIMENT-3.2)**

**MultiLevel Inheritance**

class A

{

int i;

int j;

void showij()

{

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

}

void sum()

{

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

}

}

class B extends A

{

int k;

void showk()

{

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

}

void sum()

{

System.out.println("i+j+k="+(i+j+k));

}

}

class C extends B

{

int l;

void showl()

{

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

}

void sum()

{

System.out.println("i+j+k+l="+(i+j+k+l));

}

}

class MIDemo

{

public static void main(String args[])

{

A superObj=new A();

B subObj=new B();

superObj.i=10;

superObj.j=15;

subObj.i=20;

subObj.j=25;

subObj.k=30;

System.out.println("Content of super class");

superObj.showij();

superObj.sum();

System.out.println("Contents of sub class");

subObj.showij();

subObj.showk();

subObj.sum();

C subLastObj=new C();

subLastObj.i=1;

subLastObj.j=2;

subLastObj.k=3;

subLastObj.l=4;

System.out.println("Contents of last subclass");

subLastObj.showij();

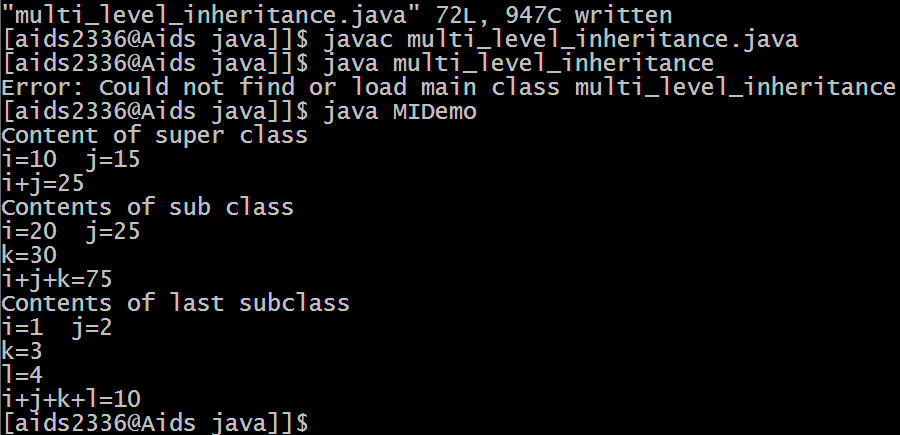
subLastObj.showk();

subLastObj.showl();

subLastObj.sum();

}

}



**(EXPERIMENT -4)**

**Multiple Inheritance**

interface Bank

{

double getROI();

}

class SBI implements Bank

{

public double getROI()

{

return 7.5;

}

}

class HDFC implements Bank

{

public double getROI()

{

return 10.0;

}

}

class HSRC implements Bank

{

public double getROI()

{

return 5.0;

}

}

class BankDemo

{

public static void main (String args[])

{

SBI sbi=new SBI();

HDFC hdfc=new HDFC();

HSRC hsrc=new HSRC();

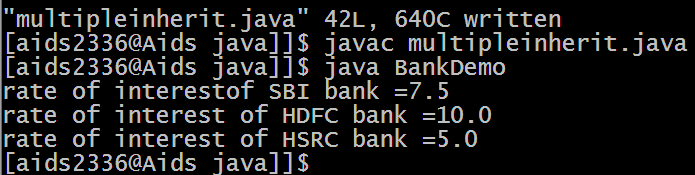
System.out.println("rate of interestof SBI bank ="+ sbi.getROI());

System.out.println("rate of interest of HDFC bank ="+ hdfc.getROI());

System.out.println("rate of interest of HSRC bank ="+ hsrc.getROI());

}

}

****

**(EXPERIMENT 5)**

**Exception Handling**

class UserExp{

public static void validate(int age) throws InvalidAgeExp

{

if(age < 18){

throw new InvalidAgeExp("Not eligable to vote");

}

else{

System.out.println("Eligable to vote");

}

}

public static void main(String args[]){

try{

validate(1);

}

catch(InvalidAgeExp e){

System.out.println(e);

}

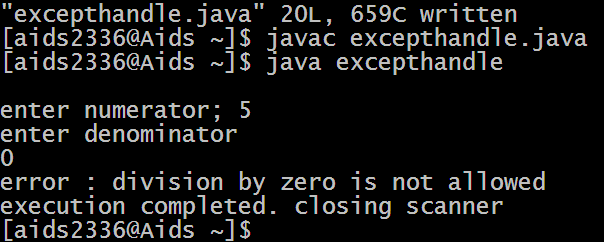
finally{

System.out.println("in finally block");

}

System.out.println("rest of code");

}}



**(EXPERIMENT 6a)**

**Multi Threading by extending Thread Class**

class NewThread extends Thread

{

NewThread()

{

super("Child thread");

this.start();

}

public void run()

{

try{

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

System.out.println("Child thread" + i);

Thread.sleep(500);

}

}

catch(InterruptedException e)

{

System.out.println(e);

}

System.out.println("Exiting Child Thread");

}

public static void main(String args[])

{

new NewThread();

try{

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

{

System.out.println("Main Thread" + i);

Thread.sleep(2000);

}

}

catch(InterruptedException e)

{

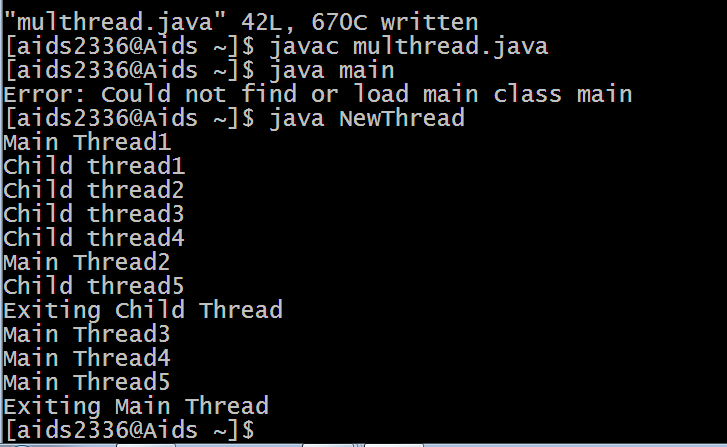
System.out.println(e);

}

System.out.println("Exiting Main Thread");

}

}



**(EXPERIMENT 6b)**

Multiple Threading by implementing Runnable

class NewThread1 implements Runnable

{

Thread t;

NewThread1(String tName)

{

t= new Thread(this, tName);

t.start();

}

public void run(){

try{

for(int i=5; i>0;i--){

Thread.sleep(500);

}

}

catch(InterruptedException e){

System.out.println(e);

}

System.out.println("exiting child thread"+Thread.currentThread().getName());

}

public static void main(String args[]){

new NewThread1("One");

new NewThread1("Two");

new NewThread1("Three");

try{

for(int i=5; i>0; i--){

System.out.println("Main thread: "+i);

Thread.sleep(3000);

}

}

catch(InterruptedException e){

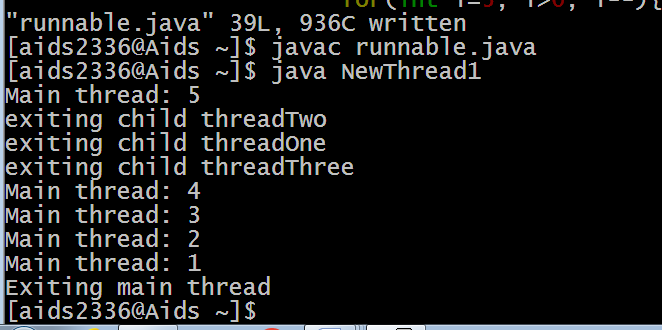
System.out.println(e);

}

System.out.println("Exiting main thread");

}

}



**(EXPERIMENT 7)**

**Synchronized method**

class Table

{

synchronized void printTable(int n)

{

try

{

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

{

System.out.println(n\*i);

Thread.sleep(500);

}

}

catch (InterruptedException e){

System.out.println(e);

}

}

}

class MyThread1 extends Thread

{

Table t;

MyThread1(Table t)

{

this.t=t;

}

public void run()

{

t.printTable(5);

}

}

class MyThread2 extends Thread

{

Table t;

MyThread2(Table t)

{

this.t=t;

}

public void run()

{

t.printTable(100);

}

}

class SyncDemo

{

public static void main(String args[])

{

Table obj = new Table();

MyThread1 t1 = new MyThread1(obj);

MyThread2 t2 = new MyThread2(obj);

t1.start();

t2.start();

try {

t1.join();

t2.join();

}

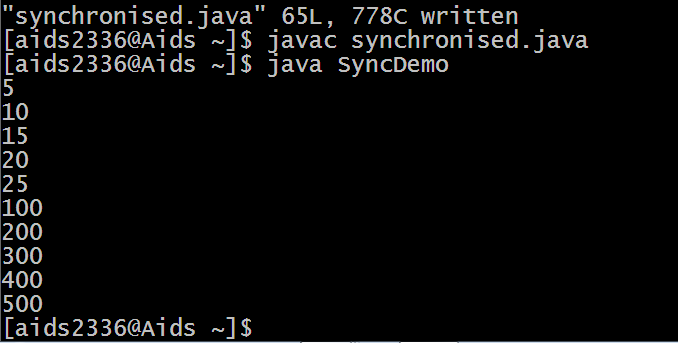
catch(InterruptedException e){

System.out.println(e);

}

}

}



**(EXPERIMENT 8)**

**Producer Consumer Problem**

class Q{

int n;

booleanvalueSet = false;

synchronized void get(){

while(!valueSet){

try{

wait();

}

catch(InterruptedException e){

System.out.println(e);

}

}

System.out.println("Got: " + n);

valueSet = false;

notify();

}

synchronized void put(int n){

while(valueSet){

try{

wait();

}

catch(InterruptedException e){

System.out.println(e);

}

}

this.n=n;

System.out.println("Put: " + n);

valueSet=true;

notify();

}

}

class Producer implements Runnable{

Q q;

Producer(Q q){

this.q=q;

new Thread(this, "Producer").start();

}

public void run(){

int i=0;

while(true){

q.put(i++);

}

}

}

class Consumer implements Runnable{

Q q;

Consumer(Q q){

this.q=q;

new Thread(this, "Consumer").start();

}

public void run(){

while(true){

q.get();

}

}

}

class PC{

public static void main(String args[]){

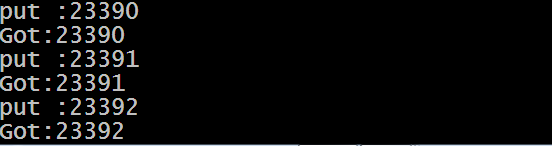
Q q = new Q();

new Producer(q );

new Consumer(q );

}

}

****