Java Programming CAT 2

**Exception handling**

Program 1:

class JavaException

{

public static void main(String args[])

{

int d = 10;

int n = 20;

int fraction = 0;

int g[] = {1};

try {

fraction = n / d;

g[20] = 100;

System.out.println("This line will not be Executed");

}

/\*catch(Exception e){

System.out.println("In the catch clock due to Exception = "+e);

}\*/

catch (ArithmeticException e) {

System.out.println("In the catch clock due to Exception = " + e);

}

catch (ArrayIndexOutOfBoundsException e) {

System.out.println("In the catch clock due to Exception = " + e);

}

finally{

System.out.println("Inside the finally block");

}

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

System.out.println("End Of Main");

}

}

Program 2:

import java.io.\*;

class MyException extends Exception {

private double amount;

public MyException(double amount) {

this.amount = amount;

}

public double getAmount() {

return amount;

}

}

class CheckingAccount {

private double balance;

private int number;

public CheckingAccount(int number) {

this.number = number;

}

public void deposit(double amount) {

balance += amount;

}

public void withdraw(double amount) throws MyException {

if(amount <= balance) {

balance -= amount;

}else {

double needs = amount - balance;

throw new MyException(needs);

}

}

public double getBalance() {

return balance;

}

public int getNumber() {

return number;

}

}

public class BankDemo {

public static void main(String [] args) {

CheckingAccount c = new CheckingAccount(101);

System.out.println("Depositing $500...");

c.deposit(500.00);

try {

System.out.println("Withdrawing $100...");

c.withdraw(100.00);

System.out.println("Withdrawing $600...");

c.withdraw(600.00);

} catch (MyException e) {

System.out.println("Sorry, but you are short $" + e.getAmount());

e.printStackTrace();

}

}

}

**Simple programs on threads:**

Program1:

**class** t1 **extends** Thread{

**int** A1[];

**int** i,odd=0;

t1(**int** A2[],**int** n){

A1=**new** **int**[n];

A1=A2;

}

**public** **void** run(){

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

**if**(A1[i]%2==1)

odd++;

System.*out*.println("\nThe odd numbers count is "+odd);

}

}

**class** t2 **extends** Thread{

**int** A1[];

**int** i,even=0;

t2(**int** A2[],**int** n){

A1=**new** **int**[n];

A1=A2;

}

**public** **void** run(){

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

**if**(A1[i]%2==0)

even++;

System.*out*.println("\nThe even numbers count is "+even);

}

}

**public** **class** simple {

**public** **static** **void** main(String[] args) {

**int** A[]=**new** **int**[10];

**int** i;

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

A[i]=(**int**)(Math.*random*()\*100);

System.*out*.print(A[i]+" ");

}

t1 th1obj=**new** t1(A,10);

t2 th2obj=**new** t2(A,10);

th1obj.start();

th2obj.run();

}

}

Program2:

**class** ThA **extends** Thread{

**public** **void** run(){

System.*out*.println("Starting Thread A");

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

System.*out*.println("\t From ThreadA: i= "+i);

System.*out*.println("Exit from A");

}

}

**class** ThB **extends** Thread{

**public** **void** run() {

System.*out*.println("Starting Thread B");

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

System.*out*.println("\t From ThreadB: i= "+i);

System.*out*.println("Exit from B");

}

}

**class** ThC **extends** Thread{

**public** **void** run() {

System.*out*.println("Starting Thread C");

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

System.*out*.println("\t From ThreadC: i= "+i);

System.*out*.println("Exit from C");

}

}

**public** **class** simplethreads {

**public** **static** **void** main(String[] args) {

ThA threadA=**new** ThA();

ThB threadB=**new** ThB();

ThC threadC=**new** ThC();

threadC.setPriority(Thread.*MAX\_PRIORITY*);

threadB.setPriority(threadA.getPriority()+1);

threadA.setPriority(Thread.*MIN\_PRIORITY*);

System.*out*.println("Started Thread A");

threadA.run();

System.*out*.println("Started Thread B");

threadB.run();

System.*out*.println("Started Thread C");

threadC.run();

System.*out*.println("End of main thread");

}

}

Synchronization:

**Program1:**

**import** java.util.Scanner;

**class** Info

{

**int** s;

Info(){

s=0;

}

**public** **synchronized** **int** READ(){

**return** s;

}

**public** **synchronized** **void** WRITE(**int** s1){

s=s1;

}

}

**class** ThreadA **extends** Thread{

Info i;

ThreadA(Info i1) {

i=i1;

}

**public** **void** run(){

**try**{Thread.*sleep*(1000);}

**catch**(Exception e){}

System.*out*.println(i.READ());

}

}

**class** ThreadB **extends** Thread

{

Info i;

Scanner scan;

**int** s;

ThreadB(Info i1) {

i=i1;

}

**public** **void** run(){

i.WRITE(10);

}

}

**public** **class** syncInfo {

**public** **static** **void** main(String[] args) {

Info In=**new** Info();

Scanner scan=**new** Scanner(System.*in*);

ThreadA t1=**new** ThreadA(In);

ThreadB t2=**new** ThreadB(In);

// t2.start();

t1.start();

t2.start();

scan.close();

}

}

Program2:

**import** java.util.Random;

**class** Buffer {

**private** **volatile** **int** num;

**private** **boolean** state = **false**;

**public** **synchronized** **void** set(**int** n) {

**try** {

**while** (state) {

wait();

}

}

**catch** (InterruptedException ix) {

ix.printStackTrace();

}

num = n;

state = **true**;

//Thread.sleep(2000);

notify();

}

**public** **synchronized** **int** get() {

**try** {

**while** (!state)

wait();

}

**catch** (InterruptedException ix) {

ix.printStackTrace();

}

state = **false**;

//System.out.println("Item " + num);

notify();

**return** num;

}}

**class** Producer **implements** Runnable {

**private** **final** Buffer syncBuffer;

Thread proth;

**private** **final** Random r = **new** Random();

**public** Producer(Buffer b) {

syncBuffer = b;

proth = **new** Thread(**this**, "Producer");

}

**public** **void** run() {

**int** i = 0;

**while** (i++ < 10) {

**int** val = r.nextInt(100);

syncBuffer.set(val);

System.*out*.println("Produced value: " + val);

}}}

**class** Consumer **implements** Runnable {

**private** **final** Buffer syncBuffer;

Thread conth;

**public** Consumer(Buffer b) {

syncBuffer = b;

conth = **new** Thread(**this**, "Consumer");

}

**public** **void** run() {

**int** i = 0;

**while** (i++ < 10)

System.*out*.println("Consumed value: " + syncBuffer.get());

}}

**public** **class** synchronizedthreads {

**public** **static** **void** main(String[] args) {

Buffer buff = **new** Buffer();

Producer p = **new** Producer(buff);

Consumer c = **new** Consumer(buff);

p.proth.start();

c.conth.start();

}

}

Program3:

**class** MakeColdCoffee{

**public** **synchronized** **void** accessMilk(){

**try**{

System.*out*.println("Milk Accessed by "+Thread.*currentThread*().getName());

System.*out*.println("Now... "+Thread.*currentThread*().getName()+" will take 5 seconds to get milk...");

Thread.*sleep*(5000);

System.*out*.println(Thread.*currentThread*().getName()+" now trying to access Coffee Powder");

}// end of try

**catch**(InterruptedException e){}

}// end of accessMilk

**public** **synchronized** **void** accessCoffeePowder(){

**try**{

System.*out*.println(Thread.*currentThread*().getName()+" accessing Coffee Powder");

Thread.*sleep*(5000);

}

**catch**(InterruptedException e){}

}//end of accessCoffeePowder

} // end of MakeColdCoffee

**class** TH1 **extends** Thread{

MakeColdCoffee obj;

TH1(String s,MakeColdCoffee coff) {

**super**(s);

obj=coff;

}

**public** **void** run() {

obj.accessMilk();

obj.accessCoffeePowder();

System.*out*.println(Thread.*currentThread*().getName()+" made the coffee");

}

}

**class** TH2 **extends** Thread{

MakeColdCoffee obj;

TH2(String s,MakeColdCoffee coff) {

**super**(s);

obj=coff;

}

**public** **void** run() {

obj.accessMilk();

obj.accessCoffeePowder();

System.*out*.println(Thread.*currentThread*().getName()+" made the coffee");

}

}

**class** synchronized2{

**public** **static** **void** main(String[] args){

MakeColdCoffee Coffee=**new** MakeColdCoffee();

TH1 t1= **new** TH1("Jhon",Coffee);

TH2 t2= **new** TH2("Kenny",Coffee);

t1.start();

t2.start();

}

}

Program4:

**class** Line {

**synchronized** **public** **void** getLine() {

System.*out*.println("Train "+ Thread.*currentThread*().getName()+" is given line");

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

System.*out*.println(i);

**try** {

Thread.*sleep*(1000);

}

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

}

} //end of getLine()

} //end of Line

**class** Train **extends** Thread{

// Reference variable of type Line.

Line line;

Train(Line line,String s)

{

**super**(s);

**this**.line = line;

}

**public** **void** run() {

line.getLine();

}

}

**class** synchronized1 {

**public** **static** **void** main(String[] args) {

Line obj = **new** Line();

// we are creating two threads which share

// same Object.

Train train1 = **new** Train(obj,"One");

Train train2 = **new** Train(obj,"Two");

// both threads start executing .

train1.start();

train2.start();

}

}

Program5:

**class** Customer{

**int** amount=10000;

**synchronized** **void** withdraw(**int** amount){

System.*out*.println("going to withdraw...");

**if**(**this**.amount<amount){

System.*out*.println("Less balance; waiting for deposit...");

**try**{

wait();

}

**catch**(Exception e){}

}

**this**.amount-=amount;

System.*out*.println("withdraw completed...");

}

**synchronized** **void** deposit(**int** amount){

System.*out*.println("going to deposit...");

**this**.amount+=amount;

System.*out*.println("deposit completed... ");

notify();

}

}

**class** interthread1{

**public** **static** **void** main(String args[]){

**final** Customer c=**new** Customer();

**new** Thread(){

**public** **void** run(){

c.withdraw(15000);}

}.start();

**new** Thread(){

**public** **void** run(){

c.deposit(10000);}

}.start();

}}

Program6:

**class** Pen{}

**class** Paper{}

**public** **class** interthread2 {

**public** **static** **void** main(String[] args)

{

**final** Pen pn =**new** Pen();

**final** Paper pr =**new** Paper();

Thread t1 = **new** Thread()

{

**public** **void** run(){

**synchronized**(pn)

{

System.*out*.println("Thread1 is holding Pen");

**try**{

Thread.*sleep*(5000);

}

**catch**(InterruptedException e){}

**synchronized**(pr)

{

System.*out*.println("Requesting for Paper");

}

}

}

};

Thread t2 = **new** Thread(){

**public** **void** run()

{

**synchronized**(pn)

{

System.*out*.println("Thread2 is holding Paper");

**try**{

Thread.*sleep*(1000);

}

**catch**(InterruptedException e){}

**synchronized**(pr)

{

System.*out*.println("requesting for Pen"); }

}

}

};

t1.start();

t2.start();

}

}

Dead lock:

Program1:

**public** **class** deadlock1 {

**public** **static** **void** main(String[] args) {

**final** String resource1 = "String1";

**final** String resource2 = "String2";

// t1 tries to lock resource1 then resource2

Thread t1 = **new** Thread()

{

**public** **void** run() {

**synchronized** (resource1) {

System.*out*.println("Thread 1: locked resource 1");

**try** {

Thread.*sleep*(5000);

}

**catch** (Exception e) {}

**synchronized** (resource2) {

System.*out*.println("Thread 1: locked resource 2");

}

/\*try {

Thread.sleep(10000);

}

catch (Exception e) {} \*/

}

} //end of run()

}; //end of t1

// t2 tries to lock resource2 then resource1

Thread t2 = **new** Thread()

{

**public** **void** run() {

/\*try {

Thread.sleep(10000);

}

catch (Exception e) {} \*/

**synchronized** (resource2)

{

System.*out*.println("Thread 2: locked resource 2");

**try** {

Thread.*sleep*(5000);

}

**catch** (Exception e) {}

**synchronized** (resource1)

{

System.*out*.println("Thread 2: locked resource 1");

}

}

} //end of run()

}; //end of t2

t1.start();

t2.start();

}

}

Program2:

**public** **class** deadlock2 {

**public** **static** Object *Lock1* = **new** Object();

**public** **static** Object *Lock2* = **new** Object();

**public** **static** **void** main(String args[]) {

ThreadDemo1 T1 = **new** ThreadDemo1();

ThreadDemo2 T2 = **new** ThreadDemo2();

T1.start();

T2.start();

}

**private** **static** **class** ThreadDemo1 **extends** Thread {

**public** **void** run() {

**synchronized** (*Lock1*) {

System.*out*.println("Thread 1: Holding lock 1...");

**try** { Thread.*sleep*(10);

}

**catch** (InterruptedException e) {}

System.*out*.println("Thread 1: Waiting for lock 2...");

**synchronized** (*Lock2*) {

System.*out*.println("Thread 1: Holding lock 1 & 2...");

}

}

}//end of run()

}//end of ThreadDemo1

**private** **static** **class** ThreadDemo2 **extends** Thread {

**public** **void** run() {

**synchronized** (*Lock2*) {

System.*out*.println("Thread 2: Holding lock 2...");

**try** { Thread.*sleep*(10); }

**catch** (InterruptedException e) {}

System.*out*.println("Thread 2: Waiting for lock 1...");

**synchronized** (*Lock1*) {

System.*out*.println("Thread 2: Holding lock 1 & 2...");

}

}

}//end of run()

}//end of ThreadDemo2

} //end of deadlock3

Program3:

**class** Text {

**private** String packet;

**public** **synchronized** **void** set(String s) {

**try** {

wait();

}

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

packet=s;

notify();

} //end of set

**public** **synchronized** String get() {

**try** {

wait();

}

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

notify();

**return** packet;

}//end of get

}//end of Text

**class** vit1 **implements** Runnable {

**private** Text data;

**int** i;

vit1(Text d) {

data=d;

}

**public** **void** run() {

String packets[]={"I packet","II packet","III packet","IV packet","End"};

System.*out*.println("Start sending the data....");

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

{

**try** {

Thread.*sleep*(1000);

}

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

data.set(packets[i]);

}

System.*out*.println("End of Sending....");

}

}

**class** vit2 **implements** Runnable {

**private** Text load;

vit2(Text l){

load=l;

}

**public** **void** run() {

**int** i;

System.*out*.println("Start receiving the data....");

String receivedMessage = load.get();

**for** (i=0;i<4;i++)

{

System.*out*.println(receivedMessage);

**try** {

Thread.*sleep*(1000);

}

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

receivedMessage = load.get();

}

System.*out*.println("End of Receiving the data");

}

}

**public** **class** deadlock3 {

**public** **static** **void** main(String[] args) {

Text data = **new** Text();

Thread sender = **new** Thread(**new** vit1(data));

Thread receiver = **new** Thread(**new** vit2(data));

sender.start();

receiver.start();

}

}

**Programs using Collections**

**Program 1:**

package collections;

import java.util.ArrayList;

import java.util.Iterator;

public class SimpleCollections {

public static void main(String[] args) {

ArrayList c;

c = new ArrayList();

System.out.println(c.getClass().getName());

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

c.add(i + " \* " + i + " = "+i\*i);

}

Iterator iter = c.iterator();

System.out.println(iter.getClass().getName());

while (iter.hasNext())

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

}}

**Program 2:**

package collections;

import java.util.\*;

class sample{}

public class ListExample{

public static void main(String args[]){

int i;

List al=new ArrayList();

al.add("Amit");

al.add(true);

al.add(10.5);

al.add('S');

al.add(10);

al.add(10.5);

sample s=new sample();

al.add(s);

System.out.println("An element at 2nd position: "+al.get(2));

//for(i=0;i<6;i++) System.out.println(al.get(i));

Iterator it=al.iterator();

while(it.hasNext())

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

}

}

**Program 3:**

package collections;

import java.util.\*;

public class LikedList\_String {

LinkedList Insert(LinkedList L,String s){

L.add(s);

System.out.println("After invoking Insert() method: "+L);

return L;

}

LinkedList Remove(LinkedList L,String s){

L.remove(s);

System.out.println("After invoking Remove method: "+L);

return L;

}

public static void main(String args[]){

LinkedList<String> ll=new LinkedList<>();

Scanner scan=new Scanner(System.in);

LikedList\_String obj=new LikedList\_String();

int ch;

String str;

do{

System.out.print("1.Insert String\n2.Remove String\n3.Exit\nEnter ur choice:");

ch=scan.nextInt();

switch(ch){

case 1:System.out.print("Enter the string:");

str=scan.next();

ll=obj.Insert(ll,str);

break;

case 2:System.out.print("Enter the string:");

str=scan.next();

ll=obj.Remove(ll, str);

break;

case 3:break;

default:System.out.println("Invalid choice....");

}

}while(ch!=3);

}

}

**Program 4:**

package collections;

import java.util.\*;

public class hashset {

public static void main(String arg[]){

HashSet hs=new HashSet();

hs.add(10);

hs.add("VIT");

hs.add(12.5);

hs.add('A');

hs.add("Vellore");

hs.add(10);hs.add(10);hs.add(10);

hs.add("10");

Iterator it1=hs.iterator();

System.out.println("Elements of HashSet");

while(it1.hasNext())

System.out.println(it1.next());

}

}

**Program 5:**

package collections;

import java.util.Iterator;

import java.util.TreeSet;

class sample1{}

public class treeset {

public static void main(String arg[]){

TreeSet ts=new TreeSet();

ts.add("aaa");

ts.add("ccc");

ts.add("ddd");

ts.add("bbb");

//ts.add(15.5);

ts.add("SCOPE");

Iterator it=ts.iterator();

System.out.println("Elements of TreeSet");

while(it.hasNext())

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

/\*sample1 s1=new sample1();

sample1 s2=new sample1();

TreeSet ts1=new TreeSet();

ts1.add(s1);

ts1.add(s2);

it=ts1.iterator();

System.out.println("Elements of TreeSet");

while(it.hasNext())

System.out.println(it.next());\*/

}

}

**Program 6:**

package collections;

import java.util.\*;

public class hashmap {

public static void main(String[] args) {

HashMap map=new HashMap();

//Adding elements to map

map.put(1,"Amit");

map.put(5,"Rahul");

map.put(2,"Jai");

map.put(6,"Amit");

map.put(7,"Arun");

map.put(7,"Banu");

//Traversing Map

Set set=map.entrySet();//Converting to Set so that we can traverse

Iterator itr=set.iterator();

while(itr.hasNext()){

//Converting to Map.Entry so that we can get key and value separately

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

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

}

//comparingByValue() in Descending Order

Map<Integer,String> map1=new HashMap();

map1.put(100,"Amit");

map1.put(101,"Vijay");

map1.put(102,"Rahul");

//Returns a Set view of the mappings contained in this map

map1.entrySet()

//Returns a sequential Stream with this collection as its source

.stream()

//Sorted according to the provided Comparator

.sorted(Map.Entry.comparingByValue(Comparator.reverseOrder()))

//Performs an action for each element of this stream

.forEach(System.out::println);

}}

**Program 7:**

package collections;

import java.util.\*;

public class treemap {

public static void main(String args[]) {

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

map.put(100,"AAAA");

map.put(101,"BBBB");

map.put(103,"DDDD");

map.put(102,"CCCC");

System.out.println("Before invoking remove() method");

Set set=map.entrySet();

Iterator itr=set.iterator();

while(itr.hasNext()){

//Converting to Map.Entry so that we can get key and value separately

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

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

}

map.remove(102);

System.out.println("After invoking remove() method");

set=map.entrySet();

itr=set.iterator();

while(itr.hasNext()){

//Converting to Map.Entry so that we can get key and value separately

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

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

}

}

}

**Program 8:**

package collections;

import java.util.\*;

public class linkedhashmap {

public static void main(String args[]){

LinkedHashMap<Integer,String> hm=new LinkedHashMap();

hm.put(100,"AAAA");

hm.put(101,"BBBB");

hm.put(103,"DDDD");

hm.put(102,"CCCC");

Set set=hm.entrySet();

Iterator itr=set.iterator();

while(itr.hasNext()){

//Converting to Map.Entry so that we can get key and value separately

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

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

}

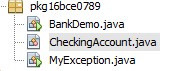
}

}

**The Quesuions:**

1. Implement a bank application where an alert message is issued when the minimum balance is going below 1000INR. Create an exception class and a Bank class for this application and test it with a Java program.

The Code:



package pkg16bce0789; import java.util.Scanner;

public class BankDemo { public static void main(String [] args) {

Scanner sc = new Scanner(System.in);

CheckingAccount c = new CheckingAccount(101);

int d = sc.nextInt();

//System.out.println("Depositing $500...");

c.deposit(d);

try {

System.out.println("Withdrawing $100...");

c.withdraw(100.00);

System.out.println("Withdrawing $600...");

c.withdraw(600.00);

} catch (MyException e) {

System.out.println("Sorry, but you are short $" + e.getAmount());

e.printStackTrace();

}

}

}

package pkg16bce0789;

class CheckingAccount { private double balance; private int number;

public CheckingAccount(int number) { this.number = number;

}

public void deposit(double amount) { balance += amount;

}

public void withdraw(double amount) throws MyException { if(amount >1000)

{

if(amount <= balance) { balance -= amount;

}else {

double needs = amount - balance; throw new MyException(needs);

}

}

else

{

double needs = amount - balance; throw new MyException(needs);

}

}

public double getBalance() { return balance;

}

public int getNumber() { return number;

}

}

import java.io.\*; class MyException extends Exception { private double amount; public MyException(double amount) { this.amount = amount;

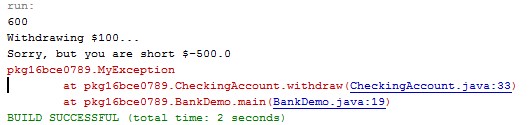
}

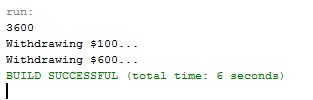
public double getAmount() { return amount;

}

}

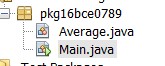
The Output:





2. Develop a Java application for calculating the average mark of ‘n’ students. Read the number of courses they have appeared for the past semester and the marks in all the courses. The number of courses should not be zero and if it is zero handle that case with a standard exception.

The Code:



package pkg16bce0789;

/\*\*

\*

\* @author OM MISHRA

\*/ class Average

{

double avg=0;

void check (double amount) throws ZeroException

{

if (a==0)

throw new ZeroException();

else

continue;

}

Average(int a[])

{

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

{

avg=avg+a[i];

}

}

}

class Main

{

public static void main(String args[])

{ int i; System.out.println("Enter number of subjects");

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

int[] a=new int[n];

System.out.println("Enter marks");

for( i=0;i<n;i++)

{

a[i]=sc.nextInt();

}

Average c = new Average(a);

c.check();

System.out.print("Average of (");

for(i=0;i<n-1;i++)

{

System.out.print(a[i]+",");

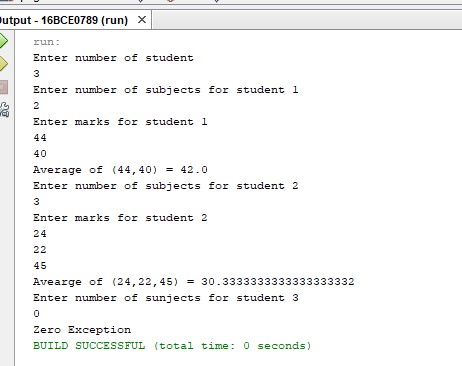
}

System.out.println(a[i]+") ="+c.avg/n);

}

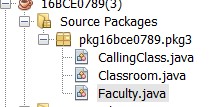
}

The Output:



3. Write a Java program to develop an application where you have a class ‘ClassRoom’ which have to be used by the (thread) objects of class ‘Faculty’ to deliver their course contents. Since the ‘Faculty’ class objects are active simultaneously, synchronize the usage of the object of ‘Classroom’.

The Code:



// A Class used to send a message class Faculty

{

public void send(String msg)

{

System.out.println("Sending\t" + msg );

try

{

Thread.sleep(1000);

}

catch (Exception e)

{

System.out.println("Thread interrupted.");

}

System.out.println("\n" + msg + "Sent");

}

}

package pkg16bce0789.pkg3;

/\*\*

\*

\* @author OM MISHRA

\*/ class Classroom extends Thread

{

private String msg; private Thread t;

Faculty faculty;

// Recieves a message object and a string

// message to be sent

Classroom(String m, Faculty obj)

{

msg = m; faculty = obj;

}

public void run()

{

// Only one thread can send a message

// at a time.

synchronized(faculty)

{

// synchronizing the snd object faculty.send(msg);

}

}

}

package pkg16bce0789.pkg3;

/\*\*

\*

\* @author OM MISHRA

\*/ class CallingClass

{

public static void main(String args[])

{

Faculty f1 = new Faculty(); Classroom C1 = new Classroom( " Good Morning " , f1 ); Classroom C2 = new Classroom( " Thank You " , f1 );

// Start two threads of ThreadedSend type

C1.start();

C2.start();

// wait for threads to end

try

{

C1.join(); C2.join();

}

catch(Exception e)

{

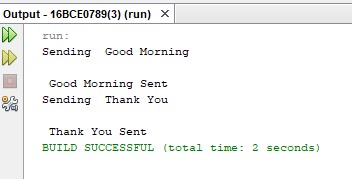
System.out.println("Interrupted");

}

}

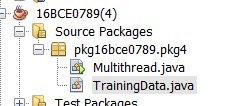
}

The Output:



4. We need to train the patient data to predict whether a new patient may or may not get the disease. Implement a data training application in Java which prepares the data set used in a disease prediction algorithm. The ‘TraingData’ class has an array of records (patient-id, patientage, diseaseid, and disease-seriousness-index) and two methods ‘writeData’ and ‘readData’. In a multi-threaded environment, if one thread is writing the data then other threads have to wait and if one thread is reading the data then other threads have to wait.

The Code:



package pkg16bce0789.pkg4;

/\*\*

\*

\* @author OM MISHRA

\*/ import java.io.\*; import java.util.\*; import java.lang.\*;

public class Multithread

{

public static void main(String[] args)

{

Scanner sc = new Scanner(System.in); //int n = 8; // Number of threads int n = sc.nextInt(); for (int i=0; i<n; i++)

{

TrainingData object = new TrainingData(); object.writeData();

object.readData(); object.start();

}

}

}

package pkg16bce0789.pkg4;

/\*\*

\*

\* @author OM MISHRA

\*/ import java.io.\*; import java.util.\*; import java.lang.\*;

class TrainingData extends Thread

{

public int Patientid[]; public int age[]; public int disid[]; public int dsi[]; static int i = 0;

public void writeData()

{

Patientid = new int[] {1,2,3,4,5,6,7,8,9}; age = new int[] {12,56,45,89,26,78,24,67,74}; disid = new int[] {1,2,1,1,2,2,1,1,1}; dsi = new int[] {1,2,3,1,1,3,2,1,2};

}

public void readData()

{

while(i!=9)

{

System.out.println("Pateint Id :"+Patientid[i]);

System.out.println("Pateint age :"+age[i]);

System.out.println("Disease Id :"+disid[i]);

System.out.println("disease-seriousness-index :"+dsi[i]);

i++;

}

}

public void run()

{ try {

// Displaying the thread that is running

int n =(int) Thread.currentThread().getId();

System.out.println ("Thread " + n + " is running");

System.out.println("Pateint id "+Patientid[(n-11)]);

System.out.println("Patient age "+age[(n-11)]);

System.out.println("Disease id "+disid[(n-11)]);

System.out.println("disease-seriousness-index "+dsi[(n-11)]);

}

catch (Exception e)

{

// Throwing an exception

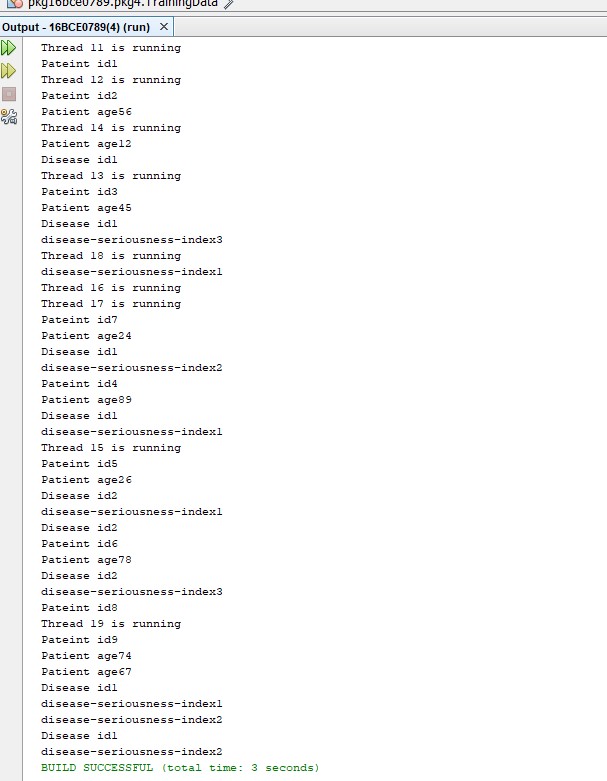
System.out.println ("Exception is caught");

}

}

}

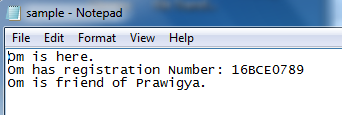
The Output:



5. Write a Java program to read the contents in a text file and display in the console.

Answer:

The Sample Text:



The Code:

package om.pkg16bce0789;

/\*\*

\*

\* @author 16BCE0789

\*/

import java.io.File;

import java.util.Scanner;

import java.lang.\*;

public class Om16BCE0789 {

/\*\*

\* @param args the command line arguments

\*/

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

File file = new File("C:\\Users\\16BCE0789\\Documents\\NetBeansProjects\\Om#16BCE0789\\sample.txt");

Scanner sc = new Scanner(file);

//BufferedReader br = new BufferedReader(new FileReader(file));

String st;

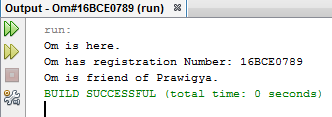
while(sc.hasNextLine())

System.out.println(sc.nextLine());

}

}

The Output:



6. Write a Java program to implement Caesar’s cipher to encrypt the data stored in a text file named, ‘input.txt’. The encrypted data should be stored in ‘cipher.txt’. Also decrypt the contents in ‘cipher.txt’ and store the plain text in ‘plain.txt’.

Answer:

The Sample:



The Code:

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package om.pkg16bce0789;

/\*\*

\*

\* @author 16BCE0789

\*/

import java.io.File;

import java.util.Scanner;

import java.lang.\*;

public class Om16BCE0789 {

/\*\*

\* @param args the command line arguments

\*/

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

File file = new File("C:\\Users\\16BCE0789\\Documents\\NetBeansProjects\\Om#16BCE0789\\input.txt");

Scanner sc = new Scanner(file);

String st="Hello World It is a great Day.";

while(sc.hasNextLine())

{

System.out.println(sc.nextLine());

st = st+sc.nextLine();

}

System.out.println(st);

int s = 4;

System.out.println("Text : "+st);

System.out.println("Shift : "+s);

System.out.println("Cipher: "+ encrypt(st,s));

System.out.println("Back to plain.txt");

System.out.println("Plian: "+ st);

}

public static StringBuffer encrypt(String st, int s)

{

StringBuffer result = new StringBuffer();

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

{

if(Character.isUpperCase(st.charAt(i)))

{

char ch = (char) (((int)st.charAt(i) + s - 65) % 26 + 65);

result.append(ch);

}

else

{

char ch = (char) (((int)st.charAt(i) + s - 97) % 26 + 97);

result.append(ch);

}

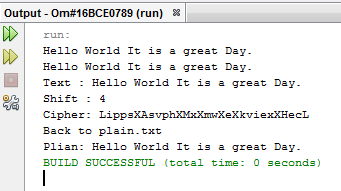
}

return result;

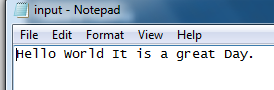
}

}

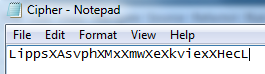
The Output:



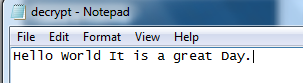
The Input File:



The Cipher File:



The Decrypt File:



7. Write a Java program to serialize the objects of the class ‘Player’ which has the following details: player\_id, player\_name, age, team\_name, run\_rate, wickets. Use appropriate classes for writing and reading the data into and from files.

Answer:

The Sample:

package pkg16bce0892;

import java.io.\*;

public class Player implements java.io.Serializable

{

int player\_id;

String name;

int age;

String team\_name;

double run\_rate;

int wickets;

//Constructor

public Player(int player\_id, String name, int age, String team\_name, double run\_rate, int wickets)

{

this.player\_id = player\_id;

this.name = name;

this.age = age;

this.team\_name = team\_name;

this.run\_rate = run\_rate;

this.wickets = wickets;

}

}

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package pkg16bce0789;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.ObjectOutputStream;

/\*\*

\*

\* @author 16BCE0789

\*/

public class Read

{

public static void main(String args[])

{

String filename = "abc.txt";

//Serialization

try

{

FileOutputStream file = new FileOutputStream(filename);

ObjectOutputStream out = new ObjectOutputStream(file);

out.writeObject(object);

out.close();

file.close();

System.out.println("File has been serialized");

}

catch(IOException ex)

{

System.out.println("IOException is caught");

}

Player object1 = null;

//Deserialization

try

{

FileInputStream file = new FileInputStream(filename);

ObjectInputStream in = new ObjectInputStream(file);

object1 = (Player)in.readObject();

in.close();

file.close();

System.out.println("Object has been deserialized");

}

catch(IOException ex)

{

System.out.println("IOExceotion caught");

}

catch(ClassNotFoundException ex)

{

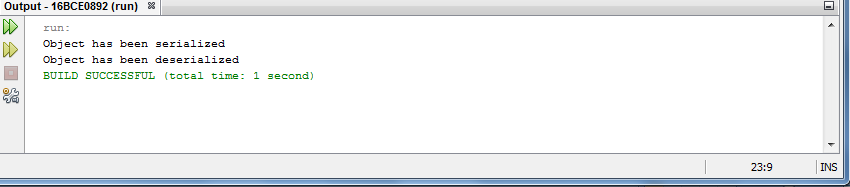
System.out.println("ClassNotFound");

}

}

}

The Output:



8. A double-ended queue is a data structure in which the enqueue and dequeue operations can be done at both the ends of the queue. Implement a doubleended queue using a suitable Java Collection object. Develop a Java application to test the same.

The Answer:

The Code:

package DoublyLinkedList;

/\*\*

\*

\* @author OM MISHRA

\*/

public class DoublyLinkedList<T> {

private Node<T> head;

private Node<T> tail;

/\*\*

\* Inserts the value at the first position (head) of LinkedList.

\*

\* @param value

\* to be inserted

\*/

public void insertFirst(final T value) {

final Node<T> node = new Node<>(value);

node.next = head;

if (head != null) {

head.previous = node;

}

head = node;

if (tail == null) {

tail = node;

}

}

/\*\*

\* Inserts the value at the last position (tail) of LinkedList.

\*

\* @param value

\*/

public void insertLast(final T value) {

final Node<T> node = new Node<>(value);

if (tail != null) {

tail.next = node;

node.previous = tail;

}

tail = node;

if (head == null) {

head = node;

}

}

/\*\*

\* Removes the node from first position (head) of LinkedList.

\*

\* @return the value of node deleted. Null if no nodes are present

\*/

public T removeFirst() {

T value = null;

if (head != null) {

value = head.value;

if (head == tail) {

tail = null;

}

head = head.next;

head.previous = null;

}

return value;

}

/\*\*

\* Removes the node from last position (tail) of LinkedList.

\*

\* @return the value of node deleted. Null if no nodes are present

\*/

public T removeLast() {

T value = null;

if (tail != null) {

value = tail.value;

if (tail == head) {

head = tail = null;

} else {

tail = tail.previous;

tail.next = null;

}

}

return value;

}

/\*\*

\* Removes the first occurance of node having the value same as input value.

\*

\* @param value

\* to be removed

\* @return deleted node's value if node is found else null;

\*/

public T remove(final T value) {

T deletedObj = null;

if (head != null) {

if (head == tail) {

if (head.value.equals(value)) {

deletedObj = head.value;

head = tail = null;

}

} else {

Node<T> node = head;

do {

if (node.value.equals(value)) {

deletedObj = node.value;

if (node.previous != null) {

node.previous.next = node.next;

}

node.next.previous = node.previous;

break;

}

node = node.next;

} while (node != null);

}

}

return deletedObj;

}

/\*\*

\* Implementation of a Node of a Doubly Linked List.

\*

\* @author Sain Technology Solutions

\*

\* @param <T>

\*/

private static class Node<T> {

T value;

Node<T> next;

Node<T> previous;

private Node(T value) {

this.value = value;

}

@Override

public String toString() {

return "Node [value=" + value + "]";

}

}

/\*\*

\* Entry point for testing LinkedList.

\*/

public static void main(String[] args) {

final DoublyLinkedList<Integer> doublyLinkedList = new DoublyLinkedList<>();

// Inserts the node with value 5 at the head position

doublyLinkedList.insertFirst(5);

// Inserts the node with value 1 at the head position, pushing the

// previously inserted node to second position

doublyLinkedList.insertFirst(1);

// Inserts the node with value 2 at the head position, pushing the

// previously inserted node to second position

doublyLinkedList.insertFirst(2);

// Inserts the node with value 3 at the tail position

doublyLinkedList.insertLast(3);

// Inserts the node with value 4 at the tail position, pushing the

// previously inserted node to second position from last

doublyLinkedList.insertLast(4);

// At this point, LinkedList will look like: 2 <=> 1 <=> 5 <=> 3 <=> 4

// Removes the node with value 2 since it is head node. This operation

// will also make node with value 1 as head node

System.out.println(doublyLinkedList.removeFirst());

// Removes the node with value 1 since it is head node. This operation

// will also make node with value 5 as head node

System.out.println(doublyLinkedList.removeFirst());

// Removes the node with value 4 since it is tail node. This operation

// will also make node with value 3 as tail node

System.out.println(doublyLinkedList.removeLast());

// Removes the node with value 3 since it is tail node. This operation

// will also make node with value 5 as tail node

System.out.println(doublyLinkedList.removeLast());

// Removes the node with value 5

System.out.println(doublyLinkedList.remove(5));

// Returns null since there is no node with value 2 as it was removed

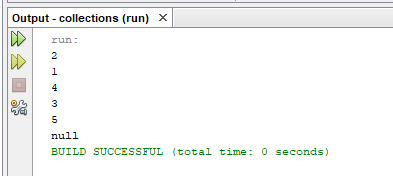
// due to earlier removeXXX method calls

System.out.println(doublyLinkedList.remove(2));

}

}

The Output:



9. Write a Java application to store the different comments given by the users on the surgical attack by our Indian army on terrorist camps in Pakistan held on 26-Feb-2019, 3.30am. The comment should be of ONE word and the duplicates need not to be stored. Use suitable Java Collection object to implement this program and test it.

The Answer:

The Code:

package Collections;

import java.util.\*;

/\*\*

\*

\* @author OM MISHRA

\*/

public class hashset {

public static void main(String arg[]){

HashSet hs=new HashSet();

hs.add("Pakisthan");

hs.add("3:30 AM");

hs.add("26-Feb-2019");

hs.add("Army");

hs.add("Attack");

hs.add("Army");

hs.add("Army");

hs.add("Firing");

hs.add(200);

Iterator it1=hs.iterator();

System.out.println("Elements of HashSet");

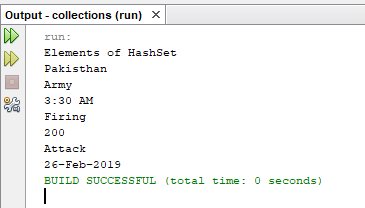
while(it1.hasNext())

System.out.println(it1.next());

}

}

The Output:



10. Implement a Lucky Draw game in which the user has to input an integer ‘input’ and multiply it with a random integer to get the product ‘p’. Depending upon the value of ‘p’ display the prize amount. Use a suitable Java Collection object to store the prize amount for each value of ‘p’. [Hint: Use modulo operation to get final value of ‘p’]

The Answer:

The Code:

package DoublyLinkedList;

import java.util.\*;

/\*\*

\*

\* @author OM MISHRA

\*/

public class Linkedhashmap {

public static void main(String args[]){

Scanner sc = new Scanner(System.in);

int n = sc.nextInt();

Random rn = new Random();

int n1 = rn.nextInt(50);

System.out.println("Random Number Generated : "+n1);

int val = n\*n1;

System.out.println("Product Formed : "+val);

int p = val%5;

System.out.println("Value of p : "+p);

LinkedHashMap<Integer,String> hm=new LinkedHashMap();

hm.put(0,"Bicycle");

hm.put(1,"Car");

hm.put(2,"Bike");

hm.put(3,"Game Set");

hm.put(4, "Dog");

Set set=hm.entrySet();

Iterator itr=set.iterator();

while(itr.hasNext()){

//Converting to Map.Entry so that we can get key and value separately

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

if(p == (int)(entry.getKey()))

{

System.out.println("The prize recieved is "+entry.getValue());

}

}

}

}

The Output:

