**INTERNATIONAL INSTITUE OF PROFESSIONAL STUDIES(IIPS) DAVV, INDORE**

**ADVANCED JAVA ASSIGNMENT- VI SEM (section-A)**

**SUBMITTED TO – DR. NITIN NAGAR SIR**

SUBMITTED BY- MUSKAN MANDLOI – [IT-2K19-34]

**INDEX:-**

1. ArrayList

2. Queue

3. Priority Queue

4. LinkedList

5. Generic and Non Generic

6. MultiThreading

7. Identity HashMap

8. HashMap

9.HashSet

10.Linked-HashMap

11. Linked-HashSet

12. Producer-Consumer

13. Abstract-Collection

14. Sorted Map

15. Graph

16. Stack

17. Vector

18. Map

19. Tree Map

20. Tree Set

21. Hash Table

22. LinkedList HashSET

23. LinkedList HashMAP

24. Navigatable Map

25. Navigatable Set

26. List

**ARRAYLIST:-**

1. Adding element in arraylist

import java.util.\*;

public class arraylist1{

public static void main(String args[]){

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

list.add("MANNY");

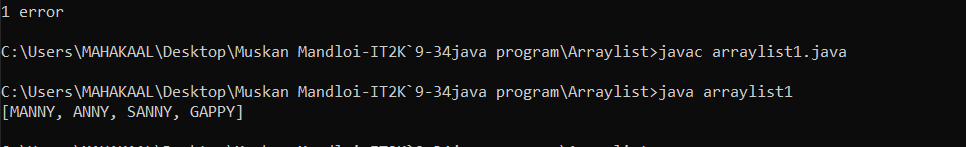
list.add("ANNY");

list.add("SANNY");

list.add("GAPPY");

System.out.println(list);

}

} 

1. Add methods in arraylist

import java.util.\*;

class arraylistadd{

public static void main(String args[]){

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

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

al.add("Ravi");

al.add("Vijay");

al.add("Ajay");

System.out.println("After invoking add(E e) method: "+al);

al.add(1, "Gaurav");

System.out.println("After invoking add(int index, E element) method: "+al);

ArrayList<String> al2=new ArrayList<String>();

al2.add("Sonoo");

al2.add("Hanumat");

al.addAll(al2);

System.out.println("After invoking addAll(Collection<? extends E> c) method: "+al);

ArrayList<String> al3=new ArrayList<String>();

al3.add("John");

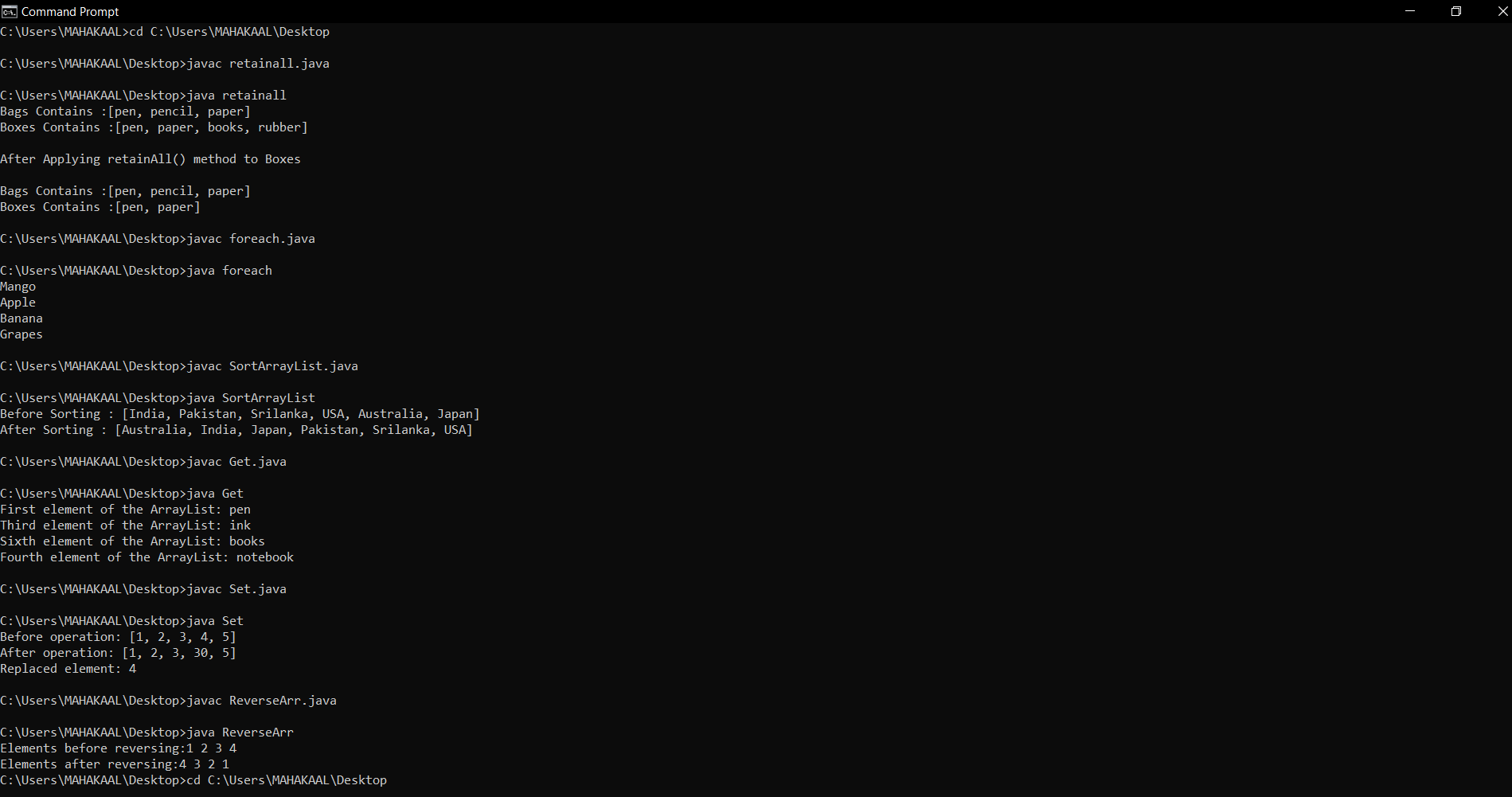
al3.add("Rahul");

al.addAll(1, al3);

System.out.println("After invoking addAll(int index, Collection<? extends E> c) method: "+al);

}

}



**QUEUE:-**

**3.Priority**

import java.util.\*;

class priority{

public static void main(String args[]){

PriorityQueue<String> queue=new PriorityQueue<String>();

queue.add("Amit");

queue.add("Vijay");

queue.add("Karan");

queue.add("Jai");

queue.add("Rahul");

System.out.println("head:"+queue.element());

System.out.println("head:"+queue.peek());

System.out.println("iterating the queue elements:");

Iterator itr=queue.iterator();

while(itr.hasNext()){

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

}

queue.remove();

queue.poll();

System.out.println("after removing two elements:");

Iterator<String> itr2=queue.iterator();

while(itr2.hasNext()){

System.out.println(itr2.next());

}

}

}

4.Priority Queue

import java.util.\*;

class PriorityQueueDemo {

public static void main(String args[]){

PriorityQueue<Integer> pQueue = new PriorityQueue<Integer>();

pQueue.add(10);

pQueue.add(20);

pQueue.add(15);

System.out.println(pQueue.peek());

System.out.println(pQueue.poll());

System.out.println(pQueue.peek());

}

}

5.PriorityAdd

import java.util.\*;

class priority{

public static void main(String args[]){

PriorityQueue<String> queue=new PriorityQueue<String>();

queue.add("Amit");

queue.add("Vijay");

queue.add("Karan");

queue.add("Jai");

queue.add("Rahul");

System.out.println("head:"+queue.element());

System.out.println("head:"+queue.peek());

System.out.println("iterating the queue elements:");

Iterator itr=queue.iterator();

while(itr.hasNext()){

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

}

queue.remove();

queue.poll();

System.out.println("after removing two elements:");

Iterator<String> itr2=queue.iterator();

while(itr2.hasNext()){

System.out.println(itr2.next());

}

}

}

6.QUEUE Example

import java.util.LinkedList;

import java.util.Queue;

public class QueueExample {

public static void main(String[] args)

{

Queue<Integer> q= new LinkedList<Integer>();

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

q.add(i);

System.out.println("Elements of queue "+ q);

int removedele = q.remove();

System.out.println("removed element-"+ removedele);

System.out.println(q);

int head = q.peek();

System.out.println("head of queue-"+ head);

int size = q.size();

System.out.println("Size of queue-"+ size);

}

}

7.Priority Peek()

import java.util.\*;

class PriorityQueueDemo {

public static void main(String args[])

{

PriorityQueue<Integer> pQueue = new PriorityQueue<Integer>();

pQueue.add(10);

pQueue.add(20);

pQueue.add(15);

System.out.println(pQueue.peek());

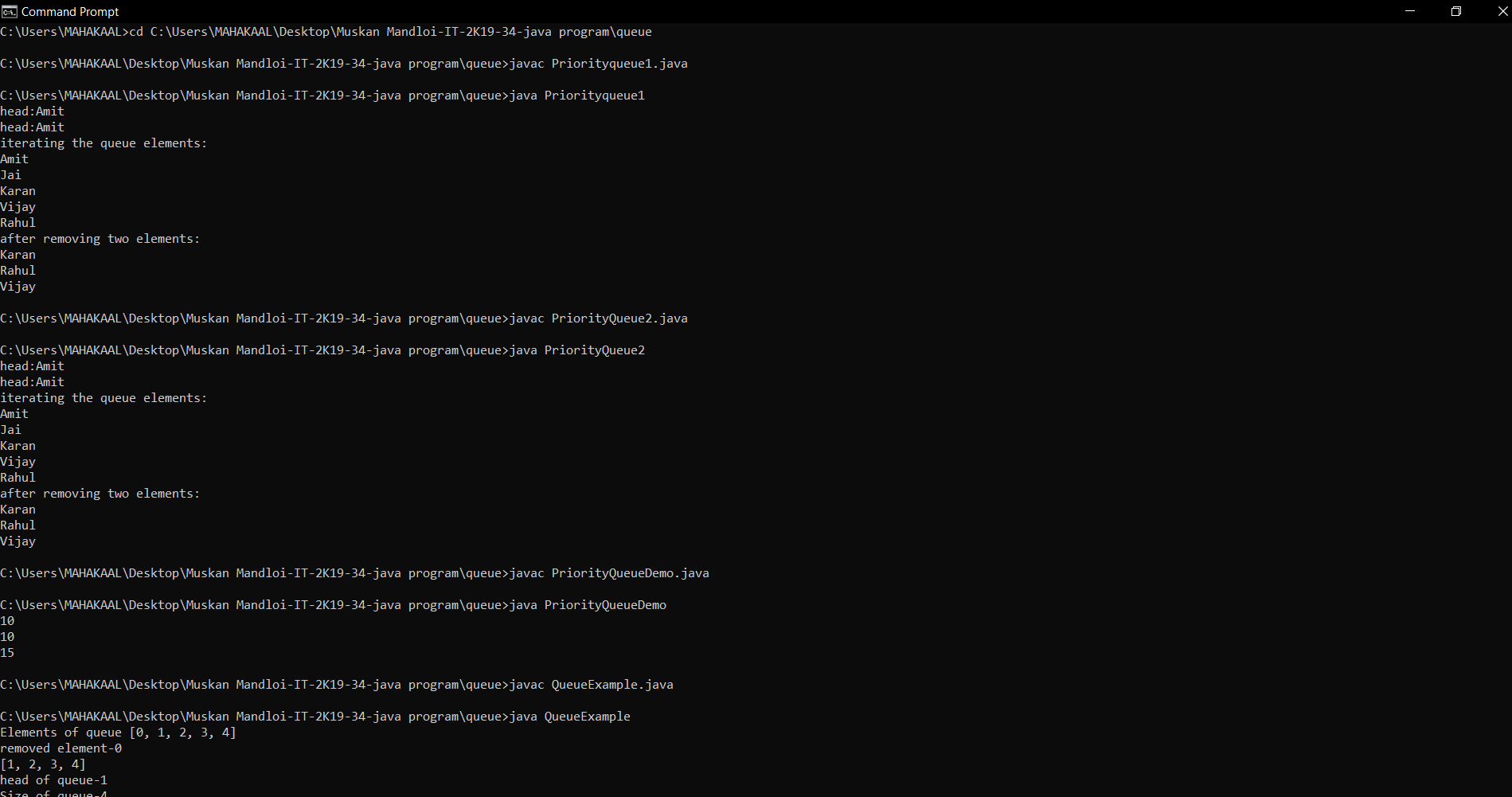
System.out.println(pQueue.poll());

System.out.println(pQueue.peek());

}

}

**OUTPUT OF ALL QUEUE PROGRAMS:-**

****

**Linkedlist:-**

8.linkedlist

import java.util.\*;

public class LinkedList1{

public static void main(String args[]){

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

al.add("Ravi");

al.add("Vijay");

al.add("Ravi");

al.add("Ajay");

Iterator<String> itr=al.iterator();

while(itr.hasNext()){

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

}

}

}

9.Methods in LinkedList

import java.util.\*;

public class LinkedList1{

public static void main(String args[]){

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

al.add("Ravi");

al.add("Vijay");

al.add("Ravi");

al.add("Ajay");

Iterator<String> itr=al.iterator();

while(itr.hasNext()){

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

}

}

}

10.Linkedlist methods()

import java.util.\*;

public class LinkedList2{

public static void main(String args[]){

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

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

ll.add("Ravi");

ll.add("Vijay");

ll.add("Ajay");

System.out.println("After invoking add(E e) method: "+ll);

ll.add(1, "Gaurav");

System.out.println("After invoking add(int index, E element) method: "+ll);

LinkedList<String> ll2=new LinkedList<String>();

ll2.add("Sonoo");

ll2.add("Hanumat");

ll.addAll(ll2);

System.out.println("After invoking addAll(Collection<? extends E> c) method: "+ll);

LinkedList<String> ll3=new LinkedList<String>();

ll3.add("John");

ll3.add("Rahul");

ll.addAll(1, ll3);

System.out.println("After invoking addAll(int index, Collection<? extends E> c) method: "+ll);

ll.addFirst("Lokesh");

System.out.println("After invoking addFirst(E e) method: "+ll);

ll.addLast("Harsh");

System.out.println("After invoking addLast(E e) method: "+ll);

}

}

11.Methods ADD

import java.util.\*;

public class LinkedList3 {

public static void main(String [] args)

{

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

ll.add("Ravi");

ll.add("Vijay");

ll.add("Ajay");

ll.add("Anuj");

ll.add("Gaurav");

ll.add("Harsh");

ll.add("Virat");

ll.add("Gaurav");

ll.add("Harsh");

ll.add("Amit");

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

ll.remove("Vijay");

System.out.println("After invoking remove(object) method: "+ll);

ll.remove(0);

System.out.println("After invoking remove(index) method: "+ll);

LinkedList<String> ll2=new LinkedList<String>();

ll2.add("Ravi");

ll2.add("Hanumat");

ll.addAll(ll2);

System.out.println("Updated list : "+ll);

ll.removeAll(ll2);

System.out.println("After invoking removeAll() method: "+ll);

ll.removeFirst();

System.out.println("After invoking removeFirst() method: "+ll);

ll.removeLast();

System.out.println("After invoking removeLast() method: "+ll);

ll.removeFirstOccurrence("Gaurav");

System.out.println("After invoking removeFirstOccurrence() method: "+ll);

ll.removeLastOccurrence("Harsh");

System.out.println("After invoking removeLastOccurrence() method: "+ll);

ll.clear();

System.out.println("After invoking clear() method: "+ll);

}

}

12.TestArrayLinkedList

import java.util.\*;

class TestArrayLinked{

public static void main(String args[]){

List<String> al=new ArrayList<String>();

al.add("Ravi");

al.add("Vijay");

al.add("Ravi");

al.add("Ajay");

List<String> al2=new LinkedList<String>();

al2.add("James");

al2.add("Serena");

al2.add("Swati");

al2.add("Junaid");

System.out.println("arraylist: "+al);

System.out.println("linkedlist: "+al2);

}

}

13.Addition By Iteration

import java.util.\*;

public class LinkedList4{

public static void main(String args[]){

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

ll.add("Ravi");

ll.add("Vijay");

ll.add("Ajay");

Iterator i=ll.descendingIterator();

while(i.hasNext())

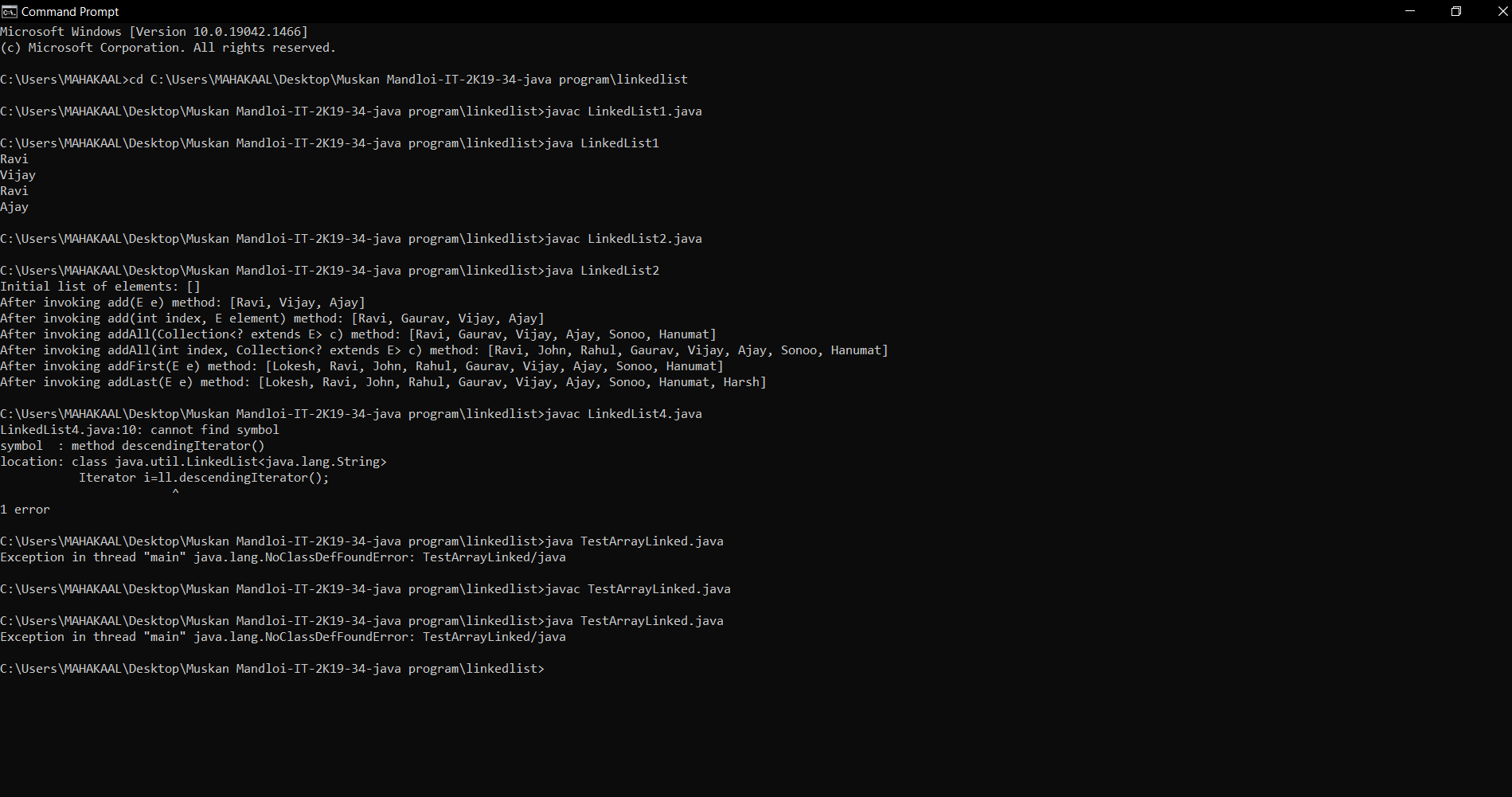
{

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

}

}

}

****

14.**STACK:-**

**1.POP**

import java.util.\*;

import java.io.\*;

public class Pop {

public static void main(String args[])

{

Stack<Integer> stack = new Stack<Integer>();

stack.push(10);

stack.push(15);

stack.push(30);

stack.push(20);

stack.push(5);

System.out.println("Initial Stack: " + stack);

System.out.println("Popped element:”+ stack.pop());

System.out.println("Popped element: "+ stack.pop());

System.out.println("Stack after pop operation “+ stack);

}

}

15.StackPush

import java.util.\*;

import java.io.\*;

public class StackDemoPush {

public static void main(String args[])

{

Stack<String> stack = new Stack<String>();

stack.push("Welcome");

stack.push("To");

stack.push("IIPS");

stack.push("For");

stack.push("MTECH");

System.out.println("Initial Stack: " + stack);

System.out.println("The element at the top of the"+ " stack is: " +stack.peek());

System.out.println("Final Stack: " + stack);

}

}

16.Push

import java.util.Stack;

public class StackEmptyMethodExample

{

public static void main(String[] args)

{

Stack<Integer> stk= new Stack<Integer>();

boolean result = stk.empty();

System.out.println("Is the stack empty? " + result);

stk.push(78);

stk.push(113);

stk.push(90);

stk.push(120);

System.out.println("Elements in Stack: " + stk);

result = stk.empty();

System.out.println("Is the stack empty? " + result);

}

}

17.StackIteration

import java.util.Iterator;

import java.util.Stack;

public class StackIterationExample

{

public static void main (String[] args)

{

Stack stk = new Stack();

stk.push("BMW");

stk.push("Audi");

stk.push("Ferrari");

stk.push("Bugatti");

stk.push("Jaguar");

//iteration over the stack

Iterator iterator = stk.iterator();

while(iterator.hasNext())

{

Object values = iterator.next();

System.out.println(values);

}

}

}

18.StackIterationPush

import java.util.Iterator;

import java.util.Stack;

public class StackIterationExample

{

public static void main (String[] args)

{

Stack stk = new Stack();

stk.push("BMW");

stk.push("Audi");

stk.push("Ferrari");

stk.push("Bugatti");

stk.push("Jaguar");

Iterator iterator = stk.iterator();

while(iterator.hasNext())

{

Object values = iterator.next();

System.out.println(values);

}

}

}

19.StackOperation

import java.util.Iterator;

import java.util.ListIterator;

import java.util.Stack;

public class StackIterationExample3

{

public static void main (String[] args)

{

Stack <Integer> stk = new Stack<Integer>();

stk.push(119);

stk.push(203);

stk.push(988);

ListIterator<Integer> ListIterator = stk.listIterator(stk.size());

System.out.println("Iteration over the Stack from top to bottom:");

while (ListIterator.hasPrevious())

{

Integer avg = ListIterator.previous();

System.out.println(avg);

}

}

}

20.StackPeek

import java.util.Stack;

public class StackPeekMethodExample

{

public static void main(String[] args)

{

Stack<String> stk= new Stack<String>();

// pushing elements into Stack

stk.push("Apple");

stk.push("Grapes");

stk.push("Mango");

stk.push("Orange");

System.out.println("Stack: " + stk);

String fruits = stk.peek();

//prints stack

System.out.println("Element at top: " + fruits);

}

}

21.StackSize

import java.util.Stack;

public class StackSizeExample

{

public static void main (String[] args)

{

Stack stk = new Stack();

stk.push(22);

stk.push(33);

stk.push(44);

stk.push(55);

stk.push(66);

boolean rslt=stk.empty();

System.out.println("Is the stack empty or not? " +rslt);

int x=stk.size();

System.out.println("The stack size is: "+x);

}

}

22.STACKSEARCH

import java.util.Stack;

public class StackSearchMethodExample

{

public static void main(String[] args)

{

Stack<String> stk= new Stack<String>();

stk.push("Mac Book");

stk.push("HP");

stk.push("DELL");

stk.push("Asus");

System.out.println("Stack: " + stk);

int location = stk.search("HP");

System.out.println("Location of Dell: " + location);

}

}

23.StackMethods()

import java.io.\*;

import java.util.\*;

class Test

{

static void stack\_push(Stack<Integer> stack)

{

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

{

stack.push(i);

}

}

static void stack\_pop(Stack<Integer> stack)

{

System.out.println("Pop Operation:");

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

{

Integer y = (Integer) stack.pop();

System.out.println(y);

}

}

static void stack\_peek(Stack<Integer> stack)

{

Integer element = (Integer) stack.peek();

System.out.println("Element on stack top: " + element);

}

static void stack\_search(Stack<Integer> stack, int element)

{

Integer pos = (Integer) stack.search(element);

if(pos == -1)

System.out.println("Element not found");

else

System.out.println("Element is found at position: " + pos);

}

24.public static void main (String[] args)

{

Stack<Integer> stack = new Stack<Integer>();

stack\_push(stack);

stack\_pop(stack);

stack\_push(stack);

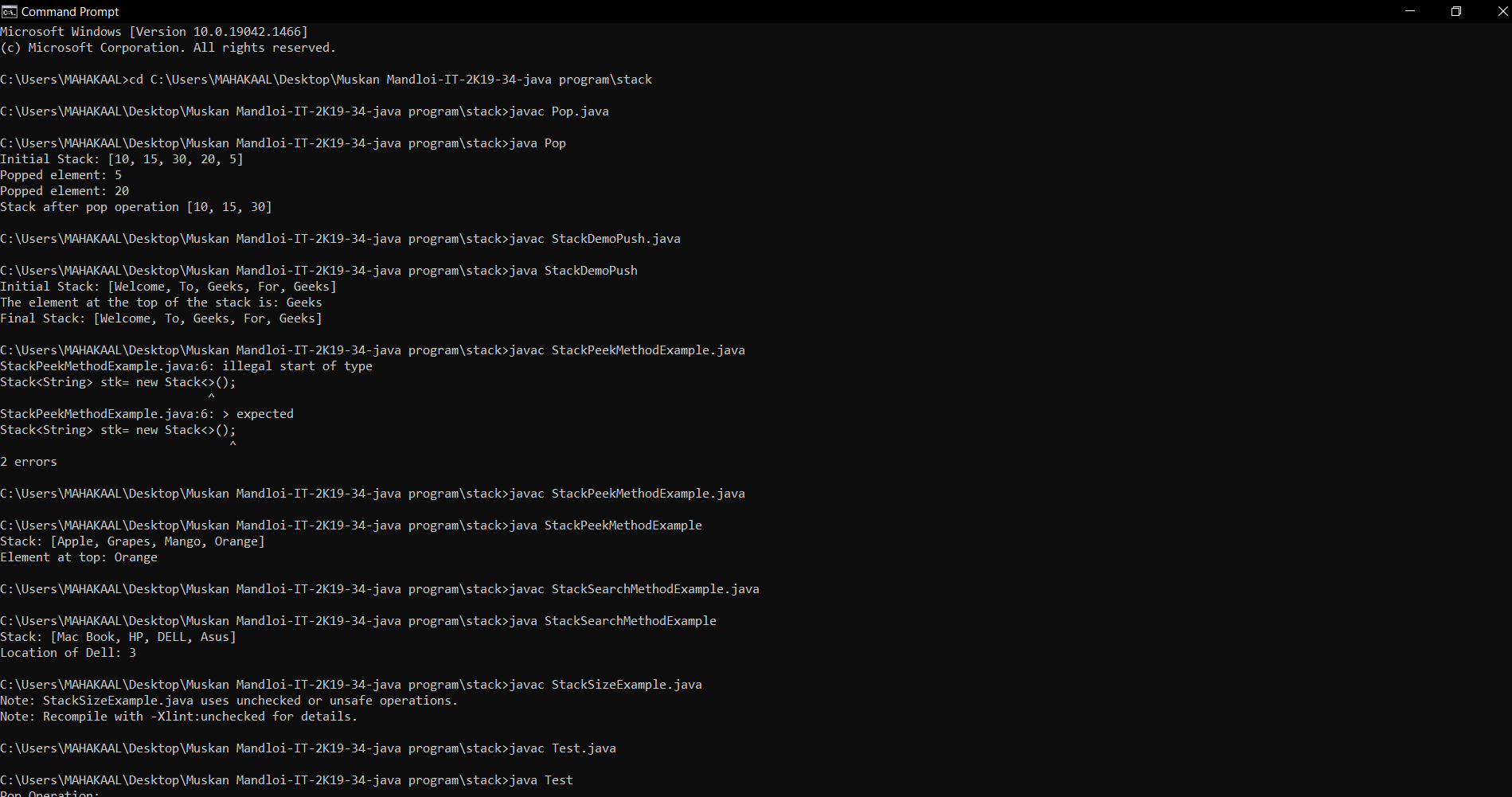
stack\_peek(stack);

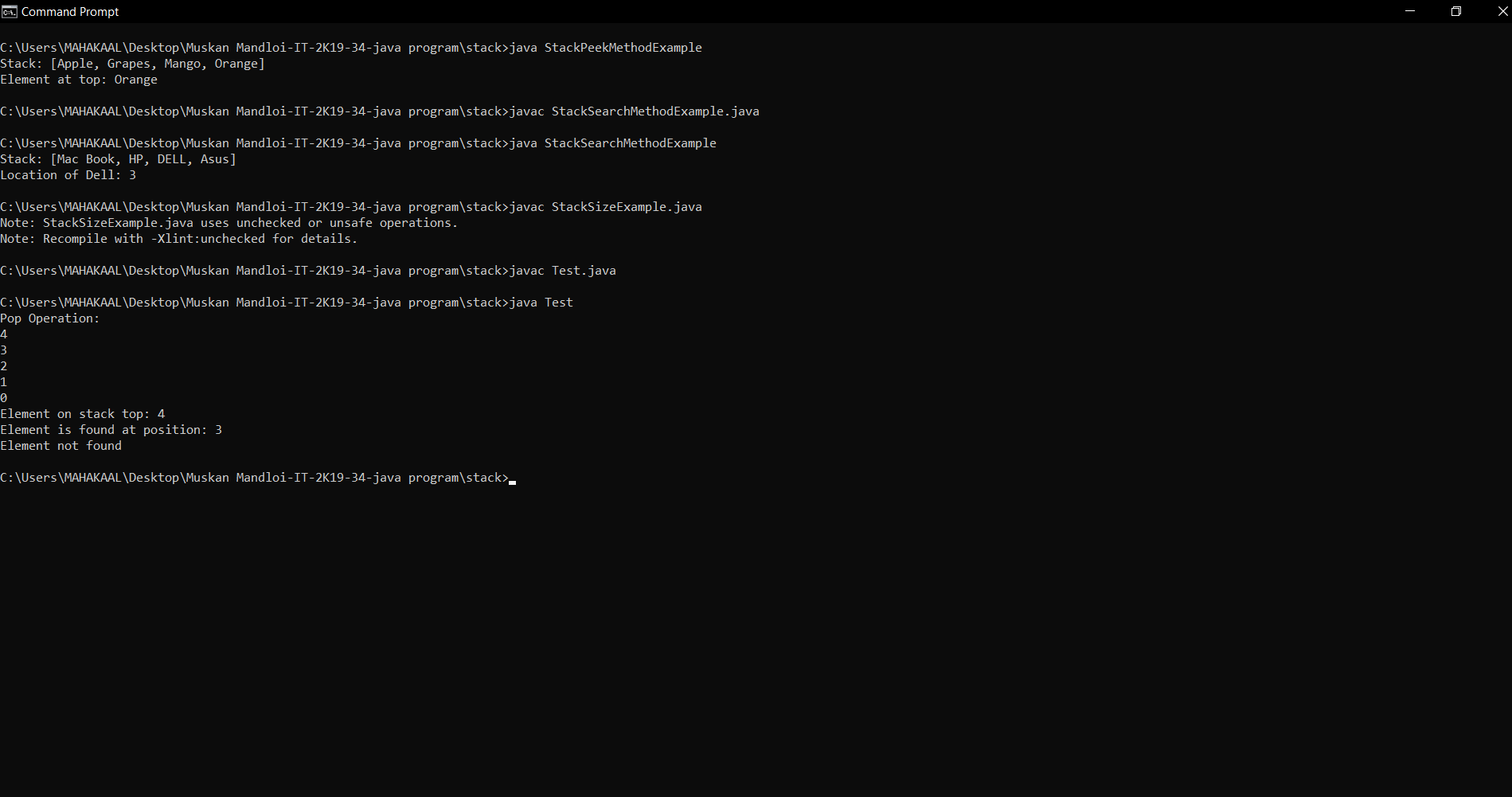
stack\_search(stack, 2);

stack\_search(stack, 6);

}

}





**25. IDENTITY HASHMAP**

import java.util.Map;

import java.util.HashMap;

import java.util.IdentityHashMap;

import java.util.\*;

public class IdentityHashMapDemo {

public static void main(String args[]) {

Map<String, String> ihm = new IdentityHashMap<String,String>();

ihm.put("Priya", "10100");

ihm.put("Shravan", "23290");

ihm.put(new String("Shreya"), "23330");

ihm.put("Anu", "45000");

ihm.put("Nirnay", "67700");

Set set = ihm.entrySet();

Iterator i = set.iterator();

while(i.hasNext()) {

Map.Entry me = (Map.Entry)i.next();

System.out.print(me.getKey() + ": ");

System.out.println(me.getValue());

}

System.out.println();

System.out.println("Size of IdentityHashMap is: " + ihm.size());

}

}

26.import java.util.\*;

public class IdentityHashMapExample1 {

public static void main(String[] args)

{

Map<Integer, String> object\_ihm

= new IdentityHashMap<Integer, String>();

object\_ihm.put(1, "Hello");

object\_ihm.put(2, "How");

object\_ihm.put(3, "Are");

object\_ihm.put(4, "You");

object\_ihm.put(5, "?");

System.out.println("The initial mappings are: "

+ object\_ihm);

String old\_value

= (String)object\_ihm.put(1, "Heyyy");

System.out.println("\nReturned value is: "

+ old\_value);

System.out.println("\nIdentityHashMap after changing the value to existing key:\n " + object\_ihm);

Map<Integer, String> newnew\_IdentityhashMap

= new IdentityHashMap<Integer, String>();

new\_IdentityhashMap.putAll(object\_ihm);

System.out.println("\nThe new IdentityHashMap (copied): "

+ new\_IdentityhashMap);

}

}

26. import java.util.\*;

public class IdentityHashMapExample2 {

public static void main(String[] args)

{

Map<Integer, String> object\_ihm

= new IdentityHashMap<Integer, String>();

object\_ihm.put(1, "Hello");

object\_ihm.put(2, "How");

object\_ihm.put(3, "Are");

object\_ihm.put(4, "You");

object\_ihm.put(5, "?");

System.out.println("The initial mappings are: "

+ object\_ihm);

String returned\_value = (String)object\_ihm.remove(3);

System.out.println("\nThe new IdentityHashMap is: "

+ object\_ihm);

}

}

27. import java.util.\*;

public class IdentityHashMapExample3 {

public static void main(String[] args)

{

Map<Integer, String> object\_ihm

= new IdentityHashMap<Integer, String>();

object\_ihm.put(1, "Hello");

object\_ihm.put(2, "How");

object\_ihm.put(3, "Are");

object\_ihm.put(4, "You");

object\_ihm.put(5, "?");

System.out.println("The initial mappings are: "

+ object\_ihm);

System.out.println("Value of key 5 is:"

+ object\_ihm.get(5));

System.out.println("Value of key 2 is:"

+ object\_ihm.get(2));

System.out.println("Set view of keys: "

+ object\_ihm.keySet());

System.out.println("The set is: "

+ object\_ihm.entrySet());

}

}

28.. import java.util.\*;

public class IdentityHashMapExample4 {

public static void main(String[] args)

{

Map<Integer, String> object\_ihm

= new IdentityHashMap<Integer, String>();

object\_ihm.put(1, "Hello");

object\_ihm.put(2, "How");

object\_ihm.put(3, "Are");

object\_ihm.put(4, "You");

object\_ihm.put(5, "?");

System.out.println("The initial mappings are: "

+ object\_ihm);

Iterator<IdentityHashMap.Entry<Integer, String> >

itr = object\_ihm.entrySet().iterator();

while (itr.hasNext()) {

IdentityHashMap.Entry<Integer, String> entry

= itr.next();

System.out.println("Key : " + entry.getKey()

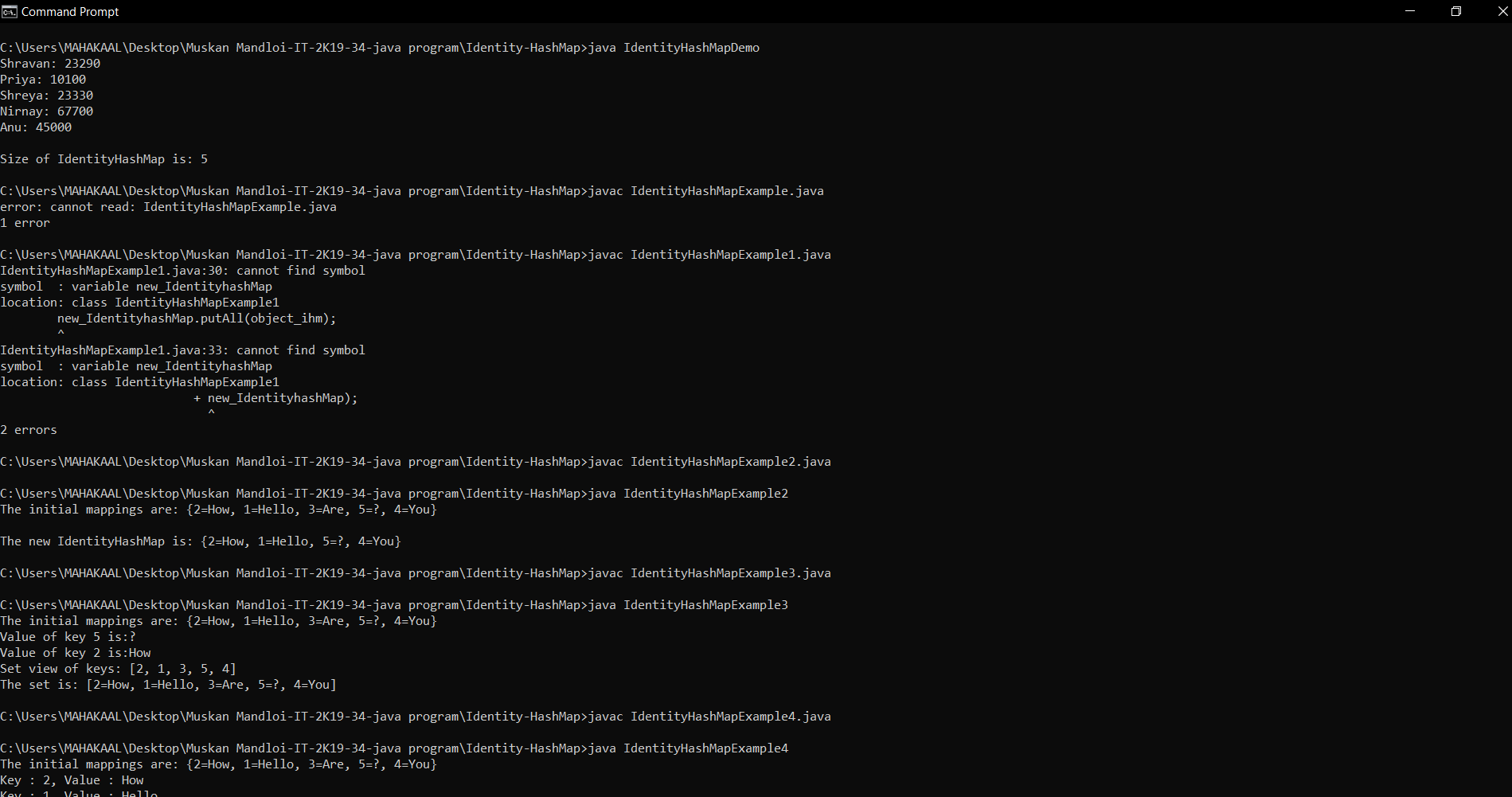
+ ", Value : "

+ entry.getValue());

}

}

}



**Multithreading**

29.public class deadlock {

public static void main(String[] args) {

final String resource1 = "ratan jaiswal";

final String resource2 = "vimal jaiswal";

Thread t1 = new Thread() {

public void run() {

synchronized (resource1) {

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

try { Thread.sleep(100);} catch (Exception e) {}

synchronized (resource2) {

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

}

}

}

};

Thread t2 = new Thread() {

public void run() {

synchronized (resource2) {

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

try { Thread.sleep(100);} catch (Exception e) {}

synchronized (resource1) {

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

}

}

}

};

t1.start();

t2.start();

}

30.package thread3;

class ThreadId extends Thread {

public void run() {

try {

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

}

catch (Exception e) {

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

}

}

}

31.public class getid{

public static void main(String[] args)

{

int n = 8;

for (int i=0; i<8; i++)

{

ThreadId object = new ThreadId();

object.start();

}

}

}

32.package thread5;

class Thread1 extends Thread {

public void run( ) {

try{

System.out.println (" First thread starts running" );

sleep(10000);

System.out.println (" First thread finishes running" );

}

catch(Exception e){ }

}

}

33.class Thread2 extends Thread {

public void run( ) {

try{

System.out.println ( "Second thread starts running");

System.out.println ( "Second thread is suspended itself ");

suspend( );

System.out.println (" Second thread runs again" );

}

catch(Exception e){ }

}

}

class method2{

public static void main (String args[ ] ){

try{

Thread1 first = new Thread1( );

Thread2 second= new Thread2( );

first.start( );

second.start( );

System.out.println("Revive the second thread" ); second.resume( );

System.out.println ("Second thread went for 10 seconds sleep " );

second.sleep (10000);

System.out.println ("Wake up second thread and finishes running" );

System.out.println ( " Demonstration is finished ");

}

catch(Exception e){ }

}

}

34.package generic;

class Helper implements Runnable {

public void run() {

try {

System.out.println("thread2 going to sleep for 5000");

Thread.sleep(5000);

}

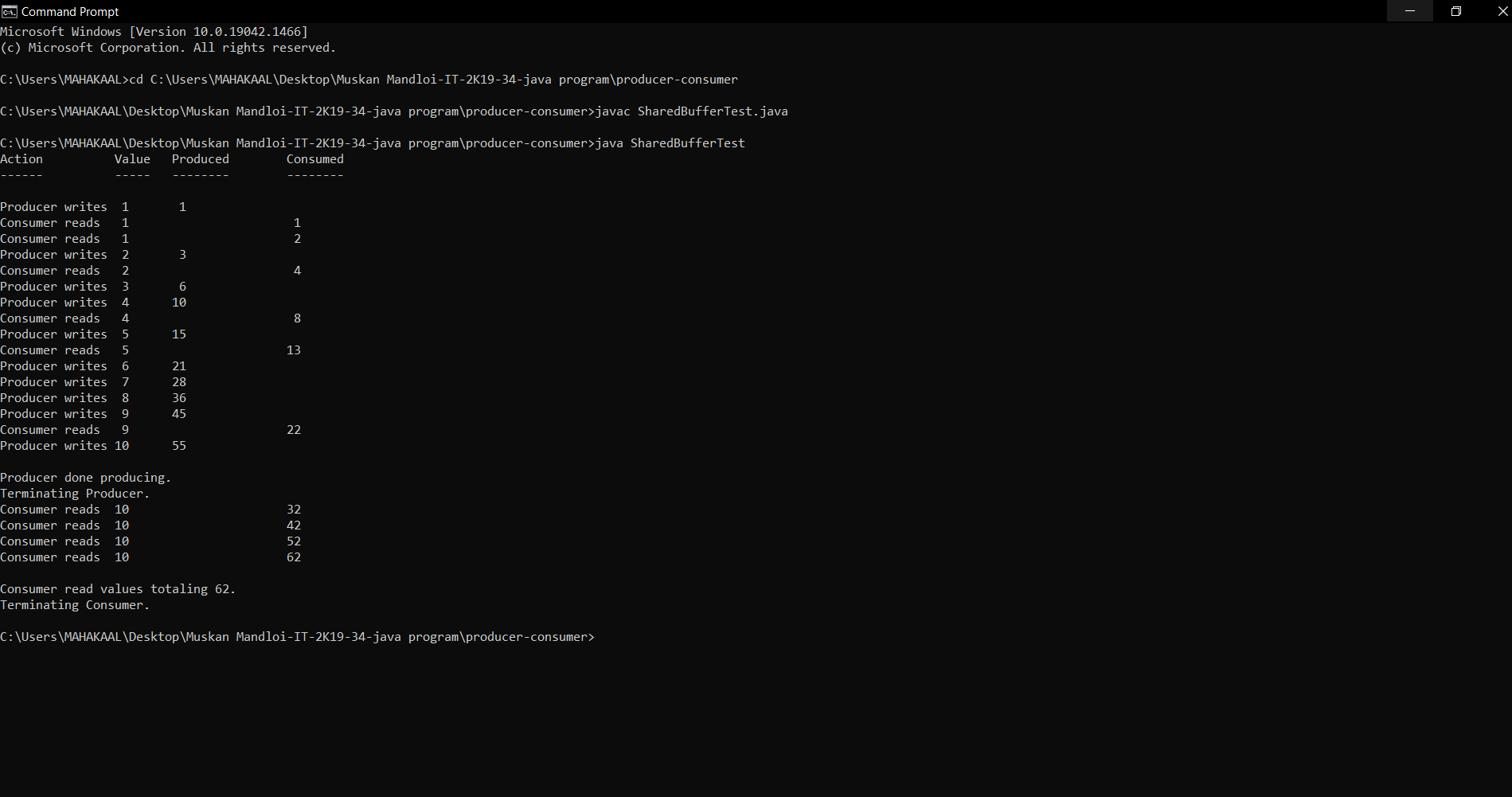
catch (InterruptedException e) {

System.out.println("Thread2 interrupted");

}

}

}



35.public class MethodTest implements Runnable {

public void run() {

}

public static void main(String[] args) {

Test obj = new Test();

Helper obj2 = new Helper();

Thread thread1 = new Thread(obj);

Thread thread2 = new Thread(obj2);

thread1.start();

thread2.start();

ClassLoader loader = thread1.getContextClassLoader();

Thread thread3 = new Thread(new Helper());

System.out.println(Thread.activeCount());

thread1.checkAccess();

Thread t = Thread.currentThread();

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

System.out.println("Thread1 name: " + thread1.getName());

System.out.println("Thread1 ID: " + thread1.getId());

System.out.println("Priority of thread1 = " + thread1.getPriority());

System.out.println(thread1.getState());

thread2 = new Thread(obj2);

thread2.start();

thread2.interrupt();

System.out.println("Is thread2 interrupted? " + thread2.interrupted() );

System.out.println("Is thread2 alive? " + thread2.isAlive());

thread1 = new Thread(obj);

thread1.setDaemon(true);

System.out.println("Is thread1 a daemon thread? " + thread1.isDaemon());

System.out.println("Is thread1 interrupted? " + thread1.isInterrupted());

System.out.println("thread1 waiting for thread2 to join");

try {

thread2.join();

}

catch (InterruptedException e) {

e.printStackTrace();

}

thread1.setName("child thread xyz");

System.out.println("New name set for thread 1" + thread1.getName());

thread1.setPriority(5);

thread2.yield();

System.out.println(thread1.toString());

Thread[] tarray = new Thread[3];

Thread.enumerate(tarray);

System.out.println("List of active threads:");

System.out.printf("[");

for (Thread thread : tarray) {

System.out.println(thread);

}

System.out.printf("]\n");

System.out.println(Thread.getAllStackTraces());

ClassLoader classLoader = thread1.getContextClassLoader();

System.out.println(classLoader.toString());

System.out.println(thread1.getDefaultUncaughtExceptionHandler());

thread2.setUncaughtExceptionHandler(thread1.getDefaultUncaughtExceptionHandler());

thread1.setContextClassLoader(thread2.getContextClassLoader());

thread1.setDefaultUncaughtExceptionHandler(thread2.getUncaughtExceptionHandler());

thread1 = new Thread(obj);

StackTraceElement[] trace = thread1.getStackTrace();

System.out.println("Printing stack trace elements for thread1:");

for (StackTraceElement e : trace) {

System.out.println(e);

}

ThreadGroup grp = thread1.getThreadGroup();

System.out.println("ThreadGroup to which thread1 belongs " + grp.toString());

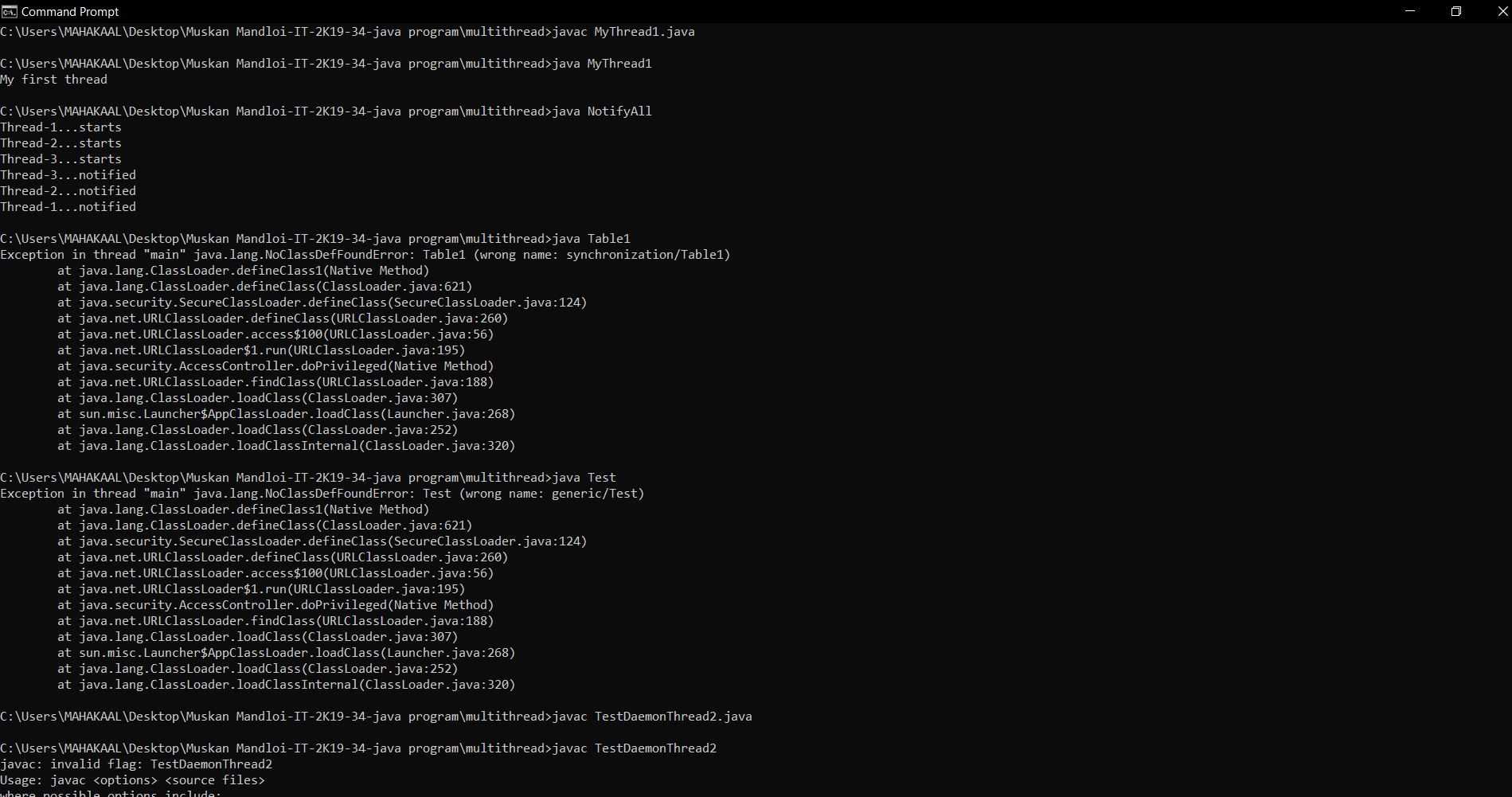
System.out.println(thread1.getUncaughtExceptionHandler());

System.out.println("Does thread1 holds Lock? " + thread1.holdsLock(obj2));

Thread.dumpStack();

}

}



36.package generic;

class Helper implements Runnable {

public void run() {

try {

System.out.println("thread2 going to sleep for 5000");

Thread.sleep(5000);

}

catch (InterruptedException e) {

System.out.println("Thread2 interrupted");

}

}

}

37.public class Test implements Runnable {

public void run() {

}

public static void main(String[] args) {

Test obj = new Test();

Helper obj2 = new Helper();

Thread thread1 = new Thread(obj);

Thread thread2 = new Thread(obj2);

thread1.start();

thread2.start();

ClassLoader loader = thread1.getContextClassLoader();

Thread thread3 = new Thread(new Helper());

System.out.println(Thread.activeCount());

thread1.checkAccess();

Thread t = Thread.currentThread();

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

System.out.println("Thread1 name: " + thread1.getName());

System.out.println("Thread1 ID: " + thread1.getId());

System.out.println("Priority of thread1 = " + thread1.getPriority());

System.out.println(thread1.getState());

thread2 = new Thread(obj2);

thread2.start();

thread2.interrupt();

System.out.println("Is thread2 interrupted? " + thread2.interrupted() );

System.out.println("Is thread2 alive? " + thread2.isAlive());

thread1 = new Thread(obj);

thread1.setDaemon(true);

System.out.println("Is thread1 a daemon thread? " + thread1.isDaemon());

System.out.println("Is thread1 interrupted? " + thread1.isInterrupted());

System.out.println("thread1 waiting for thread2 to join");

try {

thread2.join();

}

catch (InterruptedException e) {

e.printStackTrace();

}

thread1.setName("child thread xyz");

System.out.println("New name set for thread 1" + thread1.getName());

thread1.setPriority(5);

thread2.yield();

System.out.println(thread1.toString());

Thread[] tarray = new Thread[3];

Thread.enumerate(tarray);

System.out.println("List of active threads:");

System.out.printf("[");

for (Thread thread : tarray) {

System.out.println(thread);

}

System.out.printf("]\n");

System.out.println(Thread.getAllStackTraces());

ClassLoader classLoader = thread1.getContextClassLoader();

System.out.println(classLoader.toString());

System.out.println(thread1.getDefaultUncaughtExceptionHandler());

thread2.setUncaughtExceptionHandler(thread1.getDefaultUncaughtExceptionHandler());

thread1.setContextClassLoader(thread2.getContextClassLoader());

thread1.setDefaultUncaughtExceptionHandler(thread2.getUncaughtExceptionHandler());

thread1 = new Thread(obj);

StackTraceElement[] trace = thread1.getStackTrace();

System.out.println("Printing stack trace elements for thread1:");

for (StackTraceElement e : trace) {

System.out.println(e);

}

ThreadGroup grp = thread1.getThreadGroup();

System.out.println("ThreadGroup to which thread1 belongs " + grp.toString());

System.out.println(thread1.getUncaughtExceptionHandler());

System.out.println("Does thread1 holds Lock? " + thread1.holdsLock(obj2));

Thread.dumpStack();

}

}

38.public class TestInterruptingThread4 extends Thread{

public void run(){

for(int i=1;i<=2;i++){

if(Thread.interrupted()){

System.out.println("code for interrupted thread");

}

else{

System.out.println("code for normal thread");

}

}//end of for loop

}

public static void main(String args[]){

TestInterruptingThread4 t1=new TestInterruptingThread4();

TestInterruptingThread4 t2=new TestInterruptingThread4();

t1.start();

t1.interrupt();

t2.start();

}

}

40.package thread2;

import java.util.\*;

class ThreadY implements Runnable {

public void run(){class ThreadX implements Runnable{

public void run( ) {

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

System.out.println("Thread X with i = "+ -1\*i);

}

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

}

{

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

System.out.println("Thread Y with j = "+ 2\*j);

}

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

}

}

class ThreadZ implements Runnable{

public void run( ) {

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

System.out.println("Thread Z with k = "+ (2\*k-1));

}

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

}

}

49.public class thread2 {

public static void main(String []args) {

ThreadX x = new ThreadX();

Thread t1 = new Thread(x);

ThreadY y = new ThreadY();

Thread t2 = new Thread(y);

//ThreadZ z = new ThreadZ();

//Thread t3 = new Thread(z);

Thread t3 = new Thread(new ThreadZ());

t1.start();

t2.start();

t3.start();

System.out.println("... Multithreading is over ");

}

}

50.import java.io.\*;

class ThreadJoin extends Thread

{

public void run()

{

for (int j = 0; j < 2; j++)

{

try

{

Thread.sleep(300);

System.out.println("The current thread name is: " + Thread.currentThread().getName());

}

catch(Exception e)

{

System.out.println("The exception has been caught: " + e);

}

System.out.println( j );

}

}

}

public class ThreadJoinExample

{

public static void main (String argvs[])

{

ThreadJoin th1 = new ThreadJoin();

ThreadJoin th2 = new ThreadJoin();

ThreadJoin th3 = new ThreadJoin();

th1.start();

try

{

System.out.println("The current thread name is: "+ Thread.currentThread().getName());

th1.join();

}

catch(Exception e)

{

System.out.println("The exception has been caught " + e);

}

th2.start();

try

{

System.out.println("The current thread name is: " + Thread.currentThread().getName());

th2.join();

}

catch(Exception e)

{

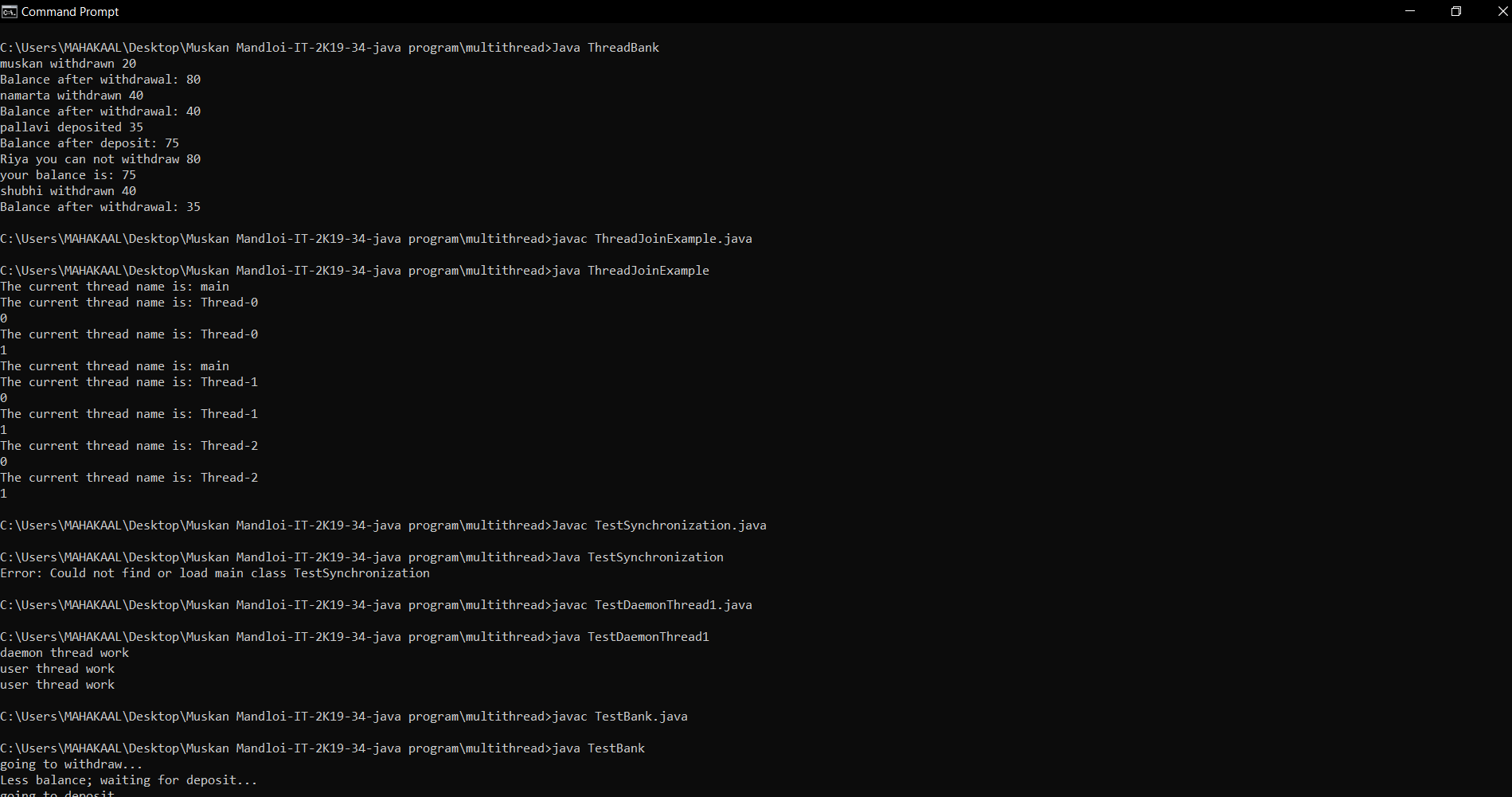
System.out.println("The exception has been caught " + e);

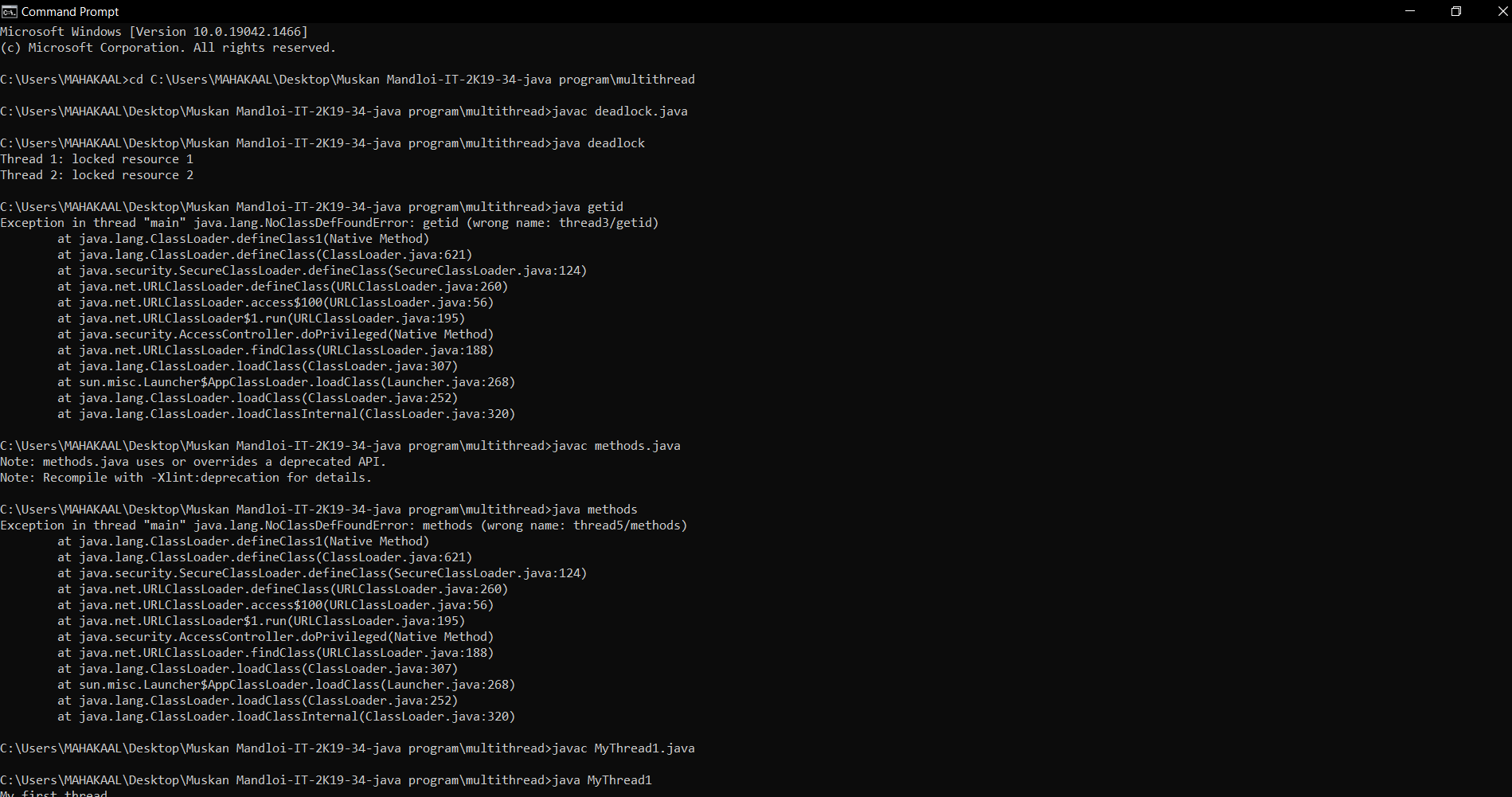
}

th3.start();

}

}





**Generic –Non Generic**

51.package generic;

import java.util.\*;

public class generic {

private Object t;

public Object get() {

return t;

}

public void set(Object t) {

this.t = t;

}

public static void main(String args[]){

generic type = new generic();

type.set("muskan");

String str = (String) type.get();

ClassCastException

}

}

52.package generic;

public class genericClass<T> {

private T t;

public T get(){

return this.t;

}

public void set(T t1){

this.t=t1;

}

public static void main(String args[]){

genericClass<String> type = new genericClass<String>();

type.set("Pankaj"); //valid

genericClass type1 = new genericClass(); //raw type

type1.set("Pankaj"); //valid

type1.set(10); //valid and autoboxing support

}

}

53.package java.lang;

import java.util.\*;

public interface genericinterface<T> {

public int compareTo(T o);

}

54. **Consumer-Producer**

import java.util.concurrent.ExecutorService;

import java.util.concurrent.Executors;

public class SharedBufferTest

{

public static void main( String[] args )

{

ExecutorService application = Executors.newFixedThreadPool( 2 );

Buffer sharedLocation = new UnsynchronizedBuffer();

System.out.println( "Action\t\tValue\tProduced\tConsumed" );

System.out.println( "------\t\t-----\t--------\t--------\n" );

try

{

application.execute( new Producer( sharedLocation ) );

application.execute( new Consumer( sharedLocation ) );

}

catch ( Exception exception )

{

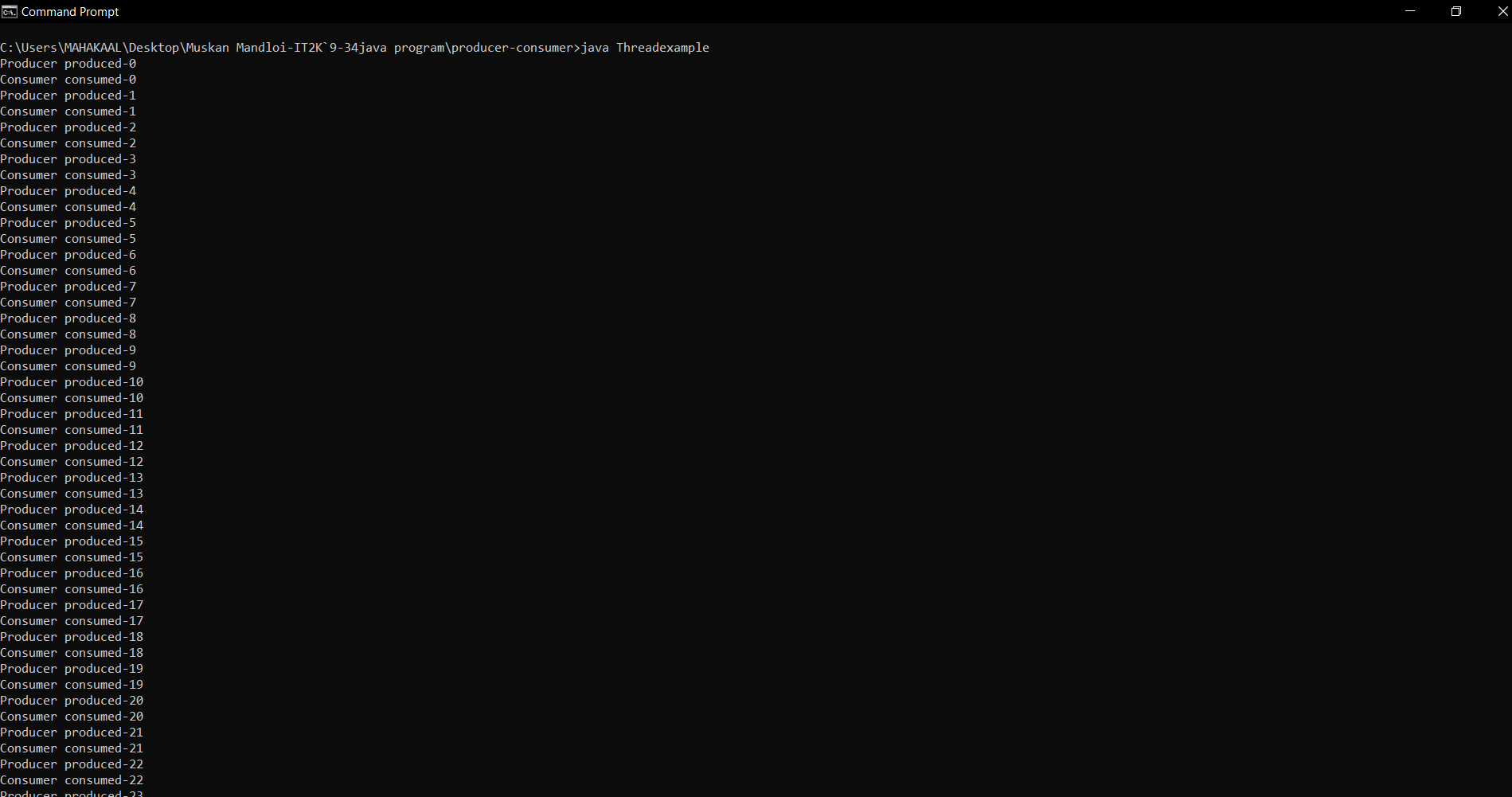
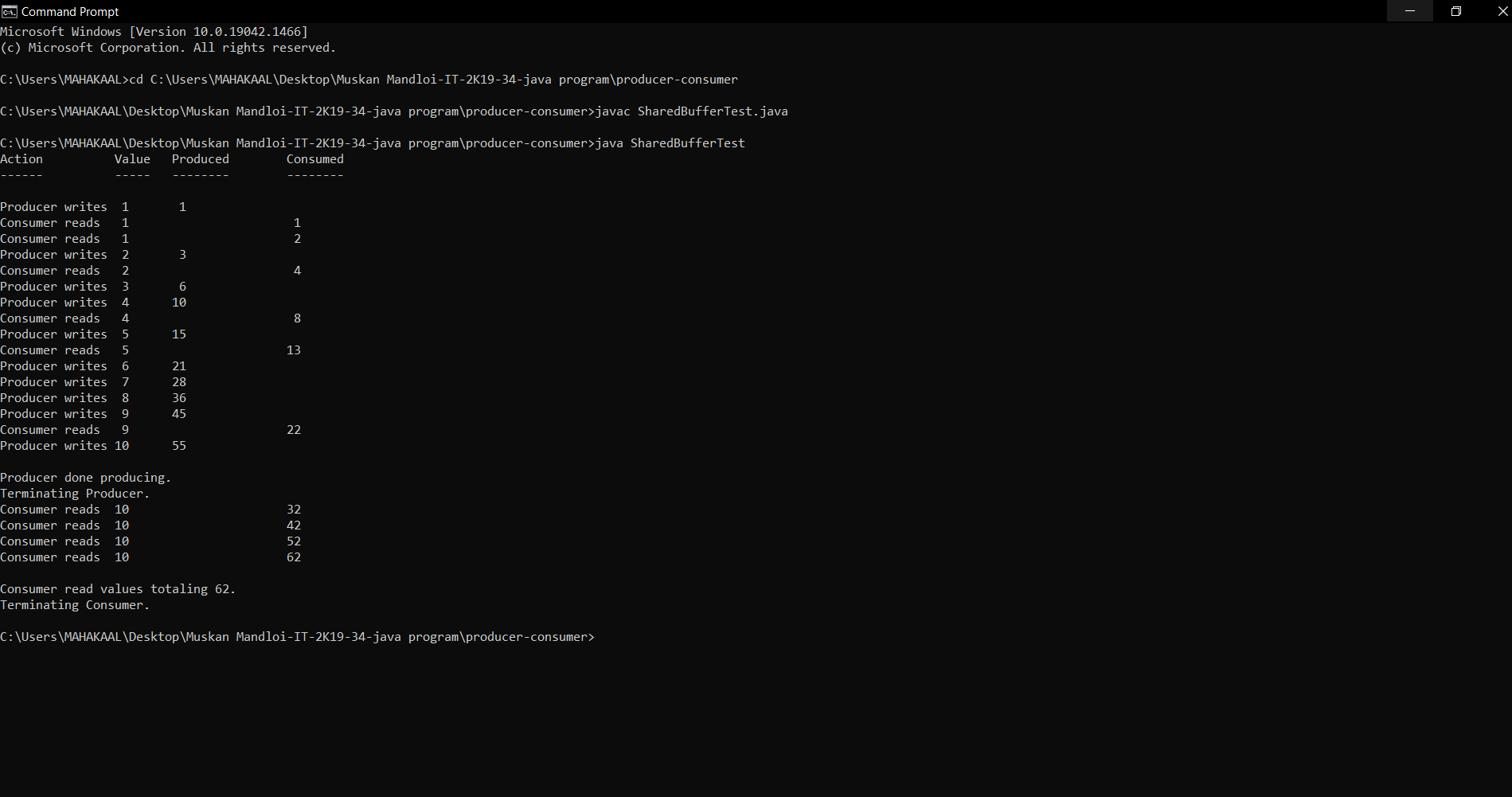
exception.printStackTrace();

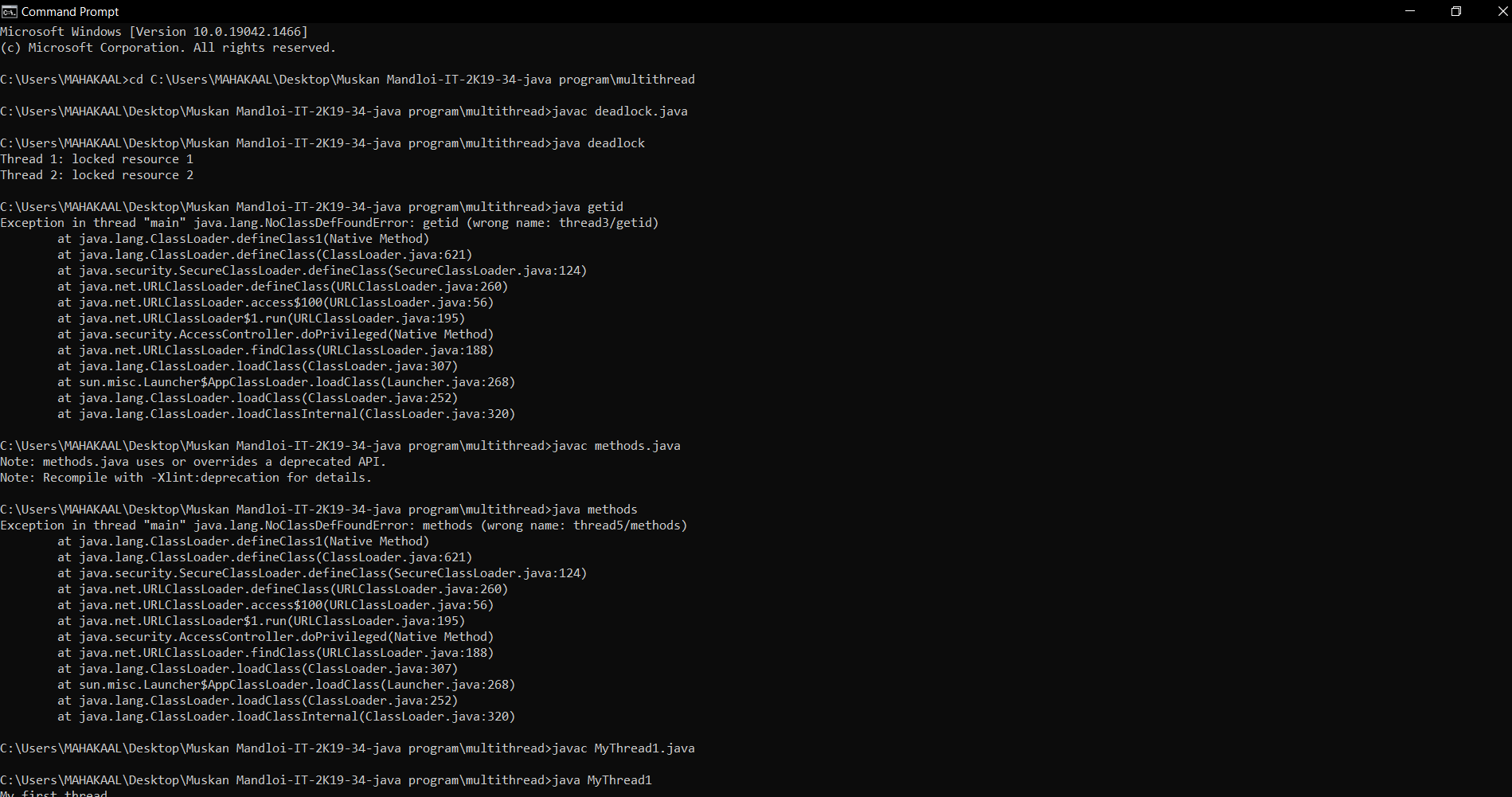
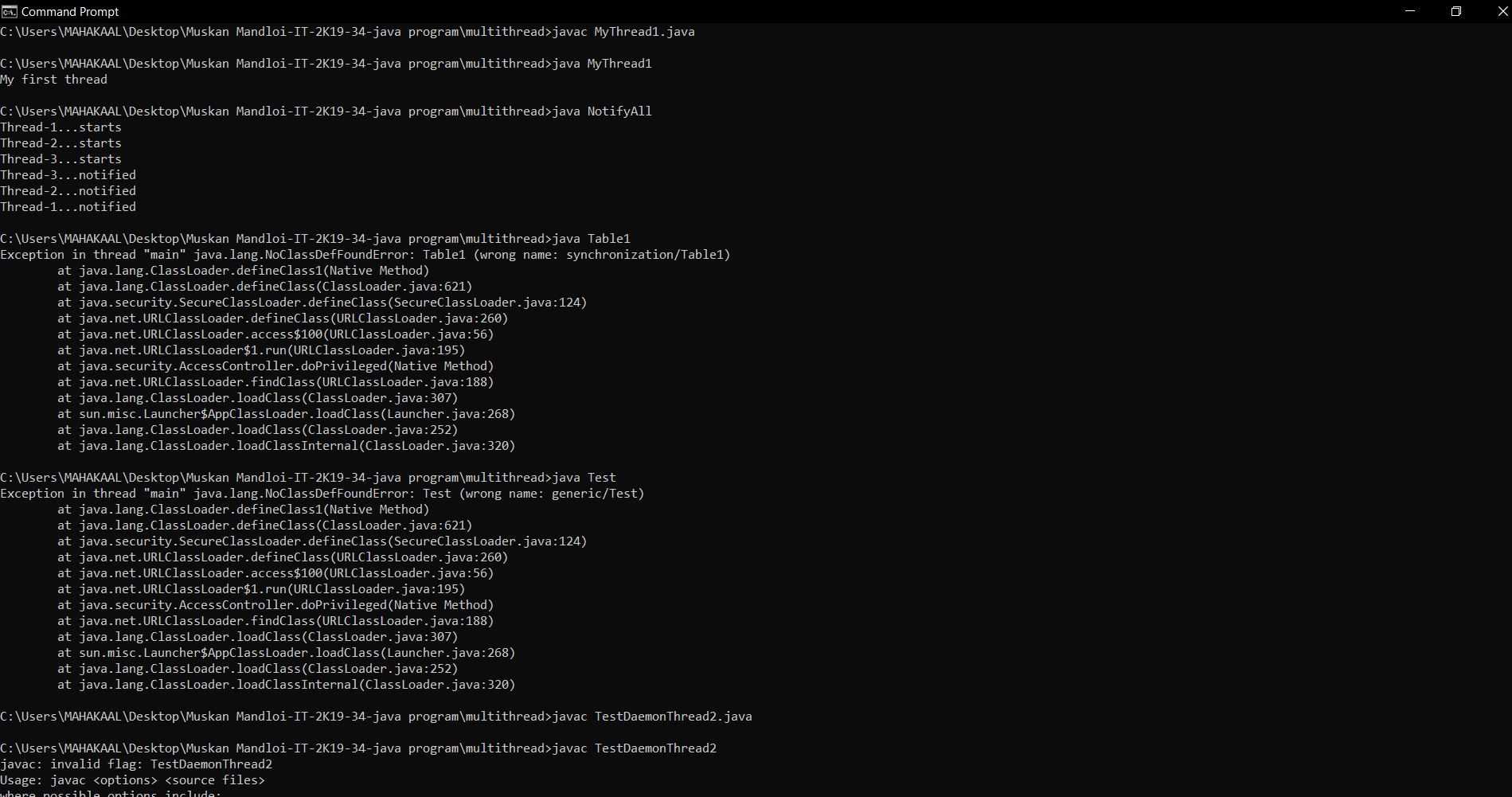
}

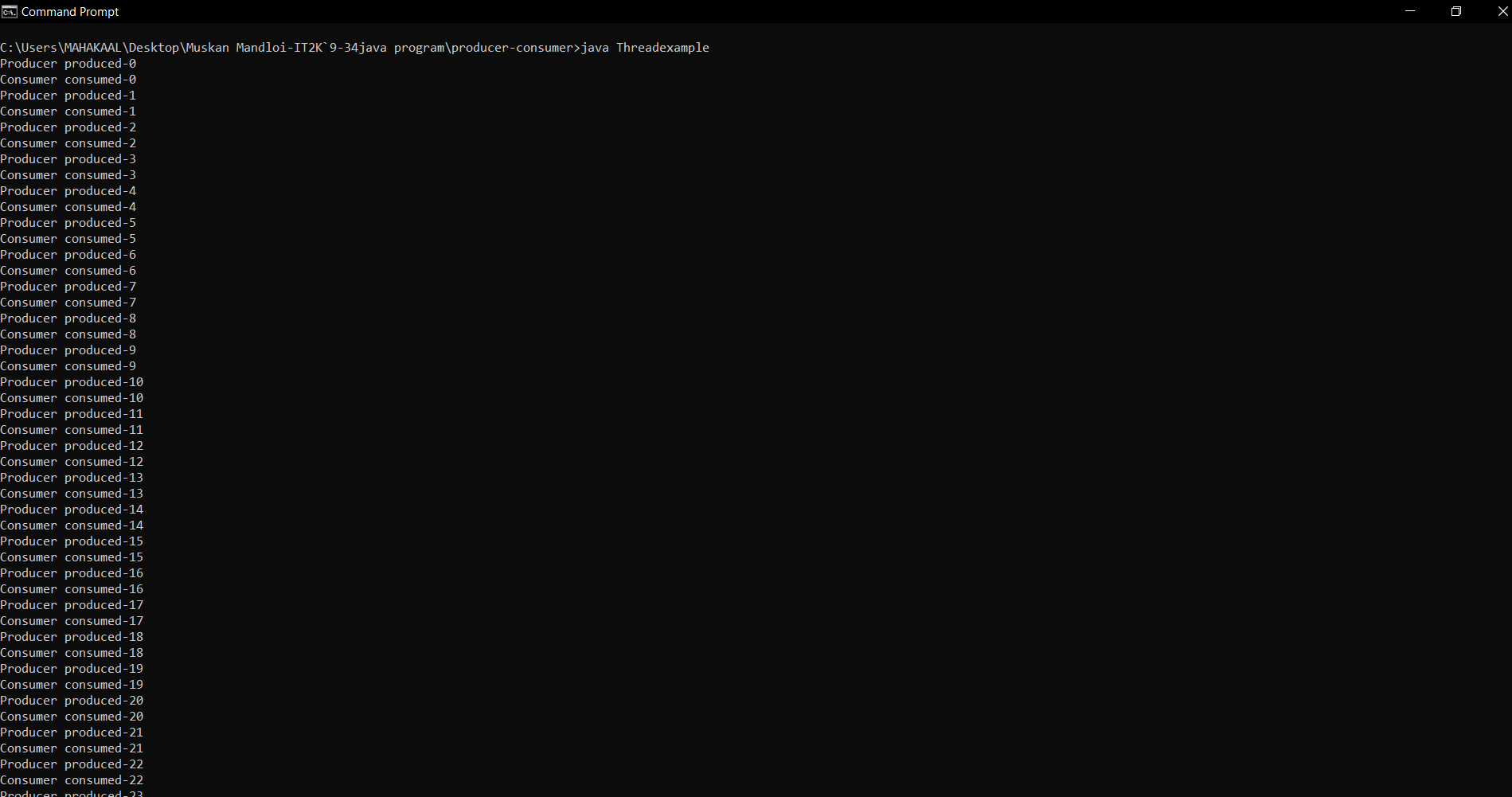
application.shutdown();

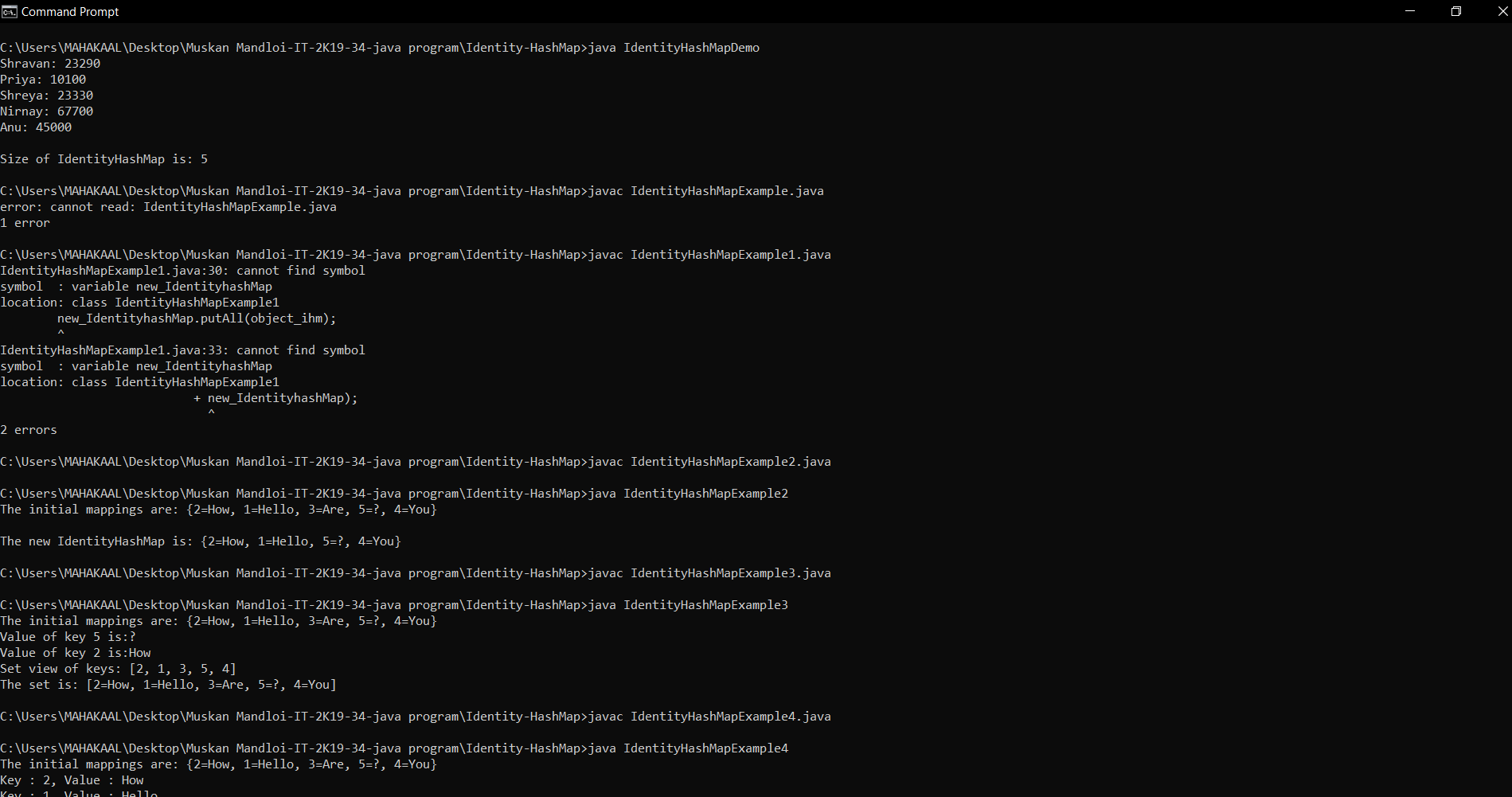
}

}









**Map :-**

55. import java.util.\*;

public class MapExample1 {

public static void main(String[] args) {

Map map=new HashMap();

map.put(1,"Amit");

map.put(5,"Rahul");

map.put(2,"Jai");

map.put(6,"Amit");

Set set=map.entrySet();

Iterator itr=set.iterator();

while(itr.hasNext()){

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

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

}

}

}

56. 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");

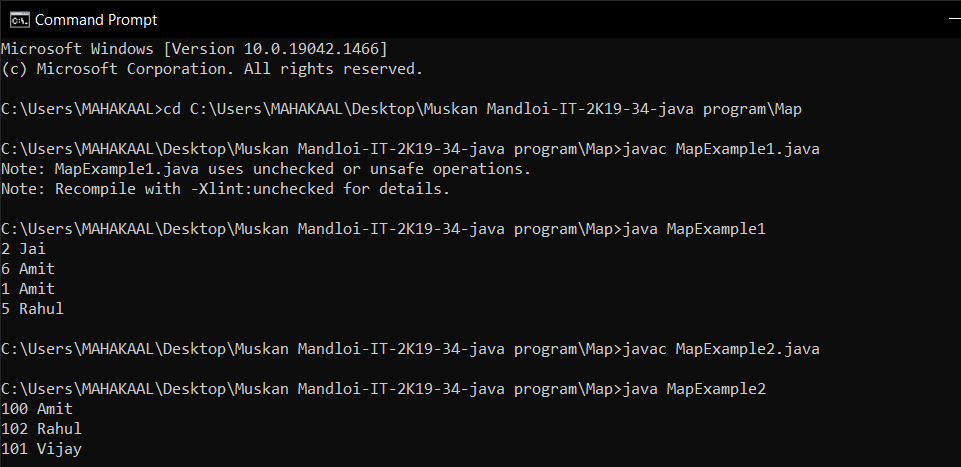
for(Map.Entry m:map.entrySet()){

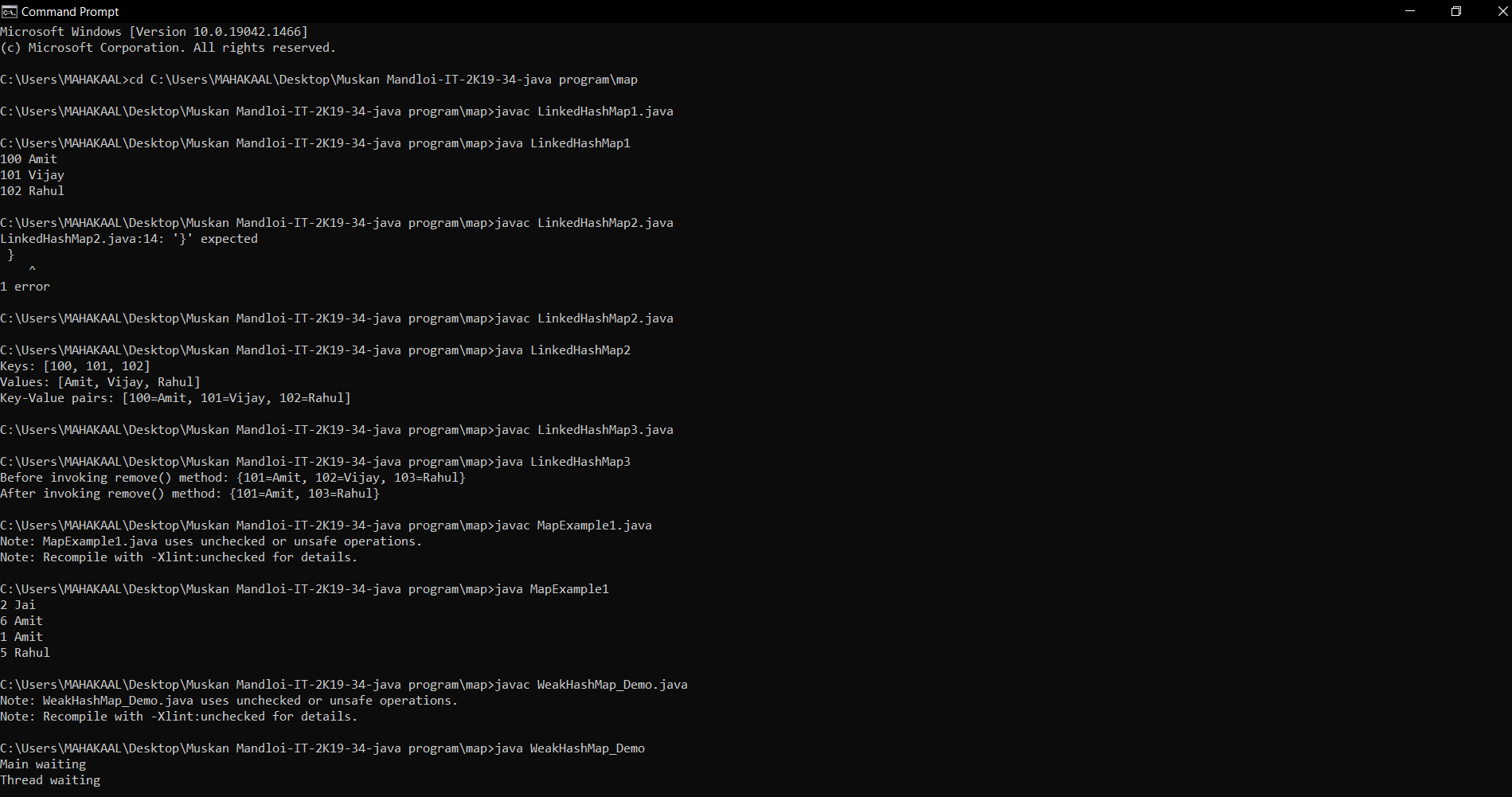
System.out.println(m.getKey()+" "+m.getValue());

}

}

}





**Linked HashMap:-**

58. import java.util.\*;

class LinkedHashMap1{

public static void main(String args[]){

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

hm.put(100,"Amit");

hm.put(101,"Vijay");

hm.put(102,"Rahul");

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

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

}

}

}

59. import java.util.\*;

class LinkedHashMap2{

public static void main(String args[]){

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

map.put(100,"Amit");

map.put(101,"Vijay");

map.put(102,"Rahul");

//Fetching key

System.out.println("Keys: "+map.keySet());

//Fetching value

System.out.println("Values: "+map.values());

//Fetching key-value pair

System.out.println("Key-Value pairs: "+map.entrySet());

}

}

60. import java.util.\*;

public class LinkedHashMap3 {

public static void main(String args[]) {

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

map.put(101,"Amit");

map.put(102,"Vijay");

map.put(103,"Rahul");

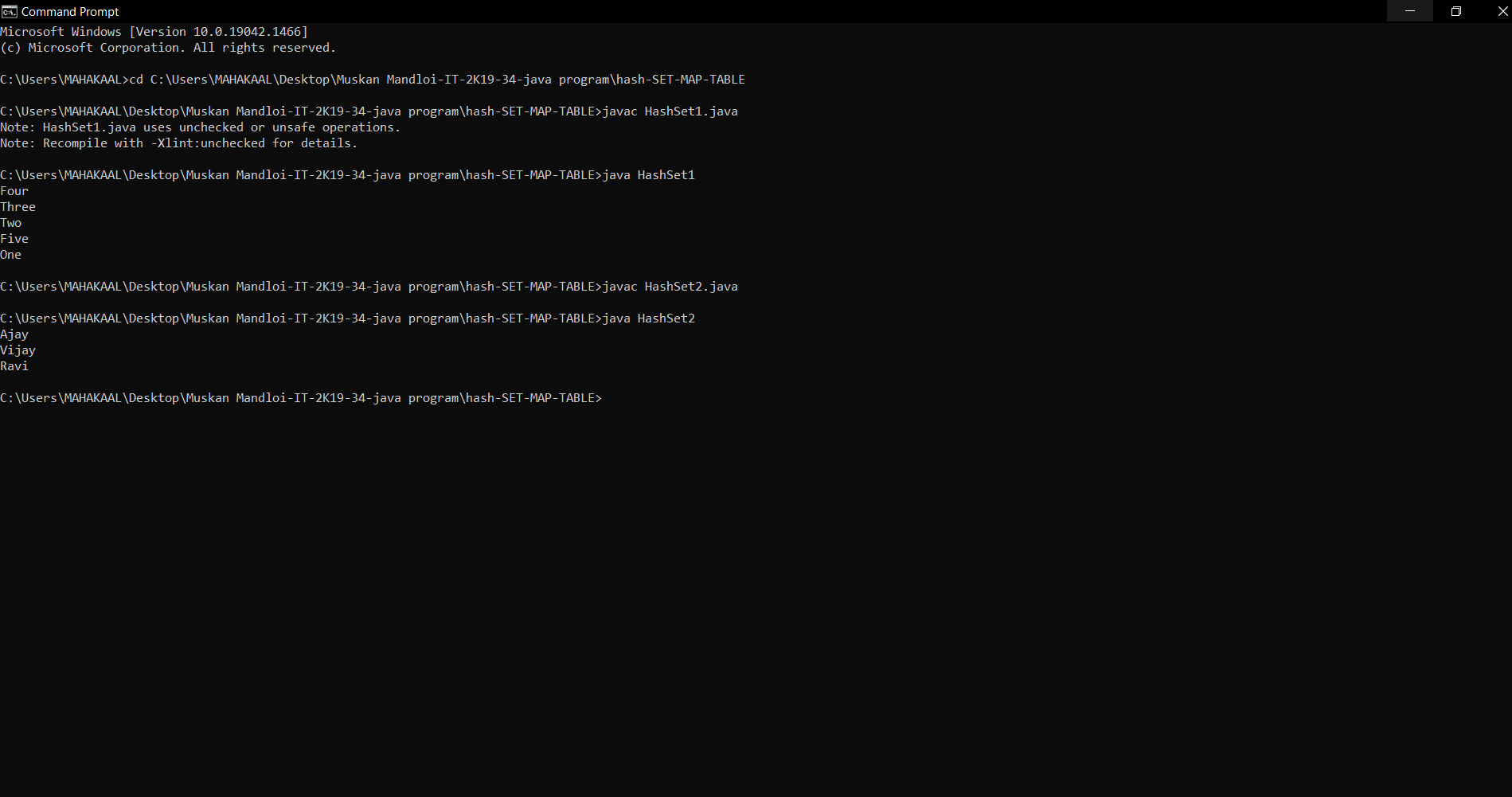
System.out.println("Before invoking remove() method: "+map);

map.remove(102);

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

}

}

****

**Linked HASHSET:-**

61. import java.util.\*;

class LinkedHashSet1{

public static void main(String args[]){

//Creating HashSet and adding elements

LinkedHashSet<String> set=new LinkedHashSet();

set.add("One");

set.add("Two");

set.add("Three");

set.add("Four");

set.add("Five");

Iterator<String> i=set.iterator();

while(i.hasNext())

{

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

}

}

}

62. import java.util.\*;

class LinkedHashSet2{

public static void main(String args[]){

LinkedHashSet<String> al=new LinkedHashSet<String>();

al.add("Ravi");

al.add("Vijay");

al.add("Ravi");

al.add("Ajay");

Iterator<String> itr=al.iterator();

while(itr.hasNext()){

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

}

}

}

**Hash Map**

63. import java.util.\*;

class LinkedHashSet2{

public static void main(String args[]){

LinkedHashSet<String> al=new LinkedHashSet<String>();

al.add("Ravi");

al.add("Vijay"); import java.util.\*;

public class WeakHashMap\_Demo {

private static Map map;

public static void main (String args[]) {

map = new WeakHashMap();

map.put(new String("Maine"), "Augusta");

Runnable runner = new Runnable() {

public void run() {

while (map.containsKey("Maine")) {

try {

Thread.sleep(500);

} catch (InterruptedException ignored) {

}

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

System.gc();

}

}

};

Thread t = new Thread(runner);

t.start();

System.out.println("Main waiting");

try {

t.join();

} catch (InterruptedException ignored) {

}

}

}

al.add("Ravi");

al.add("Ajay");

Iterator<String> itr=al.iterator();

while(itr.hasNext()){

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

}

}

}

64. import java.util.\*;

public class WeakHashMap\_Demo {

private static Map map;

public static void main (String args[]) {

map = new WeakHashMap();

map.put(new String("Maine"), "Augusta");

Runnable runner = new Runnable() {

public void run() {

while (map.containsKey("Maine")) {

try {

Thread.sleep(500);

} catch (InterruptedException ignored) {

}

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

System.gc();

}

}

};

Thread t = new Thread(runner);

t.start();

System.out.println("Main waiting");

try {

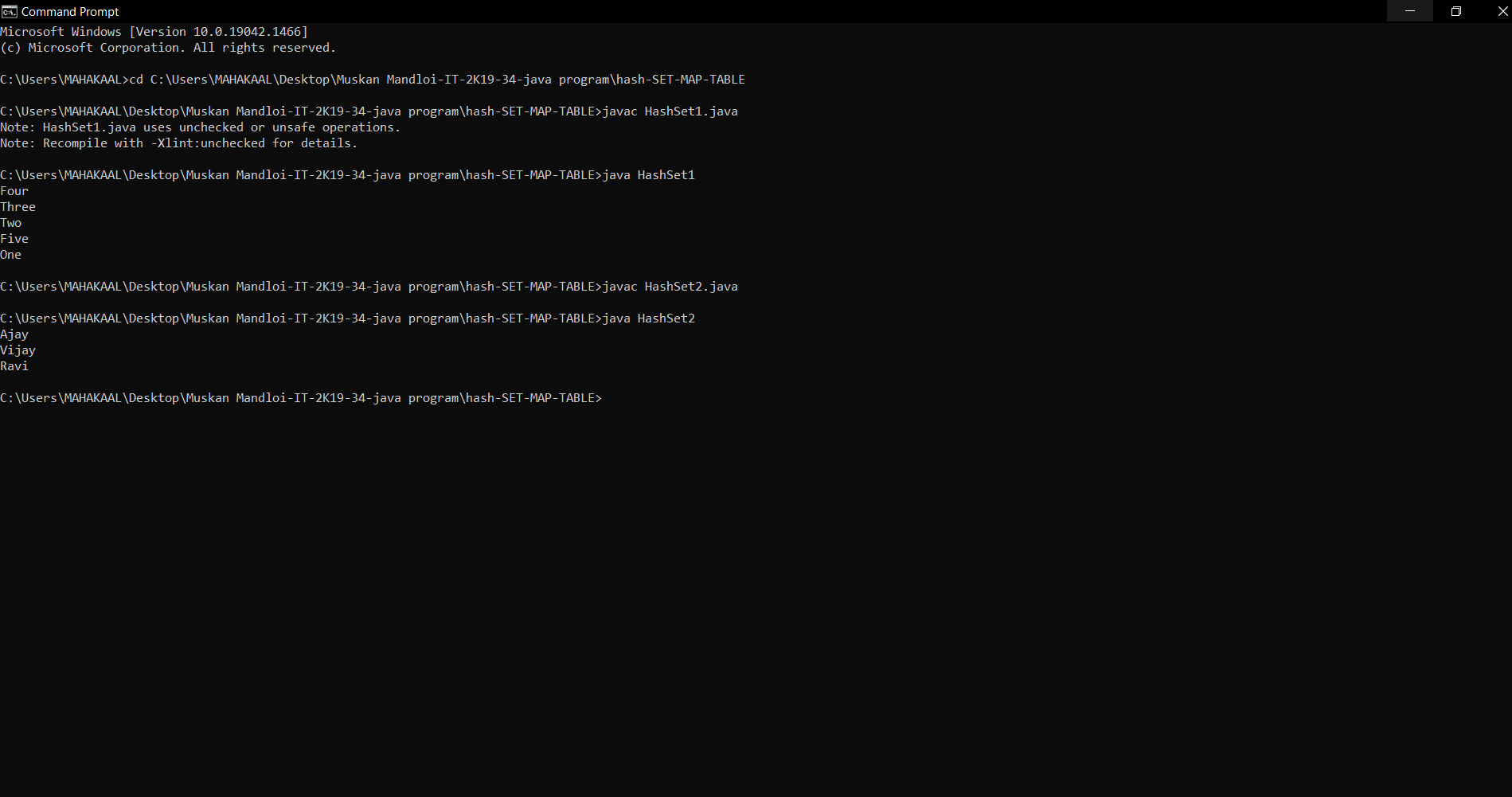
t.join();

} catch (InterruptedException ignored) {

}

}

}



65. import java.util.\*;

public class LinkedList1{

public static void main(String args[]){

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

al.add("Ravi");

al.add("Vijay");

al.add("Ravi");

al.add("Ajay");

Iterator<String> itr=al.iterator();

while(itr.hasNext()){

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

}

}

}

66. import java.util.\*;

public class LinkedList2{

public static void main(String args[]){

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

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

ll.add("Ravi");

ll.add("Vijay");

ll.add("Ajay");

System.out.println("After invoking add(E e) method: "+ll);

ll.add(1, "Gaurav");

System.out.println("After invoking add(int index, E element) method: "+ll);

LinkedList<String> ll2=new LinkedList<String>();

ll2.add("Sonoo");

ll2.add("Hanumat");

ll.addAll(ll2);

System.out.println("After invoking addAll(Collection<? extends E> c) method: "+ll);

LinkedList<String> ll3=new LinkedList<String>();

ll3.add("John");

ll3.add("Rahul");

ll.addAll(1, ll3);

System.out.println("After invoking addAll(int index, Collection<? extends E> c) method: "+ll);

ll.addFirst("Lokesh");

System.out.println("After invoking addFirst(E e) method: "+ll);

ll.addLast("Harsh");

System.out.println("After invoking addLast(E e) method: "+ll);

}

}

67. import java.util.\*;

class TestArrayLinked{

public static void main(String args[]){

List<String> al=new ArrayList<String>();//creating arraylist

al.add("Ravi");//adding object in arraylist

al.add("Vijay");

al.add("Ravi");

al.add("Ajay");

List<String> al2=new LinkedList<String>();//creating linkedlist

al2.add("James");//adding object in linkedlist

al2.add("Serena");

al2.add("Swati");

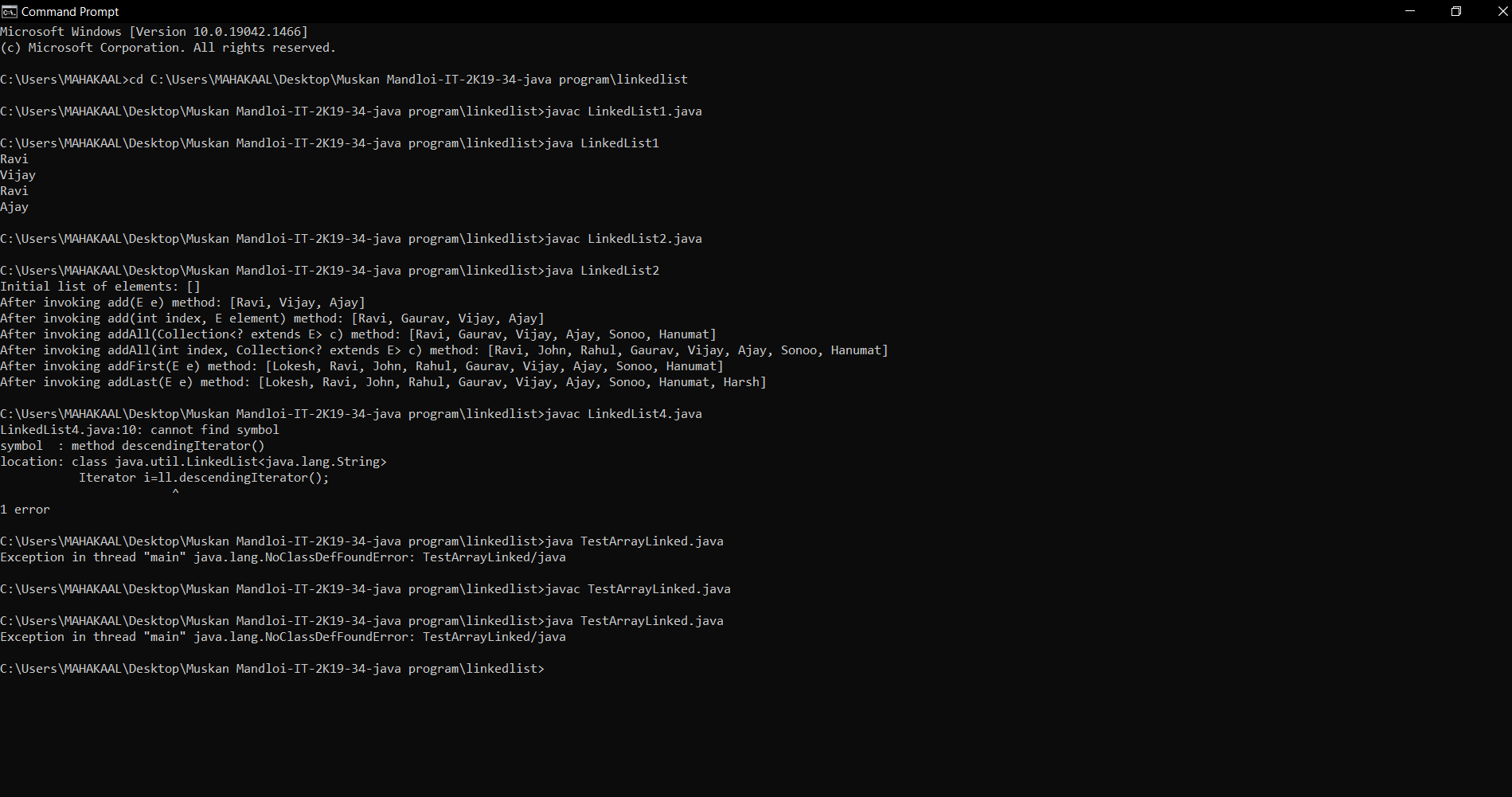
al2.add("Junaid");

System.out.println("arraylist: "+al);

System.out.println("linkedlist: "+al2);

}

}



**Abstraction collection**

68. import java.util.\*;

import java.util.AbstractCollection;

public class Abstract{

public static void main(String[] args)

{

AbstractCollection<Object>

abs = new ArrayList<Object>();

abs.add("Welcome");

abs.add("To");

abs.add("IIPS DAVV");

abs.add("IT-2K19-34");

abs.add("Muskan Mandloi");

// Displaying the Collection

System.out.println("AbstractCollection: "

+ abs);

}

}

69. import java.util.\*;

public class AddElements {

public static void main(String[] args)

{

AbstractMap<Integer, String> absMap

= new HashMap<Integer, String>();

absMap.put(1, "This");

absMap.put(2, "is");

absMap.put(3, "an");

absMap.put(4, "AbstractMap");

System.out.println("The Set view of the mappings:");

System.out.println(absMap.entrySet());

}

}

**Abstract Map**

70. import java.util.\*;

public class Traversing {

public static void main(String[] args)

{

AbstractMap<Integer, String> absMap

= new HashMap<Integer, String>();

absMap.put(1, "This");

absMap.put(2, "is");

absMap.put(3, "a");

absMap.put(4, "AbstractMap");

System.out.println("Using the entrySet() method");

for (AbstractMap.Entry<Integer, String> entry :

absMap.entrySet()) {

System.out.println("Key = " + entry.getKey()

+ ", Value = "

+ entry.getValue());

}

System.out.println(

"\nUsing the Iterator interface");

Iterator<Integer> itr = absMap.keySet().iterator();

while (itr.hasNext()) {

int key = itr.next();

System.out.println("Key = " + key+ ", Value = "+ absMap.get(key));

}

}

}

**Sorted Map**

71. import java.util.Iterator;

import java.util.Map;

import java.util.Set;

import java.util.SortedMap;

import java.util.TreeMap;

public class SortedMapExample {

public static void main(String[] args)

{

SortedMap<Integer, String> sm

= new TreeMap<Integer, String>();

sm.put(new Integer(2), "practice");

sm.put(new Integer(3), "quiz");

sm.put(new Integer(5), "code");

sm.put(new Integer(4), "contribute");

sm.put(new Integer(1), "testseries");

Set s = sm.entrySet();

Iterator i = s.iterator();.

while (i.hasNext()) {

Map.Entry m = (Map.Entry)i.next();

int key = (Integer)m.getKey();

String value = (String)m.getValue();

System.out.println("Key : " + key+ " value : " + value);

}

}

}

72. import java.io.\*;

import java.util.\*;

class Example3{

public static void main(String args[])

{

SortedMap<Integer, String> tm

= new TreeMap<Integer, String>();

tm.put(3, "Muskan ");

tm.put(2, "Mandloi");

tm.put(1, "IT-2K19-34");

System.out.println(tm);

tm.put(2, "For");

System.out.println(tm);

}

}

73. import java.io.\*;

import java.util.\*;

class Example2{

public static void main(String args[])

{

SortedMap tm1 = new TreeMap();

SortedMap<Integer, String> tm2

= new TreeMap<Integer, String>();

tm1.put(3, "IIPS ");

tm1.put(2, "DAVV");

tm1.put(1, "Indore");

tm2.put(new Integer(3), "Muskan");

tm2.put(new Integer(2), "Mandloi");

tm2.put(new Integer(1), "IT-2k19-34");

System.out.println(tm1);

System.out.println(tm2);

}

}

74. import java.io.\*;

import java.util.\*;

class Example3{

public static void main(String args[])

{

SortedMap<Integer, String> tm

= new TreeMap<Integer, String>();

tm.put(3, "Muskan ");

tm.put(2, "Mandloi");

tm.put(1, "IT-2K19-34");

System.out.println(tm);

tm.put(2, "For");

System.out.println(tm);

}

}

75. import java.io.\*;

import java.util.\*;

class Example4 {

public static void main(String args[])

{

SortedMap<Integer, String> tm

= new TreeMap<Integer, String>();

tm.put(3, "Java");

tm.put(2, "PHP");

tm.put(1, "ADA");

tm.put(4, "Web Tech");

System.out.println(tm);

tm.remove(4);

System.out.println(tm);

}

}

76. import java.util.\*;

class Example6 {

public static void main(String[] args)

{

SortedMap<String, String> tm

= new TreeMap<String, String>(new Comparator<String>() {

public int compare(String a, String b)

{

return b.compareTo(a);

}

});

tm.put("India", "1");

tm.put("Australia", "2");

tm.put("South Africa", "3");

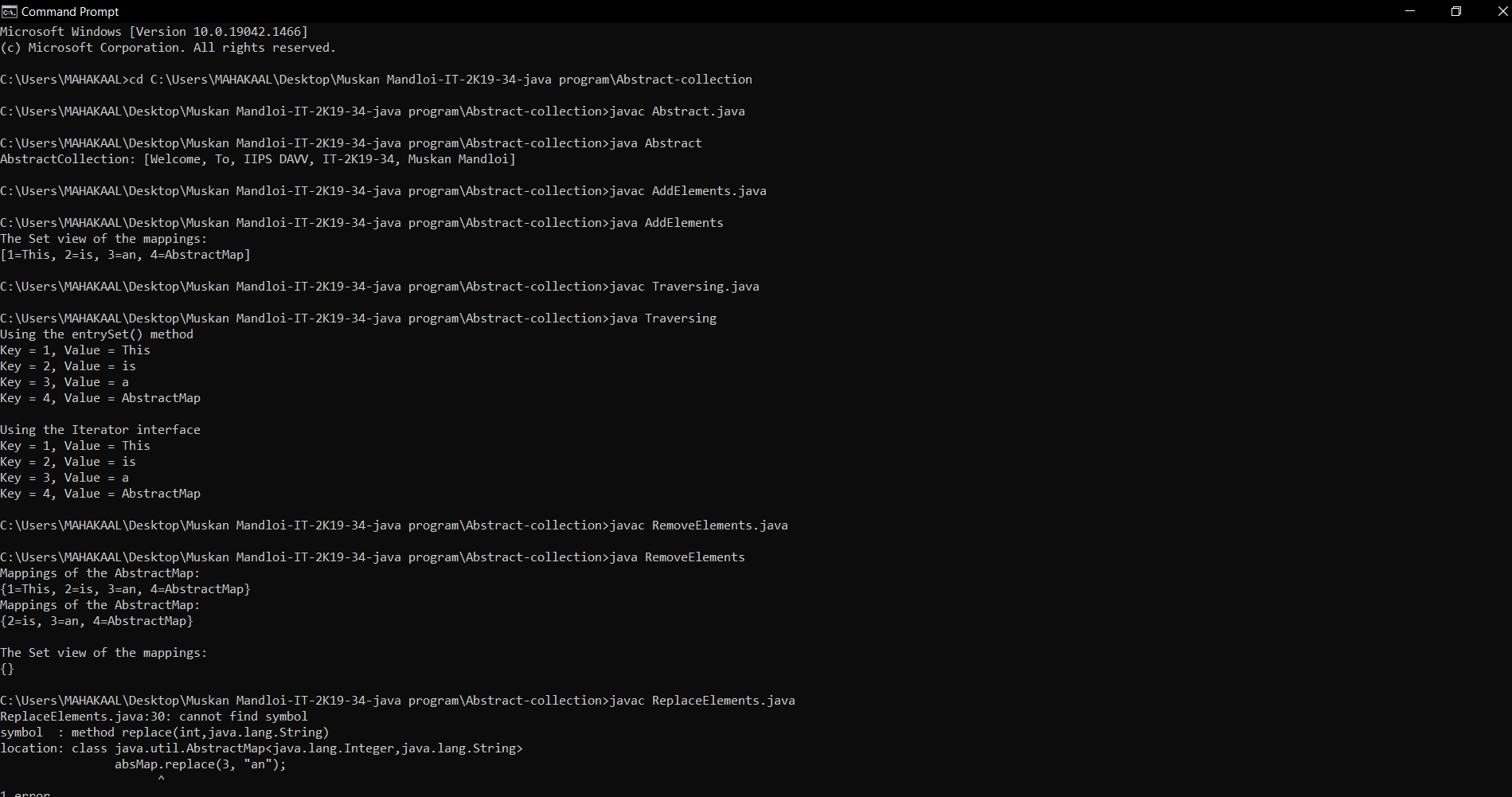
System.out.println(tm);

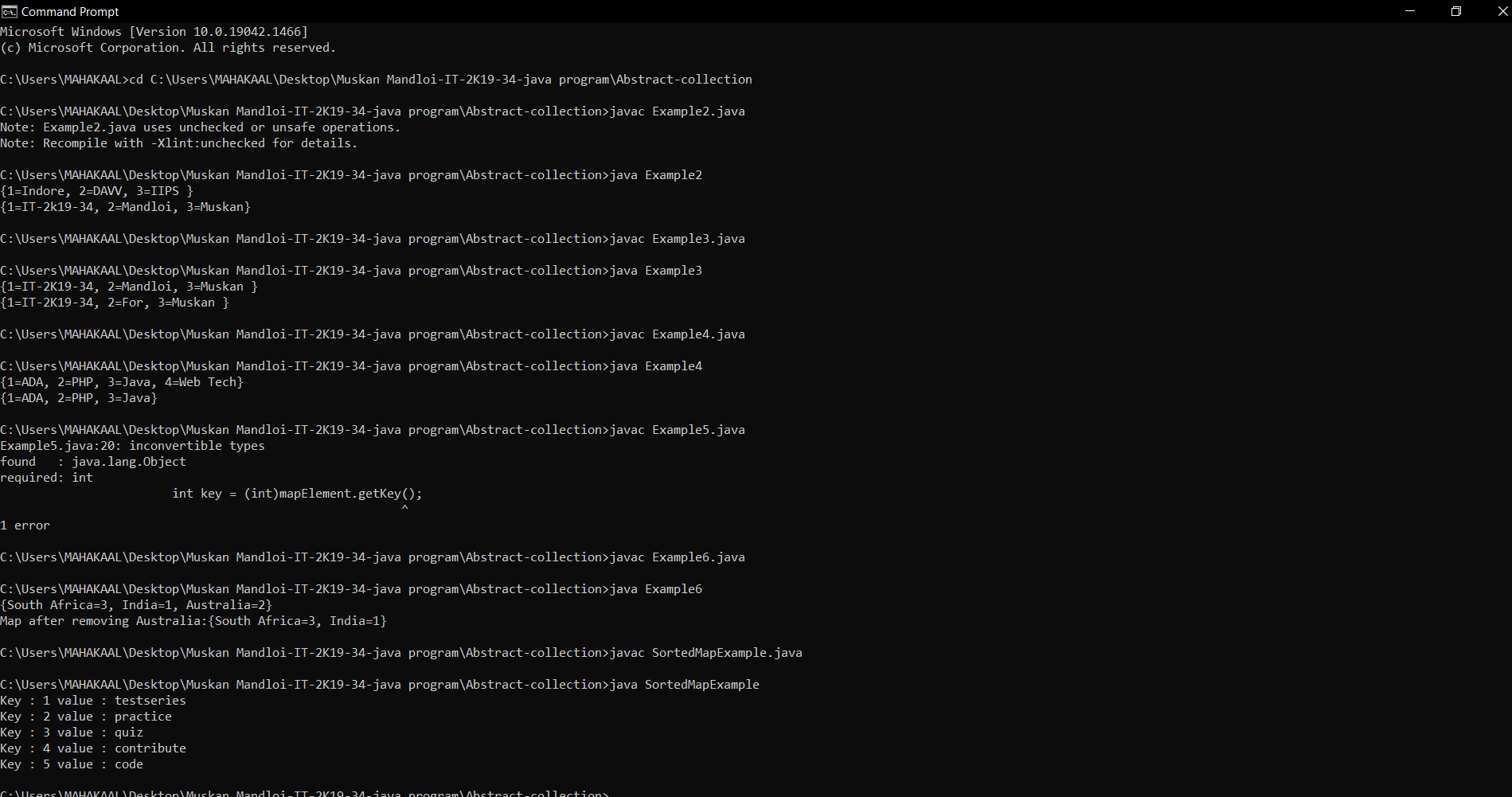
tm.remove("Australia");

System.out.println("Map after removing”+ "Australia:" + tm);

}

}





**Vector**

77. import java.util.\*;

public class VectorExample {

public static void main(String args[]) {

Vector<String> vec = new Vector<String>();

vec.add("Tiger");

vec.add("Lion");

vec.add("Dog");

vec.add("Elephant");

vec.addElement("Rat");

vec.addElement("Cat");

vec.addElement("Deer");

System.out.println("Elements are: "+vec);

}

}

78. import java.util.\*;

public class VectorExample1 {

public static void main(String args[]) {

Vector<String> vec = new Vector<String>(4);

vec.add("Tiger");

vec.add("Lion");

vec.add("Dog");

vec.add("Elephant");

System.out.println("Size is: "+vec.size());

System.out.println("Default capacity is: "+vec.capacity());

System.out.println("Vector element is: "+vec);

vec.addElement("Rat");

vec.addElement("Cat");

vec.addElement("Deer");

System.out.println("Size after addition: "+vec.size());

System.out.println("Capacity after addition is: "+vec.capacity());

System.out.println("Elements are: "+vec);

if(vec.contains("Tiger"))

{

System.out.println("Tiger is present at the index " +vec.indexOf("Tiger"));

}

else

{

System.out.println("Tiger is not present in the list.");

}

System.out.println("The first animal of the vector is = "+vec.firstElement());

System.out.println("The last animal of the vector is = "+vec.lastElement());

}

}

79. import java.util.\*;

public class VectorExample2 {

public static void main(String args[]) {

Vector<Integer> in = new Vector<Integer>();

in.add(100);

in.add(200);

in.add(300);

in.add(200);

in.add(400);

in.add(500);

in.add(600);

in.add(700);

System.out.println("Values in vector: " +in);

System.out.println("Remove first occourence of element 200: "+in.remove((Integer)200));

System.out.println("Values in vector: " +in);

System.out.println("Remove element at index 4: " +in.remove(4));

System.out.println("New Value list in vector: " +in);

in.removeElementAt(5);

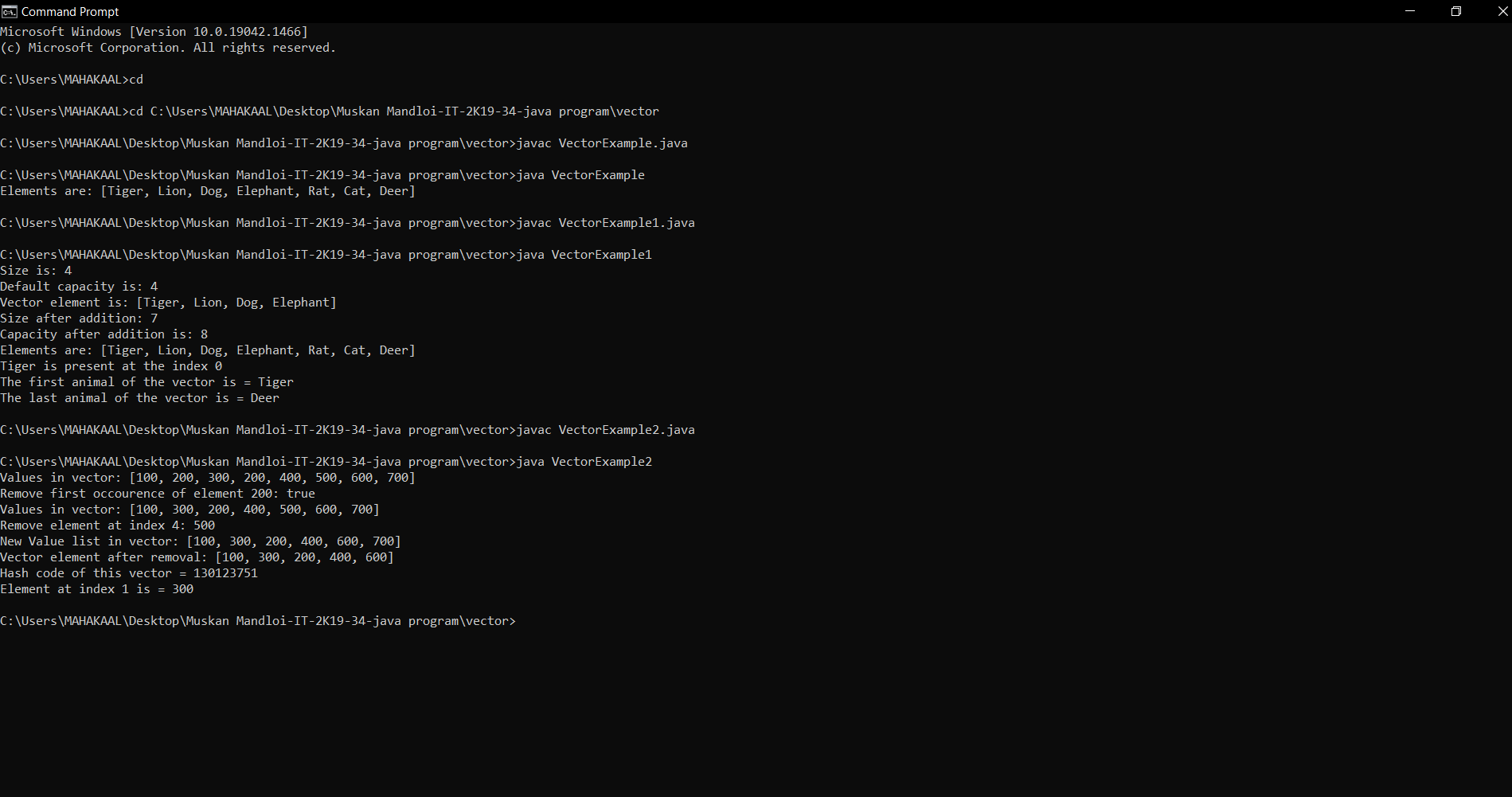
System.out.println("Vector element after removal: " +in);

System.out.println("Hash code of this vector = "+in.hashCode());

System.out.println("Element at index 1 is = "+in.get(1));

}

}

****

TREE MAP

80. import java.util.\*;

class TreeMap1{

public static void main(String args[]){

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

map.put(100,"Amit");

map.put(102,"Ravi");

map.put(101,"Vijay");

map.put(103,"Rahul");

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

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

}

}

}

81. package set;

import java.util.\*;

public class settest

{

public static void main(String[] args)

{

Set<String> words = new HashSet<String>(); // HashSet implements Set

long totalTime = 0;

Scanner in = new Scanner(System.in);

while (in.hasNext())

{

String word = in.next();

long callTime = System.currentTimeMillis();

words.add(word);

callTime = System.currentTimeMillis() - callTime;

totalTime += callTime;

}

Iterator<String> iter = words.iterator();

for (int i = 1; i <= 20 && iter.hasNext(); i++)

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

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

System.out.println(words.size() + " distinct words. " + totalTime

+ " milliseconds.");

}

}

82. import java.util.\*;

public class TreeMap2 {

public static void main(String args[]) {

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

map.put(100,"Amit");

map.put(102,"Ravi");

map.put(101,"Vijay");

map.put(103,"Rahul");

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

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

{

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

}

map.remove(102);

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

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

{

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

}

}

}

83. import java.util.\*;

class TreeMap4{

public static void main(String args[]){

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

map.put(100,"Amit");

map.put(102,"Ravi");

map.put(101,"Vijay");

map.put(103,"Rahul");

System.out.println("headMap: "+map.headMap(102));

System.out.println("tailMap: "+map.tailMap(102));

System.out.println("subMap: "+map.subMap(100, 102));

}

}

Tree Set

84. package treeSet;

import java.util.\*;

public class TreeSetTest

{

public static void main(String[] args)

{

SortedSet<Item> parts = new TreeSet<Item>();

parts.add(new Item("Toaster", 1234));

parts.add(new Item("Widget", 4562));

parts.add(new Item("Modem", 9912));

System.out.println(parts);

SortedSet<Item> sortByDescription = new TreeSet<Item>(new

Comparator<Item>()

{

public int compare(Item a, Item b)

{

String descrA = a.getDescription();

String descrB = b.getDescription();

return descrA.compareTo(descrB);

}

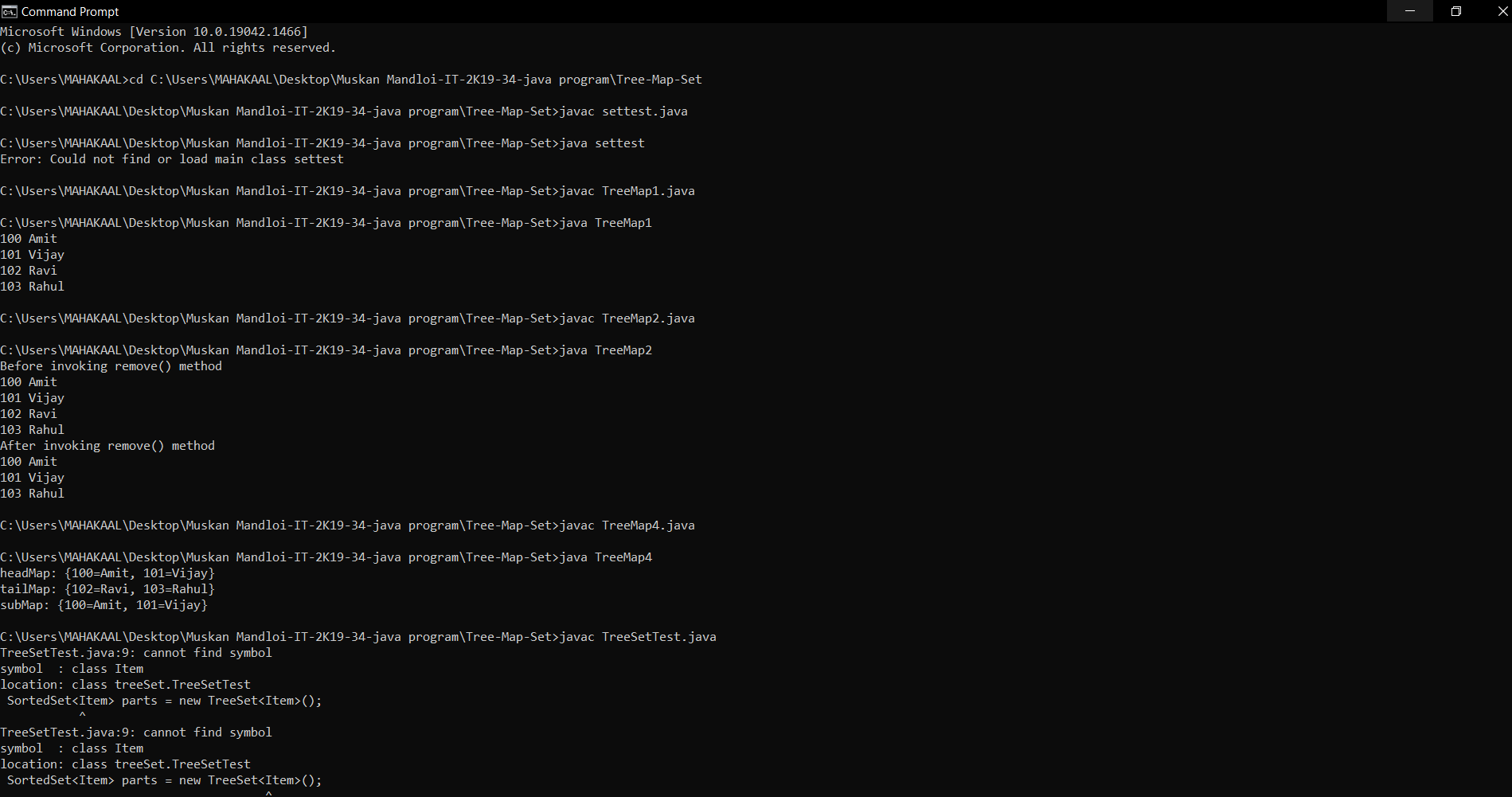
});

sortByDescription.addAll(parts);

System.out.println(sortByDescription);

}

}



**Hash Table**

84. import java.util.\*;

class HashTableExample{

public static void main(String args[]){

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

hm.put(100,"Amit");

hm.put(102,"Ravi");

hm.put(101,"Vijay");

hm.put(103,"Rahul");

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

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

}

}

}

86. import java.util.\*;

public class Hashtable2 {

public static void main(String args[]) {

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

map.put(100,"Amit");

map.put(102,"Ravi");

map.put(101,"Vijay");

map.put(103,"Rahul");

System.out.println("Before remove: "+ map);

// Remove value for key 102

map.remove(102);

System.out.println("After remove: "+ map);

}

}

87. import java.util.\*;

class Hashtable3{

public static void main(String args[]){

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

map.put(100,"Amit");

map.put(102,"Ravi");

map.put(101,"Vijay");

map.put(103,"Rahul");

System.out.println(map.getOrDefault(101, "Not Found"));

System.out.println(map.getOrDefault(105, "Not Found"));

}

}

88. import java.util.\*;

class Hashtable4{

public static void main(String args[]){

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

map.put(100,"Amit");

map.put(102,"Ravi");

map.put(101,"Vijay");

map.put(103,"Rahul");

System.out.println("Initial Map: "+map);

map.putIfAbsent(104,"Gaurav");

System.out.println("Updated Map: "+map);

map.putIfAbsent(101,"Vijay");

System.out.println("Updated Map: "+map);

}

}

**Hash Set**

89. import java.util.\*;

class HashSet1{

public static void main(String args[]){

HashSet<String> set=new HashSet();

set.add("One");

set.add("Two");

set.add("Three");

set.add("Four");

set.add("Five");

Iterator<String> i=set.iterator();

while(i.hasNext())

{

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

}

}

}

90. import java.util.\*;

class HashSet2{

public static void main(String args[]){

HashSet<String> set=new HashSet<String>();

set.add("Ravi");

set.add("Vijay");

set.add("Ravi");

set.add("Ajay");

Iterator<String> itr=set.iterator();

while(itr.hasNext()){

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

}

}

}

91. import java.util.\*;

class HashSet3{

public static void main(String args[]){

HashSet<String> set=new HashSet<String>();

set.add("Ravi");

set.add("Vijay");

set.add("Arun");

set.add("Sumit");

System.out.println("An initial list of elements: "+set);

set.remove("Ravi");

System.out.println("After invoking remove(object) method: "+set);

HashSet<String> set1=new HashSet<String>();

set1.add("Ajay");

set1.add("Gaurav");

set.addAll(set1);

System.out.println("Updated List: "+set);

set.removeAll(set1);

System.out.println("After invoking removeAll() method: "+set);

set.removeIf(str->str.contains("Vijay"));

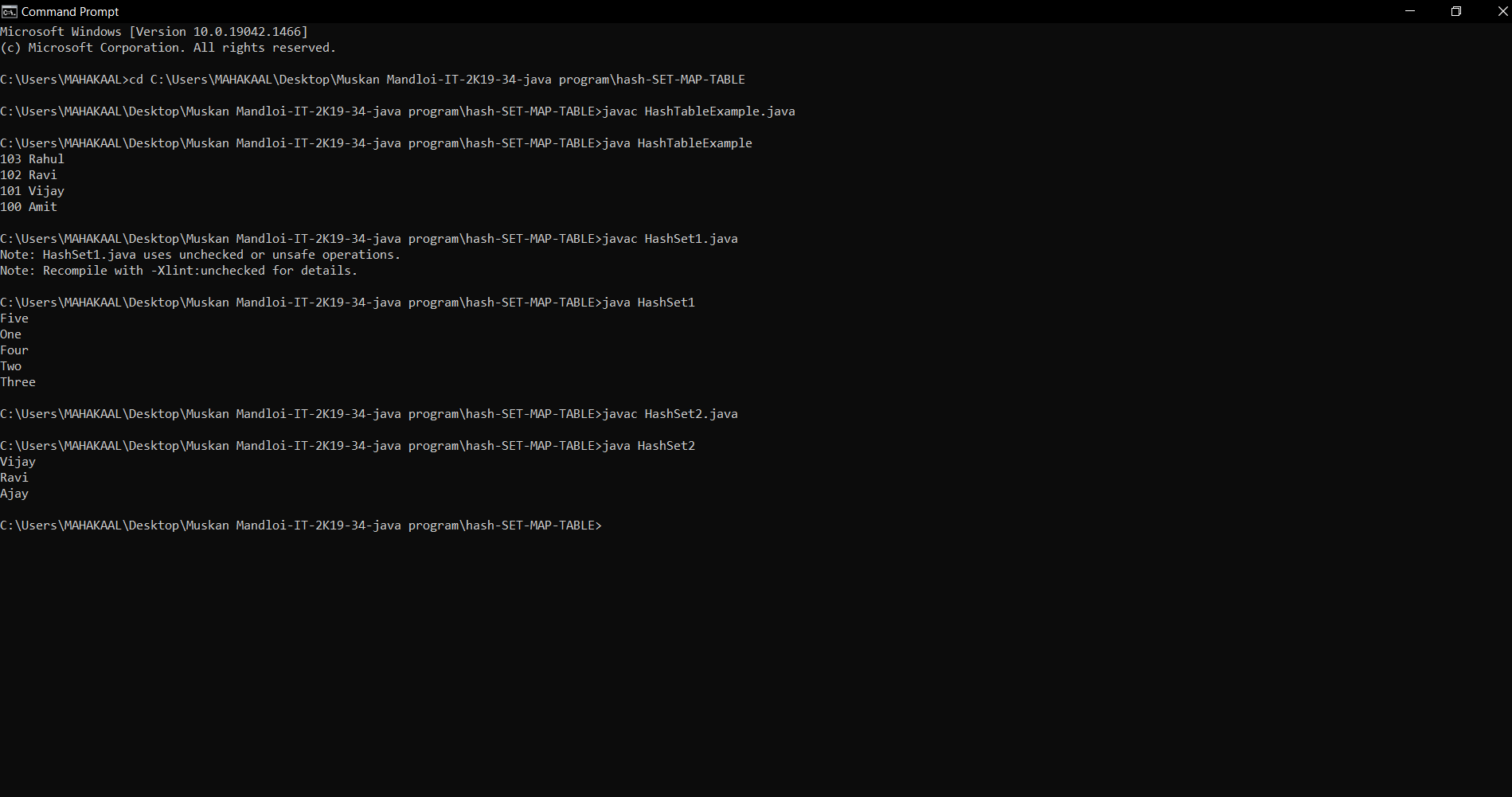
System.out.println("After invoking removeIf() method: "+set);

set.clear();

System.out.println("After invoking clear() method: "+set);

}

}



List

91. import java.util.\*;

public class indexoflinklist {

public static void main(String[] args) {

LinkedList<Integer> list = new LinkedList<Integer>();

list.add(80);

list.add(22);

list.add(36);

list.add(34);

list.add(92);

list.add(03);

System.out.println("Elements in the list are :");

for (Integer integer : list) {

System.out.println(integer);

}

System.out.println("76 is at "+list.indexOf(76)+" index");

System.out.println("Element at 3 positon is "+list.get(3));

System.out.println("Last index of 12 is "+list.lastIndexOf(12));

}

}

92. import java.util.LinkedList;

public class Linklistmethods {

public static void main(String[] args) {

LinkedList<Integer> list = new LinkedList<Integer>();

list.add(90);

list.add(12);

list.add(76);

list.add(34);

list.add(12);

list.add(23);

System.out.println("Elements in the list are :");

for (Integer integer : list) {

System.out.println(integer);

}

System.out.println("Element at index 3 removed form the list :");

list.remove(3);

System.out.println("Element at index 2 is set to 98");

list.set(2, 98);

for (Integer integer : list) {

System.out.println(integer);

}

System.out.println("First element of the list is :"+list.getFirst());

System.out.println("Last element of the list is :"+list.getLast());

}

}

93. import java.util.\*;

public class listcreate{

public static void main(String[] args) {

LinkedList<Integer> list = new LinkedList<Integer>();

list.add(90);

list.add(98);

list.add(76);

list.add(34);

list.add(12);

System.out.println("Elements in the list are :");

for (Integer integer : list) {

System.out.println(integer);

}

Scanner scn = new Scanner(System.in);

System.out.println("Where do you want to add an element and at which position:");

int i = scn.nextInt();

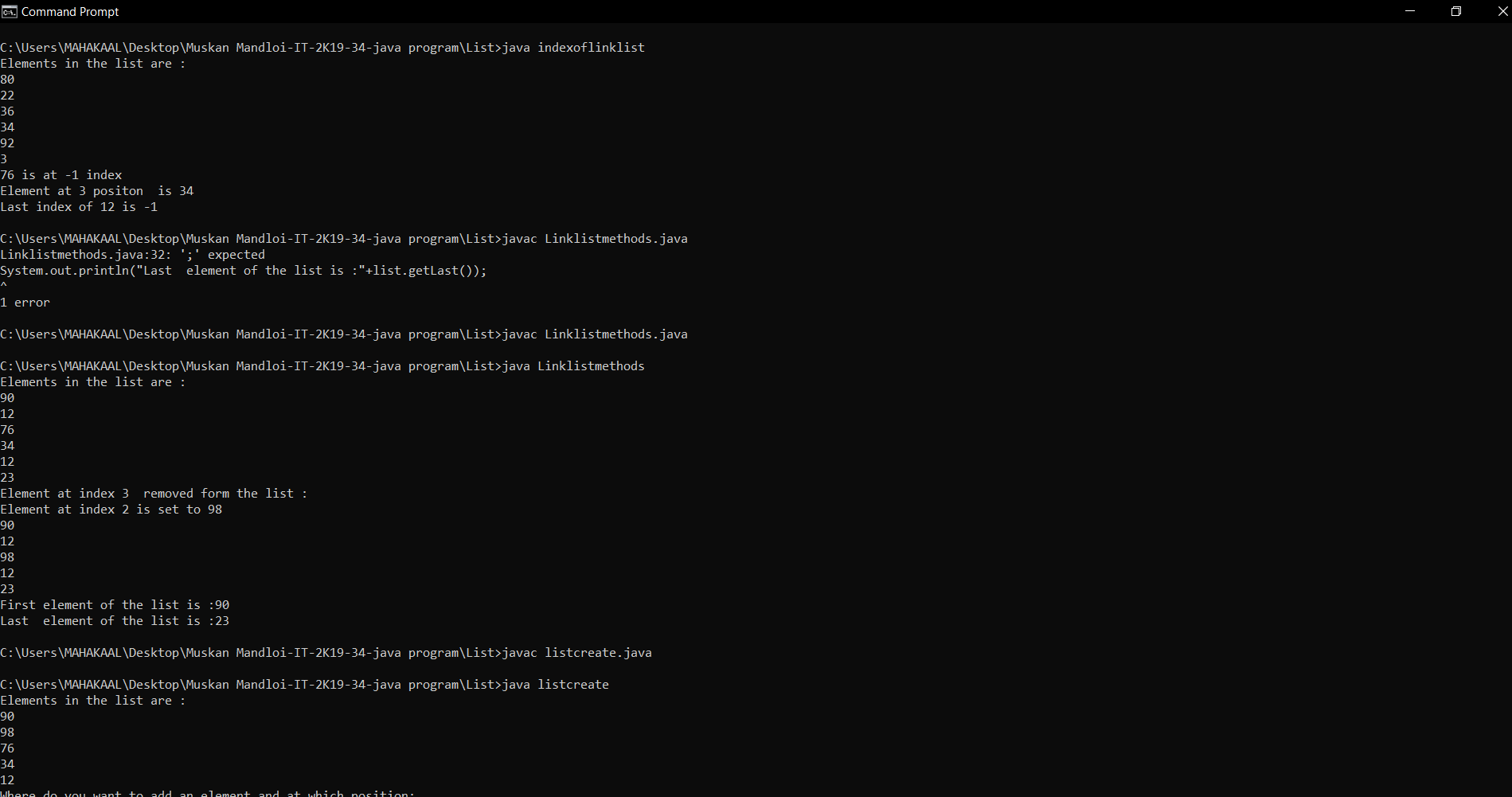
int e = scn.nextInt();

list.add(i, e);

for (Integer integer : list) {

System.out.println(integer);

}



MapKeySearch

94. import java.util.\*;

public class MapkeySearch {

public static void main(String[] args) {

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

m.put(1, "Tea");

m.put(2, "coffe");

m.put(3, "juice");

m.put(4, "soup");

System.out.println("Map is:"+m);

boolean flag = m.containsKey(1);

System.out.println("Key 1 exists in Map? : " + flag);

boolean flag2 = m.containsKey(55);

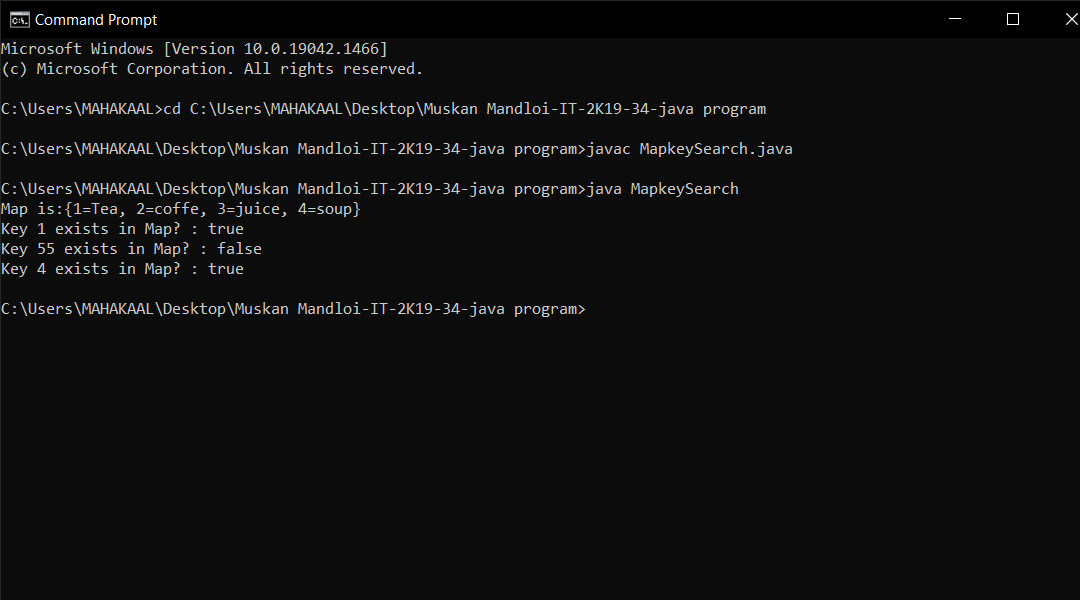
System.out.println("Key 55 exists in Map? : " + flag2);

boolean flag3 = m.containsKey(4);

System.out.println("Key 4 exists in Map? : " + flag3);

}

}



**EnumSet**

95. import java.util.EnumSet;

import java.util.Iterator;

public class acessenumelements {

enum info {

IIPS,DAVV, Java\_Assignment ,Muskan ,Mandloi

}

public static void main(String[] args) {

EnumSet<info> i = EnumSet.allOf(info.class);

Iterator<info> iterate = i.iterator();

System.out.print("EnumSet: ");

while(iterate.hasNext()) {

System.out.print(iterate.next());

System.out.print(", ");

}

}

}

96. import java.util.\*;

enum games{chess,ludo,snakeladder,carrace};

public class Enumsetcreate{

public static void main(String[] args) {

EnumSet<games> eset1 , eset2,eset3;

eset1 = EnumSet.of(games.chess,games.ludo);

eset2 = EnumSet.of(games.ludo);

eset3 = EnumSet.of(games.carrace,games.snakeladder);

System.out.println("Set 1: " + eset1);

System.out.println("Set 2: " + eset2);

System.out.println("Set 3: " + eset3);

}

}

97. import java.util.EnumSet;

public class Enumsetsize{

enum Size {

SMALL, MEDIUM, LARGE, EXTRALARGE

}

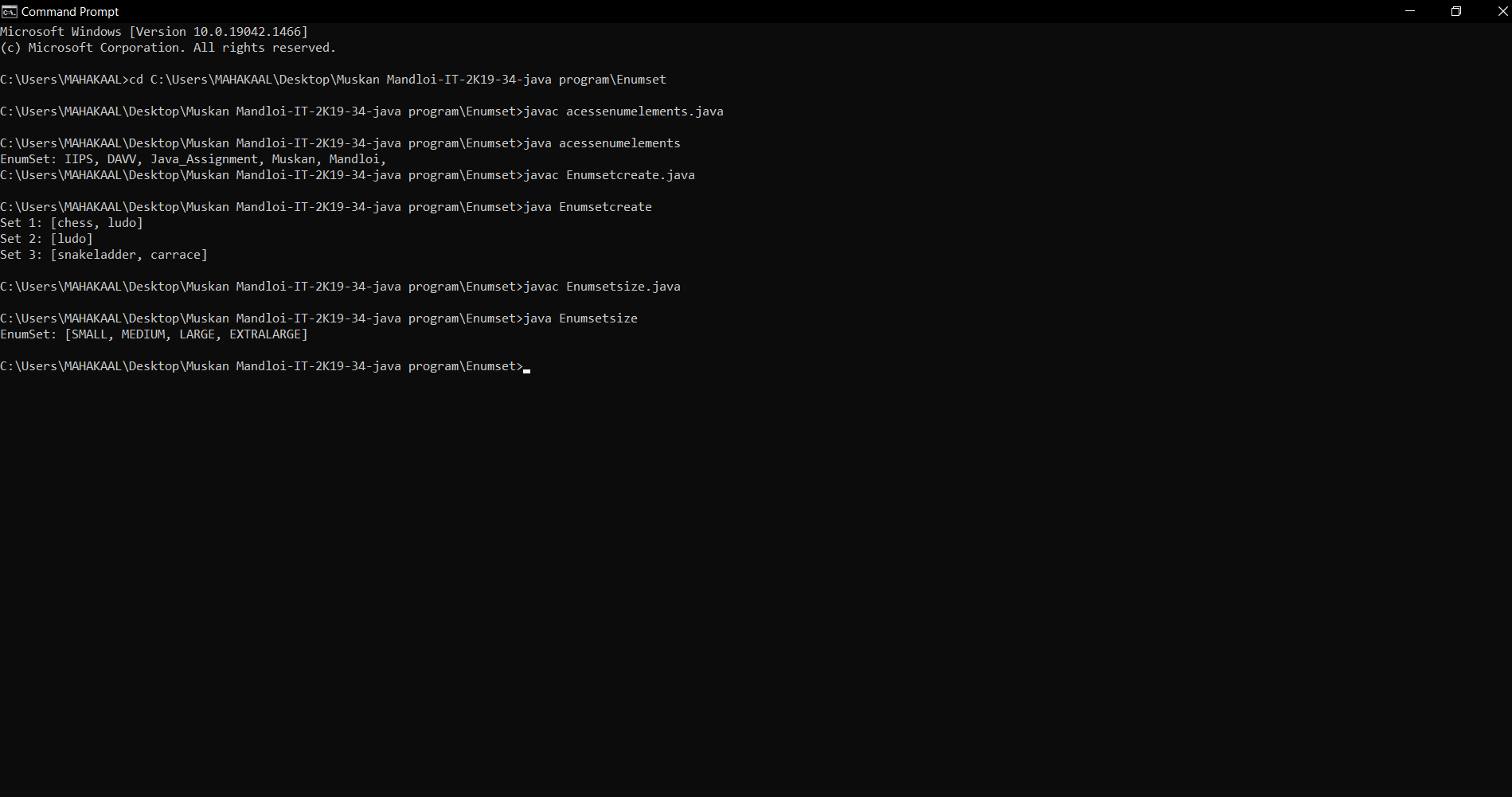
public static void main(String[] args) {

EnumSet<Size> sizes = EnumSet.allOf(Size.class);

System.out.println("EnumSet: " + sizes);

}

}



98. import java.util.\*;

public class Deque {

public static void main(String[] args)

{

Deque<String> deque

= new LinkedList<String>();

deque.add("Element 1 (Tail)");

deque.addFirst("Element 2 (Head)");

deque.addLast("Element 3 (Tail)");

deque.push("Element 4 (Head)");

deque.offer("Element 5 (Tail)");

deque.offerFirst("Element 6 (Head)");

System.out.println(deque + "\n");

deque.removeFirst();

deque.removeLast();

System.out.println("Deque after removing "

+ "first and last: "

+ deque);

}

}

99. import java.util.\*;

public class DequeExample {

public static void main(String[] args) {

Deque<String> deque=new ArrayDeque<String>();

deque.offer("arvind");

deque.offer("vimal");

deque.add("mukul");

deque.offerFirst("jai");

System.out.println("After offerFirst Traversal...");

for(String s:deque){

System.out.println(s);

}

deque.pollLast();

System.out.println("After pollLast() Traversal...");

for(String s:deque){

System.out.println(s);

}

}

}

100. import java.util.ArrayDeque;

class Example {

public static void main(String[] args) {

ArrayDeque<String> animals= new ArrayDeque<String>();

animals.add("Dog");

animals.addFirst("Cat");

animals.addLast("Horse");

System.out.println("ArrayDeque: " + animals);

}

}

101. import java.util.\*;

public class iteration {

public static void main(String[] args)

{

Deque<String> dq

= new ArrayDeque<String>();

dq.add("For");

dq.addFirst("IT-2K19-34");

dq.addLast("MUSKAN");

dq.add("MANDLOI");

for (Iterator itr = dq.iterator();

itr.hasNext();) {

System.out.print(itr.next() + " ");

}

System.out.println();

for (Iterator itr = dq.descendingIterator();

itr.hasNext();) {

System.out.print(itr.next() + " ");

}

}

}

**Graphs**

102**.** import java.util.\*;

class Graphcreate{

static void addEdge(ArrayList<ArrayList<Integer> > adj,

int u, int v)

{

adj.get(u).add(v);

adj.get(v).add(u);

}

static void

printGraph(ArrayList<ArrayList<Integer> > adj)

{

for (int i = 0; i < adj.size(); i++) {

System.out.println("\nAdjacency list of vertex"

+ i);

System.out.print("head");

for (int j = 0; j < adj.get(i).size(); j++) {

System.out.print(" -> "

+ adj.get(i).get(j));

}

System.out.println();

}

}

public static void main(String[] args)

{

int V = 5;

ArrayList<ArrayList<Integer> > adj

= new ArrayList<ArrayList<Integer> >(V);

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

adj.add(new ArrayList<Integer>());

addEdge(adj, 0, 1);

addEdge(adj, 0, 4);

addEdge(adj, 1, 2);

addEdge(adj, 1, 3);

addEdge(adj, 1, 4);

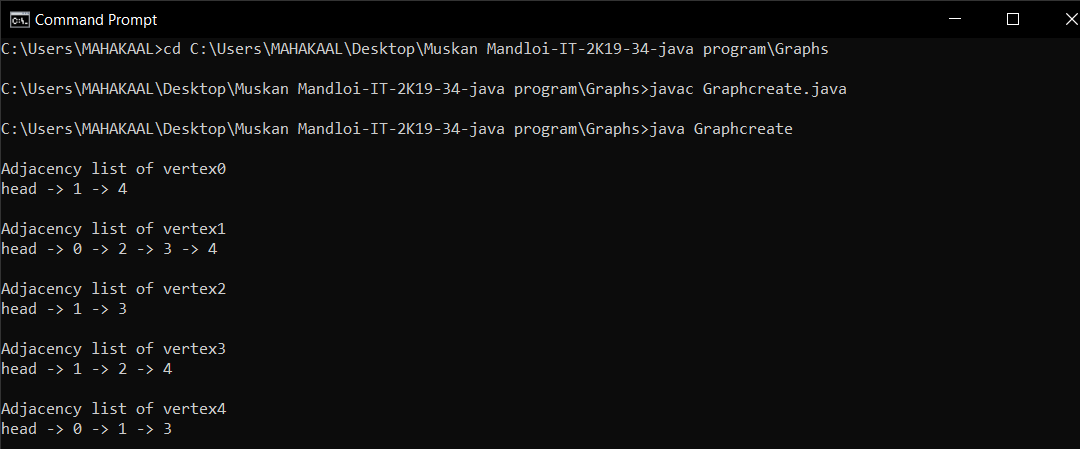
addEdge(adj, 2, 3);

addEdge(adj, 3, 4);

printGraph(adj);

}

}

****

**ARRAYDEQUE**

103. import java.util.\*;

import java.util.concurrent.\*;

public class Acessingelements {

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

{

// Create an instance of ConcurrentLinkedQueue

ConcurrentLinkedQueue<Integer> Q = new ConcurrentLinkedQueue<Integer>();

// Add numbers to end of Queue

Q.add(7855642);

Q.add(35658786);

Q.add(5278367);

Q.add(74381793);

System.out.println("Queue: " + Q);

System.out.println("Queue's head: " + Q.element());

System.out.println("Queue's head: " + Q.peek());

}

}

104. import java.util.concurrent.ArrayBlockingQueue;

import java.util.\*;

public class arrayblockingdequecreate {

public static void main(String[] args)

{

int capacity = 5;

ArrayBlockingQueue<String> queue = new ArrayBlockingQueue<String>(capacity);

queue.offer("MUSKAN MANDLOI");

queue.offer("IIPS DAVV");

queue.offer("IT-2K19-34");

queue.offer("JAVA");

queue.offer("ADVANCE");

System.out.println("Queue is " + queue);

Iterator iteratorValues = queue.iterator();

System.out.println("\nThe values:");

while (iteratorValues.hasNext()) {

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

}

}

}

105. import java.util.concurrent.\*;

import java.util.\*;

public class concurrentlinkedqueuedemo {

public static void main(String[] args)

{

ConcurrentLinkedQueue<String> queue = new ConcurrentLinkedQueue<String>();

queue.add("Kolkata");

queue.add("Patna");

queue.add("Delhi");

queue.add("Jammu");

System.out.println("ConcurrentLinkedQueue: " + queue);

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

arraylist.add("Sanjeet");

arraylist.add("Rabi");

arraylist.add("Debasis");

arraylist.add("Raunak");

arraylist.add("Mahesh");

System.out.println("Collection to be added: " + arraylist);

boolean response = queue.addAll(arraylist);

System.out.println("Collection added: " + response);

System.out.println("ConcurrentLinkedQueue: " + queue);

}

}

106. import java.util.concurrent.\*;

import java.util.\*;

public class iterateconcurentdeque {

public static void main(String[] args)

{

ConcurrentLinkedQueue<String> queue = new ConcurrentLinkedQueue<String>();

queue.add("Muskan");

queue.add("Mandloi");

queue.add("IT-2K19-34");

queue.add("JAVA\_ASSIGNMENT");

System.out.println("ConcurrentLinkedQueue : " + queue);

Iterator iterator = queue.iterator();

System.out.println("\nThe String Values of iterator are:");

while (iterator.hasNext()) {

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

}

}

}

107. import java.util.concurrent.LinkedBlockingQueue;

public class linkedblocking {

public static void main(String[] args)

{

int capacity = 15;

LinkedBlockingQueue<Integer>lbq = new LinkedBlockingQueue<Integer>(capacity);

lbq.add(1);

lbq.add(2);

lbq.add(3);

System.out.println("LinkedBlockingQueue:" + lbq);

}

}

108. import java.util.concurrent.PriorityBlockingQueue;

public class priorityblocking {

public static void main(String[] args)

{

int capacityOfQueue = 5;

PriorityBlockingQueue<Integer>PrioQueue = newPriorityBlockingQueue<Integer>(capacityOfQueue);

PrioQueue.add(464161);

PrioQueue.add(416165);

System.out.println("PrioQueue: " + PrioQueue);

int head = PrioQueue.peek();

System.out.println("Head of Queue: " + head);

}

}

109. Import java.util.concurrent.PriorityBlockingQueue;

public class priorityblockingqueuecreate {

public static void main(String[] args)

{

int capacity = 15;

PriorityBlockingQueue<Integer>pbq

= new PriorityBlockingQueue<Integer>(capacity);

pbq.add(1);

pbq.add(2);

pbq.add(3);

System.out.println("PriorityBlockingQueue:" + pbq);

}

}

110. import java.util.concurrent.LinkedBlockingQueue;

public class Removinglinkedblocking {

public static void main(String[] args)

{

int capacity = 15;

LinkedBlockingQueue<Integer>lbq

= new LinkedBlockingQueue<Integer>(capacity);

lbq.add(1);

lbq.add(2);

lbq.add(3);

System.out.println("LinkedBlockingQueue:" + lbq);

lbq.clear();

System.out.println("LinkedBlockingQueue:" + lbq);

}

}

**NavigableMap**

111. import java.util.NavigableMap;

import java.util.TreeMap;

public class Navigablemapdemo {

public static void main(String[] args)

{

NavigableMap<String, Integer> nm

= new TreeMap<String, Integer>();

nm.put("C", 888);

nm.put("Y", 999);

nm.put("A", 444);

nm.put("T", 555);

nm.put("B", 666);

nm.put("A", 555);

System.out.println("Mappings of NavigableMap : "

+ nm);

System.out.printf("Descending Set : %s%n",

nm.descendingKeySet());

System.out.printf("Floor Entry : %s%n",

nm.floorEntry("L"));

System.out.printf("First Entry : %s%n",

nm.firstEntry());

System.out.printf("Last Key : %s%n", nm.lastKey());

System.out.printf("First Key : %s%n",

nm.firstKey());

System.out.printf("Original Map : %s%n", nm);

System.out.printf("Reverse Map : %s%n",

nm.descendingMap());

}

}

112. import java.util.\*;

class navigablemapmethods {

public static void main(String args[])

{

NavigableMap<Integer, String> nmap

= new TreeMap<Integer, String>();

nmap.put(1, "MUSKAN MANDLOI");

nmap.put(3, "IT-2K19-34");

nmap.put(2, "JAVA\_ASSIGNMENT");

System.out.println("Mappings of NavigableMap : "

+ nmap);

nmap.remove(4);

System.out.println(

"\nNavigableMap, after remove operation : "

+ nmap);

nmap.clear();

System.out.println(

"\nNavigableMap, after clear operation : "

+ nmap);

}

}

113. import java.util.\*;

class LinkedHashSet1{

public static void main(String args[]){

LinkedHashSet<String> set=new LinkedHashSet();

set.add("One");

set.add("Two");

set.add("Three");

set.add("Four");

set.add("Five");

Iterator<String> i=set.iterator();

while(i.hasNext())

{

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

}

}

}

114. import java.util.\*;

class LinkedHashSet2{

public static void main(String args[]){

LinkedHashSet<String> al=new LinkedHashSet<String>();

al.add("Ravi");

al.add("Vijay");

al.add("Ravi");

al.add("Ajay");

Iterator<String> itr=al.iterator();

while(itr.hasNext()){

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

}

}

}

115. import java.util.Stack;

public class StackEmptyMethodExample

{

public static void main(String[] args)

{

Stack<Integer> stk= new Stack<Integer>();

boolean result = stk.empty();

System.out.println("Is the stack empty? " + result);

stk.push(78);

stk.push(113);

stk.push(90);

stk.push(120);

System.out.println("Elements in Stack: " + stk);

result = stk.empty();

System.out.println("Is the stack empty? " + result);

}

}

116. import java.util.Iterator;

import java.util.ListIterator;

import java.util.Stack;

public class StackIterationExample3

{

public static void main (String[] args)

{

Stack <Integer> stk = new Stack<Integer>();

stk.push(119);

stk.push(203);

stk.push(988);

ListIterator<Integer> ListIterator = stk.listIterator(stk.size());

System.out.println("Iteration over the Stack from top to bottom:");

while (ListIterator.hasPrevious())

{

Integer avg = ListIterator.previous();

System.out.println(avg);

}

}

}

117. import java.util.\*;

public class MapkeySearch {

public static void main(String[] args) {

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

m.put(1, "Tea");

m.put(2, "coffe");

m.put(3, "juice");

m.put(4, "soup");

System.out.println("Map is:"+m);

boolean flag = m.containsKey(1);

System.out.println("Key 1 exists in Map? : " + flag);

boolean flag2 = m.containsKey(55);

System.out.println("Key 55 exists in Map? : " + flag2);

boolean flag3 = m.containsKey(4);

System.out.println("Key 4 exists in Map? : " + flag3);

}

}

118. import java.util.\*;

class TreeMap3{

public static void main(String args[]){

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

map.put(100,"Amit");

map.put(102,"Ravi");

map.put(101,"Vijay");

map.put(103,"Rahul");

System.out.println("descendingMap: "+map.descendingMap());

System.out.println("headMap: "+map.headMap(102,true));

System.out.println("tailMap: "+map.tailMap(102,true));

System.out.println("subMap: "+map.subMap(100, false, 102, true));

}

}

119. package treeSet;

import java.util.\*;

public class TreeSetTest

{

public static void main(String[] args)

{

SortedSet<Item> parts = new TreeSet<Item>();

parts.add(new Item("Toaster", 1234));

parts.add(new Item("Widget", 4562));

parts.add(new Item("Modem", 9912));

System.out.println(parts);

SortedSet<Item> sortByDescription = new TreeSet<Item>(new

Comparator<Item>()

{

public int compare(Item a, Item b)

{

String descrA = a.getDescription();

String descrB = b.getDescription();

return descrA.compareTo(descrB);

}

});

sortByDescription.addAll(parts);

System.out.println(sortByDescription);

}

}

120 . import java.util.\*;

class TreeSet {

public static void main(String[] args)

{

Set<String> ts = new TreeSet<>();

ts.add("MUSKAN MANDLOI");

ts.add("IT-2K19-34");

ts.add("ADV JAVA");

ts.add("A");

ts.add("B");

ts.add("Z");

for (String value : ts)

System.out.print(value + ", ");

System.out.println();

}

}

121. import java.io.\*;

import java.util.\*;

class ArrayList7 {

public static void main(String [] args)

{

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

al.add("Ravi");

al.add("Vijay");

al.add("Ajay");

try

{

FileOutputStream fos=new FileOutputStream("file");

ObjectOutputStream oos=new ObjectOutputStream(fos);

oos.writeObject(al);

fos.close();

oos.close();

FileInputStream fis=new FileInputStream("file");

ObjectInputStream ois=new ObjectInputStream(fis);

ArrayList list=(ArrayList)ois.readObject();

System.out.println(list);

}catch(Exception e)

{

System.out.println(e);

}

}

}

