1.Write a Java program that reads a file name from the user, and then displays

information about whether the file exists, whether the file is readable, whether the

file is writable, the type of file and the length of the file in bytes.

ANSWER:

import java.io.\*;

importjava.util.\*;

classAboutFile{

public static void main(String[] args){

Scanner input = new Scanner(System.in);

System.out.println("Enter the name of the file:");

String file\_name = input.nextLine();

File f = new File(file\_name);

if(f.exists())

System.out.println("The file " +file\_name+ " exists");

else

System.out.println("The file " +file\_name+ " does not exist");

if(f.exists()){

if(f.canRead())

System.out.println("The file " +file\_name+ " is readable");

else

System.out.println("The file " +file\_name+ " is not readable");

if(f.canWrite())

System.out.println("The file " +file\_name+ " is writeable");

else

System.out.println("The file " +file\_name+ " is not writeable");

System.out.println("The file type is: " +file\_name.substring(file\_name.indexOf('.')+1));

System.out.println("The Length of the file:" +f.length());

}

}

}

2.Write a Java program that reads a file and displays the file on the screen, with a

line number before each line.

ANSWER:

importjava.util.\*;

import java.io.\*;

classRFile

{

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

{

int j=1;

charch;

Scanner scr=new Scanner(System.in);

System.out.print("\nEnter File name: ");

String str=scr.next();

FileInputStream f=new FileInputStream(str);

System.out.println("\nContents of the file are");

int n=f.available();

System.out.print(j+": ");

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

{

ch=(char)f.read();

System.out.print(ch);

if(ch=='\n')

{

System.out.print(++j+": ");

}

}

}

}

3.Write a Java program that displays the number of characters, lines and words in a

text file.

ANSWER:

import java.io.\*;

classFileDemo

{

public static void main(String args[])

{

try

{

int lines=0,chars=0,words=0;

int code=0;

FileInputStreamfis = new FileInputStream("Test.java");

while(fis.available()!=0)

{

code = fis.read();

if(code!=10)

chars++;

if(code==32)

words++;

if(code==13)

{

lines++;

words++;

}

}

System.out.println("No.of characters = "+chars);

System.out.println("No.of words = "+(words+1));

System.out.println("No.of lines = "+(lines+1));

fis.close();

}

catch(FileNotFoundException e)

{

System.out.println("Cannot find the specified file...");

}

catch(IOException i)

{

System.out.println("Cannot read file...");

}

}

}

4.Write a Java program to illustrate collection classes like (i) Array List, (ii) Iterator,

(iii)Hash map.

ANSWER:

importjava.util.ArrayList;

importjava.util.Arrays;

importjava.util.List;

public class Test {

public static void main(String[] args) {

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

distros.add("ABC");

distros.add("IJK");

distros.add("DEF");

distros.add("MNO");

for (String distro : distros) {

System.out.println(distro);

}

List<String> capitals = Arrays.asList("PQR", "RST", "EFG",

"LMN", "WXY");

for (String capital : capitals) {

System.out.println(capital);

}

}

}

5.Convert the content of a given file into the uppercase content of the same file.

ANSWER:

import java.io.\*;

importjava.util.\*;

class File

{

public static void main (String[] args)

{

try

{

FileReaderfr = new FileReader("f1.txt");

BufferedReaderbr = new BufferedReader(fr);

PrintWriter out = (new PrintWriter(new FileWriter("f2.txt")));

String s="";

while((s = br.readLine()) != null)

{

out.write(s.toUpperCase()+"\n");

}

out.close();

}

catch(Exception e)

{

e.printStackTrace();

}

}

}

**Ex .No 8**

**1.**Write a java program that implements a multi-threaded application that has three

threads. First thread generates a random integer every 1 second and if the value is even,

second thread computes the square of the number and prints. If the value is odd, the third

thread will print the value of cube of the number.

ANSWER:

import java.io.\*;

importjava.util.\*;

class First extends Thread

{

public void run()

{

int i=0;

try

{

while(i<10)

{

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

i++;

Thread.sleep(1000);

}

}

catch(Exception e3){}

}

}

class Second extends Thread

{

public void run()

{

int j=0;

try

{

while(j<10)

{

System.out.println("Hello ");

j++;

Thread.sleep(2000);

}

}

catch(Exception e2){}

}

}

class Third extends Thread

{

public void run()

{

int k=0;

try

{

while(k<10)

{

System.out.println("Welcome ");

k++;

Thread.sleep(3000);

}

}

catch(Exception e1){}

}

}

classThreeThread

{

public static void main(String args[])

{

try

{

First f=new First();

f.start();

Second s=new Second();

s.start();

Third t=new Third();

t.start();

}

catch(Exception e){}

}

}

2.A program to illustrate the concept of multi-threading that creates three threads. First

thread displays ―Good Morning‖ every one second, the second thread displays ―Hello‖

every two seconds and the third thread displays ―Welcome‖ every three seconds

ANSWER:

importjava.util.Random;

class Square extends Thread

{

int x;

Square(int n)

{

x = n;

}

public void run()

{

intsqr = x \* x;

System.out.println("Square of " + x + " = " + sqr );

}

}

class Cube extends Thread

{

int x;

Cube(int n)

{

x = n;

}

public void run()

{

int cub = x \* x \* x;

System.out.println("Cube of " + x + " = " + cub );

}

}

class Number extends Thread

{

public void run()

{

Random random = new Random();

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

{

intrandomInteger = random.nextInt(100);

System.out.println("Random Integer generated : " + randomInteger);

Square s = new Square(randomInteger);

s.start();

Cube c = new Cube(randomInteger);

c.start();

try {

Thread.sleep(1000);

} catch (InterruptedException ex) {

System.out.println(ex);

}

}

}

}

public class MultiThread{

public static void main(String args[])

{

Number n = new Number();

n.start();

}

}

**Ex .No 9**

1. Sorting using generic method

ANSWER:

importjava.util.ArrayList;

importjava.util.Arrays;

importjava.util.List;

public class SortingGenerics {

private<E> void swap(E[] a, int i, int j) {

if (i != j) {

E temp = a[i];

a[i] = a[j];

a[j] = temp;

}

}

public<E extends Comparable<E>> void selectionSort(E[] a) {

for (int i = 0; i <a.length - 1; i++) {

// find index of smallest element

int smallest = i;

for (int j = i + 1; j <a.length; j++) {

if (a[j].compareTo(a[smallest])<=0) {

smallest = j;

}

}

swap(a, i, smallest); // swap smallest to front

}

}

public static void main(String[] args){

SortingGenericsfirstsort = new SortingGenerics();

Integer[] arr = {3,4,1,5};

System.out.println("before sorting int: "+ Arrays.toString(arr));

firstsort.selectionSort(arr);

System.out.println("After sorting int : "+Arrays.toString(arr));

String[] arr1= {"acd","ded","dal","bad","cle"};

System.out.println("before sorting String: "+ Arrays.toString(arr1));

firstsort.selectionSort(arr1);

System.out.println("After sorting String : "+Arrays.toString(arr1));

Character[] arr2= {'c','e','a','d','c'};

System.out.println("before sorting char: "+ Arrays.toString(arr2));

firstsort.selectionSort(arr2);

System.out.println("After sorting char : "+Arrays.toString(arr2));

}

}

1. Stack using generic class

ANSWER:

classStackFullException extends RuntimeException {

publicStackFullException(){

super();

}

publicStackFullException(String message){

super(message);

}

}

/\*\*

\*Exception to indicate that Stack is empty.

\*/

classStackEmptyException extends RuntimeException {

publicStackEmptyException(){

super();

}

publicStackEmptyException(String message){

super(message);

}

}

/\*\*

\* Stack class(generic type)

\*/

class Stack<T> {

privateint size;

private T[] stackAr;

privateint top; // top of stack

/\*\*

\* Constructor for initializing Array.

\*/

@SuppressWarnings("unchecked")

public Stack(int size) {

this.size = size;

stackAr = (T[])new Object[size]; //Creation of Generic Stack Array

top = -1; // initialize Stack to with -1

}

/\*\*

\* Push items in stack, it will put items on top of Stack.

\*/

public void push(T value){

if(isFull()){

throw new StackFullException("Cannot push "+value+", Stack is full");

}

stackAr[++top] = value;

}

/\*\*

\* Pop items in stack, it will remove items from top of Stack.

\*/

public T pop() {

if(isEmpty()){

throw new StackEmptyException("Stack is empty");

}

returnstackAr[top--]; // remove item and decrement top as well.

}

/\*\*

\* @return true if Stack is empty

\*/

publicbooleanisEmpty(){

return (top == -1);

}

/\*\*

\* @return true if stack is full

\*/

publicbooleanisFull(){

return (top == size - 1);

}

}

/\*\* Copyright (c), AnkitMittal JavaMadeSoEasy.com \*/

/\*\*

\* Main class - StackExampleGeneric

\*/

public class Generic {

public static void main(String[] args) {

Stack<Integer> stack = new Stack<Integer>(10); // Creation of Generic Stack

stack.push(11);

stack.push(21);

stack.push(31);

stack.push(41);

stack.push(51);

System.out.print("Popped items: ");

System.out.print(stack.pop()+"");

System.out.print(stack.pop()+"");

System.out.print(stack.pop()+"");

System.out.print(stack.pop()+"");

System.out.print(stack.pop()+"");

}

}

3.Write a java program to find the maximum value from the given type of elements

using a generic function

ANSWER:

public class Max

{

public static <T extends Comparable<T>> T maximum(T x, T y, T z)

{

T max = x;

if (y.compareTo(max) > 0)

max = y;

if (z.compareTo(max) > 0)

max = z;

return max;

}

public static void main(String args[])

{

System.out.println(maximum(3, 4, 8);

System.out.println(maximum(5.6, 1.8, 9.7);

System.out.println(maximum( "Banana", "apple", "orange");

}

}