**EXERCISE:7**

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

SOLUTION:

import java.io.\*;

import javax.swing.\*;

class FileDemo

{

public static void main(String args[])

{

String filename = JOptionPane.showInputDialog("Enter filename: ");

File f = new File(filename);

System.out.println("File exists: "+f.exists());

System.out.println("File is readable: "+f.canRead());

System.out.println("File is writable: "+f.canWrite());

System.out.println("Is a directory: "+f.isDirectory());

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

try

{

char ch;

StringBuffer buff = new StringBuffer("");

FileInputStream fis = new FileInputStream(filename);

while(fis.available()!=0)

{

ch = (char)fis.read();

buff.append(ch);

}

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

System.out.println(buff);

fis.close();

}

catch(FileNotFoundException e)

{

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

}

catch(IOException i)

{

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

}

}

}

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

line number before each line.

SOLUTION:

import java.util.\*;

import java.io.\*;

class Rfile

{

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

{

int j=1;

char ch;

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.

SOLUTION:

import java.io.\*;

class File

{

public static void main(String args[])

{

try

{

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

int code=0;

FileInputStream fis = new FileInputStream("sample.txt");

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.

Array List

SOLUTION:

import java.util.ArrayList;

import java.util.Arrays;

import java.util.List;

public class Array

{

public static void main(String[] args)

{

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

distros.add("Manjaro");

distros.add("Xubuntu");

distros.add("Fedora");

distros.add("elementary");

for (String distro : distros)

{

System.out.println(distro);

}

List<String> capitals = Arrays.asList("Prague", "Bratislava", "Warsaw",

"Budapest", "Washington");

for (String capital : capitals)

{

System.out.println(capital);

}

}

}

Hash map

SOLUTION:

import java.util.HashMap;

import java.util.Map;

import java.util.Set;

public class HashMapEx

{

public static void main(String[] args)

{

Map<String, String> domains = new HashMap<>();

domains.put("de", "Germany");

domains.put("sk", "Slovakia");

domains.put("us", "United States");

domains.put("ru", "Russia");

domains.put("hu", "Hungary");

domains.put("pl", "Poland");

System.out.println(domains.get("pl"));

for (String item : domains.values())

{

System.out.println(item);

}

Set keys = domains.keySet();

System.out.println(keys);

}

}

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

SOLUTION:

import java.io.\*;

importjava.util.\*;

class FileCon

{

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();

}

}

}

**EXERCISE: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.

SOLUTION:

import java.util.\*;

class RandomNumberThread extends Thread

{

public void run()

{

Random random = new Random();

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

{

int randomInteger = random.nextInt(100);

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

if((randomInteger%2) == 0)

{

SquareThread sThread = new SquareThread(randomInteger);

sThread.start();

}

else

{

CubeThread cThread = new CubeThread(randomInteger);

cThread.start();

}

try

{

Thread.sleep(1000);

}

catch (InterruptedException e)

{

System.out.println(e);

}

}

}

}

class SquareThread extends Thread

{

int number;

SquareThread(int randomNumbern)

{

number = randomNumbern;

}

public void run()

{

System.out.println("Square of " + number + " = " + (number \* number));

}

}

class CubeThread extends Thread

{

int number;

CubeThread(int randomNumber)

{

number = randomNumber;

}

public void run()

{

System.out.println("Cube of " + number + " = " + number \* number \* number);

}

}

public class MultiThreading

{

public static void main(String args[]) {

RandomNumberThread rnThread = new RandomNumberThread();

rnThread.start();

}

}

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.

SOLUTION:

class A extends Thread

{

synchronized public void run()

{

try

{

while(true)

{

sleep(1000);

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

}

}

catch(Exception e)

{

}

}

}

class B extends Thread

{

synchronized public void run()

{

try

{

while(true)

{

sleep(2000);

System.out.println("hello");

}

}

catch(Exception e)

{

}

}

}

class C extends Thread

{

synchronized public void run()

{

try

{

while(true)

{

sleep(3000);

System.out.println("welcome");

}

}

catch(Exception e)

{

}

}

}

class ThreadDemo

{

public static void main(String args[])

{

A t1=new A();

B t2=new B();

C t3=new C();

t1.start();

t2.start();

t3.start();

}

}

**EXERCISE:9**

**1.**Sorting using generic method

SOLUTION:

import java.util.\*;

public class BubbleSortGeneric<T extends Comparable<? super T>>

{

public static void main(String[] args)

{

String[] arrayOfStrings = {"Andree", "Leana", "Faviola", "Loyce", "Quincy", "Milo", "Jamila" "Blair"};

BubbleSortGeneric<String> stringSorter = new BubbleSortGeneric<>();

stringSorter.bubbleSort(arrayOfStrings);

System.out.println(java.util.Arrays.toString(arrayOfStrings));

Double[] arrayOfDoubles = {0.35, 0.02, 0.36, 0.82, 0.27, 0.49, 0.41, 0.17};

BubbleSortGeneric<Double> doubleSorter = new BubbleSortGeneric<>();

doubleSorter.bubbleSort(arrayOfDoubles);

System.out.println(java.util.Arrays.toString(arrayOfDoubles));

}

void bubbleSort(T[] array)

{

int n = array.length;

while (n > 0)

{

int lastModifiedIndex = 0;

for (int currentIndex = 1; currentIndex < n; currentIndex++)

{

if (array[currentIndex - 1].compareTo(array[currentIndex]) > 0)

{

T temp = array[currentIndex - 1];

array[currentIndex - 1] = array[currentIndex];

array[currentIndex] = temp;

lastModifiedIndex = currentIndex;

}

}

n = lastModifiedIndex;

}

}

}

2.Stack using generic class

classStackFullException extends RuntimeException

{

publicStackFullException()

{

super();

}

publicStackFullException(String message)

{

super(message);

}

}

classStackEmptyException extends RuntimeException

{

publicStackEmptyException()

{

super();

}

publicStackEmptyException(String message)

{

super(message);

}

}

class Stack<T>

{

privateint size;

private T[] stackAr;

privateint top;

public Stack(int size)

{

this.size = size;

stackAr = (T[])new Object[size];

top = -1;

}

public void push(T value)

{

if(isFull())

{

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

}

stackAr[++top] = value;

}

public T pop()

{

if(isEmpty())

{

throw new StackEmptyException("Stack is empty");

}

returnstackAr[top--];

}

publicbooleanisEmpty()

{

return (top == -1);

}

publicbooleanisFull()

{

return (top == size - 1);

}

}

public class Generic

{

public static void main(String[] args)

{

Stack<Integer> stack = new Stack<Integer>(10);

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.

SOLUTION:

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");

}

}