1.Type casting

**public** **class** Typecasting {

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

//implicit conversion

System.***out***.println("Implicit TypeCasting");

**char** a='A';

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

**int** b=a;

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

**float** c=a;

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

**long** d=a;

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

**double** e=a;

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

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

System.***out***.println("Explicit Type Casting");

//explicit conversion

**double** x=45.5;

**int** y=(**int**)x;

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

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

}

}

2.Access modifiers.

public **class** M {

**private** **int** intvar=10;

**long** longvar=8775689;

**protected** **float** floatvar=65.87768f;

**private** **void** MethodPrivate(){

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

}

**protected** **void** MethodProtected(){

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

}

**void** MethodDefault(){

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

}

**public** **void** MethodPublic(){

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

}

}

**public** **class** N {

**public** **int** intvar=20;

**protected** **long** longvar=98775689;

**double** doublevar=6.768677678;

**public** **void** MethodPublic(){

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

}

**protected** **void** MethodProtected(){

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

}

**void** MethodDefault(){

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

}

**private** **void** MethodPrivate(){

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

}

}

**public** **class** P {

**public** **void** MethodPublic(){

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

}

**protected** **void** MethodProtected(){

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

}

**void** MethodDefault(){

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

}

**private** **void** MethodPrivate(){

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

}

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

M objM=**new** M();

N objN=**new** N();

System.***out***.println("long variable of M class: "+objM.longvar);

System.***out***.println("float variable of M class: "+objM.floatvar);

}

}

**public** **class** X {

**public** **char** charvar='c';

**private** **int** intvar=50;

**long** longvar=4587;

**protected** **float** floatvar=65.6786f;

}

**public** **class** Y **extends** N{

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

M objM=**new** M();

objM.MethodPublic();

Y objY=**new** Y();

objY.MethodProtected();

objY.MethodPublic();

X objX=**new** X();

System.***out***.println("long variable of X class: "+objX.longvar);

System.***out***.println("float variable of X class: "+objX.floatvar);

System.***out***.println("char variable of X class: "+objX.charvar);

}

}

**public** **class** Z **extends** M{

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

Z objZ=**new** Z();

objZ.MethodProtected();

objZ.MethodPublic();

N objN=**new** N();

objN.MethodPublic();

P objP=**new** P();

objP.MethodPublic();

X objX=**new** X();

System.***out***.println("long variable of X class: "+objX.longvar);

System.***out***.println("float variable of X class: "+objX.floatvar);

System.***out***.println("char variable of X class: "+objX.charvar);

}

}

3. Arithmetic Calculator.

public class Arrithmetic Calculator {

public static void main (String[ ] args)

{

double n1 , n2;

Scanner s=new Scanner (System.in);

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

n1=s.nextDouble();

n2=s.nextDouble();

System.out.println("Enter operator(+,-,\*,/)");

char op=s.next().charAt(0);

double o=0;

switch(op)

{

case '+':

o=n1+n2;

break;

case '-':

o=n1-n2;

break;

case '\*':

o=n1\*n2;

break;

case '/':

o=n1/n2;

break;

}

System.out.println("The final output is");

System.out.println();

System.out.println(n1+" "+op+" "+n2+" = "+o);

}

}

4.ReturnTypes.

**public** **class** Returntypes

{

**static** **void** add()

{

**int** a,b;

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

System.***out***.println("Enter the numbers");

a=s.nextInt();

b=s.nextInt();

**int** c=a+b;

System.***out***.println("Add method "+c);

}

**static** **int** addition()

{

**int** a,b;

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

System.***out***.println("Enter the numbers");

a=s.nextInt();

b=s.nextInt();

**int** c=a+b;

**return** c;

}

**static** **float** adds()

{

**float** a,b;

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

System.***out***.println("Enter the numbers");

a=s.nextFloat();

b=s.nextFloat();

**float** c=a+b;

**return** c;

}

**static** **char** returnchar()

{

**return** 'd';

}

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

{

*add*();

**int** addi=*addition*();

System.***out***.println(addi);

**float** addsmethod=*adds*();

System.***out***.println(addsmethod);

**char** d=*returnchar*();

System.***out***.println(d);

}

}

5.Constructor.

**class** constructors {

**int** id;

String name;

**void** display() {

System.***out***.println(id+" "+name);

}

}

**public** **class** Demo {

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

constructors e1=**new** constructors ();

constructors e2=**new** constructors ();

e1.display();

e2.display();

}

}

//parameterized constructor

**class** Std{

**int** id;

String name;

Std(**int** i,String n)

{

id=i;

name=n;

}

**void** display() {

System.***out***.println(id+" "+name);

}

}

**public** **class** parameterizedConstrDemo {

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

Std s1=**new** Std(2,"Arun");

Std s2=**new** Std(10,"Anand");

s1.display();

s2.display();

}

}

6.Collections.

**public** **class** collection {

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

//creating arraylist

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

ArrayList<String> city=**new** ArrayList<String>();

city.add("Bangalore");//

city.add("Delhi");

System.***out***.println(city);

//creating vector

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

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

Vector<Integer> vec = **new** Vector();

vec.addElement(15);

vec.addElement(30);

System.***out***.println(vec);

//creating linkedlist

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

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

LinkedList<String> names=**new** LinkedList<String>();

names.add("Alex");

names.add("John");

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

**while**(itr.hasNext()){

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

//creating hashset

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

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

HashSet<Integer> set=**new** HashSet<Integer>();

set.add(101);

set.add(103);

set.add(102);

set.add(104);

System.***out***.println(set);

//creating linkedhashset

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

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

LinkedHashSet<Integer> set2=**new** LinkedHashSet<Integer>();

set2.add(11);

set2.add(13);

set2.add(12);

set2.add(14);

System.***out***.println(set2);

}

}

}

7.Map.

**public** **class** Map {

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

//Hashmap

HashMap<Integer,String> hm=**new** HashMap<Integer,String>();

hm.put(1,"ramu");

hm.put(2,"raju");

hm.put(3,"ravi");

System.***out***.println("\nThe elements of Hashmap are ");

**for**(Entry<Integer, String> m:hm.entrySet()){

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

}

//HashTable

Hashtable<Integer,String> ht=**new** Hashtable<Integer,String>();

ht.put(4,"ramesh");

ht.put(5,"Roshan");

ht.put(6,"Jack");

ht.put(7,"John");

System.***out***.println("\nThe elements of HashTable are ");

**for**(Entry<Integer, String> n:ht.entrySet()){

System.***out***.println(n.getKey()+" "+n.getValue());

}

//TreeMap

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

map.put(8,"Anasuya");

map.put(9,"Cat");

map.put(10,"Cattle");

System.***out***.println("\nThe elements of TreeMap are ");

**for**(Entry<Integer, String> l:map.entrySet()){

System.***out***.println(l.getKey()+" "+l.getValue());

}

}

}

7.Inner class.

**public** **class** Innerclass {

**private** **int** data=10;

**void** display(){

System.***out***.println(" outer class method");

}

**class** Inner{

**private** **int** data=30;

**void** msg()

{

Innerclass.**this**.display();

System.***out***.println("data is "+data);

}

**void** display(){

System.***out***.println(" inner class method");

}

**class** Inner\_2{

**void** Inner1()

{

Innerclass.**this**.display();

System.***out***.println("data is "+data);

}

**void** Inner2()

{

System.***out***.println("2nd Inner class");

}

}

}

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

Innerclass obj=**new** Innerclass();

Innerclass.Inner in=obj.**new** Inner();

Inner.Inner\_2 i=in.**new** Inner\_2();

in.msg();

in.display();

i.Inner1();

i.Inner2();

}

}

8.string buffer and string builder.

**public** **class** stringbufferstringbuilder {

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

//methods of strings

System.***out***.println("Methods of Strings");

String sl=**new** String("Hello World");

System.***out***.println(sl.length());

//substring

String sub=**new** String("World");

System.***out***.println(sub.substring(2));

//String Comparison

String s1="what";

String s2="where";

System.***out***.println(s1.compareTo(s2));

//IsEmpty

String s4="";

System.***out***.println(s4.isEmpty());

//toLowerCase

String s5="what";

System.***out***.println(s1.toLowerCase());

//replace

String s6="where";

String replace=s2.replace('d', 'l');

System.***out***.println(replace);

//equals

String x="Welcome to Java";

String y="WeLcOmE tO JaVa";

System.***out***.println(x.equals(y));

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

System.***out***.println("Creating StringBuffer");

//Creating StringBuffer and append method

StringBuffer s=**new** StringBuffer("Welcome to Java!");

s.append("Enjoy your learning");

System.***out***.println(s);

//insert method

s.insert(0, 'w');

System.***out***.println(s);

//replace method

StringBuffer sb=**new** StringBuffer("Hello");

sb.replace(0, 2, "hEl");

System.***out***.println(sb);

//delete method

sb.delete(0, 1);

System.***out***.println(sb);

//StringBuilder

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

System.***out***.println("Creating StringBuilder");

StringBuilder sb1=**new** StringBuilder("Happy");

sb1.append("Learning");

System.***out***.println(sb1);

System.***out***.println(sb1.delete(0, 1));

System.***out***.println(sb1.insert(1, "Welcome"));

System.***out***.println(sb1.reverse());

//conversion

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

System.***out***.println("Conversion of Strings to StringBuffer and StringBuilder");

String str = "what";

// conversion from String object to StringBuffer

StringBuffer sbr = **new** StringBuffer(str);

sbr.reverse();

System.***out***.println("String to StringBuffer");

System.***out***.println(sbr);

// conversion from String object to StringBuilder

StringBuilder sbl = **new** StringBuilder(str);

sbl.append("world");

System.***out***.println("String to StringBuilder");

System.***out***.println(sbl);

}

}

9.single and multi dimensional Arrays.

**public** **class** Arrayssinglemulti {

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

//single-dimensional array

**int** a[]= {2,4,6,8,20};

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

System.***out***.println("Elements of array a: "+a[i]);

}

//multidimensional array

**int**[][] b = {

{2, 4, 6, 8},

{3, 6, 9} };

System.***out***.println("\nLength of row 1: " + b[0].length);

}

}

10.Regular Expression.

**public** **class** Regularexpresion {

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

String pattern = "[a-z]+";

String check = "Regular Expressions";

Pattern p = Pattern.*compile*(pattern);

Matcher c = p.matcher(check);

**while** (c.find())

System.***out***.println( check.substring( c.start(), c.end() ) );

}

}

11.Search a string.

**public** **class** Searchstring {

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

{

String[] str= {"raju","ramesh","ravi"};

**boolean** found=**false**;

**int** index=0;

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

System.***out***.println("Enter the string");

String a=s.nextLine();

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

{

**if**(a.equals(str[i]))

{

index=i;

found=**true**;

**break**;

}

}

**if**(found)

{

System.***out***.println(a+" found at the index "+index);

}

**else**

{

System.***out***.println(a+" not found in the array");

}

}

}

12.Threads.

**public** **class** MyThread **extends** Thread

{

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

{

System.***out***.println("concurrent thread started running..");

}

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

{

MyThread mt = **new** MyThread();

mt.start();

}

}

**public** **class** MyRunnableThread **implements** Runnable{

**public** **static** **int** myCount = 0;

**public** MyRunnableThread(){

}

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

**while**(MyRunnableThread.myCount <= 10){

**try**{

System.out.println("Expl Thread: "+(++MyRunnableThread.myCount));

Thread.sleep(100);

} **catch** (InterruptedException iex) {

System.out.println("Exception in thread: "+iex.getMessage());

}

}

}

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

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

MyRunnableThread mrt = **new** MyRunnableThread();

Thread t = **new** Thread(mrt);

t.start();

**while**(MyRunnableThread.myCount <= 10){

**try**{

System.out.println("Main Thread: "+(++MyRunnableThread.myCount));

Thread.sleep(100);

} **catch** (InterruptedException iex){

System.out.println("Exception in main thread: "+iex.getMessage());

}

}

System.out.println("End of Main Thread...");

}

}

13.sleep() and wait().

**public** **class** Syncronized {

**private** **static** Object *LOCK* = **new** Object();

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

{

Thread.*sleep*(1000);

System.***out***.println("Thread '" + Thread.*currentThread*().getName() + "is woken after sleeping for 1 second");

**synchronized** (*LOCK*)

{

*LOCK*.wait(1000);

System.***out***.println("Object '" + *LOCK* + "' is woken after" + " waiting for 1 second");

}

}

}

14.With Synchronization.

**class** Sender

{

**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");

}

}

**class** ThreadedSend **extends** Thread

{

**private** String msg;

**private** Thread t;

Sender sender;

ThreadedSend(String m, Sender obj)

{

msg = m;

sender = obj;

}

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

{

**synchronized**(sender)

{

sender.send(msg);

}

}

}

**class** SyncDemo

{

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

{

Sender snd = **new** Sender();

ThreadedSend S1 =

**new** ThreadedSend( " Hi " , snd );

ThreadedSend S2 =

**new** ThreadedSend( " Bye " , snd );

S1.start();

S2.start();

**try**

{

S1.join();

S2.join();

}

**catch**(Exception e)

{

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

}

}

}

15.Trycatch block.

**public** **class** tryblock {

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

{

**int**[] array = **new** **int**[3];

**try**

{

array[7] = 3;

}

**catch** (ArrayIndexOutOfBoundsException e)

{

System.***out***.println("Array index is out of bounds!");

}

**finally**

{

System.***out***.println("The array is of size " + array.length);

}

}

}

16.Throws.

**public** **class** MyClass

{

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

{

**int**[] array = **new** **int**[3];

**try**

{

array[7] = 3;

}

**catch** (ArrayIndexOutOfBoundsException e)

{

System.out.println("Array index is out of bounds!");

}

**finally**

{

System.out.println("The array is of size " + array.length);

}

}

}

-Throw. **public** **class** MyClass

{

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

{

**int**[] array = **new** **int**[3];

**try**

{

array[7] = 3;

}

**catch** (ArrayIndexOutOfBoundsException e)

{

System.out.println("Array index is out of bounds!");

}

**finally**

{

System.out.println("The array is of size " + array.length);

}

}

}

-Finally.

**public** **class** MyClass

{

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

{

**int**[] array = **new** **int**[3];

**try**

{

array[7] = 3;

}

**catch** (ArrayIndexOutOfBoundsException e)

{

System.out.println("Array index is out of bounds!");

}

**finally**

{

System.out.println("The array is of size " + array.length);

}

}

}

-Custom.

**public** **class** MyClass

{

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

{

**int**[] array = **new** **int**[3];

**try**

{

array[7] = 3;

}

**catch** (ArrayIndexOutOfBoundsException e)

{

System.out.println("Array index is out of bounds!");

}

**finally**

{

System.out.println("The array is of size " + array.length);

}

}

}

17.Exceptions.

**public** **class** Exceptions {

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

**int** num1,num2,num3;

num1=50;

num2=30;

**try**{

num3 = num1/num2;

System.***out***.println("Result is "+num3);

}

**catch**(ArithmeticException ae){

System.***out***.println("Numbers cannot be divided by zero");

}

**catch**(Exception ae1)

{

System.***out***.println("i am before the subclass exception");

}

**finally**

{

System.***out***.println(" this block will always executed");

}

num3=num1+num2;

System.***out***.println("Result after addition is "+num3);

}

}

18.FileHandling.

**public** **class** Fileshandling {

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

{

//Create a file

File file = **new** File("F:\\empname.txt");

**try**

{

**if** (file.createNewFile())

{

System.***out***.println("New File is created!");

}

**else**

{

**if**(file.exists())

{

System.***out***.println("File already exists.");

System.***out***.println("File path:" + file.getAbsolutePath());

System.***out***.println("File name: " + file.getName());

System.***out***.println("File class: " + file.getClass());

System.***out***.println("File parent: " + file.getParent());

System.***out***.println("File length: " + file.length());

System.***out***.println("File list: " + file.list());

}

}

} **catch** (IOException e) {

e.printStackTrace();

}

String data = "This is the data in the output file";

**try** {

FileWriter output = **new** FileWriter("F:\\empname.txt");

output.write(data);

System.***out***.println("Data is written to the file.");

output.close();

}

**catch** (Exception ex) {

ex.getStackTrace();

}

**char**[] array = **new** **char**[60];

**try** {

FileReader input = **new** FileReader("F:\\empname.txt");

input.read(array);

System.***out***.println("Data in the file:");

System.***out***.println(array);

input.close();

}

**catch**(Exception exc) {

exc.getStackTrace();

}

**boolean** b = file.delete();

**if**(b==**true**)

{

System.***out***.println("File deleted !!");

}

**else**

{

System.***out***.println("File not deleted");

}

}

}

19:Classess And Objects.

**public** **class** MyClass

{

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

{

**int**[] array = **new** **int**[2];

**try**

{

array[5] = 2;

}

**catch** (ArrayIndexOutOfBoundsException e)

{

System.***out***.println("Array index is out of bounds!");

}

**finally**

{

System.***out***.println("The array is of size " + array.length);

}

}

}

-Polymorphism.

**public** **class** MyClass

{

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

{

**int**[] array = **new** **int**[2];

**try**

{

array[5] = 2;

}

**catch** (ArrayIndexOutOfBoundsException e)

{

System.out.println("Array index is out of bounds!");

}

**finally**

{

System.out.println("The array is of size " + array.length);

}

}

}

-Inheritance.

**public** **class** MyClass

{

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

{

**int**[] array = **new** **int**[2];

**try**

{

array[5] = 2;

}

**catch** (ArrayIndexOutOfBoundsException e)

{

System.out.println("Array index is out of bounds!");

}

**finally**

{

System.out.println("The array is of size " + array.length);

}

}

}

-Encapsulation.

**public** **class** MyClass

{

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

{

**int**[] array = **new** **int**[2];

**try**

{

array[5] = 2;

}

**catch** (ArrayIndexOutOfBoundsException e)

{

System.out.println("Array index is out of bounds!");

}

**finally**

{

System.out.println("The array is of size " + array.length);

}

}

}

-Abstraction.

**public** **class** MyClass

{

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

{

**int**[] array = **new** **int**[2];

**try**

{

array[5] = 2;

}

**catch** (ArrayIndexOutOfBoundsException e)

{

System.out.println("Array index is out of bounds!");

}

**finally**

{

System.out.println("The array is of size " + array.length);

}

}

}

20: Diamond.

**public** **class** MyClass

{

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

{

**int**[] array = **new** **int**[2];

**try**

{

array[5] = 2;

}

**catch** (ArrayIndexOutOfBoundsException e)

{

System.out.println("Array index is out of bounds!");

}

**finally**

{

System.out.println("The array is of size " + array.length);

}

}

}

21.File(Read and Write)

**public** **class** File {

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

//initialize Path object

Path path = Paths.*get*("F:file.txt");

//create file

**try** {

Path createdFilePath = Files.*createFile*(path);

System.***out***.println("Created a file at : "+createdFilePath);

}

**catch** (IOException e) {

e.printStackTrace();

}

Path pathw = Paths.*get*("F:file.txt");

String question = "what is string?";

Charset charset = Charset.*forName*("ISO-8859-1");

**try** {

Files.*write*(pathw, question.getBytes());

List<String> lines = Files.*readAllLines*(pathw, charset);

**for** (String line : lines) {

System.***out***.println(line);

}

}

**catch** (IOException e) {

System.***out***.println(e);

}

}

}

22.Array Rotation.

**public** **class** Arrayrotation {

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

{

**int**[] arr= {1,2,3,4,5};

**int** n=3;

System.***out***.println("original array:");

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

{

System.***out***.println(arr[i]+" ");

}

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

{

**int** j,last;

last=arr[arr.length-1];

**for**(j=arr.length-1;j>0;j--) {

arr[j]=arr[j-1];

}

arr[0]=last;

}

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

System.***out***.println(" after rotation");

**for**(**int** i=0;i<arr.length;i++) {

System.***out***.println(arr[i]+" ");

}

}

}

24.Range Queries.

**public** **class** rangequeries {

**static** **int** *a* = 16;

**static** **int** *b* = 100000;

**static** **long** *table*[][] = **new** **long**[*b*][*a* + 1];

**static** **void** buildSparseTable(**int** arr[], **int** n)

{

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

*table*[i][0] = arr[i];

**for** (**int** j = 1; j <= *a*; j++)

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

*table*[i][j] = *table*[i][j - 1] + *table*[i + (1 << (j - 1))][j - 1];

}

**static** **long** query(**int** L, **int** R)

{

**long** answer = 0;

**for** (**int** j = *a*; j >= 0; j--)

{

**if** (L + (1 << j) - 1 <= R)

{

answer = answer + *table*[L][j];

L += 1 << j;

}

}

**return** answer;

}

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

{

**int** arr[] = { 1, 3, 7, 4, 6, 8 };

**int** n = arr.length;

*buildSparseTable*(arr, n);

System.***out***.println(*query*(1, 5));

System.***out***.println(*query*(2, 4));

System.***out***.println(*query*(6, 7));

}

}

25.Matrices.

**public** **class** Matrices {

**public** **static** **int**[][] multiplyMatrices(**int**[][] firstMatrix, **int**[][] secondMatrix, **int** r1, **int** c1, **int** c2)

{

**int**[][] product = **new** **int**[r1][c2];

**for**(**int** i = 0; i < r1; i++)

{

**for** (**int** j = 0; j < c2; j++)

{

**for** (**int** k = 0; k < c1; k++)

{

product[i][j] += firstMatrix[i][k] \* secondMatrix[k][j];

}

}

}

**return** product;

}

**public** **static** **void** displayProduct(**int**[][] product)

{

System.***out***.println("Product of two matrices is: ");

**for**(**int**[] row : product)

{

**for** (**int** column : row)

{

System.***out***.print(column + " ");

}

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

}

}

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

{

**int** r1 = 2, c1 = 3;

**int** r2 = 3, c2 = 2;

**int**[][] firstMatrix = { {2, 4, 6}, {1, 0, 3} };

**int**[][] secondMatrix = { {1, 5}, {7, 1}, {2, 6} };

**int**[][] product = *multiplyMatrices*(firstMatrix, secondMatrix, r1, c1, c2);

*displayProduct*(product);

}

}

26.Singly linked list.

**public** **class** singlelinkedlist {

**class** Node{

**int** data;

Node next;

**public** Node(**int** data) {

**this**.data = data;

**this**.next = **null**;

}

}

**public** Node head = **null**;

**public** Node tail = **null**;

**public** **void** addNode(**int** data) {

Node newNode = **new** Node(data);

**if**(head == **null**) {

head = newNode;

tail = newNode;

}

**else** {

tail.next = newNode;

tail = newNode;

}

}

**public** **void** display() {

Node current = head;

**if**(head == **null**) {

System.***out***.println("List is empty");

**return**;

}

System.***out***.println("Nodes of singly linked list: ");

**while**(current != **null**) {

System.***out***.print(current.data + " ");

current = current.next;

}

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

}

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

singlelinkedlist s = **new** singlelinkedlist();

//Add nodes to the list

s.addNode(2);

s.addNode(4);

s.addNode(6);

s.addNode(8);

//Displays the nodes present in the list

s.display();

}

}

27.Circular Linked List.

**public** **class** circularlinkedlist {

**static** **class** Node

{

**int** data;

Node next;

}

**static** Node addToEmpty(Node last, **int** data)

{

**if** (last != **null**)

**return** last;

Node temp = **new** Node();

temp.data = data;

last = temp;

last.next = last;

**return** last;

}

**static** Node addBegin(Node last, **int** data)

{

**if** (last == **null**)

**return** *addToEmpty*(last, data);

Node temp = **new** Node();

temp.data = data;

temp.next = last.next;

last.next = temp;

**return** last;

}

**static** Node addEnd(Node last, **int** data)

{

**if** (last == **null**)

**return** *addToEmpty*(last, data);

Node temp = **new** Node();

temp.data = data;

temp.next = last.next;

last.next = temp;

last = temp;

**return** last;

}

**static** Node addAfter(Node last, **int** data, **int** item)

{

**if** (last == **null**)

**return** **null**;

Node temp, p;

p = last.next;

**do**

{

**if** (p.data == item)

{

temp = **new** Node();

temp.data = data;

temp.next = p.next;

p.next = temp;

**if** (p == last)

last = temp;

**return** last;

}

p = p.next;

} **while**(p != last.next);

System.***out***.println(item + " not present in the list.");

**return** last;

}

**static** **void** traverse(Node last)

{

Node p;

**if** (last == **null**)

{

System.***out***.println("List is empty.");

**return**;

}

p = last.next;

**do**

{

System.***out***.print(p.data + " ");

p = p.next;

}

**while**(p != last.next);

}

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

{

Node last = **null**;

last = *addToEmpty*(last, 8);

last = *addBegin*(last, 2);

last = *addBegin*(last, 4);

last = *addEnd*(last, 5);

last = *addEnd*(last, 10);

last = *addAfter*(last, 8,4);

*traverse*(last);

}

}

28.Doubly linked list.

**public** **class** doublelinkeslist {

**class** Node{

**int** data;

Node previous;

Node next;

**public** Node(**int** data) {

**this**.data = data;

}

}

Node head, tail = **null**;

**public** **void** addNode(**int** data) {

Node newNode = **new** Node(data);

**if**(head == **null**) {

head = tail = newNode;

head.previous = **null**;

tail.next = **null**;

}

**else** {

tail.next = newNode;

newNode.previous = tail;

tail = newNode;

tail.next = **null**;

}

}

**public** **void** display() {

Node current = head;

**if**(head == **null**) {

System.***out***.println("List is empty");

**return**;

}

System.***out***.println("Nodes of doubly linked list: ");

**while**(current != **null**) {

System.***out***.print(current.data + " ");

current = current.next;

}

}

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

doublelinkeslist d = **new** doublelinkeslist();

d.addNode(2);

d.addNode(4);

d.addNode(6);

d.addNode(8);

d.addNode(9);

d.display();

}

}

29.Stack operation.

**public** **class** Stackoperation {

**static** **final** **int** ***MAX*** = 1000;

**int** top;

**int** a[] = **new** **int**[***MAX***];

**boolean** isEmpty()

{

**return** (top < 0);

}

Stackoperation()

{

top = -1;

}

**boolean** push(**int** x)

{

**if** (top >= (***MAX***-1))

{

System.***out***.println("Stack Overflow");

**return** **false**;

}

**else**

{

a[++top] = x;

System.***out***.println(x + " pushed into stack");

**return** **true**;

}

}

**int** pop()

{

**if** (top < 0)

{

System.***out***.println("Stack Underflow");

**return** 0;

}

**else**

{

**int** x = a[top--];

**return** x;

}

}

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

{

Stackoperation s = **new** Stackoperation();

s.push(4);

s.push(7);

s.push(9);

System.***out***.println(s.pop() + " Popped from stack");

}

}

30:Working of Queue.

**public** **class** QExample

{

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

{

Queue<String> locationsQueue = **new** LinkedList<>();

locationsQueue.add("Kolkata");

locationsQueue.add("Noida");

locationsQueue.add("Gurgaon");

locationsQueue.add("Delhi");

locationsQueue.add("Patna");

System.***out***.println("Queue is : " + locationsQueue);

System.***out***.println("Head of Queue : " + locationsQueue.peek());

locationsQueue.remove();

System.***out***.println("After removing Head of Queue : " + locationsQueue);

System.***out***.println("Size of Queue : " + locationsQueue.size());

}

}

**31.** **Longestincreasingsubsequence.**

**public** **class** Longestincreasingsubsequence {

**static** **int** *max\_ref*;

**static** **int** \_lis(**int** arr[], **int** n)

{

**if** (n == 1)

**return** 1;

**int** res, max\_ending\_here = 1;

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

{

res = *\_lis*(arr, i);

**if** (arr[i - 1] < arr[n - 1]

&& res + 1 > max\_ending\_here)

max\_ending\_here = res + 1;

}

**if** (*max\_ref* < max\_ending\_here)

*max\_ref* = max\_ending\_here;

**return** max\_ending\_here;

}

**static** **int** lis(**int** arr[], **int** n)

{

*max\_ref* = 1;

*\_lis*(arr, n);

**return** *max\_ref*;

}

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

{

**int** arr[] = { 10,20, 12, 8, 23, 22, 40, 51 };

**int** n = arr.length;

System.***out***.println("Length of lis is " + *lis*(arr, n)+ "\n");

}

}

32.linear search.

**public** **class** linearsearch {

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

**int**[] arr = {5,10,20,30,60};

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

System.***out***.println("Enter the element to be searched");

**int** searchValue = sc.nextInt();

**int** result = (**int**) *linearing*(arr,searchValue);

**if**(result==-1){

System.***out***.println("Element not in the array");

} **else** {

System.***out***.println("Element found at "+result+" and the search key is "+arr[result]);

}

}

**public** **static** **int** linearing(**int** arr[], **int** x) {

**int** arrlength = arr.length;

**for** (**int** i = 0; i < arrlength - 1; i++) {

**if** (arr[i] == x) {

**return** i;

}

}

**return** -1;

}

}

33.Binary search.

**public** **class** binarysearch {

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

**int**[] arr = {2,4,6,7,8,9};

**int** key = 9;

**int** arrlength = arr.length;

*binarySearch*(arr,0,key,arrlength);

}

**public** **static** **void** binarySearch(**int**[] arr, **int** start, **int** key, **int** length){

**int** midValue = (start+length)/2;

**while**(start<=length){

**if**(arr[midValue]<key){

start = midValue + 1;

} **else** **if**(arr[midValue]==key){

System.***out***.println("Element is found at index :"+midValue);

**break**;

}**else** {

length=midValue-1;

}

midValue = (start+length)/2;

}

**if**(start>length){

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

}

}

}

34.Exponential search.

**public** **class** Exponentialsearch {

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

**int**[] arr = {3,10,11,19,25};

**int** length= arr.length;

**int** value = 19;

**int** outcome = *exponentialSearch*(arr,length,value);

**if**(outcome<0){

System.***out***.println( "Element is not present in the array");

}**else** {

System.***out***.println( "Element is present in the array at index :"+outcome);

}

}

**public** **static** **int** exponentialSearch(**int**[] arr ,**int** length, **int** value ){

**if**(arr[0]==value){

**return** 0;

}

**int** i=1;

**while**(i<length && arr[i]<=value){

i=i\*2;

}

**return** Arrays.*binarySearch*(arr,i/2,Math.*min*(i,length),value);

}

}

35.Selection sort.

**public** **class** selectionsort {

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

**int**[] arr = {7,3,2,6,8,4};

**int** length = arr.length;

*selectionSort*(arr);

System.***out***.println("The sorted elements are:");

**for**(**int** i:arr){

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

}

}

**public** **static** **void** selectionSort(**int**[] arr){

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

**int** index =i;

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

**if**(arr[j]<arr[index]){

index =j;

}

}

**int** smallNumber = arr[index];

arr[index]= arr[i];

arr[i]= smallNumber;

}

}

}

36.bubble Sort.

**public** **class** bubblesort {

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

**int**[] arr= {15,10,35,55,20};

*bubbleSort*(arr);

**for**(**int** i=0;i<arr.length;i++){

System.***out***.println(arr[i]);

}

}

**public** **static** **void** bubbleSort(**int**[] arr){

**int** len = arr.length;

**int** temp = 0;

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

**for** (**int** j=1;j<(len);j++){

**if**(arr[j-1]>arr[j]){

temp = arr[j-1];

arr[j-1]= arr[j];

arr[j]= temp;

}

}

}

}

}

37.Insertion Sort.

**public** **class** insertionsort {

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

**int**[] arr = {2,12,5,11,4};

*insertionSort*(arr);

**for**(**int** i=0;i<arr.length;i++){

System.***out***.println(arr[i]);

}

}

**public** **static** **void** insertionSort(**int**[] arr){

**int** len = arr.length;

**for**(**int** j=1;j<len;j++){

**int** key = arr[j];

**int** i=j-1;

**while** ((i>-1) && (arr[i]>key)){

arr[i+1]=arr[i];

i--;

}

arr[i+1]=key;

}

}

}

38.Merge sort.

**class** MergeSort

{

**void** merge(**int** arr[], **int** l, **int** m, **int** r)

{

**int** n1 = m - l + 1;

**int** n2 = r - m;

/\* Create temp arrays \*/

**int** L[] = **new** **int** [n1];

**int** R[] = **new** **int** [n2];

/\*Copy data to temp arrays\*/

**for** (**int** i=0; i<n1; ++i)

L[i] = arr[l + i];

**for** (**int** j=0; j<n2; ++j)

R[j] = arr[m + 1+ j];

**int** i = 0, j = 0;

**int** k = l;

**while** (i < n1 && j < n2)

{

**if** (L[i] <= R[j])

{

arr[k] = L[i];

i++;

}

**else**

{

arr[k] = R[j];

j++;

}

k++;

}

**while** (i < n1)

{

arr[k] = L[i];

i++;

k++;

}

**while** (j < n2)

{

arr[k] = R[j];

j++;

k++;

}

}

**void** sort(**int** arr[], **int** l, **int** r)

{

**if** (l < r)

{

**int** m = (l+r)/2;

sort(arr, l, m);

sort(arr , m+1, r);

merge(arr, l, m, r);

}

}

**static** **void** printArray(**int** arr[])

{

**int** n = arr.length;

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

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

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

}

// Driver method

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

{

**int** arr[] = {10, 9, 3, 1, 5, 7};

System.***out***.println("Given Array");

*printArray*(arr);

MergeSort ob = **new** MergeSort();

ob.sort(arr, 0, arr.length-1);

System.***out***.println("\nSorted array");

*printArray*(arr);

}

}

39.Quick sort.

**public** **class** quicksort {

**int** partition(**int** arr[], **int** low, **int** high)

{

**int** pivot = arr[high];

**int** i = (low-1);

**for** (**int** j=low; j<high; j++)

{

**if** (arr[j] <= pivot)

{

i++;

**int** temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

}

**int** temp = arr[i+1];

arr[i+1] = arr[high];

arr[high] = temp;

**return** i+1;

}

**void** sort(**int** arr[], **int** low, **int** high)

{

**if** (low < high)

{

**int** pi = partition(arr, low, high);

sort(arr, low, pi-1);

sort(arr, pi+1, high);

}

}

**static** **void** printArray(**int** arr[])

{

**int** n = arr.length;

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

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

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

}

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

{

**int** arr[] = {4, 7, 9, 8, 3, 2};

**int** n = arr.length;

quicksort ob = **new** quicksort();

ob.sort(arr, 0, n-1);

System.***out***.println("sorted array");

*printArray*(arr);

}

}

40.BugFix.

**public** **class** Bugfix {

**public** **static** **void** main(String[] args) { System.***out***.println("\t Welcome to TheDesk \n"); *optionsSelection*();

}

**private** **static** **void** optionsSelection() {

String[] arr = {"1. I wish to review my expenditure",

"2. I wish to add my expenditure",

"3. I wish to delete my expenditure",

"4. I wish to sort the expenditures",

"5. I wish to search for a particular expenditure",

"6. Close the application"

};

**int**[] arr1 = {1,2,3,4,5,6};

**int** slen = arr1.length;

**for**(**int** i=0; i<slen;i++){ System.***out***.println(arr[i]);

}

ArrayList<Integer> arrlist = **new** ArrayList<Integer>();

ArrayList<Integer> expenses = **new** ArrayList<Integer>();

expenses.add(10000);

expenses.add(200);

expenses.add(5000);

expenses.add(12000);

expenses.add(1100);

expenses.addAll(arrlist);

System.***out***.println("\nEnter your choice:\t"); Scanner sc = **new** Scanner(System.***in***);

**int** options = sc.nextInt();

**for**(**int** j=1;j<=slen;j++){

**if**(options==j){

**switch** (options){

**case** 1:

System.***out***.println("Your saved expenses are listed below: \n");

System.***out***.println(expenses+"\n");

*optionsSelection*();

**break**;

**case** 2:

System.***out***.println("Enter the value to add your Expense: \n");

**int** value = sc.nextInt(); expenses.add(value);

System.***out***.println("Your value is updated\n");

expenses.addAll(arrlist);

System.***out***.println(expenses+"\n");

optionsSoptionsSelection();

**break**;

**case** 3:

System.***out***.println("You are about the delete all your expenses! \nConfirm again by selecting the same option...\n");

**int** con\_choice=sc.nextInt();

**if**(con\_choice==options)

{

expenses.clear();

System.***out***.println(expenses+"\n");

System.***out***.println("All your expenses are erased!\n");

} **else** {

System.***out***.println("Oops... try again!");

}

*optionsSelection*();

**break**;

**case** 4:

*sortExpenses*(expenses); *optionsSelection*();

**break**;

**case** 5:

*searchExpenses*(expenses); *optionsSelection*();

**break**;

**case** 6:

*closeApp*();

**break**;

**default**:

System.***out***.println("You have made an invalid choice!");

**break**;

}

}

}

}

**private** **static** **void** closeApp() {

System.***out***.println("Closing your application... \nThank you!");

}

**private** **static** **void** searchExpenses(ArrayList<Integer> arrayList) {

**int** leng = arrayList.size();

System.***out***.println("Enter the expense you need to search:\t");

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

**int** input = sc.nextInt();

//Linear Search

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

**if**(arrayList.get(i)==input) {

System.***out***.println("Found the expense " + input + " at " +i + " position");

}

}

}

**private** **static** **void** sortExpenses(ArrayList<Integer> arrayList) {

**int** arrlength = arrayList.size();

//Complete the method. The expenses should be sorted in ascending order.

Collections.sort(arrayList);

System.***out***.println("Sorted expenses: ");

**for**(Integer i: arrayList) {

System.***out***.print(i + " ");

}

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

}

}