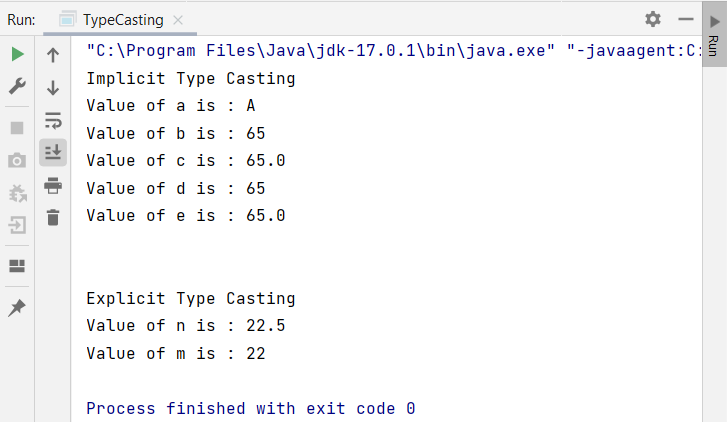
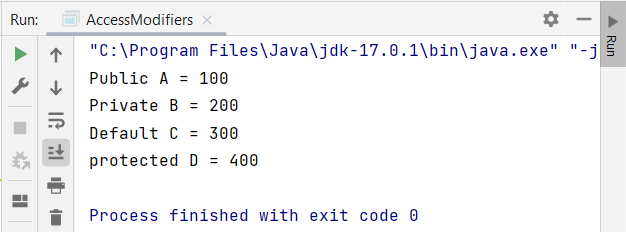
**1. Program which will take string as input and convert it into another data types.**

package com.Shivangani;  
  
public class TypeCasting  
 {  
  
 public static void main (String[] args)  
 {  
  
 System.*out*.println("Implicit Type Casting");  
 char a = 'A';  
 System.*out*.println("Value of a is : " + a);  
  
 int b = a;  
 System.*out*.println("Value of b is : " + b);  
  
 float c = a;  
 System.*out*.println("Value of c is : " + c);  
  
 long d = a;  
 System.*out*.println("Value of d is : " + d);  
  
 double e = a;  
 System.*out*.println("Value of e is : " + e);  
  
 System.*out*.println("\n");  
  
 System.*out*.println("Explicit Type Casting");  
  
 double n = 22.5;  
 int m = (int) n;  
 System.*out*.println("Value of n is : " + n);  
 System.*out*.println("Value of m is : " + m);  
  
 }  
  
 }



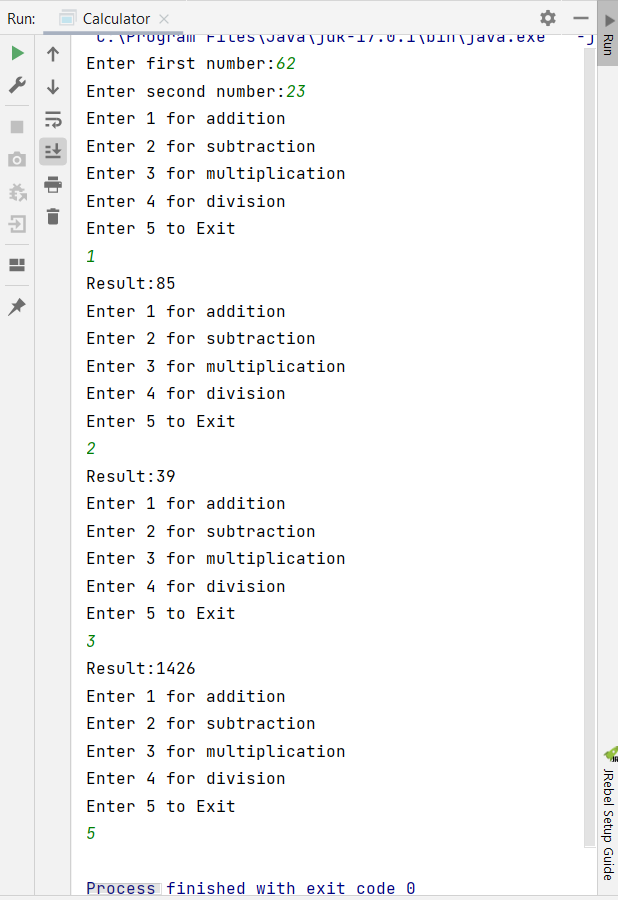
**2. Program to demonstrate the use of Access modifiers**

package com.Shivangani;  
  
public class AccessModifiers {  
 public int a = 100;  
 private int b = 200;  
 int c = 300;  
 protected int d = 400;  
  
 void show() {  
 System.*out*.println("Public A = " + a);  
 System.*out*.println("Private B = " + b);  
 System.*out*.println("Default C = " + c);  
 System.*out*.println("protected D = " + d);  
 }  
  
 public static void main(String[] args) {  
 AccessModifiers access = new AccessModifiers();  
 access.show();  
 }  
}



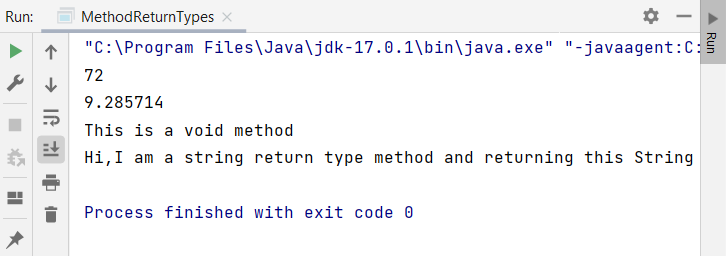
**3. Program to implement an Arithmetic Calculator.**

package com.Shivangani;  
import java.util.\*;  
public class Calculator  
{  
 public static void main(String[] args)  
 {  
 int m, n, opt, add, sub, mul;  
 double div;  
 Scanner s = new Scanner(System.*in*);  
 System.*out*.print("Enter first number:");  
 m = s.nextInt();  
 System.*out*.print("Enter second number:");  
 n = s.nextInt();  
 while(true)  
 {  
 System.*out*.println("Enter 1 for addition");  
 System.*out*.println("Enter 2 for subtraction");  
 System.*out*.println("Enter 3 for multiplication");  
 System.*out*.println("Enter 4 for division");  
 System.*out*.println("Enter 5 to Exit");  
 opt = s.nextInt();  
 switch(opt)  
 {  
 case 1:  
 add = m + n;  
 System.*out*.println("Result:"+add);  
 break;  
  
 case 2:  
 sub = m - n;  
 System.*out*.println("Result:"+sub);  
 break;  
  
 case 3:  
 mul = m \* n;  
 System.*out*.println("Result:"+mul);  
 break;  
  
 case 4:  
 div = (double)m / n;  
 System.*out*.println("Result:"+div);  
 break;  
  
 case 5:  
 System.*exit*(0);  
 }  
 }  
 }  
}



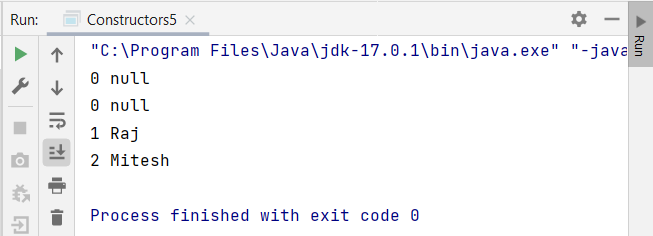
**4. Program to demonstrate different methods of different return types.**

package com.Shivangani;  
  
public class MethodReturnTypes {  
 public static void main(String[]args){  
 int mul = *multiplynumbers*(12, 6);  
 System.*out*.println(mul);  
 float d= *divide*(65,7);  
 System.*out*.println(d);  
 Void voidmethod =new Void();  
 voidmethod.*print*();  
 String str= *returnString*();  
 System.*out*.println(str);  
 }  
 private static int multiplynumbers(int a, int b) {  
 int z = a \* b;  
 return z;  
 }  
  
 int val = 150;  
 *//int return type* int operation(int val) {  
 val = val \* 10 / 100;  
 return (val);  
  
 }  
 *//return typ is float* private static float divide(float a, float b){  
 float res=a/b;  
 return res;  
 }  
  
 private static String returnString() {  
 return "Hi,I am a string return type method and returning this String";  
 }  
}  
class Void{  
 static void print(){  
 System.*out*.println("This is a void method");  
 }  
  
}



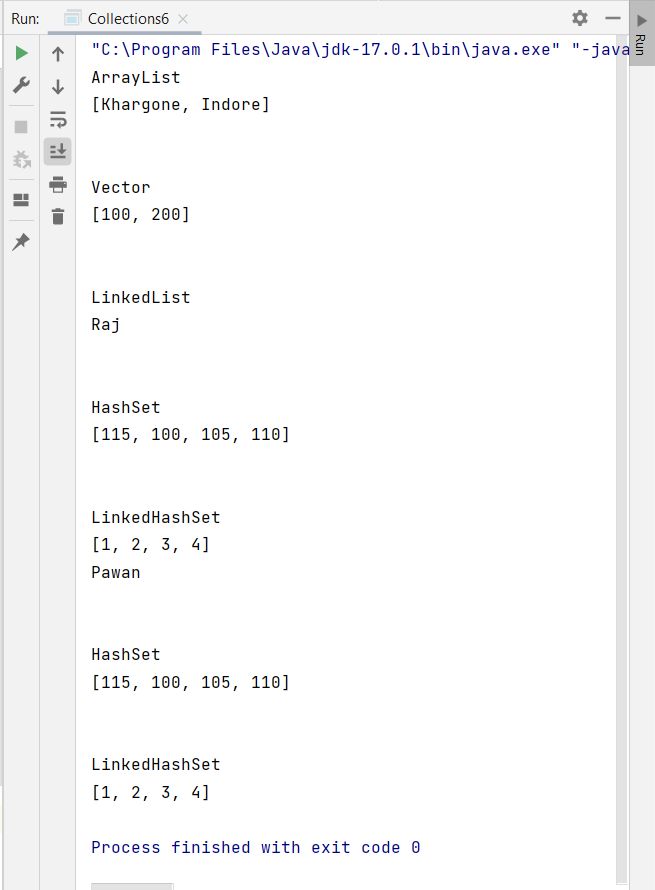
**5. Program to demonstrate the use of constructors**

package com.shivangani;  
  
class StudentInfo {  
 int id;  
 String name;  
 void display() {  
 System.*out*.println(id+" "+name);  
 }  
}  
  
class PlayerInfo {  
 int id;  
 String name;  
 PlayerInfo(int i, String n) {  
 id = i;  
 name = n;  
 }  
 void display() {  
 System.*out*.println(id + " " + name);  
 }  
}  
  
public class Constructors5 {  
 public static void main(String[] args) {  
  
 StudentInfo s1 = new StudentInfo();  
 StudentInfo s2 = new StudentInfo();  
 s1.display();  
 s2.display();  
  
 PlayerInfo p1 = new PlayerInfo(1, "Raj");  
 PlayerInfo p2 = new PlayerInfo(2, "Mitesh");  
 p1.display();  
 p2.display();  
 }  
}



**6.Program to demonstrate the use of Collections**

package com.shivangani;  
  
import java.util.\*;  
  
public class Collections6 {  
 public static void main(String [] args) {  
 *//creating array-list* System.*out*.println("ArrayList");  
 ArrayList<String>city = new ArrayList<String>();  
 city.add("Khargone");  
 city.add("Indore");  
 System.*out*.println(city);  
  
 *//creating vector* System.*out*.println("\n");  
 System.*out*.println("Vector");  
 Vector<Integer> vec = new Vector();  
 vec.addElement(100);  
 vec.addElement(200);  
 System.*out*.println(vec);  
  
 *//creating linked-list* System.*out*.println("\n");  
 System.*out*.println("LinkedList");  
 LinkedList<String> names = new LinkedList<String>();  
 names.add("Raj");  
 names.add("Pawan");  
  
 Iterator<String> itr=names.iterator();  
 while(itr.hasNext()){  
 System.*out*.println(itr.next());  
  
 *//creating hash-set* System.*out*.println("\n");  
 System.*out*.println("HashSet");  
 HashSet<Integer> set = new HashSet<Integer>();  
 set.add(100);  
 set.add(105);  
 set.add(110);  
 set.add(115);  
 System.*out*.println(set);  
  
 *//creating linked-hash-set* System.*out*.println("\n");  
 System.*out*.println("LinkedHashSet");  
 LinkedHashSet<Integer> set2 = new LinkedHashSet<Integer>();  
 set2.add(1);  
 set2.add(2);  
 set2.add(3);  
 set2.add(4);  
 System.*out*.println(set2);  
  
 }  
 }  
  
 public static List<String> emptyList() {  
 */\* TODO Auto-generated method stub \*/* return null;  
 }  
}



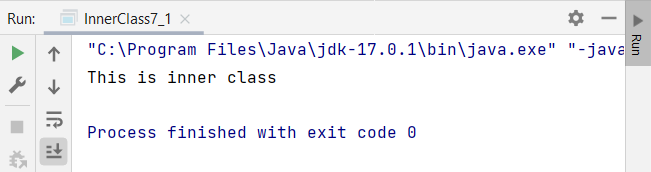
**7.Write a program to demonstrate the use of Map**

package com.shivangani;  
  
import java.util.\*;  
  
public class Map7 {  
 public static void main(String[] args) {  
 *// map  
  
 //Hashmap* HashMap<Integer,String> hm=new HashMap<Integer,String>();  
 hm.put(1,"Tim");  
 hm.put(2,"Mary");  
 hm.put(3,"Catie");  
  
 System.*out*.println("\nThe elements of Hashmap are ");  
 for(Map.Entry m:hm.entrySet()){  
 System.*out*.println(m.getKey()+" "+m.getValue());  
 }  
  
 *//HashTable* Hashtable<Integer,String> ht=new Hashtable<Integer,String>();  
  
 ht.put(4,"Ales");  
 ht.put(5,"Rosy");  
 ht.put(6,"Jack");  
 ht.put(7,"John");  
  
 System.*out*.println("\nThe elements of HashTable are ");  
 for(Map.Entry n:ht.entrySet()){  
 System.*out*.println(n.getKey()+" "+n.getValue());  
 }  
  
 *//TreeMap* TreeMap<Integer,String> map=new TreeMap<Integer,String>();  
 map.put(8,"Annie");  
 map.put(9,"Carlotte");  
 map.put(10,"Catie");  
  
 System.*out*.println("\nThe elements of TreeMap are ");  
 for(Map.Entry l:map.entrySet()){  
 System.*out*.println(l.getKey()+" "+l.getValue());  
 }  
  
 }  
  
}



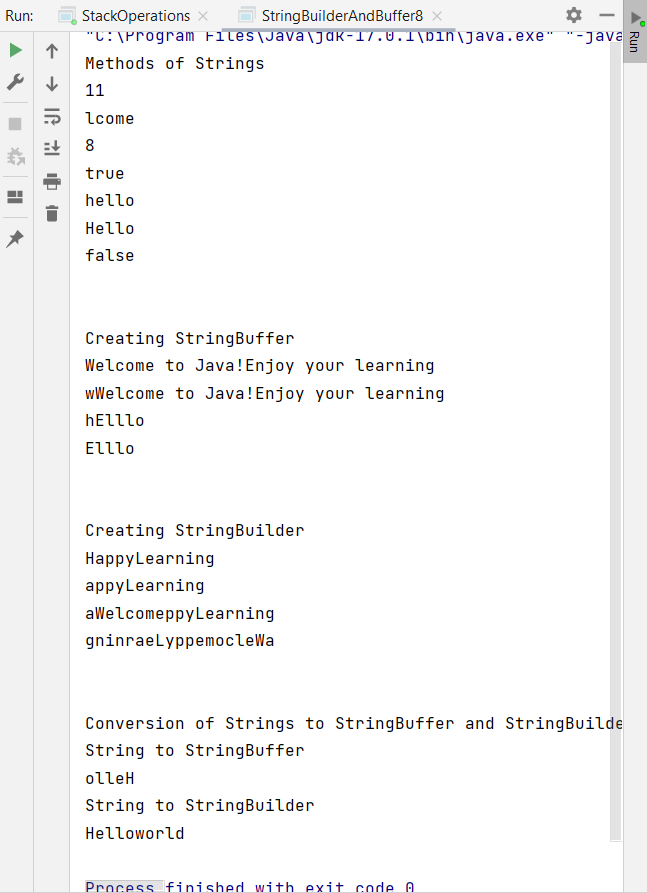
**7.1 Program to demonstrate the use of inner class.**

package com.shivangani;  
  
public class InnerClass7\_1 {  
 private String msg = "This is inner class";  
 void display() {  
 class Inner1 {  
 void msg() {  
 System.*out*.println(msg);  
 }  
 }  
 Inner1 a = new Inner1();  
 a.msg();  
 }  
  
 public static void main(String [] args) {  
 InnerClass7\_1 ob = new InnerClass7\_1();  
 ob.display();  
 }  
}



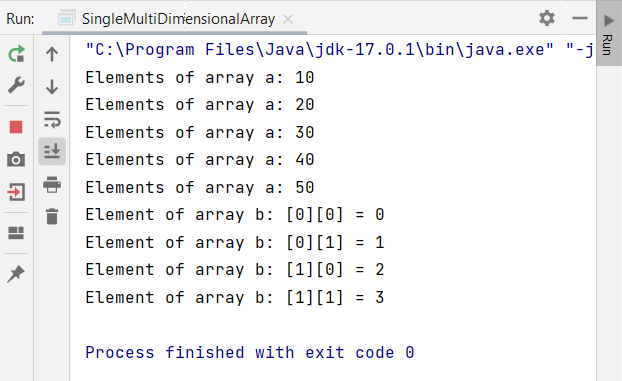
**8. Write a program to create strings and display conversions of string to string-builder and string-buffer**

package com.shivangani;  
  
public class StringBuilderAndBuffer8 {  
 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("Welcome");  
 System.*out*.println(sub.substring(2));  
  
 *//String Comparison* String s1="Hello";  
 String s2="Heldo";  
 System.*out*.println(s1.compareTo(s2));  
  
 *//IsEmpty* String s4="";  
 System.*out*.println(s4.isEmpty());  
  
 *//toLowerCase* String s5="Hello";  
 System.*out*.println(s1.toLowerCase());  
  
 *//replace* String s6="Heldo";  
 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 = "Hello";  
  
 *// 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);  
 }  
}



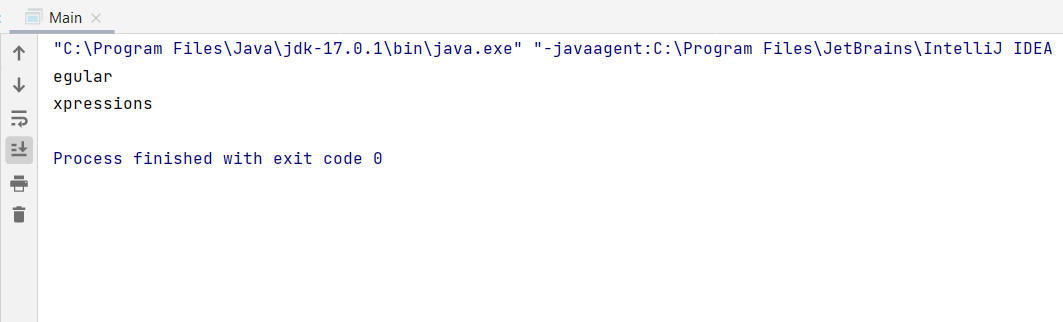
**9.Program to create single and Multidimensional arrays.**

package com.Shivangani;  
  
public class SingleMultiDimensionalArray {  
  
 public static void main(String[] args) {  
  
 *//single-dimensional array* int a[]= {10,20,30,40,50};  
 for(int i = 0;i < 5; i++) {  
 System.*out*.println("Elements of array a: " + a[i]);  
 }  
  
 *//Multi-dimensional array* int[][] b = { {0, 1}, {2, 3} };  
 for(int i = 0; i < 2; i++) {  
 for(int j = 0; j < 2; j++) {  
 System.*out*.println("Element of array b: [" + i + "][" + j + "] = " + b[i][j]);  
 }  
 }  
 }  
}



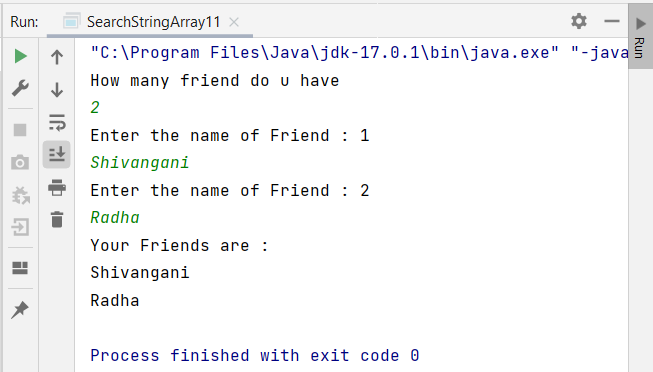
**10. Write a program to search a specific string in a given set of strings using regular expressions.**

package com.shivangani;  
  
import java.util.regex.\*;  
  
public class Main {  
  
 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. Write a program to search a string entered by a user in an array of strings.**

package com.shivangani;  
import java.util.\*;  
public class SearchStringArray {  
 public static void main(String [] args) {  
 int num;  
 Scanner sc = new Scanner(System.*in*);  
 System.*out*.println("How many friend do u have");  
  
 num = sc.nextInt();  
 String[] names = new String[num];  
  
 for(int counter = 0; counter < num; counter++) {  
 System.*out*.println("Enter the name of Friend : " + (counter + 1));  
 names[counter] = sc.next();  
 }  
 sc.close();  
 System.*out*.println("Your Friends are : ");  
 for(int counter = 0; counter < num; counter++) {  
 System.*out*.println(names[counter]);  
 }  
 }  
}

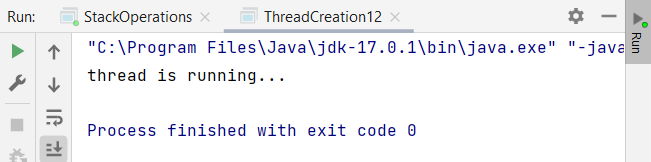


**12. Program to implement thread creation mechanisms**

package com.shivangani;

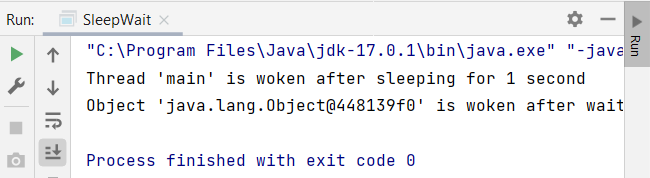
*//method 1: By extending Thread Class*public class ThreadCreation12 extends Thread{  
 public void run() {  
 System.*out*.println("thread is running...");  
  
 }  
  
 public static void main(String args[]){  
 Thread2 m1=new Thread2();  
 Thread t1 =new Thread(m1); *// Using the constructor*

*Thread(Runnable r)* t1.start();  
 }  
  
}  
*// method 2 : By implementing Runnable Interface*class Thread2 implements Runnable{  
 public void run(){  
 System.*out*.println("thread is running...");  
 }  
}



**13. Program to demonstrate the use of sleep(), wait() in threading concept**

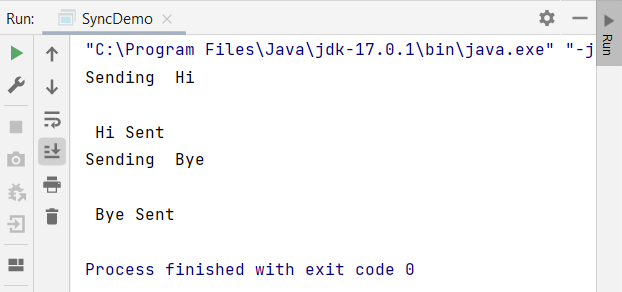
package com.shivangani;  
  
public class SleepWait {  
 private static Object *LOCK* = new Object();  
 public static void main(String args[]) throws InterruptedException  
 {  
 Thread.*sleep*(1000);  
 System.*out*.println("Thread '" + Thread.*currentThread*().getName() + "' is woken after sleeping for 1 second");  
 synchronized (*LOCK*)  
 {  
 *LOCK*.wait(2000);  
 System.*out*.println("Object '" + *LOCK* + "' is woken after" + " waiting for 1 second");  
 }  
 }  
}



**14.Program to demonstrate Multithreading with and without synchronisation.**

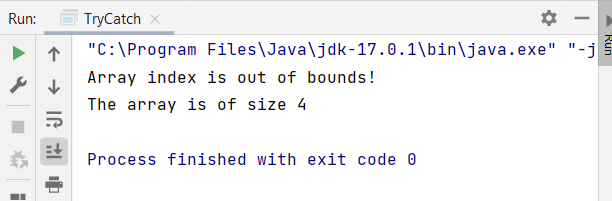
package com.Shivangani;  
public class MultithreadingSyncNoSync {  
 public void sender(String msg)  
 {  
 System.*out*.println("Sending\t" + msg );  
 try  
 {  
 Thread.*sleep*(1000);  
 }  
 catch (Exception e)  
 {  
 System.*out*.println("Thread interrupted.");

}  
 System.*out*.println("\n" + msg + "Sent");  
 }  
}  
class ThreadedSender extends Thread  
{  
 private String msg;  
 private Thread t;  
 MultithreadingSyncNoSync sender;  
 ThreadedSender(String m, MultithreadingSyncNoSync obj)  
 {  
 msg = m;  
 sender = obj;  
 }  
  
 public void run()  
 {  
 synchronized(sender)  
 {  
 sender.sender(msg);  
 }  
 }  
}  
class SyncDemo  
{  
 public static void main(String args[])  
 {  
 MultithreadingSyncNoSync snd = new MultithreadingSyncNoSync();  
 ThreadedSender S1 =  
 new ThreadedSender( " Hi " , snd );  
 ThreadedSender S2 =  
 new ThreadedSender( " Bye " , snd );  
 S1.start();  
 S2.start();  
 try  
 {  
 S1.join();  
 S2.join();  
 }  
 catch(Exception e)  
 {  
 System.*out*.println("Interrupted");  
 }  
 }  
}



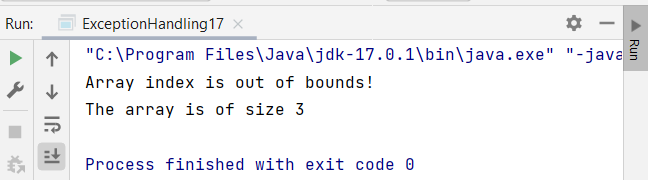
**15. Program to demonstrate the use of try and catch block**.

package com.Shivangani;  
  
public class TryCatch {  
  
 public static void main(String args[])  
 {  
 int[] arr = new int[4];  
 try  
 {  
 arr[5] = 3;  
 }  
 catch (ArrayIndexOutOfBoundsException e)  
 {  
 System.*out*.println("Array index is out of bounds!");  
 }  
  
 finally  
 {  
 System.*out*.println("The array is of size " + arr.length);  
 }  
 }  
}



**17. Program to implement the concept the exception handling and custom exception handlers**

package com.shivangani;  
  
public class ExceptionHandling17 {  
 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);  
 }  
 }  
  
}



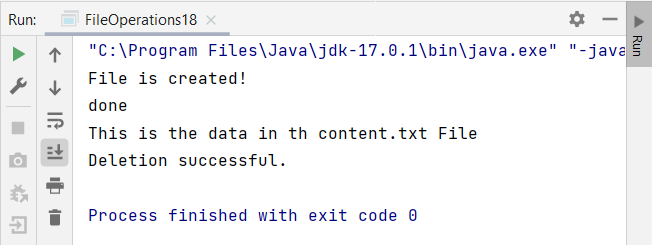
**18. Program to perform create, read, update and delete operations on files in Java.**

package com.shivangani;  
  
import java.io.\*;  
import java.nio.charset.\*;  
import java.nio.file.\*;  
import java.util.\*;  
  
public class FileOperations {  
 public static void main(String args[]) throws IOException {  
 *//CreatingFile* CreateFile cr=new CreateFile();  
 cr.createFile();  
  
 *//Updating File* UpdateFile up=new UpdateFile();  
 up.*modifyFile*("A:\\Documents//Content.txt", "100", "205");  
 System.*out*.println("done");  
  
 *//Reading File* ReadFile rd=new ReadFile();  
 List l = rd.*readFileInList*("A:\\Documents//Content.txt");  
  
 Iterator<String> itr = l.iterator();  
 while (itr.hasNext())  
 System.*out*.println(itr.next());  
  
 *// Deleting File* DeleteFile del=new DeleteFile();  
 del.delete();  
 }  
}  
  
  
 class CreateFile{

*// method to create the file*  
 void createFile() throws IOException  
 {  
 File file = new File("A:\\Documents//Content.txt");  
  
 *//Create the file* if (file.createNewFile()){  
 System.*out*.println("File is created!");  
 }else{  
 System.*out*.println("File already exists.");  
 }  
  
 *//Write Content* FileWriter writer = new FileWriter(file);  
 writer.write("This is the data in th content.txt File");  
 writer.close();  
 }  
 }  
  
class ReadFile {  
*// method to read the file*

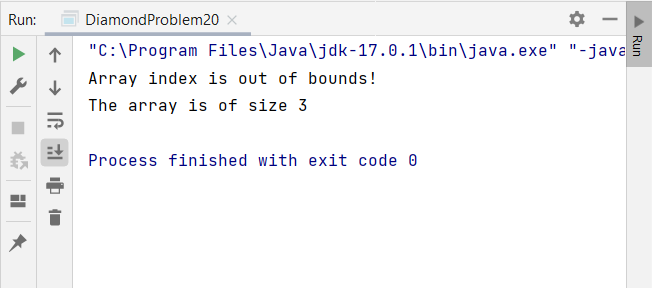
public static List<String> readFileInList(String fileName) {  
  
 List<String> lines = Collections.*emptyList*();  
  
 try {  
 lines =  
 Files.*readAllLines*(Paths.*get*(fileName), StandardCharsets.*UTF\_8*);  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 return lines;  
 }  
}  
 class UpdateFile {  
*// method to update the file:*  
 static void modifyFile(String filePath, String oldString, String newString)  
 {  
 File fileToBeModified = new File(filePath);  
 String oldContent = "";  
 BufferedReader reader = null;  
 FileWriter writer = null;  
 try  
 {  
 reader = new BufferedReader(new FileReader(fileToBeModified));  
 String line = reader.readLine();  
 while (line != null)  
 {  
 oldContent = oldContent + line + System.*lineSeparator*();  
 line = reader.readLine();  
 }  
 String newContent = oldContent.replaceAll(oldString, newString);  
 writer = new FileWriter(fileToBeModified);  
 writer.write(newContent);  
 }  
 catch (IOException e)  
 {  
 e.printStackTrace();  
 }  
 finally  
 {  
 try  
 {  
 reader.close();  
 writer.close();  
 }  
 catch (IOException e)  
 {  
 e.printStackTrace();  
 }  
 }  
 }  
}  
class DeleteFile  
{

*//method to delete the file:*  
 void delete()  
 {  
 try  
 {  
 Files.*deleteIfExists*(Paths.*get*("A:\\Documents//Content.txt"));  
 }  
 catch(NoSuchFileException e)  
 {  
 System.*out*.println("No such file/directory exists");  
 }  
 catch(DirectoryNotEmptyException e)  
 {  
 System.*out*.println("Directory is not empty.");  
 }  
 catch(IOException e)  
 {  
 System.*out*.println("Invalid permissions.");  
 }  
  
 System.*out*.println("Deletion successful.");  
 }  
}



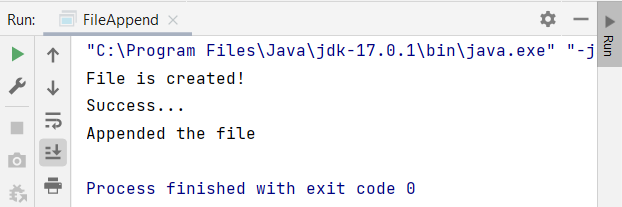
**20.Program to work and resolve the Diamond problem using the OOPs Concept**

package com.shivangani;  
  
public class DiamondProblem20 {  
 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);  
 }  
 }  
  
}



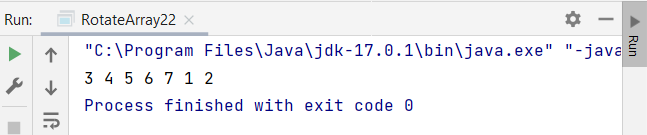
**21. Program to read, write and append a file**

package com.Shivangani;  
  
import java.io.\*;  
import java.nio.charset.StandardCharsets;  
import java.nio.file.Files;  
import java.nio.file.Paths;  
import java.util.Collections;  
import java.util.List;  
  
public class FileAppend {  
public static void main(String[] args) throws IOException {  
 CreateFile create=new CreateFile();  
 create.createFile();  
 ReadFile rd=new ReadFile();  
 rd.*readFile*("A:\\Documents//Content.txt");  
 Write wr=new Write();  
 wr.write();  
 Append app=new Append();  
 app.append();  
}  
}  
  
  
class CreateFile{  
 void createFile() throws IOException  
 {  
 File file = new File("A:\\Documents//Content.txt");  
  
 *//Create the file* if (file.createNewFile()){  
 System.*out*.println("File is created!");  
 }else{  
 System.*out*.println("File already exists.");  
 }  
  
 *//Write Content* FileWriter writer = new FileWriter(file);  
 writer.write("This is the data in th content.txt File");  
 writer.close();  
 }  
}  
  
  
class ReadFile {  
 public static List<String> readFile(String fileName) {  
  
 List<String> lines = Collections.*emptyList*();  
  
 try {  
 lines =  
 Files.*readAllLines*(Paths.*get*(fileName), StandardCharsets.*UTF\_8*);  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 return lines;  
 }  
}  
  
class Write {  
 void write()  
  
 {  
 try {  
 FileWriter fw = new FileWriter("A:\\Documents//Content.txt");  
 fw.write("Writing new content in the file");  
 fw.close();  
 } catch (Exception e) {  
 System.*out*.println(e);  
 }  
 System.*out*.println("Success...");  
 }  
}  
  
  
class Append{  
 void append(){  
  
 try {  
 String data = "Append the data";  
 File f1 = new File("A:\\Documents//Content.txt");  
 if(!f1.exists()) {  
 f1.createNewFile();  
 }  
  
 FileWriter fileWritter = new FileWriter(f1.getName(),true);  
 BufferedWriter bw = new BufferedWriter(fileWritter);  
 bw.write(data);  
 bw.close();  
 System.*out*.println("Appended the file");  
 } catch(IOException e){  
 e.printStackTrace();  
 }  
 }  
}



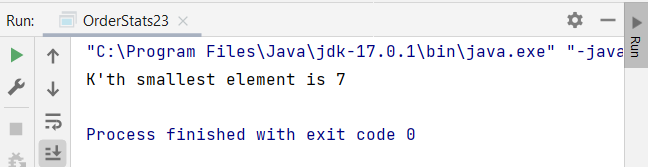
**22. Program to demonstrate the array rotation.**

package com.shivangani;  
  
class RotateArr {  
 public void rotate(int[] nums, int k) {  
 if(k > nums.length)  
 k=k%nums.length;  
 int[] result = new int[nums.length];  
 for(int i=0; i < k; i++){  
 result[i] = nums[nums.length-k+i];  
 }  
 int j=0;  
 for(int i=k; i<nums.length; i++){  
 result[i] = nums[j];  
 j++;  
 }  
 System.*arraycopy*( result, 0, nums, 0, nums.length );  
 }  
}  
public class RotateArray22  
{  
 public static void main(String[] args) {  
 RotateArr r = new RotateArr();  
 int arr[] = { 1, 2, 3, 4, 5, 6, 7 };  
 r.rotate(arr, 5);  
 for(int i=0;i<arr.length;i++){  
 System.*out*.print(arr[i]+" ");  
 }  
 }  
}



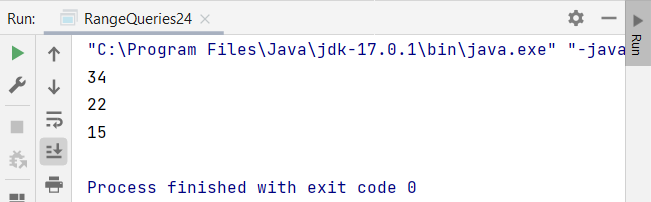
**23.Program to demonstrate order statistics.**

package com.shivangani;  
  
class KthSmallst  
{  
 int kthSmallest(int arr[], int l, int r, int k)  
 {  
 if (k > 0 && k <= r - l + 1)  
 {  
 int pos = randomPartition(arr, l, r);  
 if (pos-l == k-1)  
 return arr[pos];  
 if (pos-l > k-1)  
 return kthSmallest(arr, l, pos-1, k);  
 return kthSmallest(arr, pos+1, r, k-pos+l-1);  
 }  
 return Integer.*MAX\_VALUE*;  
 }  
 void swap(int arr[], int i, int j)  
 {  
 int temp = arr[i];  
 arr[i] = arr[j];  
 arr[j] = temp;  
 }  
 int partition(int arr[], int l, int r)  
 {  
 int x = arr[r], i = l;  
 for (int j = l; j <= r - 1; j++)  
 {  
 if (arr[j] <= x)  
 {  
 swap(arr, i, j);  
 i++;  
 }  
 }  
 swap(arr, i, r);  
 return i;  
 }  
 int randomPartition(int arr[], int l, int r)  
 {  
 int n = r-l+1;  
 int pivot = (int)(Math.*random*()) \* (n-1);  
 swap(arr, l + pivot, r);  
 return partition(arr, l, r);  
 }  
}  
public class OrderStats23  
{  
 public static void main(String[] args) {  
 KthSmallst ob = new KthSmallst();  
 int arr[] = {12, 3, 5, 7, 4, 19, 26};  
 int n = arr.length,k = 4;  
 System.*out*.println("K'th smallest element is "+ ob.kthSmallest(arr, 0, n-1, k));  
 }  
}



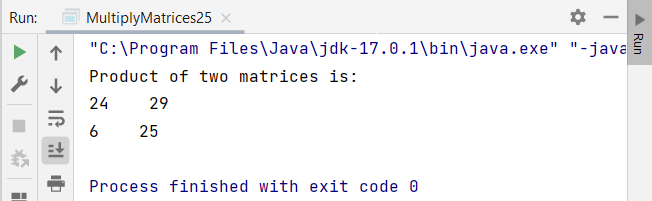
**24.Program to demonstrate range queries.**

package com.shivangani;  
  
public class RangeQueries24  
{  
 static int *k* = 16;  
 static int *N* = 100000;  
 static long *table*[][] = new long[*N*][*k* + 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 <= *k*; 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 = *k*; 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[] = { 3, 7, 2, 5, 8, 9 };  
 int n = arr.length;  
 *buildSparseTable*(arr, n);  
 System.*out*.println(*query*(0, 5));  
 System.*out*.println(*query*(3, 5));  
 System.*out*.println(*query*(2, 4));  
 }  
}



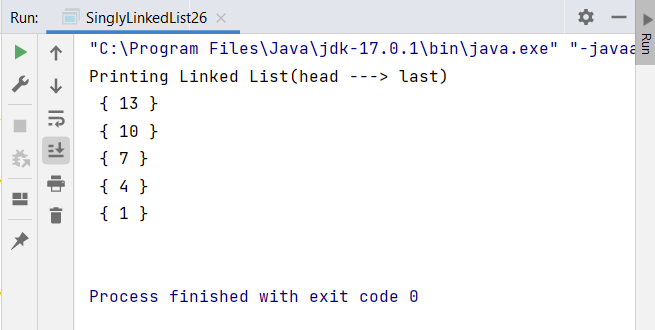
**25. Program to demonstrate working of matrices.**

package com.shivangani;  
  
public class MultiplyMatrices25  
{  
 public static void main(String[] args)  
 {  
 int r1 = 2, c1 = 3;  
 int r2 = 3, c2 = 2;  
 int[][] firstMatrix = { {3, -2, 5}, {3, 0, 4} };  
 int[][] secondMatrix = { {2, 3}, {-9, 0}, {0, 4} };  
 int[][] product = *multiplyMatrices*(firstMatrix, secondMatrix, r1, c1, c2);  
 *displayProduct*(product);  
 }  
  
 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();  
 }  
 }  
}



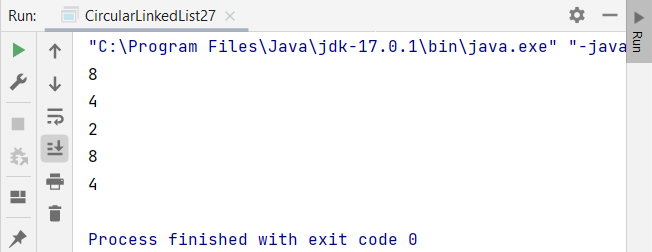
**26. Program to create and perform operations on singly Linked List**

package com.shivangani;  
  
class Node {  
 public int data;  
 public Node next;  
  
 public void displayNodeData() {  
 System.*out*.println(" { " + data + " } ");  
 }  
}  
public class SinglyLinkedList26 {  
 private Node head;  
  
 public boolean isEmpty() {  
 return(head == null);  
 }  
  
 public void insertFirst(int data) {  
 Node newNode = new Node();  
 newNode.data = data;  
 newNode.next = head;  
 head = newNode;  
 }  
  
 public Node deleteFirst() {  
 Node temp = head;  
 head = head.next;  
 return temp;  
 }  
  
 public void deleteAfter(Node after) {  
 Node temp = head;  
 while(temp.next != null && temp.data != after.data) {  
 temp = temp.next;  
 }  
  
 if(temp.next != null) {  
 temp.next = temp.next.next;  
 }  
 }  
  
 public void insertLast(int data) {  
 Node current = head;  
 while(current.next != null) {  
 current = current.next;  
 }  
  
 Node newNode = new Node();  
 newNode.data = data;  
 current.next = newNode;  
 }  
  
 public void printLinkedList() {  
 System.*out*.println("Printing Linked List(head ---> last) ");  
 Node current = head;  
 while(current != null) {  
 current.displayNodeData();  
 current = current.next;  
 }  
 System.*out*.println();  
 }  
  
 public static void main(String [] args) {  
 SinglyLinkedList26 myLinkedList = new SinglyLinkedList26();  
 myLinkedList.insertFirst(1);  
 myLinkedList.insertFirst(4);  
 myLinkedList.insertFirst(7);  
 myLinkedList.insertFirst(10);  
 myLinkedList.insertFirst(13);  
  
 Node node = new Node();  
 node.data = 1;  
 myLinkedList.deleteAfter(node);  
  
 myLinkedList.printLinkedList();  
 }  
}



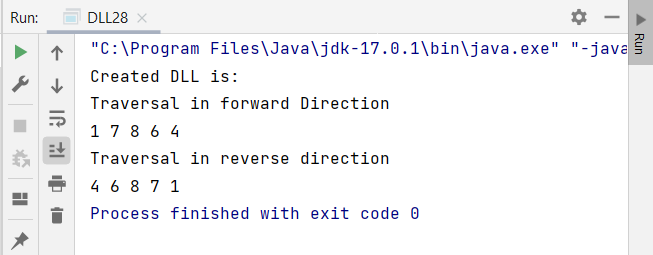
**27. Program to create and perform operations on Circular Linked list.**

package com.shivangani;  
  
public class CircularLinkedList27 {  
 static class Node {  
 int data;  
 Node next;  
 };  
  
 static Node addToEmpty(Node last, int data) {  
 if(last != null)  
 return last;  
  
 Node newNode = new Node();  
 newNode.data = data;  
 last = newNode;  
 newNode.next = last;  
  
 return last;  
 }  
  
 static Node addFront(Node last, int data) {  
 if(last == null)  
 return *addToEmpty*(last, data);  
  
 Node newNode = new Node();  
 newNode.data = data;  
 newNode.next = last.next;  
 last.next = newNode;  
  
 return last;  
 }  
  
 static Node addEnd(Node last, int data) {  
 if(last == null)  
 return *addToEmpty*(last, data);  
  
 Node newNode = new Node();  
 newNode.data = data;  
 newNode.next = last.next;  
 last.next = newNode;  
 last = newNode;  
  
 return last;  
 }  
  
 static Node afterAdd(Node last, int data, int item) {  
 if(last == null)  
 return null;  
  
 Node newNode, p;  
 p = last.next;  
 do {  
 if(p.data == item) {  
 newNode = new Node();  
 newNode.data = data;  
 newNode.next = p.next;  
  
 if(p == last)  
 last = newNode;  
 return last;  
 }  
 p = p.next;  
 }  
 while(p != last.next);  
  
 System.*out*.println(item + "The given node is not present in the list");  
 return last;  
 }  
  
 static Node deleteNode(Node last, int key) {  
 if(last == null)  
 return null;  
 if(last.data == key && last.next == last) {  
 last = null;  
 return last;  
 }  
  
 Node temp = last, d = new Node();  
  
 if(last.data == key) {  
 while(temp.next != last) {  
 temp = temp.next;  
 }  
  
 temp.next = last.next;  
 last = temp.next;  
 }  
  
 while(temp.next != last && temp.next.data != key) {  
 temp = temp.next;  
 }  
  
 if(temp.next.data == key) {  
 d = temp.next;  
 temp.next = d.next;  
 }  
 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*.println(p.data + " ");  
 p = p.next;  
 }  
 while(p != last.next);  
 }  
  
 public static void main(String [] args) {  
 Node last = null;  
 last = *addToEmpty*(last, 4);  
 last = *addEnd*(last, 2);  
 last = *addFront*(last, 8);  
  
 last = *afterAdd*(last, 5, 2);  
  
 *traverse*(last);  
  
 *deleteNode*(last, 2);  
 *traverse*(last);  
 }  
}



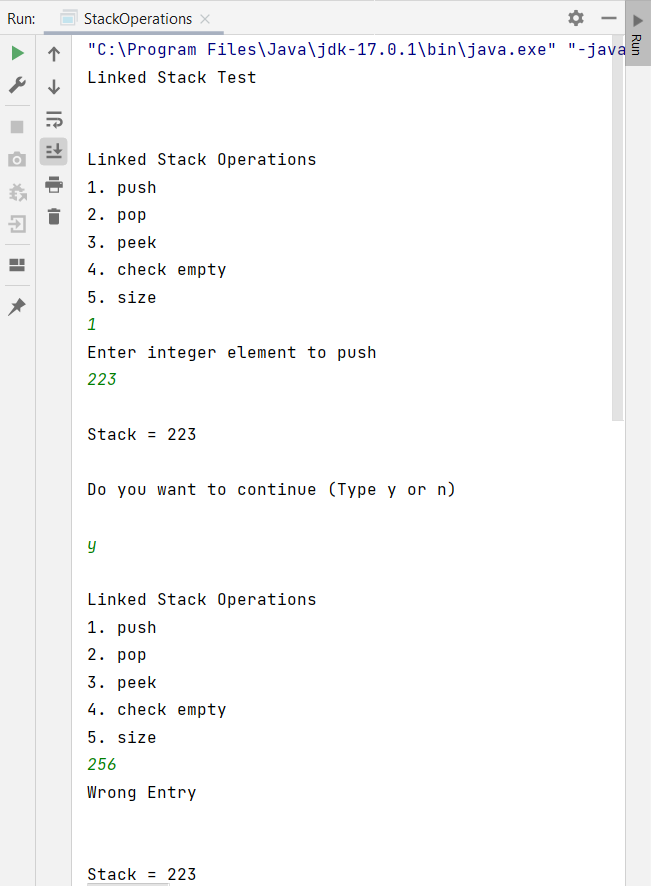
**28. Program to create and perform operations on doubly Linked list.**

package com.shivangani;  
  
public class DLL28 {  
 Node head;  
 class Node  
 {  
 int data;  
 Node prev;  
 Node next;  
 Node(int d)  
 {  
 data = d;  
 }  
 }  
 public void push(int new\_data)  
 {  
 Node new\_Node = new Node(new\_data);  
 new\_Node.next = head;  
 new\_Node.prev = null;  
 if (head != null)  
 head.prev = new\_Node;  
 head = new\_Node;  
 }  
 public void InsertAfter(Node prev\_Node, int new\_data)  
 {  
 if (prev\_Node == null)  
 {  
 System.*out*.println("The given previous node cannot be NULL ");  
 return;  
 }  
 Node new\_node = new Node(new\_data);  
 new\_node.next = prev\_Node.next;  
 prev\_Node.next = new\_node;  
 new\_node.prev = prev\_Node;  
 if (new\_node.next != null)  
 new\_node.next.prev = new\_node;  
 }  
 void append(int new\_data)  
 {  
 Node new\_node = new Node(new\_data);  
 Node last = head;  
 new\_node.next = null;  
 if (head == null)  
 {  
 new\_node.prev = null;  
 head = new\_node;  
 return;  
 }  
 while (last.next != null)  
 last = last.next;  
 last.next = new\_node;  
 new\_node.prev = last;  
 }  
 public void printlist(Node node)  
 {  
 Node last = null;  
 System.*out*.println("Traversal in forward Direction");  
 while (node != null)  
 {  
 System.*out*.print(node.data + " ");  
 last = node;  
 node = node.next;  
 }  
 System.*out*.println();  
 System.*out*.println("Traversal in reverse direction");  
 while (last != null)  
 {  
 System.*out*.print(last.data + " ");  
 last = last.prev;  
 }  
 }  
 public static void main(String[] args)  
 {  
 DLL28 dll = new DLL28();  
 dll.append(6);  
 dll.push(7);  
 dll.push(1);  
 dll.append(4);  
 dll.InsertAfter(dll.head.next, 8);  
 System.*out*.println("Created DLL is: ");  
 dll.printlist(dll.head);  
 }  
  
}



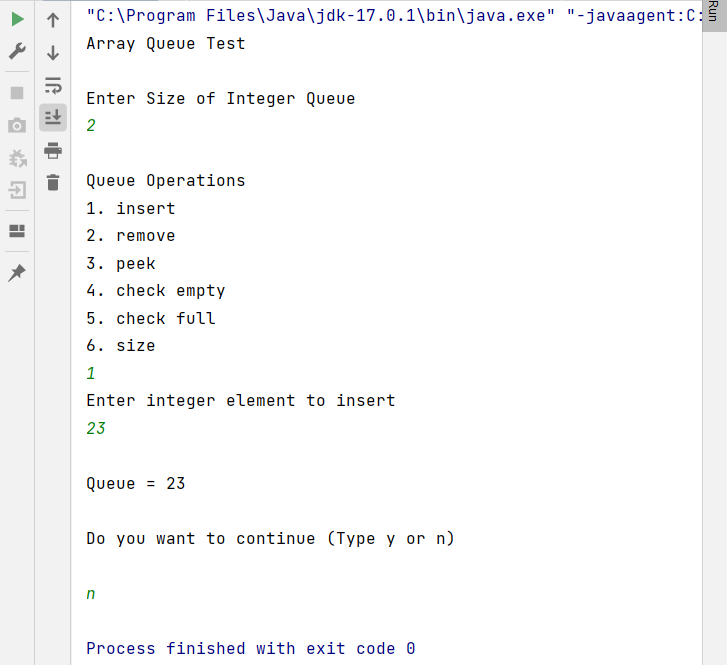
**29. Program to perform operations on stack.**

package com.shivangani;  
  
import java.util.\*;  
  
*/\* Class Node \*/*class Node  
{  
 protected int data;  
 protected Node link;  
  
 */\* Constructor \*/* public Node()  
 {  
 link = null;  
 data = 0;  
 }  
 */\* Constructor \*/* public Node(int d,Node n)  
 {  
 data = d;  
 link = n;  
 }  
 */\* Function to set link to next Node \*/* public void setLink(Node n)  
 {  
 link = n;  
 }  
 */\* Function to set data to current Node \*/* public void setData(int d)  
 {  
 data = d;  
 }  
 */\* Function to get link to next node \*/* public Node getLink()  
 {  
 return link;  
 }  
 */\* Function to get data from current Node \*/* public int getData()  
 {  
 return data;  
 }  
}  
  
*/\* Class linkedStack \*/*class linkedStack  
{  
 protected Node top ;  
 protected int size ;  
  
 */\* Constructor \*/* public linkedStack()  
 {  
 top = null;  
 size = 0;  
 }  
 */\* Function to check if stack is empty \*/* public boolean isEmpty()  
 {  
 return top == null;  
 }  
 */\* Function to get the size of the stack \*/* public int getSize()  
 {  
 return size;  
 }  
 */\* Function to push an element to the stack \*/* public void push(int data)  
 {  
 Node nptr = new Node (data, null);  
 if (top == null)  
 top = nptr;  
 else  
 {  
 nptr.setLink(top);  
 top = nptr;  
 }  
 size++ ;  
 }  
 */\* Function to pop an element from the stack \*/* public int pop()  
 {  
 if (isEmpty() )  
 throw new NoSuchElementException("Underflow Exception") ;  
 Node ptr = top;  
 top = ptr.getLink();  
 size-- ;  
 return ptr.getData();  
 }  
 */\* Function to check the top element of the stack \*/* public int peek()  
 {  
 if (isEmpty() )  
 throw new NoSuchElementException("Underflow Exception") ;  
 return top.getData();  
 }  
 */\* Function to display the status of the stack \*/* public void display()  
 {  
 System.*out*.print("\nStack = ");  
 if (size == 0)  
 {  
 System.*out*.print("Empty\n");  
 return ;  
 }  
 Node ptr = top;  
 while (ptr != null)  
 {  
 System.*out*.print(ptr.getData()+" ");  
 ptr = ptr.getLink();  
 }  
 System.*out*.println();  
 }  
}  
  
*/\* Class LinkedStackImplement \*/*public class StackOperations  
{  
 public static void main(String[] args)  
 {  
 Scanner scan = new Scanner(System.*in*);  
 */\* Creating object of class linkedStack \*/* linkedStack ls = new linkedStack();  
 */\* Perform Stack Operations \*/* System.*out*.println("Linked Stack Test\n");  
 char ch;  
 do  
 {  
 System.*out*.println("\nLinked Stack Operations");  
 System.*out*.println("1. push");  
 System.*out*.println("2. pop");  
 System.*out*.println("3. peek");  
 System.*out*.println("4. check empty");  
 System.*out*.println("5. size");  
 int choice = scan.nextInt();  
 switch (choice)  
 {  
 case 1 :  
 System.*out*.println("Enter integer element to push");  
 ls.push( scan.nextInt() );  
 break;  
 case 2 :  
 try  
 {  
 System.*out*.println("Popped Element = "+ ls.pop());  
 }  
 catch (Exception e)  
 {  
 System.*out*.println("Error : " + e.getMessage());  
 }  
 break;  
 case 3 :  
 try  
 {  
 System.*out*.println("Peek Element = "+ ls.peek());  
 }  
 catch (Exception e)  
 {  
 System.*out*.println("Error : " + e.getMessage());  
 }  
 break;  
 case 4 :  
 System.*out*.println("Empty status = "+ ls.isEmpty());  
 break;  
 case 5 :  
 System.*out*.println("Size = "+ ls.getSize());  
 break;  
 case 6 :  
 System.*out*.println("Stack = ");  
 ls.display();  
 break;  
 default :  
 System.*out*.println("Wrong Entry \n ");  
 break;  
 }  
 */\* display stack \*/* ls.display();  
 System.*out*.println("\nDo you want to continue (Type y or n) \n");  
 ch = scan.next().charAt(0);  
  
 } while (ch == 'Y'|| ch == 'y');  
 }  
}



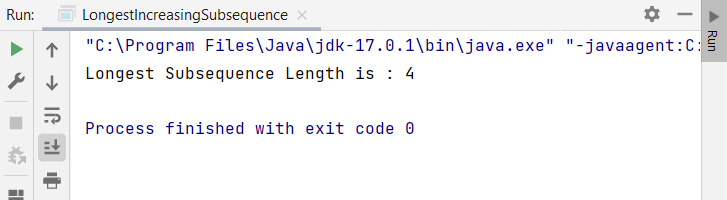
**30 Program to demonstrate working of queue.**

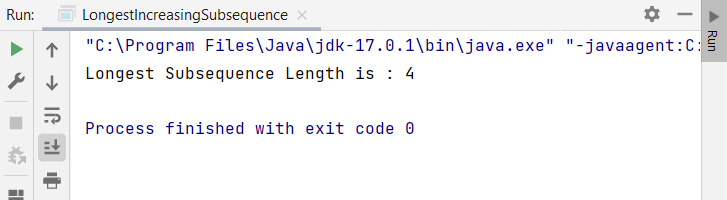
package com.Shivangani;  
import java.util.\*;  
  
class arrayQueue  
{  
 protected int Queue[] ;  
 protected int front, rear, size, len;  
  
 */\* Constructor \*/* public arrayQueue(int n)  
 {  
 size = n;  
 len = 0;  
 Queue = new int[size];  
 front = -1;  
 rear = -1;  
 }  
 */\* Function to check if queue is empty \*/* public boolean isEmpty()  
 {  
 return front == -1;  
 }  
 */\* Function to check if queue is full \*/* public boolean isFull()  
 {  
 return front==0 && rear == size -1 ;  
 }  
 */\* Function to get the size of the queue \*/* public int getSize()  
 {  
 return len ;  
 }  
 */\* Function to check the front element of the queue \*/* public int peek()  
 {  
 if (isEmpty())  
 throw new NoSuchElementException("Underflow Exception");  
 return Queue[front];  
 }  
 */\* Function to insert an element to the queue \*/* public void insert(int i)  
 {  
 if (rear == -1)  
 {  
 front = 0;  
 rear = 0;  
 Queue[rear] = i;  
 }  
 else if (rear + 1 >= size)  
 throw new IndexOutOfBoundsException("Overflow Exception");  
 else if ( rear + 1 < size)  
 Queue[++rear] = i;  
 len++ ;  
 }  
 */\* Function to remove front element from the queue \*/* public int remove()  
 {  
 if (isEmpty())  
 throw new NoSuchElementException("Underflow Exception");  
 else  
 {  
 len-- ;  
 int ele = Queue[front];  
 if ( front == rear)  
 {  
 front = -1;  
 rear = -1;  
 }  
 else  
 front++;  
 return ele;  
 }  
 }  
 */\* Function to display the status of the queue \*/* public void display()  
 {  
 System.*out*.print("\nQueue = ");  
 if (len == 0)  
 {  
 System.*out*.print("Empty\n");  
 return ;  
 }  
 for (int i = front; i <= rear; i++)  
 System.*out*.print(Queue[i]+" ");  
 System.*out*.println();  
 }  
}  
  
*/\* Class QueueImplement \*/*public class Queue  
{  
 public static void main(String[] args)  
 {  
 Scanner scan = new Scanner(System.*in*);  
  
 System.*out*.println("Array Queue Test\n");  
 System.*out*.println("Enter Size of Integer Queue ");  
 int n = scan.nextInt();  
 */\* creating object of class arrayQueue \*/* arrayQueue q = new arrayQueue(n);  
 */\* Perform Queue Operations \*/* char ch;  
 do{  
 System.*out*.println("\nQueue Operations");  
 System.*out*.println("1. insert");  
 System.*out*.println("2. remove");  
 System.*out*.println("3. peek");  
 System.*out*.println("4. check empty");  
 System.*out*.println("5. check full");  
 System.*out*.println("6. size");  
 int choice = scan.nextInt();  
 switch (choice)  
 {  
 case 1 :  
 System.*out*.println("Enter integer element to insert");  
 try  
 {  
 q.insert( scan.nextInt() );  
 }  
 catch(Exception e)  
 {  
 System.*out*.println("Error : " +e.getMessage());  
 }  
 break;  
 case 2 :  
 try  
 {  
 System.*out*.println("Removed Element = "+q.remove());  
 }  
 catch(Exception e)  
 {  
 System.*out*.println("Error : " +e.getMessage());  
 }  
 break;  
 case 3 :  
 try  
 {  
 System.*out*.println("Peek Element = "+q.peek());  
 }  
 catch(Exception e)  
 {  
 System.*out*.println("Error : "+e.getMessage());  
 }  
 break;  
 case 4 :  
 System.*out*.println("Empty status = "+q.isEmpty());  
 break;  
 case 5 :  
 System.*out*.println("Full status = "+q.isFull());  
 break;  
 case 6 :  
 System.*out*.println("Size = "+ q.getSize());  
 break;  
 default : System.*out*.println("Wrong Entry \n ");  
 break;  
 }  
 */\* display Queue \*/* q.display();  
 System.*out*.println("\nDo you want to continue (Type y or n) \n");  
 ch = scan.next().charAt(0);  
  
 } while (ch == 'Y'|| ch == 'y');  
 }  
}



**31. Longest Increasing Subsequence**

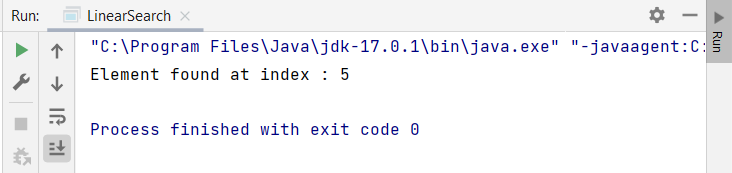
public class LongestIncreasingSubsequence {  
   
 static int max\_start;  
   
 static int lis(int arr[], int a) {  
 if (a == 1)  
 return 1;  
   
 int res, max\_end = 1;  
   
 for (int i = 1; i < a; i++) {  
 res = lis(arr, i);  
 if (arr[i - 1] < arr[a - 1]  
 && res + 1 > max\_end)  
 max\_end = res + 1;  
 }  
   
 if (max\_start < max\_end)  
 max\_start = max\_end;  
   
 return max\_end;  
 }  
   
 static int lis1(int arr[], int n) {  
 max\_start = 1;  
 lis(arr, n);  
 return max\_start;  
  
 }  
  
 public static void main(String args[])  
 {  
 int arr[] = { 10, 22, 9, 33, 21, 50, 41, 60 };  
 int n = arr.length;  
 System.out.println("Length of lis is " + lis(arr, n)  
 + "\n");  
 }  
}

****



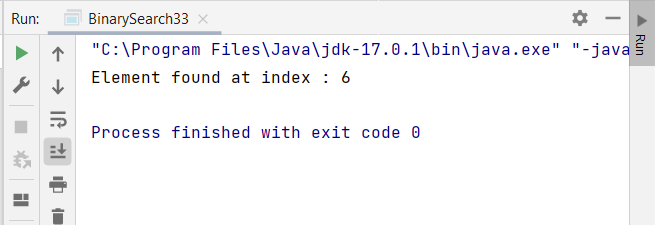
**32. Linear Search**

package com.Shivangani;  
  
public class LinearSearch {  
 static int linearSearch(int[] arr, int target) {  
 if(arr.length == 0) {  
 return -1;  
 }  
  
 for(int i = 0; i < arr.length; i++) {  
 int element = arr[i];  
 if(element == target) {  
 return i;  
 }  
 }  
 return - 1;  
 }  
  
 public static void main(String [] args) {  
 int[] nums = {23, 45, 1, 2, 8, 19, -3, 16, -11, 28};  
 int target = 19;  
 int ans = *linearSearch*(nums, target);  
 System.*out*.println("Element found at index : " + ans);  
 }  
}



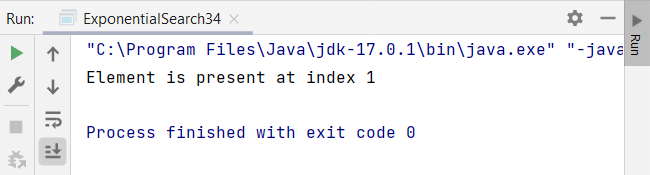
**33. Binary Search**

package com.shivangani;  
  
public class BinarySearch33 {  
 static int binarySearch(int[] arr, int target) {  
 int start = 0;  
 int end = arr.length - 1;  
  
 while(start <= end) {  
 int mid = start + (end - start) / 2;  
  
 if(target < arr[mid]) {  
 end = mid - 1;  
 }  
 else if(target > arr[mid]) {  
 start = mid + 1;  
 }  
 else {  
 return mid;  
 }  
 }  
 return -1;  
 }  
  
 public static void main(String [] args) {  
 int[] arr = {2, 3, 4, 15, 16, 18, 22, 45};  
 int target = 22;  
 int ans = *binarySearch*(arr, target);  
 System.*out*.println("Element found at index : " + ans);  
 }  
}



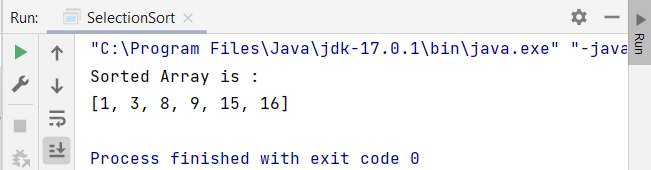
**34. Exponential Search**

package com.shivangani;  
  
import java.util.\*;  
  
public class ExponentialSearch34 {  
 static int exponentialSearch(int arr[], int n, int x) {  
  
 if (arr[0] == x)  
 return 0;  
  
 int i = 1;  
 while (i < n && arr[i] <= x)  
 i = i\*2;  
  
 return Arrays.*binarySearch*(arr, i/2,  
 Math.*min*(i, n-1), x);  
 }  
  
  
 public static void main(String args[]) {  
 int arr[] = {2, 3, 4, 10, 40};  
 int x = 3;  
 int result = *exponentialSearch*(arr,  
 arr.length, x);  
  
 System.*out*.println((result < 0) ?  
 "Element is not present in array" :  
 "Element is present at index " +  
 result);  
 }  
}



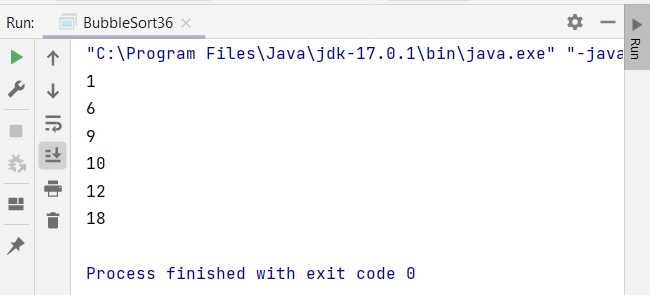
**35. Selection Sort**

package com.shivangani;  
import java.util.\*;  
public class SelectionSort {  
 public static void selectionsort(int[] arr) {  
 int size = arr.length;  
  
 for(int i = 0; i < size-1; i++) {  
 int min = i;  
 for(int j = i+1; j < size; j++) {  
 if(arr[j] < arr[min]) {  
 min = j;  
 }  
 }  
 int temp = arr[i];  
 arr[i] = arr[min];  
 arr[min] = temp;  
 }  
 }  
  
 public static void main(String [] args) {  
 int[] arr = {15, 9, 16, 1, 8, 3};  
 SelectionSort ss = new SelectionSort();  
 ss.*selectionsort*(arr);  
 System.*out*.println("Sorted Array is : ");  
 System.*out*.println(Arrays.*toString*(arr));  
 }  
}



**36. Bubble Sort**

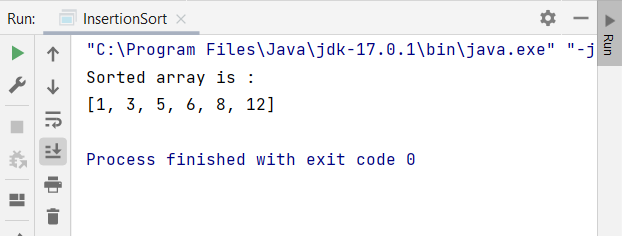
package com.shivangani;  
  
public class BubbleSort36 {  
 public static void bubblesort(int [] arr) {  
 int len = arr.length;  
 int temp = 0;  
  
 for(int i = 0; i <= len; i++) {  
 for(int j = 0; j < len-i-1; j++) {  
 if(arr[j] > arr[j+1]) {  
 temp = arr[j];  
 arr[j] = arr[j+1];  
 arr[j+1] = temp;  
 }  
 }  
 }  
 }  
  
 public static void main(String [] args) {  
 int [] arr = {10, 6, 12, 18, 9, 1 };  
 *bubblesort*(arr);  
 for(int i = 0; i < arr.length; i++) {  
 System.*out*.println(arr[i]);  
 }  
 }  
}



**37. Insertion Sort**

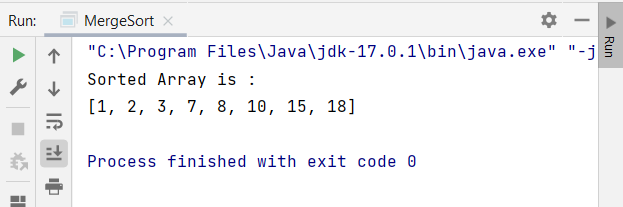
package com.Shivangani;  
  
  
 import java.util.Arrays;  
  
 public class InsertionSort {  
 public static void insertionsort(int[] arr) {  
 int size = arr.length;  
  
 for(int i = 1; i <= size-1; i++) {  
 int key = arr[i];  
 int j = i-1;  
 while(j >= 0 && arr[j] > key) {  
 arr[j+1] = arr[j];  
 j--;  
 }  
 arr[j+1] = key;  
 }  
 }  
  
 public static void main(String [] args) {  
 int[] arr = {5, 6, 12, 3, 1, 8};  
 InsertionSort is = new InsertionSort();  
 is.*insertionsort*(arr);  
 System.*out*.println("Sorted array is : ");  
 System.*out*.println(Arrays.*toString*(arr));  
 }

}



**38. Merge Sort**

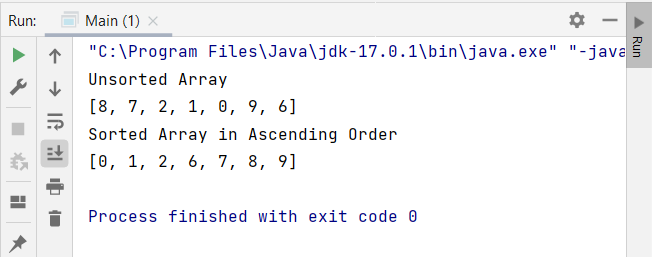
package com.Shivangani;  
import java.util.\*;  
public class MergeSort {  
 void merge(int[] arr, int top, int bottom, int mid) {  
 int a1 = bottom - top + 1;  
 int a2 = mid - bottom;  
  
 int b[] = new int[a1];  
 int c[] = new int[a2];  
  
 for(int i =0; i < a1; i++)  
 b[i] = arr[top + i];  
 for(int j = 0; j < a2; j++)  
 c[j] = arr[bottom + 1 + j];  
  
 int i, j, k;  
 i = 0;  
 j = 0;  
 k = top;  
  
 while(i < a1 && j < a2) {  
 if(b[i] <= c[j]) {  
 arr[k] = b[i];  
 i++;  
 }  
 else {  
 arr[k] = c[j];  
 j++;  
 }  
 k++;  
 }  
  
 while(i < a1) {  
 arr[k] = b[i];  
 i++;  
 k++;  
 }  
  
 while (j < a2) {  
 arr[k] = c[j];  
 j++;  
 k++;  
 }  
 }  
  
 void mergeSort(int[] arr, int left, int right) {  
 if(left < right) {  
 int mid = (left + right) / 2;  
 mergeSort(arr, left, mid);  
 mergeSort(arr, mid + 1, right);  
  
 merge(arr, left, mid, right);  
 }  
 }  
  
 public static void main(String [] args) {  
 int[] arr = {1, 10, 15, 18, 2, 3, 7, 8};  
  
 MergeSort ob = new MergeSort();  
 ob.mergeSort(arr, 0, arr.length - 1);  
  
 System.*out*.println("Sorted Array is : ");  
 System.*out*.println(Arrays.*toString*(arr));  
 }  
}



**39. Quick Sort**

package com.shivangani;  
import java.util.\*;  
  
import static com.shivangani.QuickSort.\*;  
  
public class QuickSort {  
 static int partition(int array[], int low, int high) {  
  
 *// choose the rightmost element as pivot* int pivot = array[high];  
  
 *// pointer for greater element* int i = (low - 1);  
  
 *// traverse through all elements  
 // compare each element with pivot* for (int j = low; j < high; j++) {  
 if (array[j] <= pivot) {  
  
 *// if element smaller than pivot is found  
 // swap it with the greater element pointed by i* i++;  
  
 *// swapping element at i with element at j* int temp = array[i];  
 array[i] = array[j];  
 array[j] = temp;  
 }  
  
 }

*// swapt the pivot element with the greater element specified by i* int temp = array[i + 1];  
 array[i + 1] = array[high];  
 array[high] = temp;  
  
 *// return the position from where partition is done* return (i + 1);  
 }  
  
 static void quickSort(int array[], int low, int high) {  
 if (low < high) {  
  
 *// find pivot element such that  
 // elements smaller than pivot are on the left  
 // elements greater than pivot are on the right* int pi = *partition*(array, low, high);  
  
 *// recursive call on the left of pivot  
 quickSort*(array, low, pi - 1);  
  
 *// recursive call on the right of pivot  
 quickSort*(array, pi + 1, high);  
 }  
 }  
}  
  
*// Main class*class Main {  
 public static void main(String args[]) {  
  
 int[] data = { 8, 7, 2, 1, 0, 9, 6 };  
 System.*out*.println("Unsorted Array");  
 System.*out*.println(Arrays.*toString*(data));  
  
 int size = data.length;  
  
 *// call quicksort() on array data  
 quickSort*(data, 0, size - 1);  
  
 System.*out*.println("Sorted Array in Ascending Order ");  
 System.*out*.println(Arrays.*toString*(data));  
 }  
}



**40. Fix the bugs of the application using appropriate algorithm.**

import java.util.ArrayList;  
import java.util.Collections;  
import java.util.Scanner;  
  
  
public class BugFixCode {  
   
 public static void main(String[] args) {  
   
 System.out.println("\tWelcome to Sorting and Searching World \n");  
 choiceSelection();  
  
 }  
   
 private static void choiceSelection() {  
 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(1000);  
 expenses.add(100);  
 expenses.add(5000);  
 expenses.add(500000);  
 expenses.add(1245);  
 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");  
 choiceSelection();  
 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");  
 choiceSelection();  
 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!");  
 }  
 choiceSelection();  
 break;  
   
 case 4:  
 sortExpenses(expenses);  
 choiceSelection();  
 break;  
   
 case 5:  
 searchExpenses(expenses);  
 choiceSelection();  
 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();  
 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();  
 Collections.sort(arrayList);  
 System.out.println("Sorted expenses: ");  
 for(Integer i: arrayList) {  
 System.out.print(i + " ");  
 }  
   
 System.out.println("\n");  
 }  
}

