**1. Loops**

**a) Fibonacci**

**Ans :**

**public** **class** Fibonacci {

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

{

**int** n=0;

**int** n1=1;

**int** n2;

**int** i, count =15;

**for**(i=2;i<count;++i) //

{

n2=n+n1;

n=n1;

n1=n2;

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

//System.out.print(n1+" ");

//System.out.println(n2+" ");

}

}

}

b)Ans :

**public** **class** Pingpongdivisble {

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

{

// Test Data 6, 10, 17, 60

**int** number=60;

**if**(number%3==0 && number%5==0)

{

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

}

**else**

**if**(number%3==0)

{

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

}

**else**

**if**(number%5==0)

{

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

}

**else**

{

System.***out***.println("number" +number);

}

}

}

C) Ans : -🡪

**public** **class** Swaptwonum {

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

{

**int** n1 = 20, n2=15;

System.***out***.println(" First Number n1 " +n1);

System.***out***.println( " Second Number n2 " +n2);

n1 = n1+n2;

n2= n1-n2;

n1 = n1 -n2;

System.***out***.println(" First Number n1 " +n1);

System.***out***.println( " Second Number n2 " +n2);

}

}

-🡪

**public** **class** Swapusingv {

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

{

**int** n1=10;

**int** n2=20;

System.***out***.println("Before swapping first number is " + n1);

System.***out***.println ("Before swapping second number is" +n2);

*swap*(n1,n2);

}

**private** **static** **void** swap(**int** n1, **int** n2)

{

**int** temp= n1;

n1=n2;

n2 = temp;

System.***out***.println("After swapping first number is " + n1);

System.***out***.println ("After swapping second number is" +n2);

}

}

d)Ans :

**public** **class** Factorial {

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

{

**int** fact =1;

**int** no=5;

**for**(**int** i=1;i<=no;i++)

{

fact=fact\*i;

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

}

}

}

e) Ans :

**public** **class** Prime {

**public** **boolean** isPrimeNumber(**int** number){

**for**(**int** i=2; i<=number/2; i++){

**if**(number % i == 0){

**return** **false**;

}

}

**return** **true**;

}

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

Prime p = **new** Prime();

System.***out***.println("Is 11 prime number? "+p.isPrimeNumber(11));

System.***out***.println("Is 13 prime number? "+p.isPrimeNumber(13));

System.***out***.println("Is 19 prime number? "+p.isPrimeNumber(19));

System.***out***.println("Is 15 prime number? "+p.isPrimeNumber(15));

}

}

F)Ans :

**public** **class** Armstrongnum {

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

{

**int** num = 153;

String numAsStr = **new** Integer(num).toString();

**double** convertToNum = 0 ;

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

{

System.***out***.println("value of length " + numAsStr.length());

System.***out***.println("value of numAsStr.charAt(i) " + i + " " + Integer.*parseInt*( "" + numAsStr.charAt(i)));

System.***out***.println("value of Math.pow( numAsStr.charAt(i) ,numAsStr.length() " + Math.*pow*( Integer.*parseInt*( "" + numAsStr.charAt(i)) ,numAsStr.length()));

convertToNum = convertToNum + Math.*pow*( Integer.*parseInt*( "" + numAsStr.charAt(i)) ,numAsStr.length() );

System.***out***.println(" Inside loop value of convertToNum " + convertToNum);

}

System.***out***.println("value of convertToNum " + convertToNum);

}

}

g) Ans :

**import** java.util.Scanner;

**public** **class** Cmdline{

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

{

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

System.***out***.print("Enter first number: ");

**int** firstNumber = s.nextInt();

System.***out***.print("Enter second number: ");

**int** secondNumber = s.nextInt();

**int** sum = firstNumber + secondNumber;

System.***out***.println("The result of addition was " + sum);

}

}

h)Ans :

**public** **class** Pattern {

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

{

**for**(**int** i=1; i<7; i++) //

{

**for**(**int** j=1; j<=i; j++) //

{

System.***out***.print("\*");

}

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

}

}

}

2)Arrays :

a) Ans :

**public** **class** Largestarray {

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

//array of 10 numbers

**int** numbers[] = **new** **int**[]{32,43,53,54,32,65,63,98,43,23};

//assign first element of an array to largest and smallest

**int** smallest = numbers[0];

**int** largest = numbers[0];

**for**(**int** i=1; i< numbers.length; i++)

{

**if**(numbers[i] > largest)

largest = numbers[i];

**else** **if** (numbers[i] < smallest)

smallest = numbers[i];

}

System.***out***.println("Largest Number is : " + largest);

System.***out***.println("Smallest Number is : " + smallest);

}

}

b)Ans :

**public** **class** MissingNumberArray {

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

**int**[] a1= {1,2,3,5,6,7};

**int** sumwith\_missno=0;

**for**(**int** i=0;i<a1.length;i++)

{

sumwith\_missno = sumwith\_missno + a1[i];

}

**int** length = a1.length +1;

**int** sumwithout\_missno = (length \* (length+1)) / 2;

**int** missno = sumwithout\_missno - sumwith\_missno;

System.***out***.println("Missing no is "+missno);

}

}

c) Ans :

**public** **class** Commmonnumber {

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

{

**int** a[]={10,25,43,67,98,12,32,66};

**int** b[]={20,35,63,47,98,72,92,16};

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

{

**for**(**int** j=0; j<b.length; j++)

{

**if** (a[i]==b[j])

{

System.***out***.println("The common number is: " + a[i]);

}

}

}

}

}

d) and e) and f) and g)

Answers :

**public** **class** LinearSearch {

**public** **boolean** doSearch(**int**[] input , **int** searchInt)

{

**boolean** flag = **false**;

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

{

**if**(input[i] == searchInt)

{

flag = **true**;

**break**;

}

}

**return** flag;

}

**public** **int**[] doMerge(**int**[] inputOne , **int**[] inputTwo)

{

**int**[] finalOutput = **new** **int**[inputOne.length + inputTwo.length];

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

{

finalOutput[i] = inputOne[i];

}

**for**(**int** j=0 ; j < inputTwo.length ; j++)

{

finalOutput[j+inputOne.length] = inputTwo[j];

}

**return** doSort(finalOutput);

}

**public** **int**[] doSort(**int**[] input)

{

**for**(**int** i=0;i<= input.length-2; i++)

{

**for**(**int** j=i+1 ; j <= input.length-1 ; j++)

{

**if**(input[i] > input[j])

{

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

input[i]= input[j];

input[j] = temp;

}

}

}

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

{

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

}

**return** input;

}

**public** **int**[] locateSwap(**int**[] input)

{

**int** firstPosition = 0;

**int** secondPosition = 0;

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

{

**if**(input[i] > input[i+1])

{

firstPosition = i;

**break**;

}

}

**for**(**int** j=input.length-1 ; j >= 1 ; j--)

{

**if**(input[j] < input[j-1])

{

secondPosition = j;

**break**;

}

}

**int** temp = input[firstPosition];

input[firstPosition] = input[secondPosition];

input[secondPosition] = temp;

System.***out***.println(" First Position is " + firstPosition);

System.***out***.println(" Second Position is " + secondPosition );

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

{

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

}

**return** input;

}

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

{

**int** inputElement[] = {1, 23,56, 45,22, 98,16, 48 ,59};

LinearSearch ls = **new** LinearSearch();

System.out.println(" The result of the search1 for element 22: is " + ls.doSearch(inputElement, 22));

System.out.println(" The result of the search1 for element 100: is " + ls.doSearch(inputElement, 100));

System.out.println(" The result of sorted array: is ");

ls.doSort(inputElement);

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

System.out.println(" The result of inputlocateSwap array: is ");

ls.locateSwap(inputlocateSwap);

**int** inputMerge1[] = {58, 12 , 6, 2, 19};

**int** inputMerge2[] = {32, 14 , 9, 87, 24};

ls.doMerge(inputMerge1,inputMerge2);

}

}

3.Strings

a) , b ) , c) , d), e) ,f) , g)

Ans :

**import** java.util.StringTokenizer;

**public** **class** StringComputation {

// To Reverse a String

**public** **static** String reverseString(String inputString)

{

**int** len = inputString.length();

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

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

{

array[len-1-i] = inputString.charAt(i);

}

**return** **new** String(array);

}

// c. This accepts a string like "This is nice" and converts it to a string like "This1 is2 nice3".

**public** String suffixNum(String inputString)

{

StringTokenizer stkn = **new** StringTokenizer(inputString);

StringBuffer outputString = **new** StringBuffer(10);

**int** i = 1;

**while** (stkn.hasMoreElements())

{

outputString.append( stkn.nextToken() + i + " ");

i++;

}

**return** outputString.toString();

}

// To check for a Palindrome

**public** **boolean** palindrome(String inputString)

{

**return** inputString.equals(*reverseString*(inputString)) ? **true** : **false**;

}

**public** String reverseWord(String inputString)

{

StringTokenizer stkn = **new** StringTokenizer(inputString);

StringBuffer outputString = **new** StringBuffer(10);

**while** (stkn.hasMoreElements())

{

outputString.append( *reverseString*(stkn.nextToken()) + " ");

}

**return** outputString.toString();

}

**public** String replaceSubStr(String inputString, String replaceSource, String replaceTarget)

{

String output;

output = inputString.replaceAll(replaceSource, replaceTarget);

**return** output;

}

//hashcode

**public** **int** myHashCode(String inputString)

{

**return** inputString.hashCode();

}

//equals

**public** **boolean** myEquals(String inputString1, String inputString2)

{

**return** inputString1.hashCode() == inputString2.hashCode() ? **true** : **false** ;

}

**public** **boolean** balancedParanthesis(String inputString)

{

**int** prevLen = inputString.length();

**int** curLen = 100;

**while**(prevLen !=curLen && curLen > 1)

{

prevLen = inputString.length();

inputString= inputString.replace("()", "");

// System.out.println("value of inputString is "+inputString );

curLen = inputString.length();

// System.out.println("value of curLen inside while is "+curLen );

}

//System.out.println("value of curLen outside is "+curLen );

**if**(curLen == 0)

{

**return** **true**;

}

**else**

{

**return** **false**;

}

}

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

{

StringComputation sc = **new** StringComputation();

System.***out***.println(*reverseString*("All the best for your future"));

System.***out***.println(sc.palindrome("Malayalam"));

System.***out***.println(sc.suffixNum("This is line"));

System.***out***.println(sc.replaceSubStr("This is line" , "line", "Circle"));

System.***out***.println(sc.reverseWord("This is line and circle"));

System.***out***.println(" Example 1: input String:(()) Output is:" + sc.balancedParanthesis("(())"));

System.***out***.println(" Example 2: input String:)( Output is:" + sc.balancedParanthesis(")("));

System.***out***.println(" Example 3: input String:(())) Output is:" + sc.balancedParanthesis("(()))"));

System.***out***.println("myHashCode Example 1: input String:All the best for your future Output is:" + sc.myHashCode("All the best for your future"));

System.***out***.println("myEquals Example 1: input String1 :All the best for your future\n" +

"input String2 :This is String2\n" +

" Output is:" + sc.myEquals("All the best for your future" , "This is String2"));

System.***out***.println("myEquals Example 2: input String1 :This is String2\n" +

"input String2 :This is String2\n" +

" Output is:" + sc.myEquals("This is String2" , "This is String2"));

}

}

4. File Handling :

a) i) Ans :

**import** java.util.Scanner;

**public** **class** Filehandling{

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

{

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

System.***out***.print("Enter first number: ");

**int** firstNumber = s.nextInt();

System.***out***.print("Enter second number: ");

**int** secondNumber = s.nextInt();

**int** sum = firstNumber + secondNumber;

System.***out***.println("The result of addition was " + sum);

}

}

a) ii)

iii )

iv)

Ans :

**public** **double** bufferedInputStreamReader() {

**double** result=0;

**try** {

InputStreamReader isr = **new** InputStreamReader(System.***in***);

BufferedReader br = **new** BufferedReader(isr);

System.***out***.print("Enter Number 1 ");

String str = br.readLine();

**double** Input1=0;

**try**

{

Input1 = Double.*valueOf*(str).doubleValue();

} **catch** (NumberFormatException nfe) {

System.***out***.println("Incorrect format!");

}

**if** (Input1 <= 0)

{

System.***out***.println("Radius must be positive!");

}

System.***out***.print("Enter Number 2 ");

str = br.readLine();

**double** Input2=0;

**try**

{

Input2 = Double.*valueOf*(str).doubleValue();

} **catch** (NumberFormatException nfe)

{

System.***out***.println("Incorrect format!");

}

**if** (Input2 <= 0)

{

System.***out***.println("Radius must be positive!");

}

result = Input1 - Input2;

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

} **catch** (Exception e)

{

e.printStackTrace();

}

**return** result;

}

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

{

InputOperations io = **new** InputOperations();

io.bufferedInputStreamReader();

io.multiplyDataInputStream();

io.consoleDivide();

}

}

b) and c) and d ) and e) and f )

Ans :

**public** **class** FileOperations {

**public** **int** countWords(String fileLocation)

{

File file = **new** File(fileLocation);

Scanner sc = **null**;

**try** {

sc = **new** Scanner(**new** FileInputStream(file));

} **catch** (FileNotFoundException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

**int** count=0;

**while**(sc.hasNext()){

sc.next();

count++;

}

System.***out***.println("Number of words: " + count);

**return** count;

}

**public** **int** wordOccurrence(String fileLocation, String word)

{

File file = **new** File(fileLocation);

**int** count = 0;

Scanner scanner;

**try** {

scanner = **new** Scanner(file);

**while** (scanner.hasNextLine()) {

String nextToken = scanner.next();

**if** (nextToken.equalsIgnoreCase(word))

count++;

}

} **catch** (FileNotFoundException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

System.***out***.println("Number of words: " + count);

**return** count;

}

**public** **void** replaceString(String fileLocation, String search, String replace)

{

File log= **new** File(fileLocation);

//String search = "number";

//String replace = "figure";

**try**{

FileReader fr = **new** FileReader(log);

String s;

String totalStr = "";

**try** (BufferedReader br = **new** BufferedReader(fr)) {

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

totalStr += s;

}

totalStr = totalStr.replaceAll(search, replace);

FileWriter fw = **new** FileWriter(log);

fw.write(totalStr);

fw.close();

}

}**catch**(Exception e){

e.printStackTrace();

}

}

**public** **void** copyFile(String sourceFile, String destFile)

{

File source= **new** File(sourceFile);

File dest= **new** File(destFile);

String search = "number";

//String replace = "figure";

**try**{

FileReader fr = **new** FileReader(source);

String s;

String totalStr = "";

**try** (BufferedReader br = **new** BufferedReader(fr)) {

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

totalStr += s;

}

FileWriter fw = **new** FileWriter(dest);

fw.write(totalStr);

fw.close();

}

}**catch**(Exception e){

e.printStackTrace();

}

}

**public** File[] listOfFilesinFolder(String folderpath)

{

File folder = **new** File(folderpath);

File[] listOfFiles = folder.listFiles();

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

**if** (listOfFiles[i].isFile()) {

System.***out***.println("File " + listOfFiles[i].getName());

} **else** **if** (listOfFiles[i].isDirectory()) {

System.***out***.println("Directory " + listOfFiles[i].getName());

}

}

**return** listOfFiles;

}

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

{

String fileLocation = "C:\\Users\\Anitha\\Desktop\\Files\\CountWords.txt";

String folderloc = "C:\\Users\\Anitha\\Desktop\\Files";

String destfileLocation = "C:\\Users\\Anitha\\Desktop\\Files\\DupCountWords.txt";

FileOperations fo = **new** FileOperations();

fo.countWords(fileLocation);

fo.wordOccurrence(fileLocation,"the");

fo.replaceString(fileLocation, "number","figure");

File[] fileList = fo.listOfFilesinFolder(folderloc);

fo.copyFile(fileLocation,destfileLocation);

}

}

5) Collections :

a) ,b), c) ,d) ,e) , f) ,g)

Ans :

**import** java.util.ArrayList;

**import** java.util.Arrays;

**import** java.util.Comparator;

**import** java.util.HashMap;

**import** java.util.Iterator;

**import** java.util.List;

**import** java.util.ListIterator;

**import** java.util.Map;

**import** java.util.Map.Entry;

**import** java.util.Set;

**import** java.util.Collections;

**public** **class** CollectionOperations {

ArrayList<String> myArr = **null**;

Map<String,String> myMap =**null**;

**public** **void** populateArrayList()

{

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

myArr.add("Andhra Pradesh");

myArr.add("Telangana");

myArr.add("Tamil Nadu");

myArr.add("Kerala");

myArr.add("Karnataka");

}

**public** **boolean** checkArrayList(String key)

{

**return** myArr.contains(key) ;

}

**public** **void** printArrayList()

{

Iterator<String> iterator = myArr.iterator();

**while** (iterator.hasNext()) {

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

}

}

**public** **void** printReverseArrayList()

{

ListIterator<String> listIterator = myArr.listIterator();

**while** (listIterator.hasPrevious())

{

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

}

}

**public** **void** sortSeries()

{

String[] series = **new** String[] { "a1","a2","a3","a4","b1","b2","b3","b4"};

Arrays.*sort*(series);

**int** i=0;

**for**(String temp:series){

System.***out***.println("Series " + i + " : " + temp);

i++;

}

}

**public** **void** createHashMap()

{

myMap = **new** HashMap<String,String>();

myMap.put("key1", "value1");

myMap.put("key2", "value2");

myMap.put("key3", "value3");

myMap.put("key4", "value4");

myMap.put("key5", "value5");

}

**public** String getValueHashMap(String MapKey)

{

**return** myMap.get(MapKey);

}

**public** **void** sortMapByValue()

{

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

map.put("Andhra Pradesh", 20);

map.put("Telangana", 45);

map.put("Kerala", 2);

map.put("Karnataka", 67);

map.put("Kerala", 26);

map.put("Tamil Nadu", 93);

Set<Entry<String, Integer>> set = map.entrySet();

List<Entry<String, Integer>> list = **new** ArrayList<Entry<String, Integer>>(set);

Collections.*sort*( list, **new** Comparator<Map.Entry<String, Integer>>()

{

**public** **int** compare( Map.Entry<String, Integer> o1, Map.Entry<String, Integer> o2 )

{

**return** (o2.getValue()).compareTo( o1.getValue() );

}

} );

**for**(Map.Entry<String, Integer> entry:list){

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

}

}

**public** **enum** Months\_of\_Year {

***JANUARY***,

***FEBRUARY***,

***MARCH***,

***APRIL***,

***MAY***,

***JUNE***,

***JULY***,

***AUGUST***,

***SEPTEMBER***,

***OCTOBER***,

***NOVEMBER***,

***DECEMBER***

}

**public** **void** printEnumMonths()

{

**for** (Months\_of\_Year month : Months\_of\_Year.*values*()) {

System.***out***.println("Months is: " + month);

}

}

**public** **void** copyArrayList2Array()

{

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

list.add("India");

list.add("Switzerland");

list.add("Italy");

list.add("France");

String[] countries = list.toArray(**new** String[list.size()]);

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

{

System.***out***.println(" country: " + i + " =" +countries[i] );

}

}

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

{

CollectionOperations co = **new** CollectionOperations();

co.populateArrayList();

co.checkArrayList("Telangana");

co.printArrayList();

co.printReverseArrayList();

co.sortSeries();

co.createHashMap();

co.getValueHashMap("key2");

co.sortMapByValue();

co.printEnumMonths();

co.copyArrayList2Array();

}

}

6) OOPS :

a)Ans :

**public** **abstract** **class** Shape {

**int** noofsides;

**double** area;

**double** perimeter ;

**abstract** **double** calculateArea(**double** k);

**abstract** **double** calculatePerimeter(**double** k);

**public** **void** setSides(**int** noside)

{

**this**.noofsides = noside;

}

}

c) Ans :

**import** java.util.Scanner;

**public** **class** Circle **extends** Shape **implements** ShapeConstants {

**int** noofsides;

**double** radius ;

**double** area;

**double** perimet;

Circle()

{

**super**.setSides(0);

this.noofsides = noside;

}

**double** calculateArea(**double** radius1)

{

**this**.radius = radius1;

area = ***pi*** \* Math.*pow*(**this**.radius, 2);// area = pi \*r^2

**return** area ;

}

**double** calculatePerimeter(**double** radius1)

{

**this**.radius = radius1;

perimet = 2 \* ***pi*** \* **this**.radius;

**return** perimet;

}

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

Circle cr1 = **new** Circle();

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

System.***out***.println("Enter a Radius of circle: ");

**double** rd = sc.nextDouble();

System.***out***.println("Area "+cr1.calculateArea(rd));

System.***out***.println("Perimeter "+cr1.calculatePerimeter(rd));

}

}

b) Ans :

**public** **interface** ShapeConstants {

**final** **double** ***pi*** = 3.14;

}

d) Ans :

//Overloading Methods

**public** **class** Overloading {

**public** **void** add()

{

**int** i=0;

**int** j=6;

**int** k=7;

i=j+k;

System.***out***.println("Addition of two numbers" +i);

}

**public** **void** add(**int** l,**int** m)

{

**int** n= l \* m;

System.***out***.println("multiplication of two numbers:" + n);

}

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

{

Overloading ol= **new** Overloading();

ol.add();

ol.add(3,10);

}

}

//Overriding Methods

**public** **class** Overriding {

**public** **void** add()

{

**int** i=0;

**int** j=4;

**int** k=5;

i= j+ k;

System.***out***.println("Addition of two numbers :"+ i);

}

**public** **void** add(**int** l, **int** m)

{

**int** k=0;

k=l\*m;

System.***out***.println("multiplication of two numbers"+k);

}

}

**public** **class** TestOverriding **extends** Overriding{

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

{

Overriding ovr= **new** Overriding();

ovr.add();

ovr.add(10, 5);

}

}

7) Exception Handling : a) and B) answers :

**public** **class** ErrorHandlingExample {

**public** **void** catchNPE()

{

String str=**null**;

**int** len;

**try**

{

len = str.length();

}

**catch**(NullPointerException e)

{

System.***out***.println("Inside catch block of NullPointerException");

}

**finally**

{

System.***out***.println("Inside finally block of NullPointerException");

}

}

**public** **void** canThrowNPE()

{

String str=**null**;

**int** len = str.length();

}

**public** **void** callCanThrowNPE()

{

**try**

{

canThrowNPE();

}

**catch**(Exception e)

{

System.***out***.println("Inside callCanThrowNPE(). The exception thrown is " + e.toString());

}

}

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

{

ErrorHandlingExample ehe = **new** ErrorHandlingExample();

System.***out***.println("Start: Calling ehe.catchNPE()");

ehe.catchNPE();

System.***out***.println("Done: Calling ehe.catchNPE()");

System.***out***.println("Start: Calling ehe.callCanThrowNPE()");

ehe.callCanThrowNPE();

System.***out***.println("Done: Calling ehe.callCanThrowNPE()");

}

}

8) General :

a) Ans :

**public** **final** **class** ImmutableClassExample {

**private** **int** length;

**private** **int** width;

**private** **int** getlength()

{

**return** length;

}

**private** **int** getwidth()

{

**return** width;

}

}

b) and c) and d)

Answers :

**public** **class** GeneralJavaExample {

**void** printNnumbers(**int** i)

{

**if**(i >0)

{

System.***out***.println(" The number is " + i);

printNnumbers(i-1);

}

}

**void** mygarbageCollect()

{

System.***out***.println("Inside: mygarbageCollect");

System.*gc*();

}

**private** **void** myPrivateMethod()

{

System.***out***.println("Inside: myPrivateMethod");

}

**protected** **void** myProtectedMethod()

{

System.***out***.println("Inside: myProtectedMethod");

}

**public** **void** myPublicMethod()

{

System.***out***.println("Inside: myPublicMethod");

}

**void** myDefaultMethod()

{

System.***out***.println("Inside: myDefaultMethod");

}

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

{

GeneralJavaExample gje = **new** GeneralJavaExample();

System.***out***.println("Start: printNnumbers");

gje.printNnumbers(10);

System.***out***.println("End: printNnumbers");

System.***out***.println("Start: mygarbageCollect");

gje.mygarbageCollect();

System.***out***.println("End: mygarbageCollect");

System.***out***.println("Start: myPrivateMethod");

gje.myPrivateMethod();

System.***out***.println("End: myPrivateMethod");

System.***out***.println("Start: myProtectedMethod");

gje.myProtectedMethod();

System.***out***.println("End: myProtectedMethod");

System.***out***.println("Start: myPublicMethod");

gje.myPublicMethod();

System.***out***.println("End: myPublicMethod");

System.***out***.println("Start: myDefaultMethod");

gje.myDefaultMethod();

System.***out***.println("End: myDefaultMethod");

}

}

e) Ans :

**ublic** **class** TestStatic {

**void** TestStatic()

{

System.***out***.println("Inside TestStatic() constructor ");

}

**static** **void** myStaticmethod()

{

System.***out***.println("Inside myStaticmethod() method ");

}

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

{

TestStatic ts = **new** TestStatic();

ts.*myStaticmethod*();

}

}

f) Ans :

**import** java.sql.Connection;

**import** java.sql.DriverManager;

**import** java.sql.ResultSet;

**import** java.sql.Statement;

**public** **class** JDBCdatabase {

**public** **static** Connection *con*;

**public** **static** Connection getConnection(){

**if** (*con* == **null**){

**try**

{

//Load the jdbc Driver

Class.*forName*("Oracle.Jdbc.OracleDriver");

*con* = DriverManager.*getConnection*("jdbc:oracle:thin:@localhost:1521/orcl","Admin","MyWebSite");

}

**catch**(ClassNotFoundException e)

{

e.printStackTrace();

}

**catch**(SQLException e)

{

e.printStackTrace();

}

}

**return** *con*;

}

**public** **void** getEmpData(){

**try**{

Statement st = *con*.createStatement();

ResultSet rs = st.executeQuery("select emp\_id, Salary from employees where dept\_id in(20,50)");

String query1="update emp set name='Anitha' where id=2";

String query2 = "insert into emp values (1,'Anitha','QA Tester)";

st.executeQuery(query1);

st.executeQuery(query2);

**int** count=0;

**while**(rs.next()){

count++;

**int** emp\_id= rs.getInt("Emp\_id");

**double** salary=rs.getDouble("salary");

System.***out***.println("Emp\_id"+emp\_id+ "Salary"+salary);

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

**int** id = rs.getInt("id");

String name = rs.getString("name");

String job = rs.getString("job");

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

}

rs.Close();

System.***out***.println("Total number of records"+count);

}

**catch**(SQLException e)

{

e.printStackTrace();

}

}

}