1. Loops: Below is the list of the programs that enables you to enhance your knowledge about loops(for/while) and conditional statements(if-else/switch).
2. Write a program that prints a fibonaci series that is a sequence of numbers like 0 1 1 2 3 5 8.You can vary the number of elements to be printed meaning you can print 10 numbers or 15 or 20 or any desired number.

**import** java.util.Scanner;

**class** FibonacciSeries{

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

{

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

System.***out***.println("Enter a number: ");

**int** count = a.nextInt();

**int** n1=0,n2=1,n3,i;

System.***out***.print(n1+" "+n2);

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

n3=n1+n2;

System.***out***.print(" "+n3);

n1=n2;

n2=n3;

}

}

}

1. Write a program print "ping" if a number is divisible by 3,"pong" if a number is divisible by 5, and "ping pong" if number is divisible by both, else print the number.

**import** java.util.Scanner;

**public** **class** CheckDisibility {

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

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

System.***out***.println("Enter a number: ");

**int** i = a.nextInt();

**if** ((i % 3 == 0) && (i % 5 != 0)) {

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

}

**else** **if** ((i % 3 != 0) && (i % 5 == 0)){

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

}

**if** ((i % 3 == 0) && (i % 5== 0)){

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

}

**else** **if** ((i % 3 != 0) && (i % 5 != 0)){

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

}

}

}

1. Write a program that swaps 2 given numbers. You need to have 2 separate functions in the program.
   1. One Function should swap the numbers without any third new variable.

**import** java.util.Scanner;

**public** **class** SwapNumbersWithoutTempVar {

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

{

**int** x, y;

System.***out***.println("Enter x and y");

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

x = enter.nextInt();

y = enter.nextInt();

System.***out***.println("Before Swapping x = "+x+" y = "+y);

x = x + y;

y = x - y;

x = x - y;

System.***out***.println("After Swapping x = "+x+" y = "+y);

}

}

--> Second function should swap the numbers using a third variable.

**import** java.util.Scanner;

**public** **class** SwapNumbersWithTempVar {

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

{

**int** x, y, temp;

System.***out***.println("Enter x and y");

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

x = enter.nextInt();

y = enter.nextInt();

System.***out***.println("Before Swapping x = "+x+" y = "+y);

temp = x;

x = y;

y = temp;

System.***out***.println("After Swapping x = "+x+" y = "+y);

}

}

1. Write a program to calculate factorial of a given number. You need to compute the factorial with one logic that uses recursion and another logic without recursion.

**import** java.util.Scanner;

**class** FactorialEx{

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

**int** i,fact=1;

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

System.***out***.println("Enter a number: ");

**int** number = a.nextInt();

**for**(i=1;i<=number;i++){

fact=fact\*i;

}

System.***out***.println("Factorial of "+number+" is: "+fact);

}

}

Using Recursion

**import** java.util.Scanner;

**class** FactorialExWithRecursion{

**static** **int** factorial(**int** n){

**if** (n == 0)

**return** 1;

**else**

**return**(n \* *factorial*(n-1));

}

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

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

System.***out***.println("Enter a number: ");

**int** number = a.nextInt();

**int** fact = 1;

fact = *factorial*(number);

System.***out***.println("Factorial of "+number+" is: "+fact);

}

}

1. Write a program to check if a given number is prime or not.

**import** java.util.Scanner;

**public** **class** PrimeCh

{

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

{

**int** temp;

**boolean** isPrime=**true**;

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

System.***out***.println("Enter a number: ");

**int** num = a.nextInt();

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

{

temp=num%i;

**if**(temp==0)

{

isPrime=**false**;

**break**;

}

}

**if**(isPrime)

System.***out***.println(num + " is Prime Number");

**else**

System.***out***.println(num + " is not Prime Number");

}

}

1. Write a program that accepts input from the command line and then prints them.

**import** java.util.Scanner;

**public** **class** PrintCommandLineInput {

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

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

System.***out***.println("Enter a number: ");

**int** n = a.nextInt();

**int** sum = 0;

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

sum += i;

}

System.***out***.println("The total sum is: " + sum );

}

}

1. Write a program to check an armstrong number i.e. whether the if we power up the each individual number to the total number of digits in the number and add them it should be equal to the number itself. For example 153 has 3 digits in it and if we do 1^3+5^3+3^3=153.

FYI: ^ means is to the power.

**import** java.util.Scanner;

**public** **class** ArmstrongNumberCheck {

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

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

System.***out***.println("Enter a number: ");

**int** n = num.nextInt();

**int** c=0,a,temp;

temp=n;

**while**(n>0)

{

a=n%10;

n=n/10;

c=c+(a\*a\*a);

}

**if**(temp==c)

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

**else**

System.***out***.println(" Not armstrong number");

}

}

1. Write a program that prints a pattern like below.

\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*\*

**public** **class** StarPyramid {

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

**for**(**int** i=0; i<6; i++){

**for**(**int** j=0; j<6; j++){

**if**(j<=i){

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

}

**else** {

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

}

}

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

}

}

}

1. Arrays: Below is a list of programs on java arrays. Candidates are instructed not to use any inbuilt functions.
2. Write a program to find the largest number in a given array.

**import** java.util.Scanner;

**public** **class** LargestNumberInArray {

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

**int** n;

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

System.***out***.print("Enter number of elements in the array:");

n = s.nextInt();

**int** numbers[] = **new** **int**[n];

System.***out***.println("Enter elements of array:");

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

{

numbers[i] = s.nextInt();

}

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

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

{

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

largetst = numbers[i];

}

System.***out***.println("Largest Number in given Array is : " + largetst);

}

}

1. Write a program to find the missing number in a series of sorted numbers stored in an array.

**import** java.util.ArrayList;

**public** **class** FindMissingNumberInArray {

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

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

**int** a[] = { 1,2,3,4,6,7,8 };

**int** j = a[0];

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

{

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

{

j++;

}

}

System.***out***.println("missing numbers is "+j);

}

}

1. Write a program to find the common number in any given two arrays

**public** **class** FindCommonElementsInArray {

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

**int**[] arr1 = {4,7,3,9,2,8,25,28,64};

**int**[] arr2 = {3,2,12,9,40,32,4,98,6};

**for**(**int** i=0;i<arr1.length;i++){

**for**(**int** j=0;j<arr2.length;j++){

**if**(arr1[i]==arr2[j]){

System.***out***.println(arr1[i]);

}

}

}

}

}

1. Write a program to perform a linear search on any given array. Linear search is the basic search where you look for the element to be searched in a sequential way.

**import** java.util.Scanner;

**class** LinearArraySearch

{

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

{

**int** c, n, search, array[];

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

System.***out***.println("Enter number of elements");

n = in.nextInt();

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

System.***out***.println("Enter " + n + " integers");

**for** (c = 0; c < n; c++)

array[c] = in.nextInt();

System.***out***.println("Enter value to find");

search = in.nextInt();

**for** (c = 0; c < n; c++)

{

**if** (array[c] == search)

{

System.***out***.println(search + " is present at location " + (c + 1) + ".");

**break**;

}

}

**if** (c == n)

System.***out***.println(search + " is not present in array.");

}

}

1. Write a program to sort an array.

**import** java.util.Scanner;

**public** **class** ArraySorting {

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

{

**int** n, temp;

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

System.***out***.print("Enter no. of elements you want in array:");

n = s.nextInt();

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

System.***out***.println("Enter all the elements:");

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

{

a[i] = s.nextInt();

}

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

{

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

{

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

{

temp = a[i];

a[i] = a[j];

a[j] = temp;

}

}

}

System.***out***.print("Ascending Order:");

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

{

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

}

System.***out***.print(a[n - 1]);

}

}

}

1. Write a program to locate and swap only 2 elements which are not sorted in a given sorted array. For example given an array {1, 2, 5, 6, 4} locate 5 and 4 and then swap them as they are not as per sorted order.

**import** java.util.Arrays;

**public** **class** ArraySwap2Numbers {

**public** **static** **int**[] ArraySwap2NumbersMethod(**int**[] array1) {

**int** temp;

**for** (**int** i = 0; i < array1.length - 1; i++) {

**for** (**int** j = 1; j < array1.length - i; j++) {

**if** (array1[j - 1] > array1[j]) {

temp = array1[j - 1];

array1[j - 1] = array1[j];

array1[j] = temp;

}

}

}

**return** array1;

}

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

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

System.***out***.println("Ascending Order result:" + Arrays.*toString*(*ArraySwap2NumbersMethod*(array)) + "\n");

}

}

1. Write a program to merge 2 arrays.

**import** java.util.Scanner;

**public** **class** ArrayMerge

{

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

{

**int** size1, size2, size, i, j, k;

**int** arr1[] = **new** **int**[50];

**int** arr2[] = **new** **int**[50];

**int** merge[] = **new** **int**[100];

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

System.***out***.print("Enter Array 1 Size : ");

size1 = scan.nextInt();

System.***out***.print("Enter Array 1 Elements : ");

**for**(i=0; i<size1; i++)

{

arr1[i] = scan.nextInt();

}

System.***out***.print("Enter Array 2 Size : ");

size2 = scan.nextInt();

System.***out***.print("Enter Array 2 Elements : ");

**for**(i=0; i<size2; i++)

{

arr2[i] = scan.nextInt();

}

**for**(i=0; i<size1; i++)

{

merge[i] = arr1[i];

}

size = size1 + size2;

**for**(i=0, k=size1; k<size && i<size2; i++, k++)

{

merge[k] = arr2[i];

}

System.***out***.print("Now the New Array after Merging is :\n");

**for**(i=0; i<size; i++)

{

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

}

}

}

Strings: Below is a list of programs on String computations.

* 1. Write a program to reverse a string. Donot use StringBuffer/StringBuilder inbuilt reverse function.

**import** java.util.Scanner;

**public** **class** ReverseString1 {

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

{

String original, reverse = "";

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

System.***out***.println("Enter a string to reverse");

original = in.nextLine();

**int** length = original.length();

**for** ( **int** i = length - 1 ; i >= 0 ; i-- )

reverse = reverse + original.charAt(i);

System.***out***.println("Reverse of entered string is: "+reverse);

}

}

* 1. Write a program to find a substring in a given string and then replace it with another string.

**public** **class** FindAndReplaceString {

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

String test = "This is java program";

**if** (test.contains("ja")) {

String s = test.replace("ja", "Ja");

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

}**else**{

System.***out***.println("not found!");

}

}

}

* 1. Write a program which accepts a string like "This is nice" and converts it to a string like "This1 is2 nice3".

**public** **class** CountWordsInString {

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

{

**int** count=1;

String str ="This is java program";

String a[] = str.split(" ");

**for**(String m : a){

System.***out***.print(m + count++ + " ");

}

}

}

* 1. Write a program to reverse each individual word in a sentence.

**public** **class** ReverseEachWordInString {

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

String s = "This is java program";

String[] str =s.split(" ");

StringBuffer buffer = **new** StringBuffer();

**for**(**int** i=str.length-1;i>=0;i--)

{

buffer.append(str[i]);

buffer.append(" ");

}

System.***out***.println("Reverse word string is : "+buffer.toString());

}

}

* 1. Write a program to check for a palindrome string.

**import** java.util.Scanner;

**public** **class** CheckStringIsPalindrome {

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

{

String original, reverse = "";

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

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

original = in.nextLine();

**int** length = original.length();

**for** ( **int** i = length - 1; i >= 0; i-- )

reverse = reverse + original.charAt(i);

**if** (original.equals(reverse))

System.***out***.println("String is a Palindrome");

**else**

System.***out***.println("String is not a Palindrome");

}

}

* 1. Write a program to check if a given string of parenthesis has balanced parenthesis or not. For example a string as "(())" is a valid string whereas strings like ") (" or "(()))" are not valid strings.

**import** java.util.Stack;

**public** **class** ParenthesisCheck {

**private** String parenthesis;

**public** ParenthesisCheck(String s) {

parenthesis = s;

}

**public** **boolean** validate() {

**boolean** result = **true**;

Stack<Character> stack = **new** Stack<Character>();

**char** current, previous;

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

current = **this**.parenthesis.charAt(i);

**if**(current == '(' || current == '[' || current == '{') {

stack.push(current);

} **else** **if**(current == ')' || current == ']' || current == '}') {

**if**(stack.isEmpty()) {

result = **false**;

} **else** {

previous = stack.peek();

**if**((current == ')' && previous == '(') || (current == ']' && previous == '[') || (current == '}' && previous == '{')) {

stack.pop();

} **else** {

result = **false**;

}

}

}

}

**if**(!stack.isEmpty()) {

result = **false**;

}

**return** result;

}

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

ParenthesisCheck b = **new** ParenthesisCheck("{This[(is{a}parenthesis()check)in]java}");

System.***out***.println("Valid String: " + b.validate());

}

}

* 1. Write a program to implement hash Code() and equals() methods.

**import** java.util.HashSet;

**public** **class** EqualAndHashcodeImplimentation {

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

EqualAndHashcodeImplimentation test = **new** EqualAndHashcodeImplimentation();

Equality a = **new** Equality(7);

Equality b = **new** Equality(5);

test.test1(a, b);

Equality c = **new** Equality(3);

Equality d = **new** Equality(4);

test.test2(c, d);

HashSet<EqualAndHashcodeImplimentation> f = **new** HashSet<EqualAndHashcodeImplimentation>();

f.add(test);

System.***out***.println(f.contains(test));

System.***out***.println("f.hashCode(): " + f.hashCode());

}

**public** **void** test1(Equality a, Equality b) {

**if** (a.equals(b)) {

System.***out***.println("Test1: One and Two are equal");

} **else** {

System.***out***.println("Test1: One and Two are not equal");

}

}

**public** **void** test2(Equality c, Equality d) {

**if** (c.equals(d)) {

System.***out***.println("Test2: Three and Four are equal");

} **else** {

System.***out***.println("Test2: Three and Four are not equal");

}

}

}

**class** Equality {

**private** **int** value;

Equality(**int** val) {

value = val;

}

**public** **int** getValue() {

**return** value;

}

**public** **boolean** equals(Object obj) {

**if** (**this** == obj) {

**return** **true**;

}

**if** (obj == **null**) {

**return** **false**;

}

**if** (!(obj **instanceof** Equality)) {

**return** **false**;

}

Equality other = (Equality) obj;

**if** (value != other.value) {

**return** **false**;

}

**return** **true**;

}

**public** **int** hashCode() {

**final** **int** prime = 31;

**int** result = 1;

result = prime \* result + value;

**return** result;

}

}

4. File: Below list of programs enables the candidates to learn about file handling operations via programming.

a) Write a program that takes input from user dynamically using below classes.

>>Scanner: Use this class to input 2 integer numbers and sum them.

**import** java.util.Scanner;

**public** **class** ScannerAdd {

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

**int** num1,num2;

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

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

num1 = input.nextInt ();

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

num2 = input.nextInt ();

**int** add= num1+num2;

System.***out***.println("Addition is = " +add);

}

}

>>BufferedReader and InputStreamReader: Use this classes to input 2 integer numbers and subtract them.

**import** java.io.BufferedReader;

**import** java.io.IOException;

**import** java.io.InputStreamReader;

**public** **class** BufferedReaderSub {

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

BufferedReader test = **new** BufferedReader (**new** InputStreamReader(System.***in***));

System.***out***.println("Enter 1st number:");

String first = test.readLine();

**int** x = Integer.*parseInt*(first);

System.***out***.println("Enter 2nd number:");

String second = test.readLine();

**int** y = Integer.*parseInt*(second);

**int** sub =x-y;

System.***out***.println("Sub is = " +sub);

}

}

>>DatainputStream: Use this class to input 2 integer numbers and multiply them.

**import** java.io.DataInputStream;

**import** java.io.IOException;

**public** **class** DataInputStreamMulti {

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

DataInputStream test = **new** DataInputStream(System.***in***);

System.***out***.println("Enter 1st number:");

String first = test.~~readLine~~();

**int** x = Integer.*parseInt*(first);

System.***out***.println("Enter 2nd number:");

String second = test.~~readLine~~();

**int** y = Integer.*parseInt*(second);

**int** multi= x\*y;

System.***out***.println("Multiplication is = " +multi);

}

}

>>Console: Use this class to input 2 integer numbers and divide them.

1. Write a program to count the number of words in a file.

**import** java.io.BufferedReader;

**import** java.io.FileReader;

**import** java.util.StringTokenizer;

**public** **class** CountWordsInFile {

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

FileReader fr = **new** FileReader("C:/Users/ompra/Desktop/Test.txt");

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

String line = "", str = "";

**int** a = 0;

**int** b = 0;

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

str += line + " ";

a++;

}

System.***out***.println("Total number of lines = " +a );

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

StringTokenizer st = **new** StringTokenizer(str);

**while** (st.hasMoreTokens()) {

String s = st.nextToken();

b++;

}

System.***out***.println("Total number of words in the file = " +b);

}

}

1. Write a program to search for a string in a file and then return the count of number of occurrences of the string.

**import** java.io.BufferedReader;

**import** java.io.FileReader;

**import** java.util.Scanner;

**public** **class** StringCountFile {

**public** **static** **void** main(String ar[])**throws** Exception {

BufferedReader br=**new** BufferedReader(**new** FileReader("C:/Users/ompra/Desktop/Test.txt"));

System.***out***.println("enter the string which you search");

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

String str=ob.next();

String str1="",str2="";

**int** count=0;

**while**((str1=br.readLine())!=**null**)

{

str2 +=str1;

}

**int** index = str2.indexOf(str);

**while** (index != -1) {

count++;

str2 = str2.substring(index + 1);

index = str2.indexOf(str);

}

System.***out***.println("Number of the occures="+count);

}

}

1. Write a program to search for a string in a file and then replace it with another string.

**import** java.io.BufferedReader;

**import** java.io.File;

**import** java.io.FileNotFoundException;

**import** java.io.BufferedWriter;

**import** java.io.FileReader;

**import** java.io.FileWriter;

**import** java.io.IOException;

**public** **class** FindAndReplaceTextInFile {

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

String regex = "java";

String replacement = "python";

File input = **new** File("C:\\Users\\ompra\\Desktop\\Test.txt");

File output = **new** File("C:\\Users\\ompra\\Desktop\\Test1.txt");

BufferedReader br = **null**;

BufferedWriter bw = **null**;

**try**

{

br = **new** BufferedReader (**new** FileReader(input));

bw = **new** BufferedWriter(**new** BufferedWriter (**new** FileWriter(output)));

StringBuffer entireFile = **new** StringBuffer();

**int** line=br.read();

**while** (line!=-1)

{

entireFile.append((**char**)line);

line=br.read();

}

**if** (entireFile.length()==0)

{

System.***out***.println("Input was empty.");

}

**else**

{

bw.write(entireFile.toString().replaceAll(regex, replacement));

}

}

**catch** (FileNotFoundException fnfe)

{

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

}

**catch** (IOException io)

{

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

}

**finally**

{

**if** (br!=**null**)

{

**try**

{

br.close();

}

**catch** (IOException io)

{

System.***out***.println("Problem closing input file after processing");

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

}

}

**if** (bw!=**null**)

{

bw.flush();

bw.close();

}

}

}

}

1. Write a program to show list of all file names in a folder.

**import** java.io.File;

**public** **class** FileListInFolder {

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

{

File folder = **new** File("C:/Users/ompra/Desktop/apache-tomcat-8.5.8/bin");

String[] files = folder.list();

**for** (String file : files)

{

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

}

}

}

1. Write a program to copy content of file into another file.

**import** java.io.FileNotFoundException;

**import** java.io.FileReader;

**import** java.io.FileWriter;

**import** java.io.IOException;

**public** **class** CopyFile {

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

{

FileReader fr = **null**;

FileWriter fw = **null**;

**try**{

fr = **new** FileReader("C:/Users/ompra/Desktop/T1.txt");

fw = **new** FileWriter("C:/Users/ompra/Desktop/T2.txt");

**int** c;

**while**((c = fr.read()) != -1){

fw.write(c);

}

}

**finally**{

**if** (fr != **null**){

fr.close();

}

**if**(fw != **null**){

fw.close();

}

}

}

}

5. Collections: Below list of programs revolves around computations around collection framework.

a. Write a program having different methods to perform below operations.

--> To create an arraylist and add elements to it.

**package** Array;

**import** java.util.ArrayList;

**public** **class** ArrayListAdd {

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

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

a.add(1);

a.add(2);

a.add(3);

a.add(**null**);

a.add(5);

a.add(2,7);

**for** ( Integer array : a )

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

// for ( int i=0; i<a.size(); i++ )

// System.out.println("Index " + i + " >> " + "Value " +a.get(i) );

}

--> To search for an element in the list.

**package** Array;

**import** java.util.ArrayList;

**public** **class** ArrayListCheckEle {

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

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

a.add(100);

a.add(200);

a.add(300);

a.add(400);

a.add(500);

**boolean** b= a.contains(7);

**boolean** c= a.contains(500);

System.***out***.println("ArrayList contais 7 >>" +b);

System.***out***.println("ArrayList contais 500 >>" +c);

System.***out***.println(a.indexOf(200));

}

--> To print the elements of the list using iterator.

**package** Array;

**import** java.util.ArrayList;

**import** java.util.Iterator;

**public** **class** ArrayIterator {

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

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

a.add("100");

a.add("200");

a.add("300");

a.add("400");

a.add("500");

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

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

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

}

}

--> To print the elements in the reverse order using ListIterator

**package** Array;

**import** java.util.ArrayList;

**import** java.util.ListIterator;

**public** **class** ArrayListIterator {

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

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

a.add("100");

a.add("200");

a.add("300");

a.add("400");

a.add("500");

ListIterator<String> iter = a.listIterator(a.size());

**while** (iter.hasPrevious()) {

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

}

}

}

b. Write a program to find duplicates in an array using set.

**package** Array;

**import** java.util.HashSet;

**public** **class** ArrayDuplicatesUsingSet {

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

String[] a = {"ab", "aa", "aa", "ac", "bb", "ac", "cc"};

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

**for**( String b : a){

**if** (! set.add(b)){

System.***out***.println("Duplicate Element is : " +b);

}

}

}

}

c. Write a program which takes an input series as a1,a2,a3....an,b1,b2,b3...bn and produces output as a1,b1,a2,b2,a3,b3.......,an,bn.

**public** **class** MergeTwoArray {

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

String [] a = {"a1","a2","a3","a4","a5","b1","b2","b3","b4","b5"};

String[] newarr=**new** String[a.length];

**int** count=0;

**for**(**int** i=0;i<a.length/2;i++){

newarr[count]=a[i];

count+=2;

newarr[count-1]=a[a.length/2+i];

}

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

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

}

}

d. Write a program having below methods.

--> To create a hashmap.

**package** Collections;

**import** java.util.HashMap;

**public** **class** CreateHashmap {

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

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

a.put(1, "A");

a.put(2, "B");

a.put(3, "C");

a.put(4, "D");

a.put(5, "E");

a.put(6, "F");

a.put(7, "G");

a.forEach((key, value) -> System.***out***.println(key + " : " + value));

}

}

--> To search for a key in the created map and then returns its value.

**package** Collections;

**import** java.util.HashMap;

**public** **class** CreateHashmap {

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

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

a.put(1, "A");

a.put(2, "B");

a.put(3, "C");

a.put(4, "D");

a.put(5, "E");

a.put(6, "F");

a.put(7, "G");

a.forEach((key, value) -> System.***out***.println(key + " : " + value));

String val = a.get(5);

System.***out***.println("value is : " +val);

String val1 = a.get(7);

System.***out***.println("value is : " +val1);

}

}

e. Write a program to sort a map by value.

**public** **class** SortMapValue {

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

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

map.put("a", 100);

map.put("b", 300);

map.put("c", 500);

map.put("d", 400);

map.put("e", 200);

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

Comparator<String> comparator = **new** ValueComparator<String, Integer>(map);

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

result.putAll(map);

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

}

}

package Collections;

import java.util.Comparator;

import java.util.HashMap;

class ValueComparator<K, V extends Comparable<V>> implements Comparator<K>{

HashMap<K, V> map = new HashMap<K, V>();

public ValueComparator(HashMap<K, V> map){

this.map.putAll(map);

}

@Override

public int compare(K s1, K s2) {

return -map.get(s2).compareTo(map.get(s1)); }

}

f. Write a program that defines an enum having months of the year and then prints the value of all the enum elements.

**package** Collections;

**public** **class** EnumJavaExample {

**public** **enum** year { ***January***, ***February***, ***March***, ***April***, ***May***, ***June***, ***July***, ***August***, ***September***, ***October***, ***November***, ***December***}

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

**for**( year months : year.*values*())

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

}

g. Write a program to copy arraylist to an array.

**package** Array;

**import** java.util.ArrayList;

**public** **class** ArraylistToArrayCopy {

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

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

a.add(1);

a.add(2);

a.add(3);

a.add(5);

System.***out***.println("Original arraylist " +a);

Integer [] b = **new** Integer[a.size()];

System.***out***.println("New array elements are :");

a.toArray(b);

**for** (Integer c : b){

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

}

}

}

6. OOPS Concepts: Below list of programs shapes your skill around OOPS basic concepts.

* + 1. Write a program creating an abstract class Shape with properties (noOfSides, area, perimeter) and methods (calculateArea, calculatePerimeter, setSides.

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

**double** noOfSides;

**double** area;

**double** perimeter;

**abstract** **double** setSides();

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

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

}

* + 1. Write a program creating an interface ShapeConstants with variable (pi=3.14).

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

**double** ***PI*** = 3.14;

**double** calculatePerimeter();

**double** calculateArea();

**double** setSides();

}

c. Write a program that creates a class Circle extending Shape abstract class and implementing ShapeConstants interface and has following behavior.

--> Properties: radius.

--> Constructor: To set number of sides.

--> Overrides all the methods from parents.

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

**public** **double** radius;

**public** **static** **double** *numberofside*;

**public** Circle() {

**this**(1, *numberofside*);

}

**public** Circle(**double** radius, **double** numberofside){

**this**.radius=radius;

}

@Override

**public** **double** calculateArea() {

// A = π r^2

**return** ***PI*** \* Math.*pow*(radius, 2);

}

**public** **double** calculatePerimeter() {

// P = 2πr

**return** 2 \* ***PI*** \* radius;

}

**public** **double** setSides(){

**return** *numberofside*;

}

}

**public** **class** Test {

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

**double** radius = 5;

Shape circle = **new** Circle(5, 1);

System.***out***.println("Circle radius: " + radius);

System.***out***.println("Circle Area: " + circle.calculateArea());

System.***out***.println("Circle Perimeter: " + circle.calculatePerimeter());

System.***out***.println("Circle SetSides: " + circle.setSides());

}

}

d. Write a program to depict the behavior of method overloading and overriding.

**package** oops;

**public** **class** MethodoverloadEx {

**void** sum(**int** a,**int** b)

{System.***out***.println(a+b);

}

**void** sum(**int** a,**int** b, **double** c)

{System.***out***.println(a+b+c);

}

**void** sum(**double** a,**double** b, **double** c)

{System.***out***.println(a+b+c);

}

**public** **void** method(){

System.***out***.println("This is parent class method");

}

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

MethodoverloadEx obj=**new** MethodoverloadEx();

obj.sum(20,20);

obj.sum(10,10,10.2);

obj.sum(10.5, 10.2, 10.7);

obj.method();

obj.method();

}

}

**package** oops;

**public** **class** MethodOverride **extends** MethodoverloadEx{

**public** **void** method(){

System.***out***.println("This is child class method");

}

}

7. Exception Handling: Following programs illustrates how to work around exceptions.

1. Write a program that is expected to throw a null pointer exception and in turn handles it using try catch and finally.

**package** exception;

**public** **class** NullPointerExceptionEx {

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

Integer intval = **null**;

**try** {

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

} **catch** (NullPointerException ex) {

System.***out***.println("Null pointer Exception thrown");

} **finally** {

System.***out***.println("Finally done!");

}

}

}

b. Write a program containing a function which is expected to throw any kind of exception say NullPointerException or ArithmeticException etc and then handle this function in the parent function which calls this function.

**public** **class** ExceptionEx {

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

*divideSafely*(**null**);

}

**private** **static** **void** divideSafely(String[] array) {

**try** {

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

} **catch** (ArrayIndexOutOfBoundsException e) {

System.***err***.println("Usage: ExceptionDemo <num1> <num2>");

} **catch** (NullPointerException e) {

System.***err***.println("Null Pointer Exception");

} **catch** (ArithmeticException e) {

System.***err***.println("Cannot divide by zero");

}

}

**private** **static** **int** divideArray(String[] array) {

String s1 = array[0];

String s2 = array[1];

**return** *divideStrings*(s1, s2);

}

**private** **static** **int** divideStrings(String s1, String s2) {

**int** i1 = Integer.*parseInt*(s1);

**int** i2 = Integer.*parseInt*(s2);

**return** *divideInts*(i1, i2);

}

**private** **static** **int** divideInts(**int** i1, **int** i2) {

**return** i1 / i2;

}

}

8. General: Following list of programs helps you gain knowledge on all the remaining important areas in the java programming.

1. Write an Immutable class.

**package** other;

**public** **final** **class** ImmutableEmployees { // class is final so can't create subclass

**final** String id; // instance variable of class is final

**public** ImmutableEmployees(String id){

**this**.id=id;

}

**public** String getPancardNumber(){ // no setter method so can't change instance parameter value

**return** id;

}

}

1. Write a program to print 1 to 10 numbers without using loops and you should not have more than 1 print statement.

**package** other;

**import** java.util.stream.IntStream;

**public** **class** Print1to10WithoutLoop {

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

IntStream.*range*(1, 11).forEach(n -> { System.***out***.println(n); });

}

}

Or

**package** other;

**public** **class** Print1to10WithoutLoop {

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

*print*(10);

}

**private** **static** **void** print(**int** n) {

**if**(n > 1) {

*print*(n-1);

}

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

}

}

1. Write a program implementing explicit garbage collection.

**package** other;

**public** **class** ExplicitGarbageCollection {

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

A a = **new** A(123);

a = **null**;

System.*gc*();

}

}

**class** A {

**private** **int** id;

**public** A(**int** id) {

**this**.id=id;

}

@Override

**public** **void** finalize() {

System.***out***.println(**this**.id + " id cleaned by gc()");

}

}

1. Write a program that implements various functions of different type of access modifiers (private, protected, default, public) and then access these methods within or outside the class based on the identifier scope.
2. **Private Access Modifier** >> The private access modifier is accessible only within class

EX

**package** other;

**public** **class** PrivateModifierEx {

**private** **int** a=40;

**private** **void** A(){

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

}

}

**package** other;

**public** **class** B {

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

PrivateModifierEx obj=**new** PrivateModifierEx();

System.***out***.println(obj.a);//Compile Time Error

obj.A();//Compile Time Error

}

}

1. **Default Access Modifier** >> If you don't use any modifier, it is treated as **default** bydefault. The default modifier is accessible only within package.

**package** com.wbl;

**class** DefaultModiEx {

**void** ex(){

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

}

}

**package** other;

**import** com.wbl.DefaultModiEx;

**class** DefaultModiEx1 {

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

DefaultModiEx obj = **new** DefaultModiEx();

obj.ex();//compile time error

}

}

1. **Protected Access Modifier** >> The **protected access modifier** is accessible within package and outside the package but through inheritance only.

The protected access modifier can be applied on the data member,

method and constructor. It can't be applied on the class.

**package** com.wbl;

**public** **class** ProtectedModiEx {

**protected** **void** A(){

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

}

}

**package** other;

**import** com.wbl.ProtectedModiEx;

**public** **class** ProtectedModiEx1 **extends** ProtectedModiEx {

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

ProtectedModiEx1 obj = **new** ProtectedModiEx1();

obj.A();

}

}

1. **Public Access Modifier** >> The **public access modifier** is accessible everywhere. It has the widest scope among all other modifiers.

**package** com.wbl;

**public** **class** PublicModiEx {

**public** **void** ex(){

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

}

}

**package** other;

**import** com.wbl.PublicModiEx;

**public** **class** PublicModiEx1 {

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

PublicModiEx obj = **new** PublicModiEx();

obj.ex();

}

}

1. Write a program that contains a constructor, a static block and a static method. Execute the program in order to verify the order to execution of methods and block.

**package** other;

**public** **class** StaticEx {

**public** StaticEx(**int** x){

System.***out***.println("This is constructor with parameter");

}

**static** {

System.***out***.println("This is first static block");

}

**public** StaticEx(){

System.***out***.println("This is no parameter constructor");

}

**public** **static** String *staticString* = "Static Variable";

**static** {

System.***out***.println("This is second static block and "

+ *staticString*);

}

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

StaticEx statEx = **new** StaticEx();

StaticEx statEx1 = **new** StaticEx(5);

StaticEx.*staticMethod2*();

StaticEx.*staticMethod*();

}

**static** {

*staticMethod*();

System.***out***.println("This is third static block");

}

**public** **static** **void** staticMethod() {

System.***out***.println("This is static method");

}

**public** **static** **void** staticMethod2() {

System.***out***.println("This is static method2");

}

}

f. Write a program to make use of JDBC and insert/update/select values in the database.