Java Basics

1. How to create a class, object, method, and its signature

class create

{

void isMethod()

{

String a="This is a Method";

System.out.println(a);

}

public static void main (String[] args)

{

create ob=new create ();

ob.isMethod();

}

}

2. Write a program to print your name.

class name

{

public static void main(String args[])

{

System.out.println("Shashwat Khaitan");

}

}

3. Write a program for a Single line comment, multi-line, and documentation comments.

class Comments

{

public static void main(String[] args)

{

System.out.println("//This is a Single line comment");

System.out.println("/\*This is a Multi-line comment \*/");

System.out.println("/\*\*This is a Documentation comment \*/");

}

}

4. Define variables for different Data Types int, Boolean, char, float, double and print on the Console.

class datatypes

{

public static void main(String[] args)

{

int i=111;

float f=11.1f;

double d=111.1111;

char c='C';

boolean b=true;

System.out.println("Integer="+i);

System.out.println("Float="+f);

System.out.println("Double="+d);

System.out.println("Char="+c);

System.out.println("Boolean="+b);

}

}

5. Define the local and Global variables with the same name and print both variables and understand the scope of the variables.

class variable

{

int a=11;

void localVariable()

{

int a=22;

System.out.println("Local variable a:"+a);

}

public static void main(String[] args)

{

variable ob=new variable();

System.out.println("Instance variable a:"+ob.a);

ob.localVariable();

}

}

6. Write a function to print your name and call the function from main method.

class Calling

{

static void myMethod()

{

System.out.println("Rohit Sharma");

}

public static void main(String[] args)

{

myMethod();

}

}

Operators

1. Write a function for arithmetic operators (+, -,\*,/)

class Arithmetic

{

static void addition (int a, int b)

{

int add = a + b;

System.out.println("Addition of a + b = " + add);

}

static void subtraction (int a, int b)

{

int sub = a - b;

System.out.println("Subtraction of a - b = " + sub);

}

static void multiplication (int a, int b)

{

int mul = a \* b;

System.out.println("Multiplication of a \* b = " + mul);

}

static void division (int a, int b)

{

int div = a / b;

System.out.println("Division of a / b = " + div);

}

public static void main(String[] args)

{

int a = 10;

int b = 2;

addition (a, b);

subtraction (a, b);

multiplication (a, b);

division (a, b);

}

}

2. Write a method for increment and decrement operators (++, --)

class incdec

{

static void preIncrement(int a, int b)

{

int d;

d=b+(++a);

System.out.println("Values after Pre-Increment a,b : " + a + "," + d);

}

static void postIncrement(int a, int b)

{

int d;

d=b+(a++);

System.out.println("Values after Post-Increment a,b : " + a + "," + d);

}

static void preDecrement(int a, int b)

{

int d;

d=b+(--a);

System.out.println("Values after Pre-Decrement a,b:"+a+","+d);

}

static void postDecrement(int a, int b)

{

int d;

d=b+(a--);

System.out.println("Values after Post-Decrement a,b:"+a+","+d);

}

public static void main(String[] args)

{

int a=10;

int b=5;

preIncrement(a,b);

postIncrement(a,b);

preDecrement(a,b);

postDecrement(a,b);

}

}

3. Program to equal operator and not equal operators.

import java.util.Scanner;

class equal\_unequal

{

public static void main(String[] args)

{

int num1,num2;

Scanner sc=new Scanner(System.in);

System.out.println("Enter any two numbers to check for equality: ");

num1=sc.nextInt();

num2=sc.nextInt();

if(num1==num2)

System.out.println("The two numbers are equal");

else if(num1!=num2)

System.out.println("The numbers are not equal");

}

}

4. Programs on Logical AND, OR operator and Logical NOT.

class logical

{

public static void main(String args[])

{

int a,b,c;

a=5;

b=20;

c=10;

System.out.println(a>b&&a>c);

System.out.println(a<b&&a<c);

System.out.println(a>b&&a<c);

System.out.println(a>b||a>c);

System.out.println(a<b||a<c);

System.out.println(a<b||a>c);

System.out.println(!(a>b));

System.out.println(!(a<b));

}

}

5. Program for relational operators (<, <==, >, >==)

class relational

{

public static void main (String args[])

{

int a,b;

a=15;

b=20;

System.out.println(a<b);

System.out.println(a>b);

System.out.println(a<=b);

System.out.println(a>=b);

}

}

6. Print the smaller and larger number.

class largesmall

{

public static void main (String args[])

{

int a,b;

a=20;

b=30;

if(a>b)

System.out.println("Larger number is="+a);

else

System.out.println("Smaller number is="+a);

if(b>a)

System.out.println("Larger number is="+b);

else

System.out.println("Smaller number is="+b);

}

}

Loops

1. Write a program to print “Bright IT Career” ten times using for loop.

class Loop

{

public static void main (String[] args)

{

int i;

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

{

System.out.println("Bright IT Career");

}

}

}

2. Write a java program to print 1 to 20 numbers using the while loop.

class numbers

{

public static void main (String [] args)

{

int num=1;

while(num<=20)

System.out.println(num++);

}

}

1. Write a program to print the odd and even numbers.

import java.util.Scanner;

class oddeven

{

public static void main(String[] args)

{

int n;

Scanner sc=new Scanner(System.in);

System.out.println("Enter the size of the array: ");

n=sc.nextInt();

int arr[]=new int[n];

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

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

{

arr[i]=sc.nextInt();

}

System.out.println("The Even Elements are:");

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

{

if(arr[i]%2==0)

{

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

}

}

System.out.println(" ");

System.out.println("The Odd Elements are:");

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

{

if(arr[i]%2!=0)

{

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

}

}

}

}

1. Write a program to print largest number among three numbers.

import java.util.Scanner;

class largest

{

public static void main (String args[])

{

int a,b,c;

Scanner sc=new Scanner (System.in);

System.out.println("enter the first number:");

a=sc.nextInt();

System.out.println("enter the second number:");

b=sc.nextInt();

System.out.println("enter the third number:");

c=sc.nextInt();

if(a>=b&&a>=c)

System.out.println("Largest number is="+a);

else if(b>=a&&b>=c)

System.out.println("Largest number is="+b);

else

System.out.println("Largest number is="+c);

}

}

1. Write a program to print even number between 10 and 100 using while.

class print

{

public static void main (String args[])

{

int i=10;

System.out.println("The even numbers between 10 and 100 are:");

while(i<=100)

{

System.out.println(i);

i=i+2;

}

}

}

1. Write a program to print 1 to 10 using the do-while loop statement.

class number

{

public static void main (String args[])

{

int i=1;

do

{

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

i++;

}

while(i<=10);

}

}

1. Write a program to find Armstrong number or not.

import java.util.Scanner;

class armstrong

{

public static void main (String args[])

{

int n,on,r,result=0;

Scanner sc=new Scanner (System.in);

System.out.println("enter a number:");

n=sc.nextInt();

on=n;

while (on! =0)

{

r=on%10;

result+=Math.pow(r,3);

on=on/10;

}

if(result==n)

System.out.println("The number is armstrong");

else

System.out.println("The number is not armstrong");

}

}

1. Write a program to find the prime or not.

import java.util.Scanner;

class prime

{

public static void main (String args[])

{

int n,i;

boolean flag=false;

Scanner sc=new Scanner (System.in);

System.out.println("enter a number:");

n=sc.nextInt();

for(i=2;i<n/2;i++)

{

if(n%i==0)

flag=true;

break;

}

if (! flag)

System.out.println("It is a prime number");

else

System.out.println("It is not a prime number");

}

}

1. Write a program to palindrome or not.

import java.util.Scanner;

class palindrome

{

public static void main (String args[])

{

int n,rev=0,r,on;

Scanner sc=new Scanner (System.in);

System.out.println("enter a number:");

n=sc.nextInt();

on=n;

while (n! =0)

{

r=n%10;

rev=rev\*10+r;

n=n/10;

}

if(on==rev)

System.out.println("The number is palindrome");

else

System.out.println("The number is not palindrome");

}

}

1. Program to check whether a number is EVEN or ODD using switch.

import java.util.Scanner;

class evenodd

{

public static void main (String args[])

{

int n;

System.out.println("enter a number:");

n=new Scanner (System.in).nextInt();

switch(n%2)

{

case 1:

System.out.println("The number is even");

break;

case 2:

System.out.println("The number is odd");

break;

}

}

}

1. Print gender (Male/Female) program according to given M/F using switch.

import java.util.Scanner;

class MF

{

public static void main(String args[])

{

Scanner sc=new Scanner(System.in);

System.out.print("enter a character(M/F):");

char gender=sc.next().charAt(0);

switch(gender)

{

case 'M':

System.out.println("Gender is male");

break;

case 'F':

System.out.println("Gender is female");

break;

}

}

}

1. Program for multiple if else statement (Largest number in 10,20 and 30)

class multiple

{

public static void main (String args[])

{

int a=10, b=5,c=15;

if(a>b&&a>c)

System.out.println("Largest number is="+a);

else if(b>a&&b>c)

System.out.println("Largest number is="+b);

else

System.out.println("Largest number is="+c);

}

}

Arrays

1. Write a function to add integer values of an array.

class sum

{

static int arr[]={15,12,4,9};

static int sum ()

{

int sum=0;

int i;

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

{

sum=sum+arr[i];

}

return sum;

}

public static void main (String args[])

{

System.out.println("Sum of elements in the array is="+sum ());

}

}

1. Write a function to calculate the average value of an array of integers.

class average

{

static double avg(int a[], int n)

{

int sum=0,i;

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

{

sum=sum+a[i];

}

return (double)sum/n;

}

public static void main(String args[])

{

int arr[]={5,2,3,6,9,10,12,7,4,11};

int n=arr.length;

System.out.println("Average of elements in the array="+avg(arr,n));

}

}

1. Write a program to find the index of an array element.

import java.util.Scanner;

class index

{

static int index(int arr[],int t)

{

if(arr==null)

return -1;

int len=arr.length;

int i=0;

while(i<len)

{

if(arr[i]==t)

return i;

else

i++;

}

return -1;

}

public static void main(String args[])

{

int array[]={4,6,8,7,2,14,10,20,5,3};

System.out.println("Index position of 5 is:"+index(array,5));

System.out.println("Index position of 7 is:"+index(array,7));

}

}

4. Write a function to test if array contains a specific value.

import java.util.Scanner;

class specific

{

static void contains (int arr[],int n1,int n2)

{

boolean num1=false;

boolean num2=false;

for (int i:arr)

{

if(i==n1)

num1=true;

if(i==n2)

num2=true;

}

if(num1)

System.out.println("The value contained in the array:"+n1);

else

System.out.println("The value is not contained in the array:"+n1);

if(num2)

System.out.println("The value contained in the array:"+n2);

else

System.out.println("The value is not contained in the array:"+n2);

}

public static void main (String args[])

{

int n1, n2;

int my\_arr[]={5,6,4,9,3};

Scanner sc=new Scanner (System.in);

System.out.println("enter the two specific values:");

n1=sc.nextInt();

n2=sc.nextInt();

contains (my\_arr, n1, n2);

}

}

5. Write a function to remove a specific element from an array.

import java.util.\*;

class Remove1

{

public static int[] removeElement(int[] arr, int index)

{

if (arr == null || index < 0 || index >= arr.length)

{

return arr;

}

int[] anotherArray = new int[arr.length - 1];

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

{

if (i == index)

{

continue;

}

anotherArray[k++] = arr[i];

}

return anotherArray;

}

public static void main(String[] args)

{

int[] arr = { 11, 22, 33, 44, 55 };

System.out.print("Select an index to remove a value from arr = " + Arrays.toString(arr) + ": ");

Scanner sc = new Scanner(System.in);

int index = sc.nextInt();

System.out.println("Array after removing Element at index " + index );

arr = removeElement(arr, index);

System.out.println("arr = " + Arrays.toString(arr));

}

}

6. Write a function to copy an array to another array.

import java.util.\*;

class copy

{

static void Copy(int[] arr)

{

int copy[]=arr;

System.out.println("Array after copying="+Arrays.toString(copy));

}

public static void main(String[] args)

{

int n;

Scanner sc = new Scanner(System.in);

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

n=sc.nextInt();

System.out.println("Enter the Elements:");

int my\_array[]=new int[n];

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

{

System.out.println("my\_arr="+ i);

my\_array[i] = sc.nextInt();

}

System.out.println("my\_arr = " + Arrays.toString(my\_array));

Copy(my\_array);

}

}

7. Write a function to insert an element at a specific position in the array.

import java.io.\*;

import java.lang.\*;

import java.util.\*;

class insert

{

static int[] insertX(int n, int arr[], int x, int pos)

{

int i;

int newarr[]=new int[n+1];

for(i=0;i<n+1;i++)

{

if (i<pos-1)

newarr[i]=arr[i];

else if (i==pos-1)

newarr[i]=x;

else

newarr[i]=arr[i-1];

}

return newarr;

}

public static void main(String[] args)

{

int n=10;

int i;

int arr[]={1,2,3,4,5,6,7,8,9,10};

System.out.println("Initial Array:\n" + Arrays.toString(arr));

int x=50;

int pos=5;

arr=insertX(n,arr,x,pos);

System.out.println("\nArray with " + x + " inserted at position "+ pos + ":\n" + Arrays.toString(arr));

}

}

8. Write a function to find the minimum and maximum value of an array.

import java.util.Arrays;

class MinMax

{

static int minValue(int[] arr)

{

int min=arr[0];

for(int j:arr)

{

if(j<min)

min=j;

}

return min;

}

static int maxValue(int[] arr)

{

int max=arr[0];

for(int j:arr)

{

if(j>max)

max=j;

}

return max;

}

public static void main(String[] args)

{

int my\_arr[]={11,22,33,44,55,66};

System.out.println("my\_arr="+Arrays.toString(my\_arr));

System.out.println("The Maximum value="+maxValue(my\_arr));

System.out.println("The Minimum value="+minValue(my\_arr));

}

}

9. Write a function to reverse an array of integer values.

class reverse

{

static void reverse(int a[], int n)

{

int b[]=new int[n];

int j=n;

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

{

b[j-1]=a[i];

j=j-1;

}

System.out.println("Reversed array is:");

for (int k=0;k<n;k++)

{

System.out.println(b[k]);

}

}

public static void main(String[] args)

{

int arr[]={10,20,30,40,50};

reverse(arr,arr.length);

}

}

10. Write a function to find the duplicate values of an array.

import java.util.Arrays;

class duplicate

{

static void duplicateValues(int[] arr)

{

System.out.println("Duplicate values:");

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

{

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

{

if((arr[i]==arr[j]&&(i!=j)))

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

}

}

}

public static void main(String[] args)

{

int my\_arr[]={1,2,3,4,5,2,4,3};

System.out.println("my\_arr[]="+Arrays.toString(my\_arr));

duplicateValues(my\_arr);

}

}

11. Write a program to find the common values between two arrays.

import java.io.\*;

import java.util.\*;

class common

{

static void FindCommonElemet(String[] arr1,String[] arr2)

{

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

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

{

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

{

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

set.add(arr1[i]);

break;

}

}

for(String i:set)

{

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

}

}

public static void main(String[] args)

{

String arr1[]={"Article", "in", "Geeks", "for", "Geeks" };

String arr2[]={"Geeks", "for", "Geeks" };

System.out.println("Array 1:"+ Arrays.toString(arr1));

System.out.println("Array 2:"+ Arrays.toString(arr2));

System.out.print("Common Elements:");

FindCommonElemet(arr1, arr2);

}

}

1. Write a method to remove duplicate elements from an array.

class removedup

{

public static int removeduplicates(int a[], int n)

{

if(n==0||n==1)

{

return n;

}

int temp[]=new int[n];

int j=0;

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

{

if(a[i]!=a[i + 1])

{

temp[j++]=a[i];

}

}

temp[j++]=a[n-1];

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

{

a[i]=temp[i];

}

return j;

}

public static void main(String[] args)

{

int a[]={1,1,2,2,2,4,5,9};

int n=a.length;

n=removeduplicates(a, n);

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

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

}

}

1. Write a method to find the second largest number in an array.

class largest2nd

{

static int getSecondLargest(int[] a, int total)

{

int temp;

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

{

for(int j=i+1;j<total;j++)

{

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

{

temp=a[i];

a[i]=a[j];

a[j]=temp;

}

}

}

return a[total-2];

}

public static void main(String args[])

{

int a[]={1,2,5,6,3,2};

System.out.println("Second Largest:"+getSecondLargest(a,6));

}

}

1. Write a method to find number of even number and odd numbers in an array.

import java.util.Arrays;

class EvenOddNumbers

{

static void evenNumber(int[] arr)

{

System.out.print("Even numbers in array:");

for(int i:arr)

{

if (i%2==0)

{

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

}

}

}

static void oddNumber(int[] arr)

{

System.out.print("Odd numbers in array:");

for(int j:arr)

{

if(j%2!=0)

{

System.out.print(j+ " ");

}

}

}

public static void main(String[] args)

{

int arr[]={11,22,33,44,55};

System.out.println("a[]="+Arrays.toString(arr));

evenNumber(arr);

oddNumber(arr);

}

}

1. Write a function to get the difference of largest and smallest value.

import java.util.Arrays;

class difference

{

static void difference(int arr[])

{

int max=arr[0];

int min=arr[0];

int d,i;

for(i=1;i<arr.length;i++)

{

if(arr[i]>max)

max=arr[i];

else if(arr[i]<min)

min=arr[i];

}

d=max-min;

System.out.println("Difference of largest and smallest value="+d);

}

public static void main(String args[])

{

int Array[]={2,8,7,9,4,5};

System.out.println("Array[]="+Arrays.toString(Array));

difference(Array);

}

}

1. Write a method to verify if the array contains two specified elements (12,23).

import java.util.Scanner;

class specific

{

static void contains(int arr[],int n1,int n2)

{

boolean num1=false;

boolean num2=false;

for(int i:arr)

{

if(i==n1)

num1=true;

if(i==n2)

num2=true;

}

if(num1)

System.out.println("The value contained in the array:"+n1);

else

System.out.println("The value is not contained in the array:"+n1);

if(num2)

System.out.println("The value contained in the array:"+n2);

else

System.out.println("The value is not contained in the array:"+n2);

}

public static void main(String args[])

{

int n1,n2;

int my\_arr[]={5,6,4,9,3};

Scanner sc=new Scanner(System.in);

System.out.println("enter the two specific values:");

n1=sc.nextInt();

n2=sc.nextInt();

contains(my\_arr,n1,n2);

}

}

1. Write a function to find the missing number of sorted array of 1 to 100.

import java.util.Arrays;

class missing

{

static void missing(int arr[])

{

Arrays.sort(arr);

System.out.println("a[]="+Arrays.toString(arr));

int j=0;

for(int i=1;i<100;i++)

{

if(j<arr.length&&i==arr[j])

j++;

else

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

}

}

public static void main(String args[])

{

int my\_arr[]={5,9,8,4,6,2};

missing(my\_arr);

}

}

Static

1. Write a class with 2 static variables, 2 Instance variables, 2 static methods, 2 instance methods and a main method.

2. Print instance variables in static methods

3. Print static variables in Instance methods

4. Call instance methods in static methods

5. Call static methods in instance methods

6. Print all the static, instance variables in main method

7. Call static methods and instance methods in main method.

class method

{

static int sv1=54;

static int sv2=64;

int iv1=44;

int iv2=34;

static void method()

{

method ob=new method();

System.out.println("Instance variables:"+ob.iv1+" "+ob.iv2);

}

void instanceMethod()

{

System.out.println("Static variables:"+sv1+" "+sv2);

}

static void staticMethod()

{

method ob=new method();

ob.instanceMethod();

}

void instanceMethod2()

{

method();

}

public static void main(String args[])

{

method ob1=new method();

method();

staticMethod();

ob1.instanceMethod();

ob1.instanceMethod2();

}

}

Strings

1. Different ways creating a string

2. Concatenating two strings using + operator

3. Finding the length of the string

4. Extract a string using Substring

5. Searching in strings using index of ()

6. Matching a String Against a Regular Expression With matches ()

7. Comparing strings using the methods equals (),

8. equalsIgnoreCase(), startsWith(), endsWith() and compareTo()

9. Trimming strings with trim ()

10. Replacing characters in strings with replace ()

11. Splitting strings with split ()

12. Converting Numbers to Strings with valueOf()

13. Converting integer objects to Strings

14. Converting to uppercase and lowercase

class Strings

{

public static void main(String[] args)

{

String s1 = "This is a string";

System.out.println(s1);

String s2 = new String("Creating a string object and placed in memory \"Heap\"");

System.out.println(s2);

String sc1 = "This String is ";

String sc2 = "Concatenated";

String con = sc1 + sc2;

System.out.println(con);

String sl = "String Length";

int len = sl.length();

System.out.println("String Length : " + len);

String es = "Extracting string";

String sub = es.substring(2, 10);

System.out.println("Extracted string : " + sub);

String si = "Searching index";

int i1 = si.indexOf("index");

int i2 = si.indexOf("h", 3);

System.out.println("At Index : " + i1);

System.out.println("At Index : " + i2);

String ms = "Matching a String";

boolean match = ms.matches("(.\*)String(.\*)");

System.out.println("Matches : " + match);

String cs1 = "Comparing strings";

String cs2 = "Comparing strings";

boolean e = cs1.equals(cs2);

System.out.println("Equal : " + e);

boolean eic = cs1.equalsIgnoreCase(cs2);

System.out.println("Equal(ignore case) : " + eic);

boolean sw = cs1.startsWith("Comp");

System.out.println("Starts with : " + sw);

boolean ew = cs1.endsWith("ings");

System.out.println("Ends with : " + ew);

int ct = cs1.compareTo(cs2);

System.out.println("Compared : " + ct);

String ts = " Trimming String ";

String trm = ts.trim();

System.out.println("Trimmed : " + trm);

String rs = "Replacing string";

String rpl = rs.replace("g", "n");

System.out.println("Replaced : " + rpl);

String ss = "Splitting String";

String[] spl = ss.split(" ");

for (String i : spl)

{

System.out.println("Splitted : " + i);

}

int ns = 11;

String nTs = String.valueOf(ns);

System.out.println("Converted Num to Str : " + nTs + 22);

int is = 11;

String iTs = Integer.toString(is);

System.out.println("Converted int to str : " + iTs + 33);

String a = "upper case";

String b = "LOWER CASE";

String c = a.toUpperCase();

String d = b.toLowerCase();

System.out.println("Converted to upper case : " + c);

System.out.println("Converted yo lower case : " + d);

}

}

Inheritance

A, B and C are classes A is a super class. B is a sub class of A. C is a sub class of B.

Create three methods in each class, 2 methods are specific to each class and third method (override method) should be in all three Classes A, B and C

Create a class with main method. Create an object for each class A, B and C in main method and call every method of each class using its own object/instance.

Call an overridden method with super class reference to B and C class’s objects

Runtime Polymorphism with Data Members/Instance variables, Repeat the above process only for data members.

class A

{

int n=11;

void methodA1()

{

System.out.println("This is Class A Method 1");

}

void methodA2()

{

System.out.println("This is Class A Method 2");

}

void method3()

{

System.out.println("This is override method - Class A");

}

}

class B extends A

{

int n=22;

void methodB1()

{

System.out.println("This is Class B Method 1");

}

void methodB2()

{

System.out.println("This is Class B Method 2");

}

@Override

void method3()

{

System.out.println("This is override method - Class B");

}

}

class C extends B

{

int n=33;

void methodC1()

{

System.out.println("This is Class C Method 1");

}

void methodC2()

{

System.out.println("This is Class C Method 2");

}

@Override

void method3()

{

System.out.println("This is override method - Class C");

}

}

class Inheritance

{

public static void main(String[] args)

{

A a = new A();

a.methodA1();

a.methodA2();

a.method3();

B b = new B();

b.methodB1();

b.methodB2();

b.method3();

C c = new C();

c.methodC1();

c.methodC2();

c.method3();

A orm;

orm = new B();

orm.method3();

orm = new C();

orm.method3();

A rtp;

rtp = new A();

System.out.println(rtp.n);

rtp = new B();

System.out.println(rtp.n);

rtp = new C();

System.out.println(rtp.n);

}

}

Access Modifiers

1. Create a class with PRIVATE fields, private method, and a main method. Print the fields in main method. Call the private method in main method. Create a sub class and try to access the private fields and methods from sub class.

public class pvariable

{

private String name="Shashwat";

private int age=19;

private void pvtMethod()

{

System.out.println("My Name is " + name + " and my age is " + age);

}

public static void main(String[] args)

{

pvariable obj = new pvariable();

System.out.println(obj.age);

System.out.println(obj.name);

obj.pvtMethod();

}

}

2. Create a class with DEFAULT fields and methods. Access these fields and methods from any other class in the same package.

class modifier

{

int myAge;

String myName;

void defaultMethod()

{

System.out.println("My name is " + myName + " and my age is " + myAge);

}

}

class DefaultClassFieldsMethod

{

public static void main(String[] args)

{

modifier ob=new modifier();

ob.myAge=19 ;

ob.myName="Rohit";

ob.defaultMethod();

}

}

3.Create a class with PROTECTED fields and methods. Access these fields and methods from any other class in the same package. Also, Access the PROTECTED fields and methods from child class located in a different package Access the PROTECTED fields and methods from any class in different package.

class protect

{

protected String name;

protected void protectedMethod()

{

System.out.println("This is a " + name);

}

}

class protectedClass\_03

{

public static void main(String[] args)

{

protect ob=new protect();

ob.name="Protected Method";

ob.protectedMethod();

}

}

4. Create a class with PUBLIC fields and methods. Access the public methods and fields from any class in the same package or different package.

class publicity

{

public String name = "Public";

public void publicMethod()

{

System.out.println("This is " + name);

}

}

public class PublicClassFieldsMethods

{

public static void main(String[] args)

{

publicity ob=new publicity();

ob.name = "Public Method";

ob.publicMethod();

}

}

Abstract Class

1. Create an abstract class with abstract and non-abstract methods.

2. Create a sub class for an abstract class. Create an object in the child class for the abstract class and access the non-abstract methods

3. Create an instance for the child class in child class and call abstract methods

4. Create an instance for the child class in child class and call non-abstract methods.

abstract class Animal

{

abstract void sound();

void eat()

{

System.out.println("Animal is eating");

}

}

class Lion extends Animal

{

void sound()

{

System.out.println("Lion roars");

}

public static void main(String[] args)

{

Animal A = new Lion();

A.eat();

Lion L = new Lion();

L.sound();

L.eat();

}

}

Interfaces

1. Create an interface with only one method and implement it in a class. Call the method implemented.

interface onemethod

{

void oneMethod();

}

public class InterfaceOneMethod implements onemethod

{

@Override

public void oneMethod()

{

System.out.println("My name is Shashwat Khaitan and my age is 19");

}

public static void main(String[] args)

{

InterfaceOneMethod if1 = new InterfaceOneMethod();

if1.oneMethod();

}

}

1. Create an interface with two methods but implement only one in a class. Call the method implemented.

interface twomethod

{

void methodOne();

void methodTwo();

}

class InterfaceTwoMethod implements twomethod

{

@Override

public void methodOne()

{

System.out.println("First Method");

}

@Override

public void methodTwo()

{

System.out.println("Second Method");

}

public static void main(String[] args)

{

InterfaceTwoMethod if2 = new InterfaceTwoMethod();

if2.methodOne();

if2.methodTwo();

}

}

1. Create two interfaces with one method each. Implement these two interfaces in one class.

interface twointerfaces

{

void methodOne();

}

interface method1

{

void methodTwo();

}

public class TwoInterfaceOneMethod implements twointerfaces, method1

{

@Override

public void methodOne()

{

System.out.println("This is Method one");

}

@Override

public void methodTwo()

{

System.out.println("This is Method two");

}

public static void main(String[] args)

{

TwoInterfaceOneMethod if4 = new TwoInterfaceOneMethod();

if4.methodOne();

if4.methodTwo();

}

}

1. Create two interfaces with the same method (same signature) in both the interfaces. Implement these two interfaces in one class. Call the method.

interface twointerfacesamemethod

{

void sameMethod();

}

interface method2

{

void sameMethod();

}

class TwoInterfaceSameMethod implements twointerfacesamemethod, method2

{

@Override

public void sameMethod()

{

System.out.println("This is same method in both the interfaces");

}

public static void main(String[] args)

{

TwoInterfaceSameMethod if5 = new TwoInterfaceSameMethod();

if5.sameMethod();

}

}

1. Create an interface with a default method and implement it in a class. Do not provide implementation to the default method and call the method.

interface method3

{

default void defaultMethod()

{

System.out.println("This is a default method");

}

}

class InterfaceDefaultMethod implements method3

{

public static void main(String[] args)

{

InterfaceDefaultMethod if6 = new InterfaceDefaultMethod();

if6.defaultMethod();

}

}

1. Create an interface and inherit it from the other interface.

interface inheritedinterface

{

void methodOne();

}

interface Inherited extends inheritedinterface

{

void methodTwo();

}

class InheritedInterface implements Inherited

{

@Override

public void methodOne()

{

System.out.println("This is first method");

}

@Override

public void methodTwo()

{

System.out.println("This is second method");

}

public static void main(String[] args)

{

InheritedInterface ob = new InheritedInterface();

ob.methodOne();

ob.methodTwo();

}

}

1. Create a PUBLIC interface with fields and methods, fields should have values assigned. Implement this interface to some class and print the values of the interface fields and call the interface methods.

public interface publicinterface

{

int num=111;

public void myMethod();

}

class interfacepublic implements publicinterface

{

@Override

public void myMethod()

{

System.out.println("My name is Rohitman Sharma");

}

public static void main(String[] args)

{

interfacepublic ob=new interfacepublic();

System.out.println(num);

ob.myMethod();

}

}

8. Create a PRIVATE or PROTECTED interface and print the values as above scenario

9. Create an interface with private, public, and protected fields.

10. Create an interface with static final variable.

interface privateprotected

{

public int num1=11;

static final int num2=22;

final int num3=33;

}

class InterfacePrivateProtected implements privateprotected

{

public static void main(String[] args)

{

InterfacePrivateProtected if9 = new InterfacePrivateProtected();

System.out.println(if9.num1 + " " + if9.num2);

}

}

This and super

1. Print the fields/instance members of the current class using this and without using object.

3. Call constructor of the current class using this ()

4. Call argument constructor of current class using this ()

public class CallUsingThis

{

int num;

String name;

CallUsingThis()

{

this (30, "Shashwat");

}

CallUsingThis(int num, String name)

{

this.num = num;

this.name = name;

System.out.println(num + " " + name);

}

public static void main (String[] args)

{

CallUsingThis ts1 = new CallUsingThis();

CallUsingThis ts = new CallUsingThis(12, "Computer");

}

}

2. Print the fields/instance members of the parent class using super

5. Call constructor of the parent class using super ()

class Parent

{

String name = "Shashwat";

Parent()

{

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

}

}

class Child extends Parent

{

String name = "This is computer science";

Child()

{

super();

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

}

void myMethod()

{

System.out.println("My name is " + super.name);

}

}

public class CallUsingSuper

{

public static void main (String[] args)

{

Child c = new Child();

c.myMethod();

}

}

6. Use this () and super () in methods not in constructors.

class Parent1

{

void method1()

{

System.out.println("Used this() and super() in methods");

}

void method2()

{

this.method1();

}

}

class Child extends Parent1

{

void method3()

{

super.method2();

}

}

public class ThisSuperInMethods

{

public static void main(String[] args)

{

Child c = new Child();

c.method3();

}

}

Constructors

1. Write a class with a default constructor, one argument constructor and two argument constructors. Instantiate the class to call all the constructors of that class from a main class.

class ParentClass

{

int rollNo;

String name;

String branch;

ParentClass()

{

System.out.println("Student Details");

}

ParentClass(int rollNo)

{

this.rollNo = rollNo;

System.out.println("Roll No : " + this.rollNo);

}

ParentClass(String name, String branch)

{

this.name = name;

this.branch = branch;

System.out.println("Name : " + this.name);

System.out.println("Branch : " + this.branch);

}

}

public class DefaultOneTwoArgument

{

public static void main(String... args)

{

new ParentClass();

new ParentClass(30);

new ParentClass("Shashwat", "BCA");

}

}

1. Call the constructors (both default and argument constructors) of super class from a child class.

class ChildClass extends ParentClass

{

String clg;

String city;

ChildClass()

{

super();

}

ChildClass (int rollNo)

{

super(rollNo);

}

ChildClass(String name, String branch, String clg, String city)

{

super(name, branch);

this.clg = clg;

this.city = city;

System.out.println("College : " + this.clg);

}

}

public class CallSuperFromChild

{

public static void main(String... args)

{

new ChildClass();

new ChildClass(30);

new ChildClass("Shashwat", "BCA", "AIITK","Kolkata");

}

}

3. Apply private, public, protected and default access modifiers to the constructor

4. Write constructors with return type int and String

5. Try to call the constructor multiple times with the same object.

class PublicPrivateProtectedDefault

{

int rollNo;

String name;

String clg;

String city;

{System.out.println("Student Details");}

PublicPrivateProtectedDefault()

{

this(30);

}

public PublicPrivateProtectedDefault(int rollNo)

{

this("Shashwat");

this.rollNo = rollNo;

System.out.println("rollNo : " + rollNo);

}

private PublicPrivateProtectedDefault(String name)

{

this("AIITK", "Kolkata");

this.name = name;

System.out.println("Name : " + name);

}

protected PublicPrivateProtectedDefault(String clg, String city)

{

this.clg = clg;

this.city = city;

System.out.println("college : " + clg);

System.out.println("City : " + city);

}

public static void main(String[] args)

{

PublicPrivateProtectedDefault c = new PublicPrivateProtectedDefault();

}

}

Method Overloading

1. Write two methods with the same name but different number of parameters of same type and call the methods from main method

2. Write two methods with the same name but different number of parameters of different data type and call the methods from main method

3. Write two methods with the same name and same number of parameters of same type and call from main method

4. Write two methods with the same name and same number of parameters of different type and call from main method

5. Write two methods with the same name, number of parameters and data type but different return Type.

class MethodOverloading

{

int num1;

int num2;

int rollNo;

String name;

void method1(int num1)

{

this.num1 = num1;

System.out.println("1.First number : " + num1);

}

void method1(int num1, int num2)

{

this.num1 = num1;

this.num2 = num2;

System.out.println("1.Sum of two numbers : " + (num1 + num2));

}

void method2(int rollNo)

{

this.rollNo = rollNo;

System.out.println("2.Roll number : " + rollNo);

}

void method2(int rollNo, String name)

{

this.rollNo = rollNo;

this.name = name;

System.out.println("2.Roll number : " + rollNo + " ; " + "Name : " + name);

}

void method4(int rollNo)

{

this.rollNo = rollNo;

System.out.println("4.Roll number : " + rollNo);

}

void method4(String name)

{

this.name = name;

System.out.println("4.Name : " + name);

}

public static void main(String[] args)

{

MethodOverloading mo = new MethodOverloading();

mo.method1(11);

mo.method1(10, 20);

mo.method2(30);

mo.method2(317, "Shashwat");

mo.method4(30);

mo.method4("Shashwat");

}

}

Exceptions

1. Write a program to generate Arithmetic Exception without exception handling.

2.Handle the Arithmetic exception using try-catch block.

class arithmeticExpression

{

public static void main (String args[])

{

int num1=15,num2=0,result=0;

try

{

result=num1/num2;

System.out.println("The result is:"+result);

}

catch(ArithmeticException e)

{

System.out.println("Can't be divided by Zero: "+e);

}

}

}

3. Write a method which throws exception, call that method in main class without try block.

class MethodThrowsException

{

static void throwException()

{

throw new RuntimeException("Throwing exception");

}

public static void main(String[] args)

{

throwException();

}

}

4. Write a program with multiple catch blocks.

class MultipleCatchBlocks

{

public static void main(String[] args)

{

try

{

int[] a = new int[6];

a[6] = 10 / 0;

System.out.println(a[10]);

}

catch(ArithmeticException e)

{

System.out.println("Arithmetic Exception occurs");

e.getStackTrace();

}

catch(ArrayIndexOutOfBoundsException e)

{

System.out.println("ArrayIndexOutOfBounds Exception occurs");

e.getStackTrace();

}

catch (Exception e)

{

System.out.println("Parent Exception occurs");

e.getStackTrace();

}

System.out.println("Main method ended");

}

}

5. Write a program to throw exception with your own message.

class MyCustomException extends Exception

{

public class TestCustomException2

{

public static void main(String args[])

{

try

{

throw new MyCustomException();

}

catch (MyCustomException ex)

{

System.out.println("Caught the exception");

System.out.println(ex.getMessage());

}

System.out.println("rest of the code...");

}

}

}

6. Write a program to create your own exception.

class InvalidExamException extends Exception

{

InvalidExamException(String msg)

{

System.out.println(msg);

}

}

public class CreateOwnException

{

static void exam(int marks) throws InvalidExamException

{

if (marks < 40)

{

throw new InvalidExamException("Failed in exam");

}

else

{

System.out.println("Passed in exam");

}

}

public static void main(String[] args)

{

try

{

exam(39);

}

catch (Exception i)

{

i.printStackTrace();

}

}

}

7. Write a program with finally block.

class TestFinallyBlock

{

public static void main(String args[])

{

try

{

int data=25/5;

System.out.println(data);

}

catch(NullPointerException e)

{

System.out.println(e);

}

finally

{

System.out.println("Finally block is always executed");

}

System.out.println("rest of the code");

}

}

8. Write a program to generate Arithmetic Exception.

class Exception

{

void divide(int a, int b)

{

int res=a/b;

System.out.println("Division process has been done successfully.");

System.out.println("Result came after division is:"+res);

}

public static void main(String argvs[])

{

Exception ob = new Exception();

ob.divide(1,0);

}

}

9. Write a program to generate ArrayIndexOutOfBoundException.

class ArrayIndexOutOfBound

{

public static void main(String[] args)

{

String[] arr={"Rohit","Shikar","Virat","Dhoni"};

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

{

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

}

}

}

10. Write a program to generate ClassNotFoundException.

class ClassNotFound

{

public static void main(String args[])

{

try

{

Class.forName("The Class do not Exist");

}

catch (ClassNotFoundException e)

{

e.printStackTrace();

}

}

}

11. Write a program to generate FileNotFoundException.

import java.io.\*;

public class FileNotFound

{

public static void main(String[] args)

{

try

{

BufferedReader reader = new BufferedReader(new FileReader(new File("/invalid/file/location")));

}

catch(FileNotFoundException e)

{

System.err.println("FileNotFoundException caught!");

e.printStackTrace();

}

}

}

12. Write a program to generate IOException.

import java.util.\*;

public class IOException

{

public static void main(String[] args)

{

Scanner scan = new Scanner("Hello World! Hello JavaTpoint.");

System.out.println("" + scan.nextLine());

System.out.println("Exception Output:"+scan.ioException());

scan.close();

}

}

13. Write a program to generate NoSuchFieldException.

import java.lang.reflect.\*;

public class NoSuchField

{

public static void main(String[] args)

{

try

{

String obj=new String();

Class cls=obj.getClass();

Field fld=cls.getField("name");

System.out.println("Field found:"+fld.toString());

}

catch (NoSuchFieldException e)

{

System.out.println("Exception:"+e);

}

System.out.println("Program finished");

}

}

14. Write a program to generate NoSuchMethodException.

import java.io.\*;

class NoSuchMethod

{

public void print(String myString)

{

System.out.println(myString);

}

}

class Error

{

public static void main(String[] args)

{

NoSuchMethod ob=new NoSuchMethod();

ob.print("Hello World");

}

}

15. Write a program to generate NullPointerException.

class NullPointer

{

public static void main(String[] args)

{

Object ref = null;

ref.toString();

}

}

16. Write a program to generate NumberFormatException.

class NumberFormat

{

public static void main(String[] args)

{

int a = Integer.parseInt(null);

}

}

17.Write a program to generate StringIndexOutOfBoundsException.

public class StringIndexOutOfBounds

{

public static void main(String[] args)

{

String str="Hello World";

try

{

char charAtNegativeIndex=str.charAt(-1);

char charAtLengthIndex=str.charAt(11);

}

catch (StringIndexOutOfBoundsException e)

{

System.err.println("StringIndexOutOfBoundsException caught");

e.printStackTrace();

}

}

}

18. Write a program to generate SQLException.

public void testExecuteUpdateNoAutoGenColumnName() throws SQLException {

Statement stmt = createStatement();

String[] columnNames = new String[2];

columnNames[0] = "I";

columnNames[1] = "S";

try {

stmt.executeUpdate("insert into tab1 values(2, 3, 4.1)",

columnNames);

fail("FAIL -- executeUpdate should have failed...");

} catch (SQLException ex) {

assertFailedExecuteUpdateForColumnName(ex);

}

}

JAVA IO

1. Write a program to read text from .txt file using InputStream.

import java.io.FileInputStream;

import java.io.InputStream;

public class ReadUsingInputStream

{

public static void main(String[] args)

{

try

{

InputStream fis = new FileInputStream("os.txt");

int i;

while ((i = fis.read()) != -1)

{

System.out.print((char) i);

}

fis.close();

}

catch (Exception e)

{

e.getStackTrace();

}

}

}

2. Write a program to write text to .txt file using OutputStream.

import java.io.FileOutputStream;

import java.io.OutputStream;

public class WriteUsingOutputStream

{

public static void main(String args[])

{

String data = "This text is written using OutputStream.";

try

{

OutputStream fos = new FileOutputStream("os.txt");

byte[] db = data.getBytes();

fos.write(db);

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

fos.close();

}

catch(Exception e)

{

e.getStackTrace();

}

}

}

3. Read text from a .txt file using BufferedInputStream.

import java.io.\*;

public class BufferedInputStreamExample

{

public static void main(String args[])

{

try

{

FileInputStream fin=new FileInputStream("D:\\testout.txt");

BufferedInputStream bin=new BufferedInputStream(fin);

int i;

while((i=bin.read())!=-1)

{

System.out.print((char)i);

}

bin.close();

fin.close();

}

catch(Exception e)

{System.out.println(e);}

}

}

4.Write text to a .txt file using BufferedOutputStream.

import java.io.BufferedOutputStream;

import java.io.FileOutputStream;

public class OutputStream

{

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

{

BufferedOutputStream bufferedOutput=new BufferedOutputStream(new FileOutputStream("yourFile.txt"));

bufferedOutput.write("Line one".getBytes());

bufferedOutput.write("\n".getBytes());

bufferedOutput.write(65);

bufferedOutput.close();

}

}

5. Write a program to read text from .txt file using FileReader.

import java.io.File;

import java.io.FileReader;

import java.io.IOException;

class ReaderFile

{

public static void main(String[] args)

{

File file = new File("doc.txt");

try (FileReader fr = new FileReader(file))

{

int content;

while ((content = fr.read()) != -1) {

System.out.print((char) content);

}

} catch (IOException e) {

e.printStackTrace();

}

}

}

6. Write a program to write text to .txt file using FileWriter.

import java.io.FileWriter;

public class FileWriter1

{

public static void main(String args[])

{

try

{

FileWriter fw=new FileWriter("D:\\testout.txt");

fw.write("Welcome to javaTpoint.");

fw.close();

}

catch(Exception e)

{System.out.println(e);}

System.out.println("Success");

}

}

7. Read text from a .txt file using BufferedReader.

import java.io.BufferedReader;

import java.io.File;

import java.io.FileReader;

import java.io.IOException;

class reader

{

public static void main(String[] args)

{

File file = new File("demo.txt");

try (BufferedReader br = new BufferedReader(new FileReader(file)))

{

String line;

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

{

System.out.println(line);

}

}

catch (IOException e)

{

e.printStackTrace();

}

}

}

8. Write text to a .txt file using BufferedWriter.

import java.io.\*;

public class BufferedWriterExample

{

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

{

FileWriter writer = new FileWriter("D:\\testout.txt");

BufferedWriter buffer = new BufferedWriter(writer);

buffer.write("Welcome to javaTpoint.");

buffer.close();

System.out.println("Success");

}

}

9. Write a program to read data from properties file.

import java.io.IOException;

import java.io.InputStream;

import java.util.Properties;

import java.util.Set;

public class PropertiesCache

{

   private final Properties configProp = new Properties();

   private PropertiesCache()

   {

      InputStream in =this.getClass().getClassLoader().getResourceAsStream("application.properties");

      System.out.println("Reading all properties from the file");

      try

{

          configProp.load(in);

      }

catch (IOException e)

{

          e.printStackTrace();

      }

   }

   private static class LazyHolder

   {

      private static final PropertiesCache INSTANCE = new PropertiesCache();

   }

   public static PropertiesCache getInstance()

   {

      return LazyHolder.INSTANCE;

   }

   public String getProperty(String key)

{

      return configProp.getProperty(key);

   }

   public Set<String> getAllPropertyNames()

{

      return configProp.stringPropertyNames();

   }

   public boolean containsKey(String key)

{

      return configProp.containsKey(key);

   }

}

10. Write a program to read data from excel.

import java.io.File;

import java.io.FileInputStream;

import java.io.IOException;

import org.apache.poi.hssf.usermodel.HSSFSheet;

import org.apache.poi.hssf.usermodel.HSSFWorkbook;

import org.apache.poi.ss.usermodel.Cell;

import org.apache.poi.ss.usermodel.FormulaEvaluator;

import org.apache.poi.ss.usermodel.Row;

public class ReadExcelFileDemo

{

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

{

FileInputStream fis=new FileInputStream(new File("C:\\demo\\student.xls"));

HSSFWorkbook wb=new HSSFWorkbook(fis);

HSSFSheet sheet=wb.getSheetAt(0);

FormulaEvaluator formulaEvaluator=wb.getCreationHelper().createFormulaEvaluator();

for(Row row: sheet)

for(Cell cell: row)

{

switch(formulaEvaluator.evaluateInCell(cell).getCellType())

{

case Cell.CELL\_TYPE\_NUMERIC:

System.out.print(cell.getNumericCellValue()+ "\t\t");

break;

case Cell.CELL\_TYPE\_STRING:

System.out.print(cell.getStringCellValue()+ "\t\t");

break;

}

}

System.out.println();

}

}

11. Write a program to write data to excel.

import java.io.File;

import org.apache.poi.ss.usermodel.Cell;

import org.apache.poi.xssf.usermodel.XSSFRow;

import org.apache.poi.xssf.usermodel.XSSFSheet;

import org.apache.poi.xssf.usermodel.XSSFWorkbook;

import java.io.FileOutputStream;

import java.util.Map;

import java.util.Set;

import java.util.TreeMap;

public class WriteDataToExcel

{

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

{

XSSFWorkbook workbook=new XSSFWorkbook();

XSSFSheet spreadsheet=workbook.createSheet(" Student Data ");

XSSFRow row;

Map<String, Object[]> studentData=new TreeMap<String, Object[]>();

studentData.put(

"1",

new Object[] { "Roll No", "NAME", "Year" });

studentData.put("2", new Object[] { "128", "Aditya",

"2nd year" });

studentData.put(

"3",

new Object[] { "129", "Narayana", "2nd year" });

studentData.put("4", new Object[] { "130", "Mohan",

"2nd year" });

studentData.put("5", new Object[] { "131", "Radha",

"2nd year" });

studentData.put("6", new Object[] { "132", "Gopal",

"2nd year" });

Set<String> keyid = studentData.keySet();

int rowid = 0;

for (String key : keyid)

{

row = spreadsheet.createRow(rowid++);

Object[] objectArr = studentData.get(key);

int cellid = 0;

for (Object obj : objectArr)

{

Cell cell = row.createCell(cellid++);

cell.setCellValue((String)obj);

}

}

FileOutputStream out = new FileOutputStream(new File("C:/savedexcel/GFGsheet.xlsx"));

workbook.write(out);

out.close();

}

}

Collections

1. Create an ArrayList of type String with 10 string elements.

Add 10 string elements to ArrayList and perform the below operations

Add an element to the ArrayList Iterate through the ArrayList by using Iterator object

Add an element at a specific index Remove an element from the ArrayList,

Remove at an index Update the element at a specific index

Check the element is present at a particular index

Get an element at a particular index

Find out the size of the ArrayList

Check the given element is present in the ArrayList

Remove all elements of the ArrayList

import java.util.Arrays;

import java.util.Iterator;

public class ArrayList

{

public static void main(String[] args)

{

java.util.ArrayList<String> name=new java.util.ArrayList<>(Arrays.asList("zero", "one", "two", "three", "four", "five"));

System.out.println("\njala="+name);

name.add("six");

name.add("seven");

name.add("eight");

name.add("nine");

System.out.println("\nAfter adding elements :");

System.out.println("jala="+name);

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

System.out.println("\nIterating through the ArrayList : ");

while (itr.hasNext()) {

System.out.print(itr.next() + " ");

}

System.out.println(" ");

name.add(1, "AtIndex1");

System.out.println("\nAfter adding an element at index 1 :");

System.out.println("jala="+name);

name.remove(1);

System.out.println("\nAfter removing an element at index 1 :");

System.out.println("jala="+name);

name.set(1, "ONE");

System.out.println("\nAfter updating the element at index 1 :");

System.out.println("jala="+name);

System.out.println("\nThe element is present at index :");

System.out.println(name.indexOf("three"));

System.out.println("\nThe element at index 3 :");

System.out.println(name.get(3));

System.out.println("\nSize of the ArrayList : ");

System.out.println(name.size());

System.out.println("\n'two' is present in the ArrayList : ");

System.out.println(name.contains("two"));

System.out.println("\nAfter removing all elements of the ArrayList :");

name.clear();

System.out.println("jala="+name);

}

}

1. Create a HashMap with at least 10 key value pairs of the Student ID and Name

Insert a key value mapping into the map Fetch the value of a Key Create a clone/copy of HashMap

Check if the given Key is in the Map

Check if the value is in the Map

Check if the map is empty

Print the size of the Map to the console

Print all the Keys of the map to the console

Print all the Keys of the map to the console

Remove a specific Key-value pair

Copy all the elements of the Map to another Map

class HashMap

{

public static void main(String[] args)

{

java.util.HashMap<Integer, String> jala = new java.util.HashMap<>();

jala.put(1, "Shashwat");

jala.put(2, "Rohit");

jala.put(3, "Virat");

jala.put(4, "MSD");

jala.put(5, "Aditya");

jala.put(6, "Manoj");

jala.put(7, "Mandeep");

jala.put(8, "Pawan");

jala.put(9, "Dheeraj");

jala.put(10, "Rahul");

System.out.println("key-value pairs of the Student ID and Name:");

System.out.println("jala = " + jala);

System.out.println("Name of the Student with ID no 6:");

System.out.println(jala.get(6));

System.out.println("clone/copy of HashMap:");

System.out.println(jala.clone());

System.out.println("ID No 3 is in the Map:");

System.out.println(jala.containsKey(3));

System.out.println("Student Name 'Shashwat' is in the map:");

System.out.println(jala.containsValue("Shashwat"));

System.out.println("Checking if the map is empty:");

System.out.println(jala.isEmpty());

System.out.println("The size of the Map is:");

System.out.println(jala.size());

System.out.println("All ID No of the students:");

System.out.println(jala.keySet());

System.out.println("Removed ID no 2 ,Student name= 'Rohit':");

jala.remove(2,"Rohit");

System.out.println(jala);

System.out.println("Copied all the elements of the jala to icfai:");

java.util.HashMap<Integer, String> icfai = new java.util.HashMap<>();

icfai.putAll(jala);

System.out.println("icfai="+icfai);

}

}

1. Create a HashSet with at least 10 elements of type String

Write program covering all the operations of HashSet.

import java.util.Iterator;

public class HashSet

{

public static void main(String[] args)

{

java.util.HashSet<String> jala = new java.util.HashSet<>();

jala.add("zero");

jala.add("one");

jala.add("two");

jala.add("three");

jala.add("four");

jala.add("five");

jala.add("six");

jala.add("seven");

jala.add("eight");

jala.add("nine");

jala.add("ten");

jala.add("nine");

System.out.println("\nAfter adding elements to HashSet :");

System.out.println("jala="+jala);

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

System.out.println("Iterating through the HashSet:");

while (itr.hasNext())

{

System.out.print(itr.next()+ " ");

}

System.out.println(" ");

System.out.println("clone/copy of HashSet:");

System.out.println(jala.clone());

jala.remove("zero");

System.out.println("After removing element 'zero':");

System.out.println("jala="+jala);

System.out.println("Checking if the set is empty:");

System.out.println(jala.isEmpty());

System.out.println("Size of the HashSet:");

System.out.println(jala.size());

System.out.println("Element 'one' is present in the HashSet:");

System.out.println(jala.contains("one"));

System.out.println("After removing all elements of the HashSet:");

jala.clear();

System.out.println("jala="+jala);

}

}