**ASSIGNMENTS-1**

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

Answere:A class is a group of objects which have common properties. It is a template or blueprint from which objects are created. It is a logical entity. It can't be physical.

public class Student {  
}

METHOD: A method must be declared within a class. It is defined with the name of the method, followed by parentheses ()

Signature: A signature in java is a combination of elements in a list such as constructor and methods thereby distinguishing them from other constructors and methods.

Object: The **object** is a basic building block of an [OOPs](https://www.javatpoint.com/java-oops-concepts) language. In **Java**, we cannot execute any program without creating an **object**.

2. Write a program to print your name.

public class Main {

public static void main(String[] args) {

System.out.println("Preeti!");

}

}

Output:

Preeti!

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

Single line comments

public class CommentsExample {

public static void main(String args[]) {

//Declaring a variable named num

int num = 1;

//Printing the value of the variable num

System.out.println("value if the variable num: "+num);

}

}

**Output**

value if the variable num: 1

Multiple line comments

public class CommentsExample {

/\*

Following is the main method here,

We create a variable named num.

And, print its value

\* \*/

public static void main(String args[]) {

int num = 1;

System.out.println("value if the variable num: "+num);

}

}

Documentation comments

/\*\*

\* The HelloWorld program implements an application that

\*/

public class HelloWorld {

public static void main(String[] args) {

// Prints Hello, World! on standard output.

System.out.println("Hello World!");

}

}

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

ANSWERE:

Int data type

The int data type is a 32-bit signed two's complement integer.

**Example:** int a = 100000, int b = -200000

Long datatype:

The long data type is a 64-bit two's complement integer.

**Example:** long a = 100000L, long b = -200000L

## Float Data Type

The float data type is a single-precision 32-bit IEEE 754 floating point.

Char data type

The char data type is a single 16-bit Unicode character.

**Example:** char letterA = 'A'

Boolean data type:

The Boolean data type is used to store only two possible values: true and false. This data type is used for simple flags that track true/false conditions.

**Example:** Boolean one = false

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

public class Main {

static void myMethod() {

System.out.println("preeti!");

}

public static void main(String[] args) {

myMethod();

}

}

Output:

preeti!

ASSIGNNMENT-2

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

Static int addTwoNumber(int A, int B)

{

    // Return sum of A and B

    return A - (-B);

}

Public static void main(String args[])

{

    // Given two number

    int A = 4, B = 11;

System.out.println(“sum+”addTwoNumber(A,B));

return 0;

}

Static int addTwoNumber(int A, int B)

{

    // Return sum of A and B

    return A +B);

}

Public static void main(String args[])

{

    // Given two number

    int A = 4, B = 11;

Static int multiplyTwoNumber(int A, int B)

{

    // Return sum of A and B

    return (A \*B);

}

Public static void main(String args[])

{

    // Given two number

    int A = 4, B = 11;

System.out.println(“sum+”multiplyTwoNumber(A,B));

return 0;

}

Static int divideTwoNumber(int A, int B)

{

    // Return sum of A and B

    return A/B;

}

Public static void main(String args[])

{

    // Given two number

    int A = 4, B = 11;

System.out.println(“sum+”divideTwoNumber(A,B));

return 0;

}

2. Program to equal operator and not equal operators

import java.io.\*;

public class A{

    public static void main(String[] args)

    {

        int a = 4;

        int b = 4;

        int c = 5;

        // Comparing a and b using == operator

        System.out.println("Are " + a

                           + " and " + b

                           + " equal? "

                           + (a == b));

        // Comparing b and c using == operator

        System.out.println("Are " + b

                           + " and " + c

                           + " equal? "

                           + (b == c));

    }

}

Are 4 and 4 equal? true

Are 4 and 5 equal? False

3. Write a program to find the two numbers **equal or not**.

import java.io.\*;

class A{

    public static void main(String[] args)

    {

        int firstNumber = 15;

        int secondNumber = 15;

        if (firstNumber == secondNumber)

            System.out.println("Numbers are equal");

        else

            System.out.println("Numbers are not equal");

    }

}

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

public class logicalop {

public static void main(String[] args) {

//Variables Definition and Initialization

boolean bool1 = true, bool2 = false;

//Logical AND

System.out.println("bool1 && bool2 = " + (bool1 && bool2));

//Logical OR

System.out.println("bool1 || bool2 = " + (bool1 | bool2) );

//Logical Not

System.out.println("!(bool1 && bool2) = " + !(bool1 && bool2));

}

}

Output:

bool1 && bool2 = false

bool1 || bool2 = true

!(bool1 && bool2) = true

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

public class relatiop {

public static void main(String[] args) {

//Variables Definition and Initialization

int num1 = 12, num2 = 4;

//is equal to

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

//is not equal to

System.out.println("num1 != num2 = " + (num1 != num2) );

//Greater than

System.out.println("num1 > num2 = " + (num1 > num2) );

//Less than

System.out.println("num1 < num2 = " + (num1 < num2) );

//Greater than or equal to

System.out.println("num1 >= num2 = " + (num1 >= num2) );

//Less than equal to

System.out.println(“num1 <= num2 = ” +(num1 <= num2) );

}

}

Output:

num1 == num2 = false

num1 != num2 = true

num1 > num2 = true

num1 < num2 = false

num1 >= num2 = true

num1 <= num2 = false

import java.util.Scanner;

class MyObjectExample {

    public static void main(String args[]) {

        System.out.println("Enter the num");

        Scanner a = new Scanner(System.in);

        int x = a.nextInt();

        int y = a.nextInt();

        int z = a.nextInt();

        int x1 = a.nextInt();

        int[] arr = { x, y, z, x1 };

        int large = arr[0];

        int small = arr[0];

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

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

            if (arr[i] > large) {

                large = arr[i];

            }

            if (arr[i] < small) {

                small = arr[i];

            }

        }

        System.out.println("Large Num:" + large + "=Small Num=" + small);

    }

}

// Program to print a text 5 times

class Main {

public static void main(String[] args) {

int n = 5;

// for loop

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

System.out.println("Java is fun");

}

}

}

Assignment 4

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

Ans:

Class A{

Public static void main(String args[]){

Int n=10;

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

{

System.out.println(“Bright IT career “);

}

}

}

Output:

Bright IT career

Bright IT career

Bright IT career

Bright IT career

Bright IT career

Bright IT career

Bright IT career

Bright IT career

Bright IT career

Bright IT career

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

class Main {

public static void main(String[] args) {

// declare variables

int i = 1, n = 20;

// while loop from 1 to 20

while(i <= n) {

System.out.println(i);

i++;

}

}

}

Output:

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

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

import java.util.Scanner;

public class EvenOdd {

public static void main(String[] args) {

Scanner reader = new Scanner(System.in);

System.out.print("Enter a number: ");

int num = reader.nextInt();

if(num % 2 == 0)

System.out.println(num + " is even");

else

System.out.println(num + " is odd");

}

}

Output

Enter a number: 12

12 is even

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

public class Largest {

public static void main(String[] args) {

double n1 = -4.5, n2 = 3.9, n3 = 2.5;

if( n1 >= n2 && n1 >= n3)

System.out.println(n1 + " is the largest number.");

else if (n2 >= n1 && n2 >= n3)

System.out.println(n2 + " is the largest number.");

else

System.out.println(n3 + " is the largest number.");

}

}

Output

3.9 is the largest number.

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

1. **import** java.util.Scanner;
2. **public** **class** DisplayEvenNumbersExample3
3. {
4. **public** **static** **void** main(String[] args)
5. {
6. **int** number, i;
7. Scanner sc=**new** Scanner(System.in);
8. System.out.print("Enter the limit: ");
9. number = sc.nextInt();
10. i=2;
11. System.out.print("Lit of even numbers: ");
12. //the while loop executes until the condition become false
13. **while**(i<=number)
14. {
15. //prints the even number
16. System.out.print(i +" ");
17. //increments the variable i by 2
18. i=i+2;
19. }
20. }
21. }

**Output:**

Enter the limit: 100

Lit of even numbers: 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50 52 54 56 58 60 62 64 66 68 70 72 74 76 78 80 82 84 86 88 90 92 94 96 98 100

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

package IncludeHelp;

public class Print\_1\_To\_10\_UsingWhile

{

public static void main(String args[])

{

//loop counter initialisation

int i=1;

//print statement

System.out.println("Output is : ");

//loop to print 1 to 10.

while(i<=10)

{

System.out.println(i);

i++;

}

}

}

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

import java.util.Scanner;

class Main {

public static void main(String[] args) {

int i = 1, n = 5;

// do...while loop from 1 to 5

do {

System.out.println(i);

i++;

} while(i <= n);

}

}

Output

1

2

3

4

5

8.Write a program to find Armstrong number or not

class ArmstrongExample{

public static void main(String[] args) {

int c=0,a,temp;

int n=153;//It is the number to check armstrong

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");

}

}

Output:

armstrong number

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

public class Main {

public static void main(String[] args) {

int num = 29;

boolean flag = false;

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

// condition for nonprime number

if (num % i == 0) {

flag = true;

break;

}

}

.  if (!flag)

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

else

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

}

}

Output

29 is a prime number.

10. Write a program to palindrome or not

class PalindromeExample{

public static void main(String args[]){

int r,sum=0,temp;

int n=454;//It is the number variable to be checked for palindrome

temp=n;

while(n>0){

r=n%10; //getting remainder

sum=(sum\*10)+r;

n=n/10;

}

if(temp==sum)

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

else

System.out.println("not palindrome");

}

}

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

|  |
| --- |
| Class A  {  public static void main(String args[])  {      // variable declaration      int n1 = 10, n2 = 20, n3 = 30, max;        // Largest among n1, n2 and n3      max = (n1 > n2) ?            (n1 > n3 ? n1 : n3) :            (n2 > n3 ? n2 : n3);        // Print the largest number      System.out.println("Largest number among " + n1 +                               ", " + n2 + " and " + n3 +                                   " is " + max + ". " );  }  } |

**Output:**

Largest number among 10,20 and 30 is 30

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

import java.util.Scanner;

class CheckOddEvenswitch2{

public static void main(String args[]){

Scanner scan=new Scanner(System.in); //create a scanner object for input

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

int num=scan.nextInt();//get input from the user for num

switch(num%2){//this will return either 0 or 1

case 0:

System.out.println(num+" is a Even number");

break;

case 1:

System.out.println(num+" is a Odd number");

}

}

}

Case 1

Enter the integer number: 23

23 is a odd number

Case 2

Enter the integer number: 30

30 is a even number

Assignment-4

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

import java.io.\*;

import java.lang.\*;

import java.util.\*;

class GFG {

public static int[] addX(int n, int arr[], int x)

{

int i;

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

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

newarr[i] = arr[i];

newarr[n] = x;

return newarr;

}

public static void main(String[] args)

{

int n = 10;

int i;

// initial array of size 10

int arr[]

= { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };

// print the original array

System.out.println("Initial Array:\n"

+ Arrays.toString(arr));

// element to be added

int x = 50;

// call the method to add x in arr

arr = addX(n, arr, x);

// print the updated array

System.out.println("\nArray with " + x

+ " added:\n"

+ Arrays.toString(arr));

}

}

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

class GFG {

// Function that return average of an array.

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

{

// Find sum of array element

int sum = 0;

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

sum += a[i];

return (double)sum / n;

}

//driver code

public static void main (String[] args)

{

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

int n = arr.length;

System.out.println(average(arr, n));

}

}

Output

6

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

import java.util.\*;

public class index {

// Linear-search function to find the index of an element

public static int findIndex(int arr[], int t)

{

// if array is Null

if (arr == null) {

return -1;

}

// find length of array

int len = arr.length;

int i = 0;

// traverse in the array

while (i < len) {

// if the i-th element is t

// then return the index

if (arr[i] == t) {

return i;

}

else {

i = i + 1;

}

}

return -1;

}

// Driver Code

public static void main(String[] args)

{

int[] my\_array = { 5, 4, 6, 1, 3, 2, 7, 8, 9 };

// find the index of 5

System.out.println("Index position of 5 is: "

+ findIndex(my\_array, 5));

// find the index of 7

System.out.println("Index position of 7 is: "

+ findIndex(my\_array, 7));

}

}

Output:

Index position of 5 is: 0

Index position of 7 is: 6

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

public class Exercise5 {

public static boolean contains(int[] arr, int item) {

for (int n : arr) {

if (item == n) {

return true;

}

}

return false;

}

public static void main(String[] args) {

int[] my\_array1 = {

1789, 2035, 1899, 1456, 2013,

1458, 2458, 1254, 1472, 2365,

1456, 2265, 1457, 2456};

System.out.println(contains(my\_array1, 2013));

System.out.println(contains(my\_array1, 2015));

}

}

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

function removeItemFromArray(array, n) {

const newArray = [];

for ( let i = 0; i < array.length; i++) {

if(array[i] !== n) {

newArray.push(array[i]);

}

}

return newArray;

}

const result = removeItemFromArray([1, 2, 3 , 4 , 5], 2);

console.log(result);

Output

[1, 3, 4, 5]

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

public class Test {

public static void main(String[] args)

{

int a[] = { 1, 8, 3 };

// Create an array b[] of same size as a[]

int b[] = new int[a.length];

// Copy elements of a[] to b[]

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

b[i] = a[i];

// Change b[] to verify that

// b[] is different from a[]

b[0]++;

System.out.println("Contents of a[] ");

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

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

System.out.println("\n\nContents of b[] ");

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

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

}

}

Output

Contents of a[]

1 8 3

Contents of b[]

2 8 3

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

// Java Program to Insert an element

// at a specific position in an Array

import java.io.\*;

import java.lang.\*;

import java.util.\*;

class GFG {

// Function to insert x in arr at position pos

public static int[] insertX(int n, int arr[],

int x, int pos)

{

int i;

// create a new array of size n+1

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

// insert the elements from

// the old array into the new array

// insert all elements till pos

// then insert x at pos

// then insert rest of the elements

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;

}

// Driver code

public static void main(String[] args)

{

int n = 10;

int i;

// initial array of size 10

int arr[]

= { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };

// print the original array

System.out.println("Initial Array:\n"

+ Arrays.toString(arr));

// element to be inserted

int x = 50;

// position at which element

// is to be inserted

int pos = 5;

// call the method to insert x

// in arr at position pos

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

// print the updated array

System.out.println("\nArray with " + x

+ " inserted at position "

+ pos + ":\n"

+ Arrays.toString(arr));

}

}

Output:

Initial Array:

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

Array with 50 inserted at position 5:

[1, 2, 3, 4, 50, 5, 6, 7, 8, 9, 10]

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

|  |
| --- |
| import java.io.\*;    class GFG {        static int getMin(int arr[], int n)      {          int res = arr[0];            for (int i = 1; i < n; i++)              res = Math.min(res, arr[i]);          return res;      }        static int getMax(int arr[], int n)      {          int res = arr[0];            for (int i = 1; i < n; i++)              res = Math.max(res, arr[i]);          return res;      }        // Driver code      public static void main (String[] args)      {          int arr[] = { 12, 1234, 45, 67, 1 };          int n = arr.length;          System.out.println( "Minimum element"             + " of array: " + getMin(arr, n));          System.out.println( "Maximum element"             + " of array: " + getMax(arr, n));      }  } |

**Output:**

Minimum element of array: 1

Maximum element of array: 1234

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

|  |
| --- |
| public class A{      static void rvereseArray(int arr[],                      int start, int end)      {          int temp;            while (start < end)          {              temp = arr[start];              arr[start] = arr[end];              arr[end] = temp;              start++;              end--;          }      }        /\* Utility that prints out an      array on a line \*/      static void printArray(int arr[], int size)      {          for (int i = 0; i < size; i++)               System.out.print(arr[i] + " ");             System.out.println();      }      public static void main(String args[]) {           int arr[] = {1, 2, 3, 4, 5, 6};          printArray(arr, 6);          rvereseArray(arr, 0, 5);          System.out.print("Reversed array is \n");          printArray(arr, 6)  }}} |

**Output :**

1 2 3 4 5 6

Reversed array is

6 5 4 3 2 1

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

import java.util.HashMap;

import java.util.Map;

import java.util.Map.Entry;

public class FindDuplicatedInArray

{

// Driver program

public static void main(String[] args)

{

int arr[] = {12, 11, 40, 12, 5, 6, 5, 12, 11};

int n = arr.length;

printDuplicates(arr, n);

}

// function to find and print duplicates

private static void printDuplicates(int[] arr, int n)

{

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

int count = 0;

boolean dup = false;

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

if(map.containsKey(arr[i])){

count = map.get(arr[i]);

map.put(arr[i], count + 1);

}

else{

map.put(arr[i], 1);

}

}

for(Entry<Integer,Integer> entry : map.entrySet())

{

// if frequency is more than 1

// print the element

if(entry.getValue() > 1){

System.out.print(entry.getKey()+ " ");

dup = true;

}

}

// no duplicates present

if(!dup){

System.out.println("-1");

}

}

}

Output

5 12 11

Assignment -5

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

public class Demo{

static int my\_count=2;

public void increment(){

my\_count++;

}

public static void main(String args[]){

Demo obj\_1=new Demo();

Demo obj\_2=new Demo();

obj\_1.increment();

obj\_2.increment();

System.out.println("The count of first object is "+obj\_1.my\_count);

System.out.println("The count of second object is "+obj\_2.my\_count);

}

}

Output

The count of first object is 4

The count of second object is 4

public class VariableExample{

int myVariable;

static int data = 30;

public static void main(String args[]){

int a = 100;

VariableExample obj = new VariableExample();

System.out.println("Value of instance variable myVariable: "+obj.myVariable);

System.out.println("Value of static variable data: "+VariableExample.data);

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

}

}

Output

Value of instance variable myVariable: 0

Value of static variable data: 30

Value of local variable a: 100

public class Demo{

public static void main(String args[]){

Student s1 = new Student();

s1.showData();

Student s2 = new Student();

s2.showData();

//Student.b++;

//s1.showData();

}

}

class Student {

int a; //initialized to zero

static int b; //initialized to zero only when class is loaded not for each object created.

Student(){

//Constructor incrementing static variable b

b++;

}

public void showData(){

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

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

}

//public static void increment(){

//a++;

//}

}

Main method:

class OverloadMain

{

public static void main(int a) //overloaded main method

{

System.out.println(a);

}

public static void main(String args[])

{

System.out.println("main method incoked");

main(6);

}

}

Output:

main method invoked

6

2. Print instance variables in static methods

public class Test {

public int instanceVariable = 10;

public static void main(String args[]) {

Test test = new Test();

System.out.println(test.instanceVariable);

}

}

Output:

10

4. Call instance methods in static methods

Static methods can always call instance methods - so long as they have a reference to an instance on which to call the method.

For example:public static void main(String[] args) {

String foo = "hello";

System.out.println(foo.length());

}

length() is an instance method on String, main is a static method,

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

public class Demo{

public static void main(String args[]){

Student s1 = new Student();

s1.showData();

Student s2 = new Student();

s2.showData();

//Student.b++;

//s1.showData();

}

}

class Student {

int a; //initialized to zero

static int b; //initialized to zero only when class is loaded not for each object created.

Student(){

//Constructor incrementing static variable b

b++;

}

public void showData(){

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

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

}

//public static void increment(){

//a++;

//}

}

1. Different ways creating a string

We can create a String by –

Step 1 − Assigning a string value wrapped in " " to a String type variable.

String message = "Hello Welcome “;

Step 2 − Creating an object of the String class using the new keyword by passing the string value as a parameter of its constructor.

String message = new String ("Hello Welcome “);

Step 3 − Passing a character array to the String constructor.

char arr[] = {'H','e','l','l','o'};

String message = new String(arr);

1. Concatenating two strings using + operator.

import java.util.Scanner;

public class StringExample {

   public static void main(String args[]) {

      Scanner sc = new Scanner(System.in);

      System.out.println("Enter the first string: ");

      String str1 = sc.next();

      System.out.println("Enter the second string: ");

      String str2 = sc.next();

      //Concatenating the two Strings

      String result = str1+str2;

      System.out.println(result);

   }

}

**Output**

Enter the first string:

Krishna

Enter the second string:

Kasyap

KrishnaKasyap

Java

1. Finding the length of the string

import java.io.\*;

public class Test {

public static void main(String args[]) {

String Str1 = new String("Welcome ");

String Str2 = new String("Tutorials" );

System.out.print("String Length :" );

System.out.println(Str1.length());

System.out.print("String Length :" );

System.out.println(Str2.length());

}

}

4. Extract a string using Substring

public class Substr1 {

public static void main(String args[])

{

// Initializing String

String Str = new String("Welcome ");

// using substring() to extract substring

// returns geeksforgeeks

System.out.print("The extracted substring is : ");

System.out.println(Str.substring(10));

}

}

5.Searching in strings using indexOf()

public class Index1 {

public static void main(String args[])

{

// Initialising String

String gfg = new String("Welcome to geeksforgeeks");

System.out.print("Found g first at position : ");

// Initial index of 'g' will print

// prints 11

System.out.println(gfg.indexOf('g'));

}

}

Found g first at position : 11

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

import java.io.\*;

public class Test {

public static void main(String args[]) {

String Str = new String("Welcome to Tutorialspoint.com");

System.out.print("Return Value :" );

System.out.println(Str.matches("(.\*)Tutorials(.\*)"));

System.out.print("Return Value :" );

System.out.println(Str.matches("Tutorials"));

System.out.print("Return Value :" );

System.out.println(Str.matches("Welcome(.\*)"));

}

}

## Output

Return Value :true

Return Value :false

Return Value :true

1. Comparing strings using the methods equals(),

public class JavaExample{

public static void main(String args[]){

String str1= new String("Hello");

String str2= new String("Hi");

String str3= new String("Hello");

System.out.println("str1 equals to str2:"+str1.equals(str2));

System.out.println("str1 equals to str3:"+str1.equals(str3));

System.out.println("str1 equals to Welcome:"+str1.equals("Welcome"));

System.out.println("str1 equals to Hello:"+str1.equals("Hello"));

System.out.println("str1 equals to hello:"+str1.equals("hello"));

}

}

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

public class JavaExample{

public static void main(String args[]){

String str1= new String("Apple");

String str2= new String("MANGO");

String str3= new String("APPLE");

System.out.println("str1 equals to str2:"+str1.equalsIgnoreCase(str2));

System.out.println("str1 equals to str3:"+str1.equalsIgnoreCase(str3));

System.out.println("str1 equals to Welcome:"+str1.equalsIgnoreCase("Welcome"));

System.out.println("str1 equals to Apple:"+str1.equalsIgnoreCase("Apple"));

System.out.println("str2 equals to mango:"+str2.equalsIgnoreCase("mango"));

}

}

public class StringComparison {

public static void main(String[] args) {

String str1 = "run";

String str2 = new String("run");

String str3 = "gun";

String str4 = "sun";

//equal so returns 0

System.out.println(str1.compareTo(str2));

// Comparison with in a condition

// that's how generally used

if(str1.compareTo(str2) == 0){

System.out.println("str1 is equal to str2");

}else{

System.out.println("str1 is not equal to str2");

}

//str1 > str3 so returns positive integer

System.out.println(str1.compareTo(str3));

// with condition

if(str1.compareTo(str3) > 0){

System.out.println("str1 is greater than str3");

}else{

System.out.println("str1 is less than str3");

System.out.println(str1.compareTo(str4));

}

}

Output

0

str1 is equal to str2

11

str1 is greater than str3

-1

public class StringComparison {

public static void main(String[] args) {

String str = "This is a test string";

// Should be true

System.out.println(str.startsWith("This"));

// test start at index 10, so returns true

System.out.println(str.startsWith("test", 10));

// returns false

System.out.println(str.endsWith("test"));

// returns true

System.out.println(str.endsWith("test string"));

// returns true

System.out.println(str.endsWith("string"));

}

}

Output

true

true

false

true

true

10 Converting to uppercase and lowercase

import java.util.Scanner;

public class UpperToLower1 {

private static Scanner sc;

public static void main(String[] args) {

String uppStr;

sc= new Scanner(System.in);

System.out.print("\nEnter Uppercase String to convert = ");

uppStr = sc.nextLine();

String uppStr2 = uppStr.toLowerCase();

System.out.println("\nThe Lowercase String = " + uppStr2);

}}

Assignment-8

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.

class Test {

// private variables

private int age;

private String name;

// initialize age

public void setAge(int age) {

this.age = age;

}

// initialize name

public void setName(String name) {

this.name = name;

}

// access age

public int getAge() {

return this.age;

}

// access name

public String getName() {

return this.name;

}

}

class Main {

public static void main(String[] args) {

// create an object of Test

Test test = new Test();

// set value of private variables

test.setAge(24);

test.setName("Programiz");

// get value of private variables

System.out.println("Age: " + test.getAge());

System.out.println("Name: " + test.getName());

}

}

**Output**

Age: 24

Name: Programiz

package pack;

class A{

void msg(){System.out.println("Hello");}

package mypack;

import pack.\*;

class B{

public static void main(String args[]){

A obj = new A();//Compile Time Error

obj.msg();

}

}

3. Create a class with PROTECTED fields and methods. Access these fields and methods  from any other class in the different package.

Addition.java

package abcpackage;

public class Addition {

protected int addTwoNumbers(int a, int b){

return a+b;

}

}

Test.java

package xyzpackage;

import abcpackage.\*;

class Test extends Addition{

public static void main(String args[]){

Test obj = new Test();

System.out.println(obj.addTwoNumbers(11, 22));

}

}

Output:33

3.Access the public methods and fields from any class in the same package or different  package

public class HelloWorld

{

private String format;

public String getFormat() {

return this.format;

}

public void setFormat(String format) {

this.format = format;

}

}

Assignment-9

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

abstract class AbstractDemo { // Abstract class

private int i = 0;

public void display() { // non-abstract method

System.out.print("Welcome to Tutorials Point");

}

}

public class InheritedClassDemo extends AbstractDemo {

public static void main(String args[]) {

AbstractDemo demo = new InheritedClassDemo();

demo.display();

}

}

|  |
| --- |
| abstract class Content {      int a;        // Constructor of Content      public Content(int a) { this.a = a; }        // Abstractact Method in Abstract class      abstract int mutiply(int val);  }    // It is the Child class of abstract class Content  class GFG extends Content {      GFG() { super(2); }      public int mutiply(int val) { return this.a \* val; }  }    // Main class  public class Main {      // Main Method in java      public static void main(String args[])          Content c = new GFG;          System.out.println(c.mutiply(3));      }  } |

**Output**

6

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

abstract class Language {

// fields and methods

}

Language obj = new Language();

abstract class Language {

abstract void method1();

void method2() {

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

}

}

ASSIGNMENT-10

1. Create an interface with only one method and implement it in a class. Call the method  implemented.
2. **interface** printable{
3. **void** print();
4. }
5. **class** A6 **implements** printable{
6. **public** **void** print()
7. {System.out.println("Hello");}
9. **public** **static** **void** main(String args[]){
10. A6 obj = **new** A6();
11. obj.print();
12. }

}

1. Use Interface instances to call the implemented method in the implemented class

public interface InterfaceDemo{

    default public void displayNameDefault(String name){

       System.out.println("Your name is : " + name);

   }

    public void displayName(String name);

    public void displayNameAndDesignation(String name, String designation);

}

public class InterfaceDemoImpl implements InterfaceDemo{

    public void displayName(String name) {

         System.out.println(name);

    }

      public void displayNameAndDesignation(String name, String designation) {

           System.out.println("Name:" + name + "\n"+ "Designation:" + designation);

     }

}

public class CallInterfaceMethod {

     public static void main(String args[]){

         InterfaceDemo demo = new InterfaceDemoImpl();

         demo.displayName("Adithya");

         demo.displayNameAndDesignation("Adithya", "Java Developer");

         demo.displayNameDefault("Adithya");

      }

}

**Output**

Adithya

Name:Adithya

Designation:Java Developer

Your name is : Adithya

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

interface Pet{

public void test();

}

class Dog implements Pet{

public void test(){

System.out.println("Interface Method Implemented");

}

public static void main(String args[]){

Pet p = new Dog();

p.test();

}

}

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

public interface InterfaceX

{

public int geek();

}

public interface InterfaceY

{

public String geek();

}

public class Testing implements InterfaceX, InterfaceY

{

public String geek()

{

return "hello";

}

}

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

|  |
| --- |
| class sample implements intfB  {      @Override      public void geekName()      {          System.out.println("Rohit");      }        @Override      public void geekInstitute()      {          System.out.println("JIIT");      }        public static void main (String[] args)      {          sample ob1 = new sample();            // calling the method implemented          // within the class.          ob1.geekName();          ob1.geekInstitute();      }  } |

Output:

Rohit

JIIT

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

**interface** Sayable{

default void say() {

saySomething();

}

// **Private method** inside **interface**.

**private** void saySomething() {

System.out.println("Hello... I'm **private method**");

}

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

If the fields of the interface are private, you cannot access them in the implementing class.

In general, the protected fields can be accessed in the same class or, the class inheriting it. But, we do not inherit an interface we will implement it.

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

class Student{

int rollno;

String name;

float fee;

Student(int rollno,String name,float fee){

this.rollno=rollno;

this.name=name;

this.fee=fee;

}

void display(){System.out.println(rollno+" "+name+" "+fee);}

}

class TestThis2{

public static void main(String args[]){

Student s1=new Student(111,"ankit",5000f);

Student s2=new Student(112,"sumit",6000f);

s1.display();

s2.display();

}}

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

class Animal{

String color="white";

}

class Dog extends Animal{

String color="black";

void printColor(){

System.out.println(color);//prints color of Dog class.

System.out.println(super.color);//prints color of Animal class.

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

class A{

A(){System.out.println("hello a");}

A(int x){

this();

System.out.println(x);

}

}

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

// Java code for using this() to

// invoke current class constructor

class Test

{

int a;

int b;

//Default constructor

Test()

{

this(10, 20);

System.out.println("Inside default constructor \n");

}

//Parameterized constructor

Test(int a, int b)

{

this.a = a;

this.b = b;

System.out.println("Inside parameterized constructor");

}

public static void main(String[] args)

{

Test object = new Test();

}

}

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

class Animal{

Animal(){System.out.println("animal is created");}

}

class Dog extends Animal{

Dog(){

super();

System.out.println("dog is created");

}

}

class TestSuper3{

public static void main(String args[]){

Dog d=new Dog();

}}

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

We can use super() as well this() only once inside constructor. If we use super() twice or this() twice or super() followed by this() or this() followed by super(), then immediately we get compile time error i.e, we can use either super() or this() as first statement inside constructor and not both.

ASSIGNMENT-13

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

public class MethodOverloading

{

void add(int a,int b)

{

System.out.println(a+b);

}

void add(int a, int b, int c, int d)

{

System.out.println(a+b+c+d);

}

public static void main(String args[]){

MethodOverloading obj=new MethodOverloading();

obj.add(10,100);

obj.add(25,35,45,55);

}

}

Output:

110

160

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

public class MethodOverloading

{

void add(int a,int b)

{

System.out.println(a+b);

}

void add(float a,float b)

{

System.out.println(a+b);

}

public static void main(String args[]){

MethodOverloading obj=new MethodOverloading();

obj.add(15f,15f);

obj.add(25,25);

}

}

Output:

30.0

50

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

Class Test{

public int division(int a, int b){

int result = a/b;

return result;

}

public double division (float a, float b){

double result = a/b;

return result;

}

}

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

public class ExceptionHandled

{

public static void main(String args[])

{

int a =10, b = 0, c;

System.out.println("Hello 1");

try

{

c = a/b;

System.out.println(c);

}

catch(ArithmeticException e)

{

System.out.println("Do not divide by zero sir." + e );

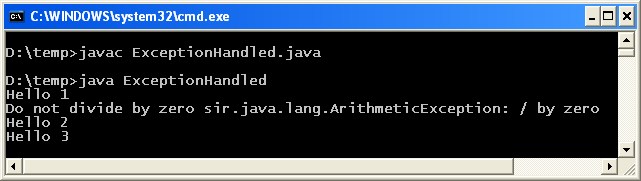
}

System.out.println("Hello 2");

System.out.println("Hello 3");

}

}

[](https://i0.wp.com/way2java.com/wp-content/uploads/2014/09/ima12.png)

1. Handle the Arithmetic exception using try-catch block

package TIHException;

public class HandleArithmeticException {

public static void main(String[] args) {

**try**{

int firstNum = 25;

int secondNum = 20%2;

int div = firstNum/secondNum;

System.out.println("Result : "+ div);

}**catch**(**ArithmeticException** ae){

System.out.println("Arithmetic Exception occurred in code");

}

System.out.println("After division");

}

}

1. Write a method which throws exception, Call that method in main class without try block

// Java program to demonstrate working of try,

// catch and finally

// Java program to demonstrate working of try,

// catch and finally

class Division {

public static void main(String[] args)

{

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

try {

result = a / (b - c);

System.out.println("result" + result);

}

catch (ArithmeticException e) {

System.out.println("Exception caught:Division by zero");

}

finally {

System.out.println("I am in final block");

}

}

} }

}

1. Write a program with multiple catch blocks

public class MultipleCatchBlock1 {

public static void main(String[] args) {

try{

int a[]=new int[5];

a[5]=30/0;

}

catch(ArithmeticException e)

{

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

}

catch(ArrayIndexOutOfBoundsException e)

{

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

}

catch(Exception e)

{

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

}

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

}

}

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

class TestCustomException1{

static void validate(int age)throws InvalidAgeException{

if(age<18)

throw new InvalidAgeException("not valid");

else

System.out.println("welcome to vote");

}

public static void main(String args[]){

try{

validate(13);

}catch(Exception m){System.out.println("Exception occured: "+m);}

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

}

}

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

package TIHException;

public class HandleArithmeticException {

public static void main(String[] args) {

**try**{

int firstNum = 25;

int secondNum = 20%2;

int div = firstNum/secondNum;

System.out.println("Result : "+ div);

}**catch**(**ArithmeticException** ae){

System.out.println("Arithmetic Exception occurred in code");

}

System.out.println("After division");

}

}

**OUTPUT**

Arithmetic Exception occurred in code

After division

1. Write a program to generate ArrayIndexOutOfBoundException

public class ArrayIndexOutOfBoundException {

public static void main(String[] args) {

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

//Declaring 4 elements in the String array

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

//Here, no element is present at the iteration number arr.length, i.e 4

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

//So it will throw ArrayIndexOutOfBoundException at iteration 4

}

}

}

Output:

ArrayIndexOutOfBoundsException in Java

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

try {

// create a reader

FileInputStream fis = new FileInputStream(new File("input.txt"));

// read one byte at a time

int ch;

while ((ch = fis.read()) != -1) {

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

}

// close the reader

fis.close();

} catch (IOException ex) {

ex.printStackTrace();

}

This

is

an

example

file.

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

import java.io.\*;

class FileExample {

public static void main(String[] args)

throws IOException

{

int i;

// create a fileoutputstream object

FileOutputStream fout = new FileOutputStream("../files/name3.txt",

true);

// we need to transfer this string to files

String st = "TATA";

char ch[] = st.toCharArray();

for (i = 0; i < st.length(); i++) {

// we will write the string by writing each

// character one by one to file

fout.write(ch[i]);

}

// by doing fout.close() all the changes which have

// been made till now in RAM had been now saved to

// hard disk

fout.close();

}

}

* 1. Read text from a .txt file using BufferedInputStream

package beginnersbook.com;

import java.io.\*;

public class ReadFileDemo {

public static void main(String[] args) {

//Specify the path of the file here

File file = new File("C://myfile.txt");

BufferedInputStream bis = null;

FileInputStream fis= null;

try

{

//FileInputStream to read the file

fis = new FileInputStream(file);

/\*Passed the FileInputStream to BufferedInputStream

\*For Fast read using the buffer array.\*/

bis = new BufferedInputStream(fis);

/\*available() method of BufferedInputStream

\* returns 0 when there are no more bytes

\* present in the file to be read\*/

while( bis.available() > 0 ){

System.out.print((char)bis.read());

}

}catch(FileNotFoundException fnfe)

{

System.out.println("The specified file not found" + fnfe);

}

catch(IOException ioe)

{

System.out.println("I/O Exception: " + ioe);

}

finally

{

try{

if(bis != null && fis!=null)

{

fis.close();

bis.close();

}

}catch(IOException ioe)

{

System.out.println("Error in InputStream close(): " + ioe);

}

}

}

}

4.Write a program to read data from properties file

import java.io.FileOutputStream;

import java.io.IOException;

import java.util.Properties;

public class CreatingPropertiesFile {

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

//Instantiating the properties file

Properties props = new Properties();

//Populating the properties file

props.put("Device\_name", "OnePlus7");

props.put("Android\_version", "9");

props.put("Model", "GM1901");

props.put("CPU", "Snapdragon855");

//Instantiating the FileInputStream for output file

String path = "D:\\ExampleDirectory\\myFile.properties";

FileOutputStream outputStrem = new FileOutputStream(path);

//Storing the properties file

props.store(outputStrem, "This is a sample properties file");

System.out.println("Properties file created......");

}

}

Output

Properties file created.....

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

{

//obtaining input bytes from a file

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

//creating workbook instance that refers to .xls file

HSSFWorkbook wb=new HSSFWorkbook(fis);

//creating a Sheet object to retrieve the object

HSSFSheet sheet=wb.getSheetAt(0);

//evaluating cell type

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

for(Row row: sheet) //iteration over row using for each loop

{

for(Cell cell: row) //iteration over cell using for each loop

{

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

{

case Cell.CELL\_TYPE\_NUMERIC: //field that represents numeric cell type

//getting the value of the cell as a number

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

break;

case Cell.CELL\_TYPE\_STRING: //field that represents string cell type

//getting the value of the cell as a string

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

break;

}

}

System.out.println();

}

}

}

Output:

Name Age Height

Swarit 23.0 5"

Puneet 25.0 6'1"

Swastik 22.0 5'5"

Tejas 12.0 4'9"

Reading XLSX File

All steps will remain same except file format.

Table: employee.xslx

How to Read Excel File in Java

Example of read excel file (.xlsx)

In this example we use XSSFWorkbook class.

import java.io.File;

import java.io.FileInputStream;

import java.util.Iterator;

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

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

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

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

public class XLSXReaderExample

{

public static void main(String[] args)

{

try

{

File file = new File("C:\\demo\\employee.xlsx"); //creating a new file instance

FileInputStream fis = new FileInputStream(file); //obtaining bytes from the file

//creating Workbook instance that refers to .xlsx file

XSSFWorkbook wb = new XSSFWorkbook(fis);

XSSFSheet sheet = wb.getSheetAt(0); //creating a Sheet object to retrieve object

Iterator<Row> itr = sheet.iterator(); //iterating over excel file

while (itr.hasNext())

{

Row row = itr.next();

Iterator<Cell> cellIterator = row.cellIterator(); //iterating over each column

while (cellIterator.hasNext())

{

Cell cell = cellIterator.next();

switch (cell.getCellType())

{

case Cell.CELL\_TYPE\_STRING: //field that represents string cell type

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

break;

case Cell.CELL\_TYPE\_NUMERIC: //field that represents number cell type

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

break;

default:

}

}

System.out.println("");

}

}

catch(Exception e)

{

e.printStackTrace();

}

}

}

Output:

Employee ID Employee Name Salary Designation Department

1223.0 Harsh 20000.0 Marketing Manager Marketing

3213.0 Vivek 15000.0 Financial Advisor Finance

6542.0 Krishna 21000.0 HR Manager HR

9213.0 Sarika 34000.0 Sales Manager Sales

Reading a particular cell value from a excel file (.xlsx)

Table: EmployeeData.xlsx

How to Read Excel File in Java

Example

In the following example, we read the value of the 2nd row and the 2nd column. The row and column counting start from 0. So the program returns "Software Engineer."

How to Read Excel File in Java

//reading value of a particular cell

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.IOException;

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

import org.apache.poi.ss.usermodel.\*;

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

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

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

public class ReadCellExample

{

public static void main(String[] args)

{

ReadCellExample rc=new ReadCellExample(); //object of the class

//reading the value of 2nd row and 2nd column

String vOutput=rc.ReadCellData(2, 2);

System.out.println(vOutput);

}

//method defined for reading a cell

public String ReadCellData(int vRow, int vColumn)

{

String value=; //variable for storing the cell value

Workbook wb=; //initialize Workbook null

try

{

//reading data from a file in the form of bytes

FileInputStream fis=new FileInputStream("C:\\demo\\EmployeeData.xlsx");

//constructs an XSSFWorkbook object, by buffering the whole stream into the memory

wb=new XSSFWorkbook(fis);

}

catch(FileNotFoundException e)

{

e.printStackTrace();

}

catch(IOException e1)

{

e1.printStackTrace();

}

Sheet sheet=wb.getSheetAt(0); //getting the XSSFSheet object at given index

Row row=sheet.getRow(vRow); //returns the logical row

Cell cell=row.getCell(vColumn); //getting the cell representing the given column

value=cell.getStringCellValue(); //getting cell value

return value; //returns the cell value

}

}

Output:

Software Engineer

5. 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 {

// any exceptions need to be caught

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

{

// workbook object

XSSFWorkbook workbook = new XSSFWorkbook();

// spreadsheet object

XSSFSheet spreadsheet

= workbook.createSheet(" Student Data ");

// creating a row object

XSSFRow row;

// This data needs to be written (Object[])

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;

// writing the data into the sheets...

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);

}

}

// .xlsx is the format for Excel Sheets...

// writing the workbook into the file...

FileOutputStream out = new FileOutputStream(

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

workbook.write(out);

out.close();

}

}

Assignment=16

 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

import java.util.\*;

class JavaExample{

public static void main(String args[]){

ArrayList<String> alist=new ArrayList<String>();

alist.add("Steve");

alist.add("Tim");

alist.add("Lucy");

alist.add("Pat");

alist.add("Angela");

alist.add("Tom");

//displaying elements

System.out.println(alist);

//Adding "Steve" at the fourth position

alist.add(3, "Steve");

//displaying elements

System.out.println(alist);

}

}

**Output:**

[Steve, Tim, Lucy, Pat, Angela, Tom]

[Steve, Tim, Lucy, Steve, Pat, Angela, Tom]

Remove all elements of the ArrayList

import java.util.ArrayList;

public class JavaExample {

public static void main(String[] args) {

ArrayList<String> names = new ArrayList<String>();

names.add("Jim");

names.add("Jack");

names.add("Ajeet");

names.add("Chaitanya");

names.set(0, "Lucy");

System.out.println(names);

}

}

import java.util.\*;

class JavaExample{

public static void main(String args[]){

ArrayList<String> alist=new ArrayList<String>();

alist.add("Steve");

alist.add("Tim");

alist.add("Lucy");

alist.add("Pat");

alist.add("Angela");

alist.add("Tom");

//displaying elements

System.out.println(alist);

//Removing "Steve" and "Angela"

alist.remove("Steve");

alist.remove("Angela");

//displaying elements

System.out.println(alist);

//Removing 3rd element

alist.remove(2);

//displaying elements

System.out.println(alist);

}

}

Output:

[Steve, Tim, Lucy, Pat, Angela, Tom]

[Tim, Lucy, Pat, Tom]

[Tim, Lucy, Tom]

Or

import java.util.\*;

public class JavaExample {

public static void main(String args[]) {

/\* Creating ArrayList of type "String" which means

\* we can only add "String" elements

\*/

ArrayList<String> obj = new ArrayList<String>();

/\*This is how we add elements to an ArrayList\*/

obj.add("Ajeet");

obj.add("Harry");

obj.add("Chaitanya");

obj.add("Steve");

obj.add("Anuj");

// Displaying elements

System.out.println("Original ArrayList:");

for(String str:obj)

System.out.println(str);

/\* Add element at the given index

\* obj.add(0, "Rahul") - Adding element "Rahul" at first position

\* obj.add(1, "Justin") - Adding element "Justin" at second position

\*/

obj.add(0, "Rahul");

obj.add(1, "Justin");

// Displaying elements

System.out.println("ArrayList after add operation:");

for(String str:obj)

System.out.println(str);

//Remove elements from ArrayList like this

obj.remove("Chaitanya"); //Removes "Chaitanya" from ArrayList

obj.remove("Harry"); //Removes "Harry" from ArrayList

// Displaying elements

System.out.println("ArrayList after remove operation:");

for(String str:obj)

System.out.println(str);

//Remove element from the specified index

obj.remove(1); //Removes Second element from the List

// Displaying elements

System.out.println("Final ArrayList:");

for(String str:obj)

System.out.println(str);

}

}

Output:

Original ArrayList:

Ajeet

Harry

Chaitanya

Steve

Anuj

ArrayList after add operation:

Rahul

Justin

Ajeet

Harry

Chaitanya

Steve

Anuj

ArrayList after remove operation:

Rahul

Justin

Ajeet

Steve

Anuj

Final ArrayList:

Rahul

Ajeet

Steve

Anuj

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

Write program covering all the operations of HashSet

// Java program to illustrate

// Java.util.HashMap

import java.util.HashMap;

public class GFG {

public static void main(String[] args)

{

// Create an empty hash map

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

// Add elements to the map

map.put("vishal", 10);

map.put("sachin", 30);

map.put("vaibhav", 20);

// Print size and content

System.out.println("Size of map is:- "

+ map.size());

System.out.println(map);

// Check if a key is present and if

// present, print value

if (map.containsKey("vishal")) {

Integer a = map.get("vishal");

System.out.println("value for key"

+ " \"vishal\" is:- " + a);

}

}

}

Output:

[South Africa, Australia, India]

List contains India or not:true

List after removing Australia:[South Africa, India]

Iterating over list:

South Africa

India