https://www.javatpoint.com/interface-in-java

www.javatpoint.com

Interface in Java - Javatpoint

Interface in Java, why use interface with examples and marker/tagged java interface with difference between abstract class and java interface, understanding relationship between class and interfaces, java interface example,what is marker in java, tagged interface in java.

https://www.javatpoint.com/nested-interface

www.javatpoint.com

Java Nested Interface - javatpoint

An interface which is declared within another interface or class is known as nested interface. The nested interfaces are used to group related interfaces so that they can be easy to maintain. The nested interface must be referred by the outer interface or class. It can't be accessed directly.

https://www.tutorialspoint.com/Can-we-define-a-class-inside-a-Java-interface

tutorialspoint.com

Can we define a class inside a Java interface

Can we define a class inside a Java interface - Yes you can define a class inside an interface In general if the methods of the interface use this class and if we are not using it anywhere else we will declare a class within an interface Exampleinterface Library void issueBook Book b voi...

https://www.javatpoint.com/java-default-methods

www.javatpoint.com

Java 8 Default Methods - javatpoint

Java 8 Default Methods with examples and topics on functional interface, anonymous class, lambda for list, lambda for comparable, default methods, method reference, java date and time, java nashorn, java optional, stream, filter etc.

https://www.javatpoint.com/java-9-interface-private-methods

www.javatpoint.com

Java 9 Interface Private Methods - javatpoint

Java 9 Interface Private Methods with examples and topics on functional interface, anonymous class, lambda for list, lambda for comparable, default methods, method reference, java date and time, java nashorn, java optional, stream, filter etc

interface behaviour{

void user();

void name();

void details();

void location();

}

abstract class Good implements behaviour{

Good(){

System.out.println("ABstract constructor");

}

public void name(){

System.out.println("Gowtham");

}

}

class Bad extends Good{

int x;

Bad(){

//super();

System.out.println("Bad constructor");

}

Bad(int x){

this();

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

}

public void user(){

System.out.println("Good");

}

public void details(){

System.out.println("Active");

}

public void location(){

System.out.println("Coimbatore");

}

}

class Test5{

public static void main(String[] args){

Bad ob=new Bad(5);

ob.user();

ob.name();

ob.details();

ob.location();

}

}

https://www.javatpoint.com/java-inner-class

www.javatpoint.com

Java Inner Class - javatpoint

Java inner class or nested class with member inner class, anonymous inner class, local inner class and java static nested class.

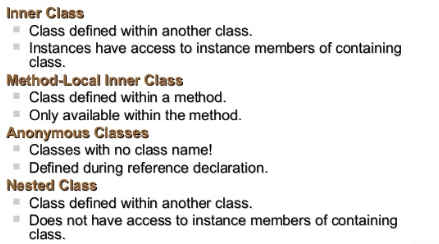
https://www.tutorialspoint.com/java/java\_innerclasses.htm

www.tutorialspoint.com

Java Inner classes

Java Inner classes - Learn Java in simple and easy steps starting from basic to advanced concepts with examples including Java Syntax Object Oriented Language, Methods, Overriding, Inheritance, Polymorphism, Interfaces, Packages, Collections, Networking, Multithreading, Generics, Multimedia, Serialization, GUI.

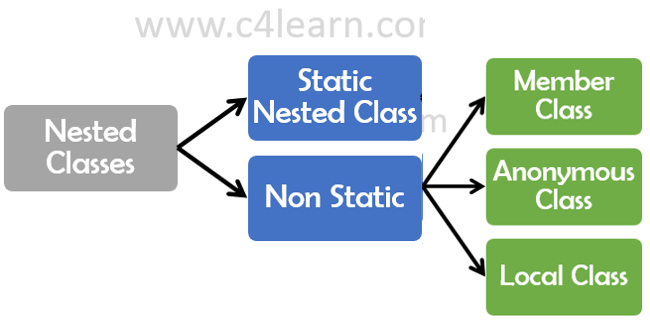
<https://topjavatutorial.com/wp-content/uploads/2016/02/Java-inner-class-definitions.png>



http://www.c4learn.com/java/java-nested-class-overview/

Learn Programming Language Step By Step

Java nested class overview - Java Tutorials - c4learn.com

When we declares a class within the body of another class or interface then class is called as nested class. Three types of inner classes – There are two types of nested classes : static inner classes and non-static nested classes. Non-static nested classes are called inner classes. Three Types of the Inner Classes are …

Why can't we instantiate an interface or an abstract class in java?

https://www.javatpoint.com/array-in-java

www.javatpoint.com

Java Array - Javatpoint

Java array or array in java with single dimensional and multidimensional array with examples and copying array, array length, passing array to method in java and so forth.

/\*\*

\* StringArray

\*/

import java.util.Scanner;

public class StringArray {

// static String a[]={"Bike","Car","pulsar","R15"};

static String a[] = new String[2];

// a[0]="Bike";

public static void main(String[] args) {

selectOptions();

// create();

// display();

// getOne();

// update();

// display();

// remove();

// display();

}

static void selectOptions() {

System.out.println("Select Options");

System.out.println("1.Create\t2.Display\t3.Getone\t4.Update\t5.Remove");

System.out.println("Select Your Option");

Scanner scan = new Scanner(System.in);

int myOption = scan.nextInt();

switch (myOption) {

case 1:

create();

selectOptions();

break;

case 2:

display();

selectOptions();

break;

case 3:

getOne();

selectOptions();

break;

case 4:

update();

selectOptions();

break;

case 5:

remove();

selectOptions();

break;

default:

selectOptions();

break;

}

}

static void getOne() {

System.out.println("getOne");

System.out.println("Enter Name to find:");

Scanner scan = new Scanner(System.in);

String findName = scan.nextLine();

for (String var : a) {

if (findName.equals(var)) {

System.out.println(findName);

break;

}

}

}

static void create() {

System.out.println("create");

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

System.out.println("Enter name:");

Scanner scan = new Scanner(System.in);

String inputName = scan.nextLine();

a[i] = inputName;

}

}

static void display() {

System.out.println("display");

for (String var : a) {

System.out.println(var);

}

}

static void update() {

System.out.println("update");

System.out.println("Enter Name to update:");

Scanner scan = new Scanner(System.in);

String findName = scan.nextLine();

System.out.println("Enter New Name:");

scan = new Scanner(System.in);

String updateName = scan.nextLine();

for (int i = 1; i < a.length; i++) {

if (findName.equals(a[i])) {

a[i] = updateName;

break;

}

}

}

static void remove() {

System.out.println("Remove");

System.out.println("Enter Name to Remove:");

Scanner scan = new Scanner(System.in);

String findName = scan.nextLine();

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

if (findName.equals(a[i])) {

a[i] = "";

break;

}

}

}

}

import java.util.Arrays;

/\*\*

\* FindMins

\*/

public class FindMins {

public static void main(String[] args) {

int[] num = new int[] { 34, 45, 21, 12, 54, 67, 15 };

Arrays.sort(num);

System.out.println(num[0]+","+num[1]);

int[] arr = { 4, 1, 2, 0, 6, -1, 2, 0 };

int smallest = Integer.MAX\_VALUE;

int smaller = Integer.MAX\_VALUE;

int i = 0;

if (arr.length > 2) {

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

if (arr[i] < smallest) {

smaller = smallest;

smallest = arr[i];

} else if (arr[i] < smaller && arr[i] > smallest) {

smaller = arr[i];

}

}

System.out.println("Smallest number is " + smallest);

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

} else {

System.out.println("Invalid array !");

}

}

}

Collapse

public class ArrayMin {

public static void main(String[] args) {

int a[] = { 23,4, 43, 5,2, 54 };

int min = a[0];

int secondMin = a[0];

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

if (min > a[i]) {

secondMin = min;

min = a[i];

// System.out.println("Min values:" + min);

}

// secondMin=min;

else if (secondMin > a[i]) {

secondMin = a[i];

// System.out.println("Min values:" + secondMin);

}

}

System.out.println("Min: " + min);

System.out.println("SecondMin: " + secondMin);

}

// min(a);

}

http://www.javawithus.com/tutorial/array-of-objects

http://www.tutorialdost.com/Java-Programming-Tutorial/15-Java-Array-of-Object.aspx

tutorialdost.com

Java Array of Objects

An object is a variable of class type. As we know, an array is a collection of similar type, threrfore an array can be a collection of class type.

// two column view in pgm

/\*\*

\* StringArray

\*/

import java.util.Scanner;

public class ObjectArray {

// static String a[]={"Bike","Car","pulsar","R15"};

static Employee employees[] = new Employee[2];

// a[0]="Bike";

public static void main(String[] args) {

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

employees[i] = new Employee();//this will call constructor.

}

selectOptions();

// create();

// display();

// getOne();

// update();

// display();

// remove();

// display();

}

static void selectOptions() {

System.out.println("\*\*\*\*\*\* Enter Your Option \*\*\*\*\*\*\*\*");

System.out.println(

"\* 1.Create\t\t\t\*\n\* 2.Display\t\t\t\*\n\* 3.Getone\t\t\t\*\n\* 4.Update\t\t\t\*\n\* 5.Remove\t\t\t\*\n\* 6.Exit\t\t\t\*");

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

System.out.println("Your Selected Option: ");

Scanner scan = new Scanner(System.in);

int myOption = scan.nextInt();

switch (myOption) {

case 1:

create();

selectOptions();

break;

case 2:

display();

selectOptions();

break;

case 3:

getOne();

selectOptions();

break;

case 4:

update();

selectOptions();

break;

case 5:

remove();

selectOptions();

break;

case 6:

exit();

break;

default:

selectOptions();

break;

}

}

static void exit() {

System.exit(0);

}

static void getOne() {

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

System.out.println("Enter Name to find:");

Scanner scan = new Scanner(System.in);

String findName = scan.nextLine();

for (Employee var : employees) {

if (findName.equals(var.getName())) {

System.out.println(findName);

break;

}

}

}

static void create() {

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

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

System.out.println("Enter id:");

Scanner scan = new Scanner(System.in);

int inputId = scan.nextInt();

System.out.println("Enter name:");

scan = new Scanner(System.in);

String inputName = scan.nextLine();

employees[i].setId(inputId);

employees[i].setName(inputName);

}

}

static void display() {

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

// Employee e1 = new Employee();

// e1.setId(0);

// e1.setName("Employee0");

// employees[0]=e1;

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

// System.out.println(employees[i]);

// }

for (Employee var : employees) {

System.out.println(var.getId() + " " + var.getName());

}

}

static void update() {

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

System.out.println("Enter Name to update:");

Scanner scan = new Scanner(System.in);

String findName = scan.nextLine();

System.out.println("Enter New Name:");

scan = new Scanner(System.in);

String updateName = scan.nextLine();

for (int i = 1; i < employees.length; i++) {

if (findName.equals(employees[i].getName())) {

employees[i].setName(updateName);

break;

}

}

}

static void remove() {

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

System.out.println("Enter Name to Remove:");

Scanner scan = new Scanner(System.in);

String findName = scan.nextLine();

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

if (findName.equals(employees[i].getName())) {

employees[i].setName("");

break;

}

}

}

}

import java.util.\*;

import java.util.Scanner;

public class FinalAtm{

static int a[]={1001, 50000};

public static void main(String[] args){

selectoptions();

}

static void selectoptions(){

System.out.println("1.withdraw\t2.updatepin");

Scanner scan= new Scanner(System.in);

int myOption = scan.nextInt();

switch (myOption){

case 1:

withdraw();

selectoptions();

break;

case 2:

updatepin();

break;

default:

System.out.println("Retry");

break;

}

scan.close();

}

static void updatepin(){

System.out.println("Enter old pin");

Scanner scan1= new Scanner(System.in);

int oldPin = scan1.nextInt();

System.out.println("Enter New pin:");

int newPin = scan1.nextInt();

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

if(oldPin==(a[i])){

a[i]=newPin;

System.out.println("Success");

selectoptions();

}}

scan1.close();

}

static void withdraw(){

System.out.println("Enter your pin");

Scanner inp = new Scanner(System.in);

int pin= inp.nextInt();

for (int var : a) {

if (pin==var) {

System.out.println("Enter Amount");

int wd = inp.nextInt();

int c = a[1] - wd;

System.out.println("Available balance Rs." + c); } }} }

//package com.logics;

import java.util.Scanner;

class Atmm {

int a = 50000;

int pin;

int san;

void start(int p)

{

pin=p;

options();

}

void options() {

// System.out.println("Insertcard");

// System.out.println("Enter your pin");

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

// pin = s.nextInt();

// System.out.print("1.withdraw 2.Deposit 3.pinchange 4.updatepin");

// Scanner t = new Scanner(System.in);

// k = t.nextInt();

// t.close();

// System.out.println();

System.out.println("Insertcard");

System.out.println("Enter your pin");

Scanner key = new Scanner(System.in);

int enterpin = key.nextInt();

if (enterpin == pin) {

System.out.println("Select Your Options");

System.out.println("1.withdraw 2.Deposit 3.pinchange 4.updatepin");

Scanner inp=new Scanner(System.in);

int number=inp.nextInt();

// System.out.print("1.withdraw 2.Deposit 3.pinchange 4.updatepin");

// Scanner t = new Scanner(System.in);

// k = t.nextInt();

// t.close();

// System.out.println();

switch (number) {

case 1:

// System.out.print("1.withdraw 2.Deposit 3.pinchange 4.updatepin");

// Scanner kk = new Scanner(System.in);

// pin = kk.nextInt();

// System.out.println("Withdrawl");

wd();

break;

case 2:

System.out.println("Deposit");

dp();

options();

break;

case 3:

pinchange();

break;

case 4:

updatepin();

options();

break;

default:

System.out.println("Atm is not working properly,please visit another Atm");

break;

}

// t.close();

} else {

System.out.println("Retry");

}

}

void wd() {

// // System.out.print("1.withdraw 2.Deposit 3.pinchange 4.updatepin");

// Scanner kk = new Scanner(System.in);

// int g = kk.nextInt();

System.out.println("Enter your amount");

Scanner u = new Scanner(System.in);

int wd = u.nextInt();

int c = a - wd;

System.out.println("Available balance Rs." + c);

u.close();

}

void dp() {

System.out.println("Deposit cash");

Scanner v = new Scanner(System.in);

int dp = v.nextInt();

int l = a + dp;

System.out.println("Available balance Rs." + l);

v.close();

}

void pinchange() {

System.out.println("Enter your current pin");

Scanner w = new Scanner(System.in);

int epin = w.nextInt();

int pinchange;

if (pin == epin) {

System.out.println("Enter new pin number");

Scanner x = new Scanner(System.in);

pinchange = x.nextInt();

pin=pinchange;

System.out.println("pin changed successfully " + pin);

Bank b=new Bank();

b.rerun(pinchange);

x.close();

} else {

System.out.println("No match");

}

//start(pin);

}

void updatepin() {

System.out.println("update");

System.out.println("Enter pin to update:");

Scanner scan = new Scanner(System.in);

pin = scan.nextInt();

System.out.println("Enter New pin :");

scan = new Scanner(System.in);

int pinchange = scan.nextInt();

if(pinchange==pin){

pin=pinchange;

System.out.println("Success");

wd();

}

}

// System.out.println("Updated Successfully");

}

class Bank {

public static void main(String[] args) {

rerun(1234);

}

static void rerun(int pin)

{

Atmm o1 = new Atmm();

o1.start(pin);

}

}

import com.sun.xml.internal.bind.v2.runtime.RuntimeUtil.ToStringAdapter;

class Employee{

private int id;

private String name;

private String update;

Employee(int id ,String name,String update){

this.id=id;

this.name=name;

this.update=update;

}

Employee(){

}

/\*\*

\* @return the name

\*/

public String getName() {

return name;

}

/\*\*

\* @param name the name to set

\*/

public void setName(String name) {

this.name = name;

}

/\*\*

\* @return the id

\*/

public int getId() {

return id;

}

/\*\*

\* @param id the id to set

\*/

public void setId(int id) {

this.id = id;

}

//@Override

public String toString() {

return id +" "+ name;

}

/\*\*

\* @return the update

\*/

public String getUpdate() {

return update;

}

/\*\*

\* @param update the update to set

\*/

public void setUpdate(int update) {

this.update = update;

}

}

https://www.tutorialspoint.com/java/util/java\_util\_arrays.htm

www.tutorialspoint.com

Java.util.Arrays Class

Java.util.Arrays Class - Learning Java.util Packages in simple and easy steps : A beginner's tutorial containing complete knowledge of all the classes, interfaces, enumerations and exceptions have been explained with examples for beginners to advanced java programmers

import java.util.Arrays;

/\*\*

\* ArrayClass

\*/

public class ArrayUtilClass {

public static void main(String[] args) {

ArrayUtilClass arrayUtilClass=new ArrayUtilClass();

System.out.println(arrayUtilClass);

System.out.println(arrayUtilClass.hashCode());

int[] intArray = { 1, 2, 3, 4, 5 };

System.out.println(intArray);

String intArrayString = Arrays.toString(intArray);

System.out.println(intArrayString);

String[] ob={"tuts","point"};

Arrays.sort(ob);

System.out.println(Arrays.toString(ob));

System.out.println(Arrays.deepToString(ob));

int src=3;

System.out.println(Arrays.binarySearch(intArray,0,intArray.length,src));

System.out.println(intArray[Arrays.binarySearch(intArray,0,intArray.length,src)]);

String arr[] = new String[3];

// System.out.println(Arrays.toString(arr));

Arrays.fill(arr, "18");

System.out.println(Arrays.toString(arr));

System.out.println(intArray.hashCode());

System.out.println(ob.hashCode());

System.out.println(arr.hashCode());

System.out.println(Arrays.equals(arr, ob));

String a[] = new String[]{"abc","klm","xyz","pqr"};

String a1[] = {"abc","klm","xyz","pqr"};

}

}

https://dzone.com/articles/top-10-methods-java-arrays

dzone.com

Top 10 Methods for Java Arrays - DZone Java

The following are top 10 methods for Java Array. They are the most voted questions from stackoverflow. 0. Decalre an array String[] aArray = new String[5];...

<https://mvnrepository.com/artifact/org.apache.commons/commons-lang3/3.8.0031>

package com.kgisl.qs1;

import java.util.Arrays;

import org.apache.commons.lang3.ArrayUtils;

/\*\*

\* Hello world!

\*

\*/

public class App {

// static String[] names = new String[2];

static String[] names ={"element0","element1"};

public static void main(String[] args) {

displayNames();

displayOneName();

updateName();

deleteName();

}

static void create() {

System.out.println("create");

}

static void displayNames() {

System.out.println("displayNames");

System.out.println(ArrayUtils.toString(names));

}

static void displayOneName() {

System.out.println("displayOneName");

String findName = "element0";

// System.out.println(ArrayUtils.indexOf(names, findName));

System.out.println(names[ArrayUtils.indexOf(names, findName)]);

}

static void updateName() {

System.out.println("updateName");

String findName = "element0";

String updateName = "element0000";

//System.out.println(ArrayUtils.indexOf(names, findName));

names[ArrayUtils.indexOf(names, findName)] = updateName;

displayNames();

}

static void deleteName() {

System.out.println("deleteName");

String findName = "element0000";

// String[] newArray = ArrayUtils.remove(names, ArrayUtils.indexOf(names, findName));

// System.out.println(ArrayUtils.toString(newArray));

String[] newArray=ArrayUtils.removeElement(names, findName);

System.out.println(ArrayUtils.toString(newArray));

}}

package com.kgisl.javaarraylist;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.List;

/\*\*

\* Hello world!

\*

\*/

public class App {

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

static ArrayList<String> names1 = new ArrayList<String>();

static String name1 = "Devi";

static String name2 = "Kavin";

public static void main(String[] args) {

create();

display();

getName();

removeif();

// removeNameInLastIndex();

// removeName();

updateName();

// display();

toarray();

toarrayAsType();

arrayClone();

arrayContains();

clearAll();

}

static void create() {

// names.ensureCapacity(3);

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

names.add("Giri");

names.add("Sasi");

names.add("Kannan");

names.add("Sasi");

names.add("Gowtham");

names.add("Sasi");

names1.add(name1);

names1.add(name2);

names.addAll(names1);

}

static void display() {

System.out.println(names);

}

static void getName() {

String findName = "Sasi";

int findNameIdx = names.indexOf(findName);

System.out.println(names.get(findNameIdx));

}

static void removeName() {

String findName = "Sasi";

int findNameIdx = names.indexOf(findName);

names.remove(findNameIdx);

System.out.println(names);

}

static void removeNameInLastIndex() {

String findName = "Sasi";

int findNameIdx = names.lastIndexOf(findName);

names.remove(findNameIdx);

System.out.println(names);

}

static void updateName() {

String findName = "Kannan";

String updateName = "Kannan R";

int findNameIdx = names.indexOf(findName);

names.set(findNameIdx, updateName);

System.out.println(names);

}

static void toarray() {

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

Object[] arr = names.toArray();

System.out.println(Arrays.toString(arr));

}

static void toarrayAsType() {

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

String[] myArray = names.toArray(new String[0]);

// String[] arr = (String)names.toArray();

System.out.println(Arrays.toString(myArray));

}

static void arrayClone() {

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

Object arrayClone = names.clone();

System.out.println(arrayClone.toString());

}

static void arrayContains() {

String findName = "Sasi 1";

System.out.println(names.contains(findName));

}

static void removeif(){

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

names.removeIf( x -> x == "Sasi" );

System.out.println(names);

}

// static void removeRange(){

// System.out.println("\*\*\*\*\*\*\*\* removeRange \*\*\*\*\*\*\*\*");

// // names.removeRange(1, 4);

// // System.out.println(names);

// List<Integer> al = new ArrayList<Integer>();

// al.add(10);

// al.add(20);

// al.add(30);

// al.add(1);

// al.add(2);

// al.removeRange(1, 4);

// }

static void clearAll() {

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

names.clear();

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

}

}

**package com.kgisl.pgm;**

**import java.util.ArrayList;**

**import java.util.Arrays;**

**import java.util.Collections;**

**import java.util.Scanner;**

**public class App**

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

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

**al.add("kannan");**

**al.add("nivi");**

**al.add("ramya");**

**Collections.sort(al);**

**System.out.println("sorting A 2 z"+al);**

**Collections.sort(al,Collections.reverseOrder()); //reverseOrder**

**System.out.println("reverseOrder"+al);**

**}}**

**package com.kgisl.pgm;**

**import java.util.ArrayList;**

**import java.util.Arrays;**

**import java.util.Collections;**

**import java.util.Scanner;**

**public class App**

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

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

**al.add("kannan");**

**al.add("nivi");**

**al.add("ramya");**

**al.add("mahi");**

**al.add("nidhya");**

**al.add("mohana");**

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

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

**System.out.println(al.subList(2,4 ));**

**//hear 2is start value and upto 4 index**

**System.out.println(al.subList(5,6 ));**

**}}**

**1. Write a C# Sharp program that takes a number and a width also a number, as input and then displays a triangle of that width, using that number.**

**Test Data**

**Enter a number: 6**

**Enter the desired width: 6**

**Expected Output :**

**666666**

**66666**

**6666**

**666**

**66**

**6**

**2. Write a C# Sharp program that takes two numbers as input and perform an operation (+,-,\*,x,/) on them and displays the result of that operation.**

**Test Data**

**Input first number: 20**

**Input operation: -**

**Input second number: 12**

**Expected Output :**

**20 - 12 = 8**

**3. Write a C# Sharp program to find the largest of three numbers.**

**Test Data :**

**Input the 1st number :25**

**Input the 2nd number :63**

**Input the 3rd number :10**

**Expected Output :**

**The 2nd Number is the greatest among three**

**4. Write a C# Sharp program to find the eligibility of admission for a professional course based on the following criteria:**

**Marks in Maths >=65**

**Marks in Phy >=55**

**Marks in Chem>=50**

**Total in all three subject >=180**

**or**

**Total in Math and Subjects >=140**

**Test Data :**

**Input the marks obtained in Physics :65**

**Input the marks obtained in Chemistry :51**

**Input the marks obtained in Mathematics :72**

**Expected Output :**

**The candidate is eligible for admission.**

**5. Write a program in C# Sharp which is a Menu-Driven Program to perform a simple calculation.**

**Test Date and Expected Output**

**Enter the first Integer :10**

**Enter the second Integer :2**

**Here are the options :**

**1-Addition.**

**2-Substraction.**

**3-Multiplication.**

**4-Division.**

**5-Exit.**

**Input your choice :3**

**The Multiplication of 10 and 2 is: 20**

**6. Write a program in C# Sharp to read n number of values in an array and display it in reverse order.**

**Test Data :**

**Input the number of elements to store in the array :3**

**Input 3 number of elements in the array :**

**element - 0 : 2**

**element - 1 : 5**

**element - 2 : 7**

**Expected Output:**

**The values store into the array are:**

**2 5 7**

**The values store into the array in reverse are :**

**7 5 2**

**7. Write a program in C# Sharp to find the sum of all elements of the array.**

**Test Data :**

**Input the number of elements to be stored in the array :3**

**Input 3 elements in the array :**

**element - 0 : 2**

**element - 1 : 5**

**element - 2 : 8**

**Expected Output :**

**Sum of all elements stored in the array is : 15**

**8. Write a program in C# Sharp to count the frequency of each element of an array.**

**Test Data :**

**Input the number of elements to be stored in the array :3**

**Input 3 elements in the array :**

**element - 0 : 25**

**element - 1 : 12**

**element - 2 : 43**

**Expected Output :**

**Frequency of all elements of array :**

**25 occurs 1 times**

**12 occurs 1 times**

**43 occurs 1 times**

**9. Write a program in C# Sharp to find maximum and minimum element in an array.**

**Test Data :**

**Input the number of elements to be stored in the array :3**

**Input 3 elements in the array :**

**element - 0 : 45**

**element - 1 : 25**

**element - 2 : 21**

**Expected Output :**

**Maximum element is : 45**

**Minimum element is : 21**

**10. Write a program in C# Sharp to sort elements of the array in descending order.**

**Test Data :**

**Input the size of array : 3**

**Input 3 elements in the array :**

**element - 0 : 5**

**element - 1 : 9**

**element - 2 : 1**

**Expected Output :**

**Elements of the array in sorted descending order:**

**9 5 1**

**11. Write a program in C# Sharp to delete an element at desired position from an array.**

**Test Data :**

**Input the size of array : 5**

**Input 5 elements in the array in ascending order:**

**element - 0 : 1**

**element - 1 : 2**

**element - 2 : 3**

**element - 3 : 4**

**element - 4 : 5**

**Input the position where to delete: 3**

**Expected Output :**

**The new list is : 1 2 4 5**

**12. Write a program in C# Sharp to find the second largest element in an array.**

**Test Data :**

**Input the size of array : 5**

**Input 5 elements in the array :**

**element - 0 : 2**

**element - 1 : 9**

**element - 2 : 1**

**element - 3 : 4**

**element - 4 : 6**

**Expected Output :**

**The Second largest element in the array is: 6**

**package com.kgisl.ex3;**

**import java.lang.reflect.Field;**

**import java.util.ArrayList;**

**public class Test {**

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

**ArrayList<Integer> list = new ArrayList<Integer>(5);**

**for (int i = 0; i < 10; i++) {**

**list.add(i);**

**System.out.format("Size: %2d, Capacity: %2d%n",**

**list.size(), getCapacity(list));**

**}**

**}**

**static int getCapacity(ArrayList<Integer> list) throws Exception {**

**// Field x = Student1.class.getDeclaredField("rollno");**

**Field d = ArrayList.class.getDeclaredField("elementData");**

**d.setAccessible(true);**

**return ((Object[]) d.get(list)).length;**

**}**

**}**

**package com.kgisl.qs1;**

**public class Student {**

**private int rollno;**

**private String name;**

**Student(int rollno, String name) {**

**this.rollno = rollno;**

**this.name = name;**

**}**

**/\*\***

**\* @return the name**

**\*/**

**public String getName() {**

**return name;**

**}**

**/\*\***

**\* @param name the name to set**

**\*/**

**public void setName(String name) {**

**this.name = name;**

**}**

**/\*\***

**\* @return the rollno**

**\*/**

**public int getRollno() {**

**return rollno;**

**}**

**/\*\***

**\* @param rollno the rollno to set**

**\*/**

**public void setRollno(int rollno) {**

**this.rollno = rollno;**

**}**

**@Override**

**public String toString() {**

**return "rollno:" + rollno + " name:" + name;**

**}**

**}**

**package com.kgisl.qs1;**

**import java.util.ArrayList;**

**import java.util.List;**

**public class StudentDemo {**

**public static ArrayList<Student> studentList = new ArrayList<Student>();**

**static Student newStudent=new Student(502, "name");**

**static Student updateStudent=new Student(502, "updatename");**

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

**StudentDemo s=new StudentDemo();**

**s.loadStudents();**

**s.getAllStudents();**

**s.getStudent(12);**

**s.newStudent(newStudent);**

**s.updateStudent(502,updateStudent);**

**s.removeStudent(101);**

**}**

**StudentDemo(){**

**loadStudents();**

**}**

**public void loadStudents() {**

**studentList.add(new Student(101, "ramu"));**

**studentList.add(new Student(12, "Divya"));**

**studentList.add(new Student(11, "Janani"));**

**}**

**public List<Student> getAllStudents() {**

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

**return studentList;**

**}**

**public Student getStudent(int rollno) {**

**for (Student student : studentList) {**

**if (student.getRollno() == rollno) {**

**System.out.println(String.valueOf(student.getRollno())+student.getName());**

**return student;**

**}**

**}**

**return getStudent(0);**

**}**

**public void newStudent(Student student) {**

**studentList.add(newStudent);**

**getAllStudents();**

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

**}**

**public void updateStudent(int rollno, Student updateStudent) {**

**for (Student student : studentList) {**

**if (student.getRollno() == rollno) {**

**System.out.println(studentList.indexOf(student));**

**studentList.set(studentList.indexOf(student),updateStudent);**

**}**

**}**

**getAllStudents();**

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

**}**

**public void removeStudent(int rollno) {**

**for (Student student : studentList) {**

**if (student.getRollno() == rollno) {**

**System.out.println(studentList.indexOf(student));**

**studentList.remove(studentList.indexOf(student));**

**}**

**break;**

**}**

**getAllStudents();**

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

**}**

**}**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Unit test:**

[**https://howtodoinjava.com/junit-4/**](https://howtodoinjava.com/junit-4/)

**package com.kgisl.qs1;**

**/\*\***

**\* Calculator**

**\*/**

**public class Calculator {**

**public int add(int x, int y) {**

**return x + y;**

**}**

**public int sub(int x, int y) {**

**return x - y;**

**}**

**public int mul(int x, int y) {**

**return x \* y;**

**}**

**public int div(int x, int y) {**

**return x / y;**

**}**

**}**

**package com.kgisl.qs1;**

**import static org.junit.Assert.assertEquals;**

**import org.junit.Test;**

**/\*\***

**\* CalculatorTest**

**\*/**

**public class CalculatorTest {**

**Calculator c = new Calculator();**

**@Test**

**public void addTest() {**

**int actual=c.add(10, 20);**

**int expected=30;**

**assertEquals(expected, actual);**

**}**

**@Test**

**public void subTest() {**

**int actual=c.sub(10, 20);**

**int expected=-10;**

**assertEquals(expected, actual);**

**}**

**@Test**

**public void mulTest() {**

**int actual=c.mul(10, 20);**

**int expected=200;**

**assertEquals(expected, actual);**

**}**

**@Test**

**public void divTest() {**

**int actual=c.div(100, 20);**

**int expected=5;**

**assertEquals(expected, actual);**

**}**

**}**

**package com.kgisl.qs1;**

**import static org.junit.Assert.assertEquals;**

**import java.util.Arrays;**

**import java.util.Collection;**

**import java.util.List;**

**import org.junit.Test;**

**import org.junit.runner.RunWith;**

**import org.junit.runners.Parameterized;**

**import org.junit.runners.Parameterized.Parameters;**

**/\*\***

**\* CalculatorTest**

**\*/**

**@RunWith(Parameterized.class)**

**public class CalculatorParamsTest {**

**private int fn;**

**private int sn;**

**Calculator c = new Calculator();**

**public CalculatorParamsTest(int fn, int sn) {**

**this.fn = fn;**

**this.sn = sn;**

**}**

**@Parameters**

**public static Collection input() {**

**return Arrays.asList(new Object[][] { { 10, 20 }, { 30, 40 } });**

**}**

**@Test**

**public void addTest() {**

**int actual = c.add(fn, sn);**

**int expected = fn + sn;**

**assertEquals(expected, actual);**

**}**

**@Test**

**public void subTest() {**

**int actual = c.sub(fn, sn);**

**int expected = fn - sn;**

**assertEquals(expected, actual);**

**}**

**@Test**

**public void mulTest() {**

**int actual = c.mul(fn, sn);**

**int expected = fn \* sn;**

**assertEquals(expected, actual);**

**}**

**@Test**

**public void divTest() {**

**// int actual = c.div(fn, sn);**

**// int expected = fn/sn;**

**// assertEquals(expected, actual);**

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

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

**int actual = c.div(fn, sn);**

**int expected = 10 / 20;**

**assertEquals(expected, actual);**

**}**

**}**

**By default maven uses maven-surefire-plugin:2.12.4. Some of the maven commands may not work with the following error.**

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

**Failed to execute goal org.apache.maven.plugins:maven-surefire-plugin:2.12.4:test**

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

**Add the following plugin to get rid of this error**

**<plugin>**

**<groupId>org.apache.maven.plugins</groupId>**

**<artifactId>maven-surefire-plugin</artifactId>**

**<version>3.0.0-M3</version>**

**</plugin>**

**Maven commands to run test classes and methods**

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

**Run all test methods with in test package**

**mvn test**

**Run all test methods in a class**

**mvn -Dtest=StudentDemoTest test**

**Run one test method in a class**

**mvn -Dtest=StudentDemoTest#getAllStudentsTest test**

**Run multiple test methods in a class**

**mvn -Dtest=StudentDemoTest#getAllStudentsTest+getStudentTest test**

**Run all test methods in multiple test classes**

**mvn -Dtest="StudentDemoTest,CalculatorTest" test**

**Run selected test methods in multiple test classes**

**mvn -Dtest="StudentDemoTest#getAllStudentsTest+getStudentTest,CalculatorTest" test**

**Run all test methods in multiple test classes using expression**

**mvn -Dtest="\*Test" test**

**Run all test Clases from subdirectory, eg: /doc/ You can use command:**

**mvn -Dtest=\*/doc/\* test**

**package com.kgisl.qs1;**

**import java.util.ArrayList;**

**import java.util.List;**

**import org.junit.BeforeClass;**

**import org.junit.FixMethodOrder;**

**import org.junit.runners.MethodSorters;**

**import org.junit.Test;**

**// @FixMethodOrder(MethodSorters.NAME\_ASCENDING)**

**public class StudentDemoTest {**

**public static ArrayList<Student> studentList = new ArrayList<Student>();**

**Student newStudent = new Student(502, "name");**

**Student updateStudent = new Student(502, "updatename");**

**private static StudentDemo studentDemo;**

**@BeforeClass**

**public static void before() {**

**studentDemo = new StudentDemo();**

**}**

**@Test**

**public void T1getAllStudentsTest() {**

**List<Student> actual = studentDemo.getAllStudents();**

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

**// assertEquals(fn+sn, actual);**

**}**

**@Test**

**public void T2getStudentTest() {**

**Student actual = studentDemo.getStudent(12);**

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

**// assertEquals(fn+sn, actual);**

**}**

**// newStudent**

**@Test**

**public void T3newStudentTest() {**

**studentDemo.newStudent(newStudent);**

**// assertEquals(fn+sn, actual);**

**}**

**@Test**

**public void T4updateStudentTest() {**

**studentDemo.updateStudent(502, updateStudent);**

**// assertEquals(fn+sn, actual);**

**}**

**@Test**

**public void T5removeStudentTest() {**

**studentDemo.removeStudent(101);**

**// assertEquals(fn+sn, actual);**

**}**

**}**

[**https://www.guru99.com/junit-parameterized-test.html**](https://www.guru99.com/junit-parameterized-test.html)

[**https://www.geeksforgeeks.org/static-methods-vs-instance-methods-java/**](https://www.geeksforgeeks.org/static-methods-vs-instance-methods-java/)

**Do it Yourself**

**Create ArrayList from array**

**How do You remove repeated elements from ArrayList?**

**Java - Method executed prior to Default Constructor**

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

**public class ChkCons {**

**int var = getVal();**

**ChkCons() {**

**System.out.println("I'm Default Constructor.");**

**}**

**public int getVal() {**

**System.out.println("I'm in Method.");**

**return 10;**

**}**

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

**ChkCons c = new ChkCons();**

**}**

**}**

**OUTPUT :**

**I'm in Method.**

**I'm Default Constructor.**

**Instance variable initialization expressions such as int var = getVal(); are evaluated after the super class constructor is executed but prior to the execution of the current class constructor's body.**

**Therefore getVal() is called before the body of the ChkCons constructor is executed**

**In what order do static blocks and initialization blocks execute when using inheritance?**

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

**public class Parent {**

**public Parent() {**

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

**}**

**static {**

**System.out.println("Parent static block");**

**}**

**{**

**System.out.println("Parent initialisation block");**

**}**

**}**

**public class Child extends Parent {**

**{**

**System.out.println("Child initialisation block");**

**}**

**static {**

**System.out.println("Child static block");**

**}**

**public Child() {**

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

**}**

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

**new Child();**

**}**

**}**

**The output of the above code will be**

**Parent static block**

**Child static block**

**Parent initialization block**

**Parent Constructor**

**Child initialization block**

**Child Constructor**

**There are several rules in play**

**static blocks are always run before the object is created, so that's why you see print messages from both parents and child static blocks**

**now, when you are calling constructor of the subclass (child), then this constructor implicitly calls super(); before executing it's own constructor. Initialization block comes into play even before the constructor call, so that's why it is called first. So now your parent is created and the program can continue creating child class which will undergo the same process.**

**Explanations:**

**Static block of parent is executed first because it is loaded first and static blocks are called when the class is loaded.**

**//junit parameterised test**

**package com.kgisl.qs1;**

**​**

**/\*\***

**\* Calculator**

**\*/**

**public class Calculator {**

**​**

**public int add(int x, int y) {**

**return x + y;**

**}**

**}**

**​**

**package com.kgisl.qs1;**

**​**

**import static junit.framework.Assert.assertEquals;**

**​**

**import java.util.Arrays;**

**import java.util.Collection;**

**​**

**import org.junit.Before;**

**​**

**// import static junit.framework.Assert.assertEquals;**

**​**

**import org.junit.Test;**

**import org.junit.runner.RunWith;**

**import org.junit.runners.Parameterized;**

**import org.junit.runners.Parameterized.Parameters;**

**​**

**/\*\***

**\* CalculatorTest**

**\*/**

**@RunWith(Parameterized.class)**

**public class CalculatorTest {**

**private int fn;**

**private int sn;**

**private int exp;**

**private Calculator calculator;**

**​**

**public CalculatorTest(int fn, int sn, int exp) {**

**System.out.println("Parameterised constructor runs");**

**this.fn = fn;**

**this.sn = sn;**

**this.exp = exp;**

**}**

**​**

**@Before**

**public void initialise() {**

**System.out.println("Before methods run");**

**calculator = new Calculator();**

**}**

**​**

**@Parameters**

**public static Collection input() {**

**System.out.println("List object created");**

**return Arrays.asList(new Object[][] { { 1, 2, 3 }, { 11, 22, 33 } });**

**}**

**​**

**@Test**

**public void addTest() {**

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

**assertEquals(exp, calculator.add(fn, sn));**

**}**

**}**

**import static org.junit.Assert.assertEquals;**

**// import static junit.framework.Assert.assertEquals;**

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

**JUnit 3.X: junit.framework.Assert**

**JUnit 4.X: org.junit.Assert**

**Prefer the newest one, especially when running JDK5 and higher with annotation support.**

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

**The old method (of JUnit 3) was to mark the test-classes by extending junit.framework.TestCase. That inherited junit.framework.Assert itself and your test class gained the ability to call the assert methods this way.**

**Since version 4 of JUnit, the framework uses Annotations for marking tests. So you no longer need to extend TestCase. But that means, the assert methods aren't available. But you can make a static import of the new Assert class. That's why all the assert methods in the new class are static methods. So you can import it this way:**

**import static org.junit.Assert.\*;**

**After this static import, you can use this methods without prefix.**

**At the redesign they also moved to the new package org.junit that follows better the normal conventions for package naming.**

**Do it Yourself**

**Combine two arraylists**

**find unique values**

**find common values**

**Collection<String> listOne = new ArrayList(Arrays.asList("a", "b", "c", "d", "e", "f", "g"));**

**Collection<String> listTwo = new ArrayList(Arrays.asList("a", "b", "d", "e", "f", "gg", "h"));**

[**https://git-scm.com/download/win**](https://git-scm.com/download/win)

**import java.util.Arrays;**

**import java.util.List;**

**import java.util.ArrayList;**

**class HowForEachWorks1 {**

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

**ArrayList list1=new ArrayList();**

**List<Integer> numberList = Arrays.asList(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10);**

**for (Integer i : numberList) {**

**if ((i & 1)==0) {**

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

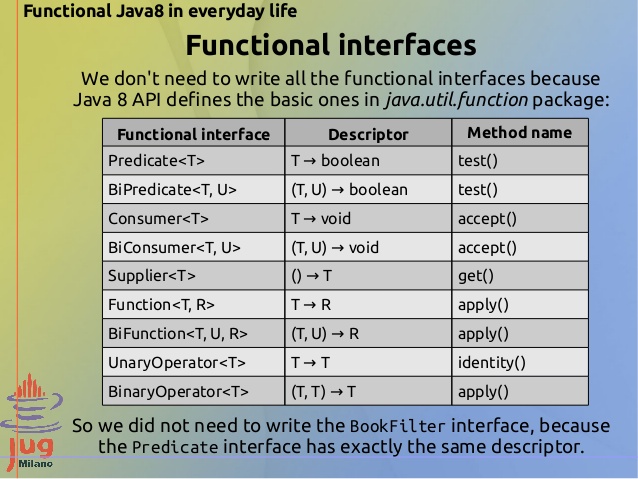
**}**

**}**

**}**

**}**

[**https://image.slidesharecdn.com/java8-141003090600-phpapp02/95/functional-java-8-in-everyday-life-9-638.jpg?cb=1412327296**](https://image.slidesharecdn.com/java8-141003090600-phpapp02/95/functional-java-8-in-everyday-life-9-638.jpg?cb=1412327296)

****

[**https://www.slideshare.net/martyhall/java-8-tutorial-streams-part-1**](https://www.slideshare.net/martyhall/java-8-tutorial-streams-part-1)

[**https://www.javatpoint.com/java-8-stream**](https://www.javatpoint.com/java-8-stream)

[**https://2.bp.blogspot.com/-tFyvqQwhpFQ/WZp6467TqrI/AAAAAAAAPnY/AfdnCrqViQ4dJQ6jUAXnF-thOTuT-apYQCLcBGAs/s640/What%2Bare%2Bthe%2Bcore%2Bstream%2Boperations%2Bof%2BJava%2B8%2BStream%2B%2BStreams%2Bin%2BJava%2B8.jpg**](https://2.bp.blogspot.com/-tFyvqQwhpFQ/WZp6467TqrI/AAAAAAAAPnY/AfdnCrqViQ4dJQ6jUAXnF-thOTuT-apYQCLcBGAs/s640/What%2Bare%2Bthe%2Bcore%2Bstream%2Boperations%2Bof%2BJava%2B8%2BStream%2B%2BStreams%2Bin%2BJava%2B8.jpg)

**import java.util.\*;**

**import java.util.Arrays;**

**import java.util.function.Consumer;**

**class HowForEachWorks {**

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

**List<String> disneyCharacters = Arrays.asList("Micky", "Donald", "Minnie", "Aurora", "Winnie", "Pluto");**

**/\*Consumer<String> con = new Consumer<String>() {**

**public void accept(String s) {**

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

**}**

**};\*/**

**//Consumer<String> con = s -> System.out.println(s);**

**//disneyCharacters.forEach(con);**

**//disneyCharacters.forEach(s->System.out.println(s));**

**// call by method**

**disneyCharacters.forEach(System.out::println);**

**}**

**}**

**```**

**```**

**package com.kg.lambdaapp;**

**import java.util.Arrays;**

**import java.util.List;**

**import java.util.ArrayList;**

**class HowForEachWorks1 {**

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

**ArrayList list1=new ArrayList();**

**List<Integer> numberList = Arrays.asList(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10);**

**for (Integer i : numberList) {**

**if ((i & 1)==0) {**

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

**}**

**}**

**}**

**}**

**This is classic OOP style of hiding method implementations from the caller.**

**The caller simply passes a variable to the method which then does something with the value of the variable**

**and returns another value or produces a side effect as it is in our case.**

**class LambdaDemo {**

**public void printSomething(String something) {**

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

**}**

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

**LambdaDemo lambdaDemo = new LambdaDemo();**

**String something = "Learning Lambda";**

**lambdaDemo.printSomething(something);**

**}**

**}**

**see an equivalent implementation that uses behavior passing other than variable passing.**

**To achieve this, we have to create a functional interface that defines that abstracts the behavior instead of a method.**

**A functional interface is an interface that has only one method:**

**package com.kg.lambdaapp;**

**interface Printer {**

**void print(String val);**

**}**

**class LambdaDemo {**

**public void printSomething(String something, Printer printer) {**

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

**printer.print(something);**

**}**

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

**LambdaDemo lambdaDemo = new LambdaDemo();**

**Printer printer = new Printer() {**

**@Override**

**public void print(String val) {**

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

**}**

**};**

**String something = "Learning Lambda";**

**lambdaDemo.printSomething(something, printer);**

**}**

**}**

**public void print(String toPrint) {**

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

**}**

**lambda expressions**

**package com.kg.lambdaapp;**

**interface Printer {**

**void print(String val);**

**}**

**class LambdaDemo {**

**public void printSomething(String something, Printer printer) {**

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

**printer.print(something);**

**}**

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

**LambdaDemo lambdaDemo = new LambdaDemo();**

**Printer printer = (String toPrint) -> {**

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

**};**

**String something = "Learning Lambda";**

**lambdaDemo.printSomething(something, printer);**

**}**

**}**

**(String toPrint) -> {**

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

**}**

**JAVA 8**

[**https://beginnersbook.com/2017/10/java-8-interface-changes-default-method-and-static-method/**](https://beginnersbook.com/2017/10/java-8-interface-changes-default-method-and-static-method/)

**package com.kgfsl.stream;**

**import java.util.ArrayList;**

**import java.util.Comparator;**

**import java.util.List;**

**import java.util.\*;**

**import java.util.stream.Collectors;**

**import java.util.Optional;**

**import javax.xml.transform.stream.StreamSource;**

**import java.util.stream.\*;**

**import java.util.stream.Stream;**

**class Sample {**

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

**List<Person> persons = Arrays.asList(new Person("ani", 21), new Person("vani", 18), new Person("vani", 99),**

**new Person("azar", 11));**

**// Find Maximum value**

**persons.stream().max(Comparator.comparing(Person::getName))**

**.ifPresent(p -> System.out.println("sort by alphabet order person max" + p));**

**// Find Minimum value**

**persons.stream().min(Comparator.comparing(Person::getName))**

**.ifPresent(s -> System.out.println("sort by alphabet order person min " + s));**

**// using filter**

**List<Person> l1 = persons.stream().filter(s -> s.getName().endsWith("i")).collect(Collectors.toList());**

**l1.stream().forEach(System.out::println);**

**// count**

**long n = persons.stream().filter(e -> e.getName().endsWith("i")).count();**

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

**// foreach and distinct**

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

**List<String> d = persons.stream().map(Person::getName).distinct().collect(Collectors.toList());**

**d.forEach(System.out::println);**

**// skip**

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

**persons.stream().skip(2).forEach(System.out::println);**

**// limit**

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

**persons.stream().limit(3).forEach(System.out::println);**

**// allmatch**

**boolean b1 = persons.stream().allMatch(p1 -> p1.getAge() > 20 && p1.getName().startsWith("v"));**

**System.out.println(b1);**

**// nonematch**

**boolean b2 = persons.stream().noneMatch(p1 -> p1.getAge() > 20 && p1.getName().startsWith("m"));**

**System.out.println(b2);**

**// anymatch**

**boolean b3 = persons.stream().anyMatch(p1 -> p1.getAge() > 20 && p1.getName().startsWith("v"));**

**System.out.println(b3);**

**// string reduce**

**String[] myArray = { "this", "is", "a", "sentence" };**

**String result = Arrays.stream(myArray).reduce("", (a, b) -> a + b);**

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

**// number reduce**

**int[] myArray1 = { 1, 2, 3, 4 };**

**int result1 = Arrays.stream(myArray1).reduce(0, (a, b) -> a + b);**

**System.out.println(result1);**

**// boolean b5=persons.stream().filter(s->s.getName().endsWith("i")).findAny();**

**// l1.stream().forEach(System.out::println);**

**// find any**

**Optional<Person> anyEmpAbove40 = persons.stream().filter(emp -> emp.getAge() > 40).findAny();**

**if (anyEmpAbove40.isPresent()) {**

**System.out.println("Any Employee above age 40: " + anyEmpAbove40.get());**

**}**

**// find first**

**Optional<Person> o1 = persons.stream().filter(emp -> emp.getAge() > 20).findFirst();**

**if (o1.isPresent()) {**

**System.out.println("Any Employee above age 20: " + o1.get());**

**}**

**// sort**

**List<Person> slist = persons.stream().sorted(Comparator.comparing(Person::getAge)).collect(Collectors.toList());**

**slist.forEach(System.out::println);**

**// Person[]**

**// type=persons.stream().filter(s->s.getName().endsWith("i")).toArray(Person[]::new);**

**// Stream.of(l1).forEach(System.out::println);**

**// peek**

**List<Integer> list = Arrays.asList(10, 11, 12);**

**list.stream().peek(i -> System.out.println(i \* i)).collect(Collectors.toList());**

**// map**

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

**List<String> c1 = persons.stream().map(Person::getName).collect(Collectors.toList());**

**System.out.println(c1);**

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

**// parallelstream**

**/\***

**\* Stream<Person> stream = persons.parallelStream(); List<Person> evenNumbersArr**

**\* = stream.filter(emp -> emp.getAge() > 40).collect(Collectors.toList());**

**\***

**\* for(Person even:evenNumbersArr) { System.out.println(even); }**

**\*/**

**// maptoint and sum**

**int sum1 = persons.stream().mapToInt(Person::getAge).sum();**

**System.out.println(sum1);**

**// maptodouble**

**double sum2 = persons.stream().mapToDouble(Person::getAge).sum();**

**System.out.println(sum2);**

**// maptolong**

**long sum3 = persons.stream().mapToLong(Person::getAge).sum();**

**System.out.println(sum3);**

**// average**

**double sum4 = persons.stream().mapToDouble(Person::getAge).average().getAsDouble();**

**System.out.println(sum4);**

**// collect**

**List<Person> c2 = persons.stream().collect(Collectors.toList());**

**System.out.println(c2);**

**// stream builder**

**Stream.Builder<String> b = Stream.builder();**

**b.accept("a");**

**b.accept("b");**

**b.accept("c");**

**b.accept("d");**

**Stream<String> s = b.build();**

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

**// flat map**

**String[][] data = new String[][] { { "a", "b" }, { "c", "d" }, { "e", "f" } };**

**Stream<String[]> temp = Arrays.stream(data);**

**Stream<String> stringStream = temp.flatMap(x -> Arrays.stream(x));**

**Stream<String> stream1 = stringStream.filter(x -> "a".equals(x.toString()));**

**stream1.forEach(System.out::println);**

**}**

**}**

**///\_\_model class\_\_\_\_\_\_\_\_**

**package com.kgfsl.stream;**

**class Person**

**{**

**private String name;**

**private int age;**

**Person(String name,int age)**

**{**

**this.name=name;**

**this.age=age;**

**}**

**public String getName()**

**{**

**return name;**

**}**

**public int getAge()**

**{**

**return age;**

**}**

**public String toString()**

**{**

**return " name: "+name+" age: "+age;**

**}**

**}**

**Do it Yourself**

**concurrency, parallelism, thread and process**

**socket vs port**

**find odd even in single line java**

import java.util.ArrayList;

import java.util.Arrays;

import java.util.List;

import java.util.function.Consumer;

import java.util.function.Predicate;

/\*\*

\* LambdaDemo

\*/

public class LambdaDemo1 {

public static void main(String[] args) {

List<Integer> numberList = Arrays.asList(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10);

Predicate<Integer> odd=new Predicate<Integer>() {

@Override

public boolean test(Integer t) {

return t%2!=0;

}

};

// Predicate<Integer> odd=x->x%2!=0;

// Predicate<Integer> even=x->x%2==0;

numberList.stream().filter(odd).forEach(System.out::println);

numberList.stream().filter(odd.negate()).forEach(System.out::println);

}

}

\*Comparable\*

A comparable object is capable of comparing itself with another object. The class itself must implements the java.lang.Comparable interface in order to be able to compare its instances.

\*Comparator\*

A comparator object is capable of comparing two different objects. The class is not comparing its instances, but some other class’s instances. This comparator class must implement the java.util.Comparator interface.

<https://www.javatpoint.com/Comparator-interface-in-collection-framework>

<https://www.javatpoint.com/difference-between-comparable-and-comparator>

<https://www.digizol.com/2008/07/java-sorting-comparator-vs-comparable.html>

https://docs.oracle.com/javase/8/docs/api/?java/lang/FunctionalInterface.html

JSR

JCP JEP

Java 8 vs java 9

open jdk vs oracle jdk

Java 8 / 9 features

Do it Yourself

id, name, college, department, year, mark1, mark2, mark3

First 3 colleges

Last 3 colleges

College wise toppers

Top 3 departments

Department wise average

<https://blog.idrsolutions.com/2015/02/java-8-method-references-explained-5-minutes/>

db: https://www.heidisql.com/downloads/releases/HeidiSQL\_10.1\_64\_Portable.zip