**Program 1 :**

interface Abc {

public default void dis1() {

System.out.println("Abc Default IMPLEMENTATION");

}

}

interface Abc1 {

public default void dis1() {

System.out.println("Abc1 Default IMPLEMENTATION");

}

}

class Xyz implements Abc1 {

}

class Hello {

public static void main(String args[]) {

System.out.println("Welcome to java 8");

Xyz obj = new Xyz();

obj.dis1();

}

}

Program 2 :

interface Abc {

public static void dis1() {

System.out.println("Abc Default IMPLEMENTATION");

}

}

class Xyz implements Abc {

}

class Hello {

public static void main(String args[]) {

System.out.println("Welcome to java 8");

Xyz obj = new Xyz();

obj.dis1();

}

}

Error : error: cannot find symbol

Program 3 :

interface Abc {

public static void dis1() {

System.out.println("Abc Default IMPLEMENTATION");

}

}

class Xyz implements Abc {

public static void dis1() {

System.out.println("Abc Ovveride without Annotation");

}

}

class Hello {

public static void main(String args[]) {

System.out.println("Welcome to java 8");

Xyz obj = new Xyz();

obj.dis1();

Abc.dis1();

}

}

OutPut :

Welcome to java 8

Abc Ovveride without Annotation

Abc Default IMPLEMENTATION

Program 4:

interface Abc {

public static void dis1() {

System.out.println("Abc Default IMPLEMENTATION");

}

}

class Xyz implements Abc {

@Override

public static void dis1() {

System.out.println("Abc Ovveride without Annotation");

}

}

class Hello {

public static void main(String args[]) {

System.out.println("Welcome to java 8");

Xyz obj = new Xyz();

obj.dis1();

Abc.dis1();

}

}

OutPut : error: method does not override or implement a method from a supertype

Program 5 :

@FunctionalInterface

interface Abc {

void dis1();

}

class Hello {

public static void main(String args[]) {

}

}

O/P : No Error

Program 6 :

interface Abc {

void dis1();

default void dis2() {};

}

class Hello {

public static void main(String args[]) {

}

}

O/P : No Error

Program 7:

@FunctionalInterface

interface Abc {

void dis1();

void dis2();

}

class Hello {

public static void main(String args[]) {

}

}

O/P : Hello.java:1: error: Unexpected @FunctionalInterface annotation

@FunctionalInterface

^

Abc is not a functional interface

multiple non-overriding abstract methods found in interface Abc

1 error

Program 8 :

class Outer {

void dis1() {

System.out.println("Outer Class Disc Method");

}

class Inner1 {

void dis2() {

System.out.println("Inner Non static class Disc2 Method");

}

}

}

class Hello {

public static void main(String args[]) {

Outer out = new Outer();

out.dis1(); // Calles Outer class Method but cannot call Inner class methods with this Object

Outer.Inner1 in1 = out.new Inner1();

Outer.Inner1 in2 = new Outer().new Inner1();

in2.dis2();

in1.dis2();

}

}

O/P

Outer Class Disc Method

Inner Non static class Disc2 Method

Inner Non static class Disc2 Method

Program 9 :

class Outer {

void dis1() {

System.out.println("Outer Class Disc Method");

}

class Inner1 {

void dis2() {

System.out.println("Inner Non static class Disc2 Method");

}

}

static class Inner2 {

void dis3() {

System.out.println("Inner static class Disc3 Method");

}

}

}

class Hello {

public static void main(String args[]) {

Outer out = new Outer();

out.dis1(); // Calles Outer class Method but cannot call Inner class methods with this Object

Outer.Inner1 in1 = out.new Inner1();

Outer.Inner1 in2 = new Outer().new Inner1();

in2.dis2();

in1.dis2();

Outer.Inner2 in3 = new Outer.Inner2();

in3.dis3();

}

}

O/P:

Outer Class Disc Method

Inner Non static class Disc2 Method

Inner Non static class Disc2 Method

Inner static class Disc3 Method

Program 10 :

interface Abc {

void dis1();

}

class Hello {

public static void main(String args[]) {

Abc obj = new Abc() {

public void dis1() {

System.out.println("Disc Overrided");

}

};

obj.dis1();

}

}

O/P : Disc Overrided

Prgram 11:

interface Abc {

void dis1();

}

class Hello {

public static void main(String args[]) {

Abc obj = new Abc();

obj.dis1();

}

}

O/P :

Hello.java:7: error: Abc is abstract; cannot be instantiated

Abc obj = new Abc();

^

1 error

Program 12:

interface Abc {

void dis1();

}

class Hello {

public static void main(String args[]) {

Abc obj = new Abc(){

};

obj.dis1();

}

}

O/P:

Hello.java:7: error: <anonymous Hello$1> is not abstract and does not override abstract method dis1() in Abc

Abc obj = new Abc(){

^

1 error

Prgram 13 :

interface Abc {

void dis1();

}

class Hello {

public static void main(String args[]) {

Abc obj = new Abc(){

public void dis1(){

System.out.println("Dis1 method overrided");

}

};

obj.dis1();

// lAMBDA EXPRESSION

Abc obj1 = ()->System.out.println("Dis1 method overrided in lambda");

obj1.dis1();

}

}

O/P:

Dis1 method overrided

Dis1 method overrided in lambda

Program 14:

interface Emp{

void empDetails(int id,String name,float salary);

}

class Hello {

public static void main(String args[]) {

// lAMBDA EXPRESSION with Params

Emp obj1 = (id,name,salary)->{

System.out.println("Emp Details are");

System.out.println("Id "+ id);

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

System.out.println("Salary : "+ salary);

};

obj1.empDetails(1,"Abhi",15000);

}

}

O/p:

Emp Details are

Id 1

Name : Abhi

Salary : 15000.0

Program 15 :

import java.util.\*;

class Hello {

public static void main(String args[]) {

List<Integer> ll = new ArrayList();

ll.add(20); ll.add(30); ll.add(40); ll.add(50); ll.add(60);

System.out.println("Before java 8");

Iterator<Integer> li = ll.iterator();

while(li.hasNext()) {

System.out.println(li.next());

}

System.out.println("After 8");

ll.forEach((val)->System.out.println(val));

}

}

O/P:

Before java 8

20

30

40

50

60

After 8

20

30

40

50

60

Program 16:

import java.util.\*;

class Hello {

public static void main(String args[]) {

List<Integer> ll = new ArrayList();

ll.add(20); ll.add(70); ll.add(40); ll.add(50); ll.add(60);

System.out.println("Before sort");

ll.forEach((val)->System.out.println(val));

Collections.sort(ll);

System.out.println("Before sort");

ll.forEach((val)->System.out.println(val));

}

}

O/P:

Before sort

20

70

40

50

60

Before sort

20

40

50

60

70

Program 17 :

**public** **class** Employee **implements** Comparable<Employee>{

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

**private** String name;

**private** **float** salary;

**public** **int** compareTo(Employee emp) {

// return this.id-emp.id; sort id by Desc

//return emp.id - this.id; // sort id by asc

**return** **this**.name.compareTo(emp.name);

//return emp.name.compareTo(this.name);

// return (int)(this.salary - emp.salary);

// return (int)(emp.salary - this.salary);

}

**public** Employee() {

**super**();

// **TODO** Auto-generated constructor stub

}

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

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

**int** result = 1;

result = prime \* result + id;

result = prime \* result + ((name == **null**) ? 0 : name.hashCode());

result = prime \* result + Float.*floatToIntBits*(salary);

**return** result;

}

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

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

**return** **true**;

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

**return** **false**;

**if** (getClass() != obj.getClass())

**return** **false**;

Employee other = (Employee) obj;

**if** (id != other.id)

**return** **false**;

**if** (name == **null**) {

**if** (other.name != **null**)

**return** **false**;

} **else** **if** (!name.equals(other.name))

**return** **false**;

**if** (Float.*floatToIntBits*(salary) != Float.*floatToIntBits*(other.salary))

**return** **false**;

**return** **true**;

}

**public** Employee(**int** id, String name, **float** salary) {

**super**();

**this**.id = id;

**this**.name = name;

**this**.salary = salary;

}

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

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

**this**.name = name;

}

**public** **float** getSalary() {

**return** salary;

}

**public** **void** setSalary(**float** salary) {

**this**.salary = salary;

}

@Override

**public** String toString() {

**return** "Employee [id=" + id + ", name=" + name + ", salary=" + salary

+ "]";

}

}

**import** java.util.ArrayList;

**import** java.util.Collections;

**import** java.util.List;

**public** **class** DemoTest {

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

// **TODO** Auto-generated method stub

System.***out***.println("Welcome to Java 8");

Employee emp11 = **new** Employee(1,"Abhi",15000);

Employee emp12 = **new** Employee(2,"Aarthi",15000);

Employee emp13 = **new** Employee(3,"Pooja",15000);

List<Employee> listEmp = **new** ArrayList<>();

listEmp.add(emp11); listEmp.add(emp12); listEmp.add(emp13);

listEmp.forEach((val)-> System.***out***.println(val));

Collections.*sort*(listEmp);

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

listEmp.forEach((val)-> System.***out***.println(val));

}

}

O/P : Employee [id=1, name=Abhi, salary=15000.0]

Employee [id=2, name=Aarthi, salary=15000.0]

Employee [id=3, name=Pooja, salary=15000.0]

After Sort

Employee [id=3, name=Pooja, salary=15000.0]

Employee [id=2, name=Aarthi, salary=15000.0]

Employee [id=1, name=Abhi, salary=15000.0]

Program 18 :

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

**public** **class** StreamDemo {

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

// **TODO** Auto-generated method stub

Integer abc[] = {1,3,4,5,2,8,10,12,23,35,67};

Stream<Integer> stream = Stream.*of*(abc); //Creating the stream

stream.forEach((val)->System.***out***.println(val));

}

}

o/P :

1,3,4,5,2,8,10,12,23,35,67

Program 19:

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

**public** **class** StreamDemo {

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

// **TODO** Auto-generated method stub

Integer abc[] = {1,3,4,5,2,8,10,12,23,35,67};

Stream<Integer> stream = Stream.*of*(abc); //Creating the stream

stream.sorted().forEach((val)->System.***out***.println(val));

}

}

O/P:

1 2 3 4 5 8 10 12 23 35 67