Q1. Write a program to show Interface Example in java?

Ans : interface printable{

void print();

}

class A6 implements printable{

public void print(){

System.out.println("Hello");

}

public static void main(String args[]){

A6 obj = new A6();

obj.print();

 }

}

Q2. Write a program a Program with 2 concrete method and 2 abstract method in java ?

Ans :

abstract class arithmetic\_operation {

abstract void printInfo();

}

class add extends arithmetic\_operation {

void printInfo()

{

int a = 3;

int b = 4;

System.out.println(a + b);

}

}

class sub extends arithmetic\_operation {

void printInfo()

{

int c = 4;

int d = 5;

System.out.println(c - d);

}

}

class abstraction {

// Main Function

public static void main(String args[])

{

arithmetic\_operation n = new add();

n.printInfo();

arithmetic\_operation y = new sub();

y.printInfo();

}

}

Q3. Write a program to show the use of functional interface in java?

Ans : @FunctionalInterface

interface Test{

void display(String message);

}

public class TestImplementation implements Test{

public void display(String message){

System.out.println(message);

}

public static void main(String[] args) {

TestImplementation obj = new TestImplementation();

obj.display("Hello World!");

}

}

Q4.What is an interface in Java?

Ans : An interface in Java is a blueprint of a class. It has static constants and abstract methods.

The interface in Java is *a mechanism to achieve*[*abstraction*](https://www.javatpoint.com/abstract-class-in-java). There can be only abstract methods in the Java interface, not method body. It is used to achieve abstraction and multiple [inheritance in Java](https://www.javatpoint.com/inheritance-in-java).

In other words, you can say that interfaces can have abstract methods and variables. It cannot have a method body.

Q5.What is the use of interface in Java?

Ans : An interface in Java has remained a complex topic for many beginners to understand. The first thing which puzzles many programmers is the fact that *you cannot define any method inside interface*, it a just declaration. By rule, all method inside interface must be abstract (Well, this rule is changing in Java 8 to allow lambda expressions, now interface can have one non-abstract method, also known as a default method). So, if you can't define anything, *Why we need an interface?* what's the use of an interface, if we are anyway going to write a class and override them to provide behaviour, Can't we declare those methods inside the class itself without using interface etc. Well, if you are thinking in terms of behaviour then you are really missing the point of interface.

Q6.What is the lambda expression of Java 8?

Ans : A lambda expression is a short block of code which takes in parameters and returns a value. Lambda expressions are similar to methods, but they do not need a name and they can be implemented right in the body of a method.

import java.util.ArrayList;

public class Main {

public static void main(String[] args) {

ArrayList<Integer> numbers = new ArrayList<Integer>();

numbers.add(5);

numbers.add(9);

numbers.add(8);

numbers.add(1);

numbers.forEach( (n) -> { System.out.println(n); } );

}

}

Q7.Can you pass lambda expressions to a method? When?

Ans : A lambda expression passed in a method that has an argument of type of functional interface. If we need to pass a lambda expression as an argument, the type of parameter receiving the lambda expression argument must be of a functional interface type.

In the below example, the lambda expression can be passed in a method which argument's type is "TestInterface".

interface TestInterface {

   boolean test(int a);

}

class Test {

   // lambda expression can be passed as first argument in the check() method

   static boolean check(TestInterface ti, int b) {

      return ti.test(b);

   }

}

public class LambdaExpressionPassMethodTest {

   public static void main(String arg[]) {

      // lambda expression

      boolean result = Test.check((x) -> (x%2) == 0, 10);

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

   }

}

Q8.What is the functional interface in Java 8?

Ans : A functional interface is an interface that contains only one abstract method. They can have only one functionality to exhibit. From Java 8 onwards, lambda expressions can be used to represent the instance of a functional interface. A functional interface can have any number of default methods.

Q9.What is the benefit of lambda expressions in Java 8?

Ans : Lambda oﬀers many beneﬁts, including the following:

* No servers to manage. ...
* Continuous scaling. ...
* Millisecond metering. ...
* Increases innovation. ...
* Modernize your applications. ...
* Support for developers.

Q10.Is it mandatory for a lambda expression to have parameters?

Ans : The lambda must contain the same number of parameters as the delegate type. Each input parameter in the lambda must be implicitly convertible to its corresponding delegate parameter.