Core Specific- Java Assignment 4

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

Ans:

```
1. interface printable{
2. void print();
3. }
4. class Ak implements printable{
5. public void print(){System.out.println("Hello");}
6.
7. public static void main(String args[]){
8. Ak obj = new Ak();
9. obj.print();
10. }
11.}
```

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

Ans:

```
1. // Abstract class example
2. abstract class AbstractExample {
3.
4.     // Abstract method
5.     abstract void display();
6.
7.     // Concrete method 1
8.     void show()
9.     {
10.        System.out.println("Concrete method 1 of abstract class");
11.    }
12.     // Concrete method 2
13.    void showB()
14.    {
15.        System.out.println("Concrete method 2 of abstract class");
16.    }
```

```
17.
18. }
19.
20. // Subclass of abstract class
21. class SubClass extends AbstractExample {
22.
23.     // Implementing the abstract method 1
24.     void display()
25.     {
26.         System.out.println("Abstract method1 implemented");
27.     }
28.     // Implementing the abstract method 2
29.     void displayB()
30.     {
31.         System.out.println("Abstract method2 implemented");
32.     }
33. }
34.
35.
36. /**
37.  * Main class
38.  */
39. public class AbstractClass{
40.
41.     public static void main(String args[])
42.     {
43.         // Creating an object of the subclass
44.         SubClass obj = new SubClass();
45.
46.         // Calling the abstract method
47.         obj.display();
48.
49.         // Calling the concrete method
50.         obj.show();
51.         // Calling the abstract method
52.         obj.displayB();
53.
```

```
54.         // Calling the concrete method
55.         obj.showB();
56.
57.     }
58. }
```

Output:

```
Abstract method1 implemented
A concrete method1 of abstract class
Abstract method2 implemented
A concrete method2 of abstract class
```

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

Ans:

```
class Test {

  public static void main(String args[])

  {

    new Thread(new Runnable() {

      @Override public void run()

      {

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

      }

    }).start();

  }

}
```

Core Specific- Java Assignment 4

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.* There can be only abstract methods in the Java interface, not method body. It is used to achieve abstraction and multiple inheritance in Java.

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

Java Interface also **represents the IS-A relationship**.

It cannot be instantiated just like the abstract class.

Since Java 8, we can have **default and static methods** in an interface.

Since Java 9, we can have **private methods** in an interface.

Q5. What is the use of interface in Java?

Ans:

Interfaces are used in Java to achieve abstraction. By using the implements keyword, a java class can implement an interface. In general terms, an interface can be defined as a container that stores the signatures of the methods to be implemented in the code segment. It improves the levels of Abstraction.

Q6. What is the lambda expression of Java 8?

Ans:

Lambda Expressions were added in Java 8. 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.

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

Ans:

Lambdas are purely a call-site construct: the recipient of the lambda does not need to know that a Lambda is involved, instead it accepts an Interface with the appropriate method.

In other words, you define or use a functional interface (i.e. an interface with a single method) that accepts and returns exactly what you want.

Since Java 8 there is a set of commonly-used interface types in java.util.function.
For this specific use case there's java.util.function.IntBinaryOperator with a single int applyAsInt(int left, int right) method, so you could write your method like this:

```
static int method(IntBinaryOperator op){
   return op.applyAsInt(5, 10);
}
```

But you can just as well define your own interface and use it like this:

```
public interface TwoArgIntOperator {
   public int op(int a, int b);
}
//elsewhere:
static int method(TwoArgIntOperator operator) {
   return operator.op(5, 10);
}
```

Then call the method with a lambda as parameter:

```
public static void main(String[] args) {
   TwoArgIntOperator addTwoInts = (a, b) -> a + b;
   int result = method(addTwoInts);
   System.out.println("Result: " + result);
}
```

Using your own interface has the advantage that you can have names that more clearly indicate the intent.

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 expressions improve code readability and do not require interpretation. Lambdas allow you to write concise code. It encourages the use of functional programming. It simplifies variable scope and encourages code reusability.

Core Specific- Java Assignment 4

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

Ans:

No Parameter can be passed.

Syntax:

```
() -> System.out.println("Hello");
```

It takes interface of the following form:

```
interface Test1
{
   void print()
}
```