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.
⇒Key points about interfaces in Java are:

- An interface is declared using the `interface` keyword.
- It can contain method signatures (without implementation) and constant fields.
- All methods in an interface are by default public and abstract.
- Classes implement interfaces using the `implements` keyword, and they must provide implementation for all methods defined in the interface.
- A class can implement multiple interfaces, allowing for multiple inheritance of behaviours.

Q5.What is the use of interfaces in Java?

Ans)Here are some of the key uses of interfaces:

- **Defining Contracts:** Interfaces define a set of methods that a class must implement.
- **Achieving Abstraction** : Interfaces enable the abstraction of behaviour from implementation.

- **Enabling Polymorphism** : Interfaces allow objects of different classes to be treated uniformly if they implement the same interface. This promotes code reuse and simplifies the handling of multiple implementations.

- **Promoting Loose Coupling :** By programming to interfaces, classes are decoupled from specific implementations.

Q6.What is the lambda expression of Java 8?

Ans)Lambda expressions in Java 8 introduced a concise way to express functional interfaces. A lambda expression is a way to represent an anonymous function (a function without a name) as an expression. It provides a more compact syntax for writing functional interfaces, which are interfaces with a single abstract method.

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

Ans)Yes, you can pass lambda expressions as arguments to methods in Java.

This feature is particularly useful in situations where you need to pass behaviour as a parameter, such as when working with functional programming constructs like the “forEach” method in Java's Stream API or when implementing event handlers in graphical user interface (GUI) programming.

Here's an example of passing a lambda expression to the “forEach” method:

```
List<String> names = Arrays.asList("Alice", "Bob",
"Charlie");
names.forEach(name -> System.out.println(name));
```

In this example, the lambda expression `name → System.out.println(name)` is passed as an argument to the `forEach` method. It defines the behaviour of what should be done with each element in the list.

Q8.What is the functional interface in Java 8?

Ans)In Java 8, a functional interface is an interface that contains exactly one abstract method. Functional interfaces are also known as SAM (Single Abstract Method) interfaces or lambda interfaces.

Java 8 provides the “@FunctionalInterface” annotation to explicitly mark an interface as a functional interface.

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

Ans)Lambda expressions in Java 8 provide several benefits:

1. Concise syntax: Lambda expressions allow you to write more compact code by reducing the boilerplate of anonymous inner classes.
2. Readability: Lambda expressions make code more readable and expressive, especially for simple functional-style operations.
3. Functional programming: Lambda expressions enable functional programming concepts like higher-order functions, function composition, and method chaining.

4. Improved performance: Lambda expressions can help optimise code execution by enabling more efficient and parallel processing.

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

Ans)No, it is not mandatory for a lambda expression to have parameters. Lambda expressions in Java can be parameterless or can have one or more parameters, depending on the functional interface they are associated with.