Assignment Questions 4

Q1.1. Write a program to show Interface Example in java?

```
// Defining an interface
interface Shape {
    void draw(); // Abstract method
}

// Implementing the interface
class Circle implements Shape {
    @Override
    public void draw() {
        System.out.println("Drawing a circle");
    }
}

class Rectangle implements Shape {
    @Override
    public void draw() {
        System.out.println("Drawing a rectangle");
    }
}

public class InterfaceExample {
    public static void main(String[] args) {
        Shape circle = new Circle();
```

```
        circle.draw(); // Calling the draw() method of Circle class

        Shape rectangle = new Rectangle();
        rectangle.draw(); // Calling the draw() method of Rectangle
            class
    }
}
```

Output:

Drawing a circle

Drawing a rectangle

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

```
// Abstract class
abstract class Vehicle {
    // Concrete method
    public void startEngine() {
        System.out.println("Engine started");
    }

    // Concrete method
    public void stopEngine() {
        System.out.println("Engine stopped");
```

```
    }

    // Abstract method
    public abstract void accelerate();

    // Abstract method
    public abstract void brake();
}

// Concrete class
class Car extends Vehicle {
    @Override
    public void accelerate() {
        System.out.println("Car accelerating");
    }

    @Override
    public void brake() {
        System.out.println("Car braking");
    }
}

public class AbstractExample {
    public static void main(String[] args) {
        Car car = new Car();
        car.startEngine(); // Concrete method from Vehicle
            class
```

```
        car.accelerate(); // Concrete method from Car class
        car.brake(); // Concrete method from Car class
        car.stopEngine(); // Concrete method from Vehicle
            class
    }
}
```

Output:

```
Engine started
Car accelerating
Car braking
Engine stopped
```

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

```
// Functional interface
interface MathOperation {
    int operate(int a, int b);
}

public class FunctionalInterfaceExample {
    public static void main(String[] args) {
        // Using lambda expression to implement the
            functional interface
```

```
        MathOperation addition = (a, b) -> a + b;

        // Using method reference to implement the
           functional interface
        MathOperation subtraction =
           FunctionalInterfaceExample::subtract;

        int result1 = addition.operate(10, 5);
        System.out.println("Addition: " + result1);

        int result2 = subtraction.operate(10, 5);
        System.out.println("Subtraction: " + result2);
    }

    // Method to perform subtraction
    public static int subtract(int a, int b) {
        return a - b;
    }
}
```

Output:
Addition: 15
Subtraction: 5

Q4.What is an interface in Java?

An interface is a reference type that defines a contract of methods that a class must implement. It specifies a set of methods that a class implementing the interface must provide. An interface acts as a blueprint for classes, defining the methods they should have without providing the implementation details.

Q5.What is the use of interface in Java?

Interfaces in Java serve several important purposes and provide significant benefits.

Key uses of interfaces in Java:

- Abstraction and Polymorphism.
- Contractual Obligation.
- Multiple Inheritance.
- Loose Coupling and Modularity.
- API Design and Frameworks.
- Unit Testing and Mocking:

Q6.What is the lambda expression of Java 8?

In Java 8, lambda expressions were introduced as a new feature to provide a concise way of implementing functional interfaces. A lambda expression is an anonymous function that can be used to represent a method implementation directly in the code. Lambda expressions simplify the syntax for

implementing single-method interfaces, making the code more readable and expressive.

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

Yes, we can pass lambda expressions as arguments to methods in Java. This is possible when the method parameter type is a functional interface, an interface with a single abstract method. Since lambda expressions can be used to implement the abstract method of a functional interface, they can be passed as arguments to methods expecting that interface type. This allows for the implementation of the method's functionality to be defined at the point of invocation, providing flexibility and customization.

Q8.What is the functional interface in Java 8?

In Java 8, a functional interface is an interface that has exactly one abstract method. Functional interfaces are also known as SAM (Single Abstract Method) interfaces or lambda interfaces. The introduction of functional interfaces in Java 8 is closely associated with the addition of lambda expressions, which provide a concise way to implement the abstract method of a functional interface.

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

Lambda expressions introduced in Java 8 bring several benefits to the language and facilitate the adoption of functional programming concepts.

Benefits of lambda expressions in Java 8:

- Concise and Readable Code
- Functional Programming Support
- Improved API Design
- Enhanced Collections and Streams
- Increased Productivity

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

No, it is not mandatory for a lambda expression to have parameters. Lambda expressions can be defined without any parameters, as long as the target functional interface does not require any input arguments. The presence or absence of parameters in a lambda expression depends on the signature of the single abstract method in the functional interface being implemented.