PRACTICAL 5 – LAMBDA EXPRESSIONS (ADVANCED JAVA)

AIM

To study and implement Lambda Expressions in Java and perform the following operations: a) Print “Hello World” using Lambda Expression
b) Lambda Expression with single parameter
c) Lambda Expression with multiple parameters to add two numbers
d) Lambda Expressions for unit conversion
e) Lambda Expression with and without return keyword
f) Lambda Expression to concatenate two strings

SOFTWARE REQUIREMENTS

• Operating System: Windows / Linux
• JDK: Version 8 (mandatory for Lambda Expressions)
• IDE: NetBeans IDE 7.x

THEORY

Lambda Expressions were introduced in Java 8 to support functional programming. A Lambda Expression is a short block of code that takes parameters and returns a value. It eliminates the need to write anonymous classes and makes code concise and readable. A Lambda Expression works with a Functional Interface, which contains exactly one abstract method.

PART A: Lambda Expression to Print Hello World

```
interface Message {
    void show();
}

public class LambdaHelloWorld {
    public static void main(String[] args) {
        Message msg = () -> System.out.println("Hello World");
        msg.show();
    }
}
```

PART B: Lambda Expression with Single Parameter

```
interface Printer {
    void print(String message);
}

public class LambdaSingleParameter {
    public static void main(String[] args) {
        Printer p = msg -> System.out.println(msg);
        p.print("Welcome to Advanced Java");
    }
}
```

PART C: Lambda Expression with Multiple Parameters

```
interface Addition {
    int add(int a, int b);
}

public class LambdaAddition {
    public static void main(String[] args) {
        Addition sum = (a, b) -> a + b;
        System.out.println("Sum: " + sum.add(10, 20));
    }
}
```

PART D: Lambda Expression for Unit Conversion

```
interface Converter {
    double convert(double value);
}

public class LambdaFahrenheitToCelsius {
    public static void main(String[] args) {
        Converter c = f -> (f - 32) * 5 / 9;
        System.out.println("Celsius: " + c.convert(98.6));
    }
}

interface DistanceConverter {
    double convert(double km);
}

public class LambdaKilometersToMiles {
    public static void main(String[] args) {
        DistanceConverter d = km -> km * 0.621371;
        System.out.println("Miles: " + d.convert(10));
    }
}
```

PART E: Lambda Expression With and Without return Keyword

```
interface Square {
    int calculate(int x);
}

public class LambdaReturnDemo {
    public static void main(String[] args) {
        Square s1 = x -> x * x;
        Square s2 = x -> {
            return x * x;
        };

        System.out.println("Without return: " + s1.calculate(5));
        System.out.println("With return: " + s2.calculate(6));
    }
}
```

PART F: Lambda Expression to Concatenate Two Strings

```
interface Concatenate {
    String join(String a, String b);
}

public class LambdaStringConcat {
    public static void main(String[] args) {
```

```
                Concatenate c = (a, b) -> a + b;
                System.out.println(c.join("Advanced ", "Java"));
            }
        }
```

NETBEANS 7 – DETAILED EXECUTION STEPS

1. Install JDK 8 and configure it in NetBeans
2. Open NetBeans IDE 7
3. File → New Project → Java → Java Application
4. Project Name: LambdaDemo
5. Uncheck 'Create Main Class'
6. Finish
7. Create a package named lambdademo
8. Create separate Java classes for each program
9. Paste respective code
10. Run each file using Shift + F6

RESULT

Thus, Lambda Expressions were successfully implemented for various operations in Java.