Assignment 19

Ans 1. 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. They are used only in functional interfaces.

Syntax:

- The simplest lambda expression contains a single parameter and an expression: (parameter)->expression
- To use more than one parameter, wrap them in parentheses: (parameter1,parameter 2)->expression.

Ans 2. A lambda expression when passed in a method that has an argument of type of functional interface. Then 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.

Ans 3. 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.

Ans 4. Uses of lamda expression:

It helps to iterate, filter and extract data from collection. The Lambda expression is used to provide the implementation of an interface which has functional interface. It saves a lot of code. In case of lambda expression, we don't need to define the method again for providing the implementation.

- **Enables functional programming**: All new JVM based languages take advantage of the functional paradigm in their applications, but programmers forced to work with Object-Oriented Programming till lambda expressions came. Hence, lambda expressions enable us to write functional code.

- **Readable and concise code**: People have started using lambda expressions and reported that it can help to remove a huge number of lines from their code.

  Syntax: (arg1, arg2...) -> { body }
  Or
  (type1 arg1,type2 arg2…)->{body}

Ans 5. 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. The return value of the lambda must be implicitly convertible to the delegate's return type