Lambda Expression
Assignment

Q1.What is the lambda expression of java 8?
Ans. Java Lambda Expression is a way to define an anonymous function or method that can be passed around as a value. It allows you to write code that is more concise, flexible, and easier to read.
java Lambda Expression was introduced in Java 8, and they are a shorthand notation for creating instances of functional interfaces. A functional interface is an interface that has only one abstract method and can be used to represent a single behavior or action.
Example-:
(parameters) -> expression

or

(parameters) -> { statements; }

Here, the "*parameters*" specify the input arguments for the anonymous function, and the "*expression*" or "*statements*" specify the behaviour of the function.

The arrow symbol "->*"* separates the parameters from the body of the function.

Q2. Can you pass lambda expressions to a method ?When?
Ans. *forEach() method is of Iterable interface that is used to iterate through a collection. Here it takes an argument of Consumer type interface. This is a functional interface having only one abstract method called accept(). Since it is a functional interface, a lambda expression can be passed.*

Example-:

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

class abc {
        static void fun(Test1 t) { t.print(); }
    public static void main(String[] args)
    {
                fun(() -> System.out.println("Hello"));
```

```
    }
}
```

Q3. What is the function 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. *Runnable*, *ActionListener*, *and Comparable* are some of the examples of functional interfaces.

Example:

```
class Test {
    public static void main(String args[])
    {
                new Thread(new Runnable() {
            @Override public void run()
            {
                System.out.println("New thread created");
            }
        }).start();
    }
}
```

Q4. Why do we use lambda expressions in Java?

Ans. There are various reasons for addition of lambda expression in Java platform but the most beneficial of them is that we can easily distribute processing of collection over multiple threads. Prior to Java 8, if the processing of elements in a collection had to be done in parallel, the client code was supposed to perform the necessary steps and not the collection. In Java 8, using lambda expression and Stream API we can pass processing logic of elements into methods provided by collections and now collection is responsible for parallel processing of elements and not the client.

Q5. Is it mandatory for a lambda expression to have parameters?

Ans. Lambda syntax only requires parentheses around more than one parameter, or when there is no parameter at all.

This lambda has no parameters and returns a String as an expression. This lambda has no parameters and returns a String (using an explicit return statement, within a block). return is a control-flow statement.