1. What is the lambda expression of Java 8?

   A lambda expression in Java is a concise way of representing an anonymous function that can be passed as a parameter, stored in a variable, or returned from a method. It is introduced in Java 8 to enable functional programming.

   Syntax :

   ```
   (parameters) -> expression
   (parameters) -> { statements  }
   ```

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

   Yes, we can pass lambda expressions to a method, especially when the method accepts a functional interface (an interface with a single abstract method) as an argument.

   Example :

   ```
   @FunctionalInterface
   interface Calculator {
       int operate(int a, int b);
   }

   class Example {
       public static int compute(int a, int b, Calculator calculator) {
           return calculator.operate(a, b);
       }

       public static void main(String[] args) {
           // Passing lambda expressions
           System.out.println(compute(5, 3, (x, y) -> x + y)); // Addition
           System.out.println(compute(5, 3, (x, y) -> x * y)); // Multiplication
       }
   }
   ```

3. What is the functional interface in Java 8?

   A functional interface is an interface with exactly one abstract method. It can have default or static methods. Functional interfaces are the target types for lambda expressions.It is annotated with @FunctionalInterface.

   Example :

```
@FunctionalInterface
interface Greeting {
   void sayHello(String name);
}

class Example {
   public static void main(String[] args) {
      Greeting greet = (name) -> System.out.println("Hello, " + name);
      greet.sayHello("Java");
   }
}
```

4. Why do we use lambda expressions in Java?

   Lambda expressions are used in Java for the following reasons:

   - Conciseness: Simplifies code by replacing verbose anonymous classes.
   - Improved Readability: Enhances clarity when working with functional interfaces.
   - Functional Programming: Allows functions to be treated as first-class citizens.

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

   It is not mandatory for a lambda expression to have parameters. If the method being implemented does not require parameters, the lambda expression can omit them.

   Example :

```
Runnable runnable = () -> System.out.println("hi");
runnable.run();
```