Lambda Expression

1.What is the lambda expression in java 8?

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.

Which is equal to following code:

```
public int value(int a, int b){
return a + b;
}
```

It's also called an anonymous function because you are essentially writing the code you write in function but

2.Can you pass the lambda expression to a method?when?

Yes, you can pass a lambda expression to a method provided it is expecting a functional interface. For
example, if a method is accepting a Runnable, Comparable or Comparator then you can pass a lambda
expression to it because all these are functional interfaces in Java 8.

3.What is the functional interface in java 8?

A functional interface in Java 8 is an interface with a single abstract method. For example, Comparator
which has just one abstract method called compare() or Runnable which has just one abstract method called
run(). There are many more general purpose functional interfaces introduced in JDK on java.util.function
package. They are also annotated with @FunctionalInterface but that's optional.

4.What is the benefit of lambda expressions in java 8?

The main benefit of lambda expression in Java 8 is that now it's easier to pass a code block to a method.
Earlier, the only way to do this was wrapping the code inside an Anonymous class, which requires a lot of
boilerplate code.

5.It is mandatory for lambda expression to have parameters

No, it's not mandatory for a lambda expression to have parameters, you can define a lambda expression
without parameters as shown below:

```
() -> System.out.println("lambdas without parameter");
```

You can pass this code to any method which accepts a functional interface.