1. What is lambda expression of Java 8?

**Ans:-** 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.

For eg:- (a,b)->a+b is lambda expression.

public int value(int a. Int b){

return a+b;

}

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

Ans- Yes, you can pass a lambda expression to a method provided it is expecting a functional interface means interface allows only one abstract method. 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.

1. What is the functional interface in Java 8 ?

Ans:- 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 interface introduced in JDK on java.util.function package.

They are also annotated with @FunctionalInterface but its an optional.

1. What is the benefit of Lambda expression in Java 8?

Ans:- The main benefit 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.

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

Ans:- No. its not mandatory for a lambda expression to have parameters, you can define a lambda expression without parameters as shoen below:

() -> System.out.println(“Lambdas without parameter”);

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