**210. Introduction to Lambda Expression.**

1. Lambda Expression are used for defying anonymous methods or nameless functions or methods.

2. Lambda Expression requires interface reference. It cannot be done with class reference.

3. Interface should be a functional interface compulsorily.

4. For example:

Car c = () ->  
 {  
 System.*out*.println("Booooooooooonk!");  
 };

5. As functional interface contains only one function the lambda expression always overridden the only function in the interface.

6. They are very handy.

**211. Parameters in Lambda Expression.**

1. -> operator represents the lambda expression.

2. For example:

MyLambda m = (s)->{System.*out*.println(s);};

3. It’s okay without {} brackets.

**212. Capture in Lambda Expression.**

1. Lambda Expression can access only those variables which are final or effectively final(variables that are not modified inside the method).

2. You can access and modify the instance variables. But you cannot modify the variables inside the class. They may or may not be final.

3. The lambda expressions are similar to inner class. Inner class can’t access variable of the method.

4. We can pass lambda expression as a parameter.

5. Exampl:

class LambdaDemp2 implements MyLambda1  
{  
 @Override  
 public void display()  
 {  
 System.*out*.println("Success!");  
 }  
}  
  
class UseLambda2  
{  
 public void useLambda(LambdaDemp2 l)  
 {  
 l.display();  
 }  
}  
  
  
  
  
class LambdaDemo1  
{  
 public static void main(String[] args)  
 {  
// MyLambda m = (s)->System.out.println(s);  
// m.display("Shreyash");  
 new UseLambda2().useLambda(new LambdaDemp2());  
 }  
}

**213. Method Reference.**

1. You can assign any method definition to functional interface method.

2. We can assign constructor as lambda expression.

ml = Print1::new;

3. For static method.

Ml = Print1::display;

4. For instance method

Ml = new Print1()::print;

5. We can make compact code using lambda expression.

6. It is more like achieving polymorphism without writing too much code.