**Lambda Type Inference**

Before Java 8 you would have to specify what interface to implement, when making anonymous interface implementations. Here is the anonymous interface implementation example from the beginning of this text:

stateOwner.addStateListener(new StateChangeListener() {

public void onStateChange(State oldState, State newState) {

// do something with the old and new state.

}

});

With lambda expressions the type can often be *inferred* from the surrounding code. For instance, the interface type of the parameter can be inferred from the method declaration of the addStateListener()method (the single method on the StateChangeListener interface). This is called *type inference*. The compiler infers the type of a parameter by looking elsewhere for the type - in this case the method definition. Here is the example from the beginning of this text, showing that the StateChangeListenerinterface is not mentioned in the lambda expression:

stateOwner.addStateListener(

(oldState, newState) -> System.out.println("State changed")

);

In the lambda expression the parameter types can often be inferred too. In the example above, the compiler can infer their type from the onStateChange() method declaration. Thus, the type of the parameters oldState and newState are inferred from the method declaration of the onStateChange()method

**Reference source:** [**http://tutorials.jenkov.com/java/lambda-expressions.html**](http://tutorials.jenkov.com/java/lambda-expressions.html)