Serious Coding

Want to take your Command Pattern coding to the next level? You can use Java's lambda expressions to skip the step of creating all those concrete command objects. With lambda expressions, instead of instantiating the concrete command objects, you can use *function objects* in their place. In other words, we can use a function object *as a command*. And, while we're at it, we can delete all those concrete Command classes, too.

Let's take a look at how you'd use lambda expressions as commands to simplify our previous code:

The updated code, using lambda expressions:

```
public class RemoteLoader {

    public static void main(String[] args) {
        RemoteControl remoteControl = new RemoteControl();

        Light livingRoomLight = new Light("Living Room");
        ...
        LightOnCommand livingRoomLightOn =
                               new LightOnCommand(livingRoomLight);
        LightOffCommand livingRoomLightOff =
                               new LightOffCommand(livingRoomLight);

        ...
        remoteControl.setCommand(0,() -> livingRoomLight.on(),
                                   () -> livingRoomLight.off());
        ...
    }
}
```

We create the Light object like normal...

But we can remove the concrete LightOnCommand and LightOffCommand objects.

Instead we'll write the concrete commands as lambda expressions that do the same work as the concrete command's execute() method was doing: that is, turning the light on or turning the light off.

Later, when you click one of the remote's buttons, the remote calls the execute() method of the command object in the slot for that button, which is represented by this lambda expression.

Once we've replaced the concrete commands with lambda expressions, we can delete all those concrete command classes (LightOnCommand, LightOffCommand, HottubOnCommand, HottubOffCommand, etc.). If you do this for every concrete command, you'll reduce the total number of classes in the remote control application from 22 to 9.

Note that you can only do this if your Command interface has *one* abstract method. As soon as we add a second abstract method, the lambda shorthand no longer works.

If you like this technique, check out your favorite Java reference for more information on the lambda expression.