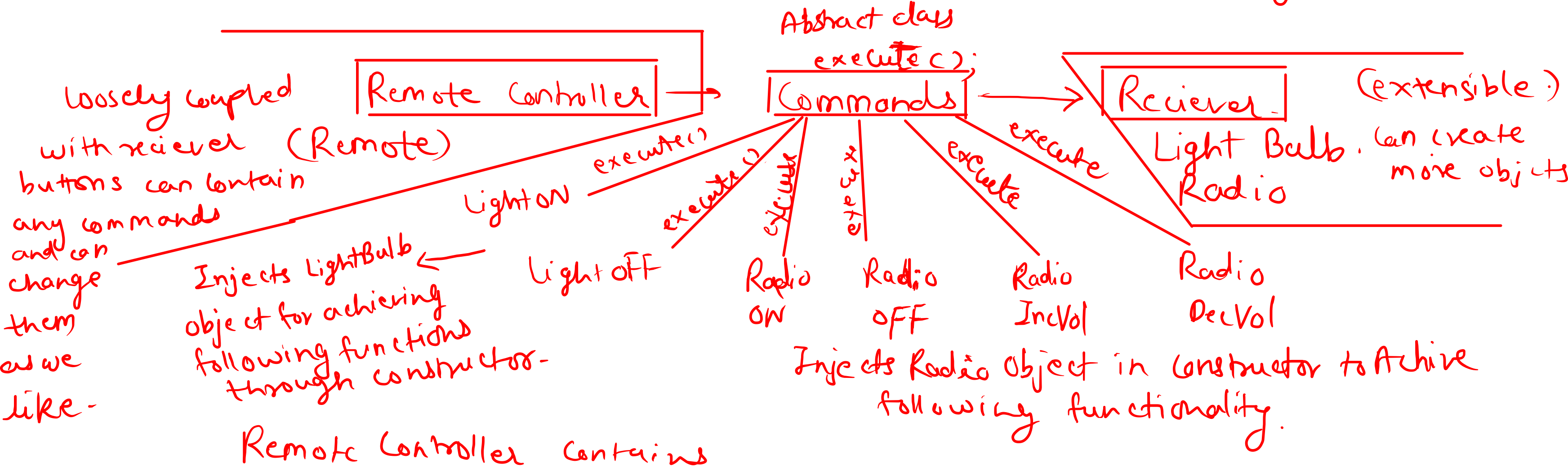
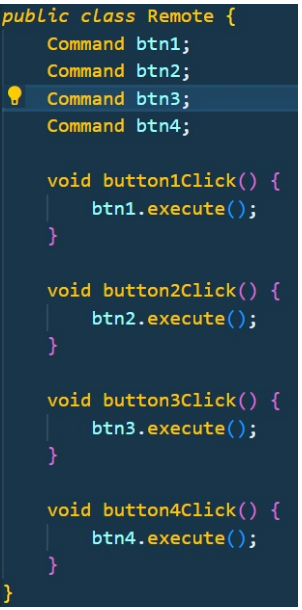
Command Design Pattern. (Loose Coupling.)

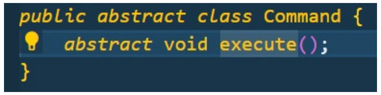

Abstract class
execute();
Commands
Remote Controller
Reciever
Light Bulb.
Radio
(extensible)
Can create more objects
loosely coupled with reciever (Remote) buttons can contain any commands and can change them as we like.
execute()
execute()
execute
execute
execute
execute
Light ON
Light OFF
Radio ON
Radio OFF
Radio IncVol
Radio DecVol
Injects LightBulb object for achieving following functions through constructor.
Injects Radio Object in constructor to Achive following functionality.

Remote Controller contains
Command btn1, btn2, btn3, btn4.

```
ClickBtn1 ( ) { btn1.execute. }   ClickBtn3 ( ) { btn3.execute. }
ClickBtn2 ( ) { btn2.execute. }   ClickBtn4 ( ) { btn4.execute. }
```

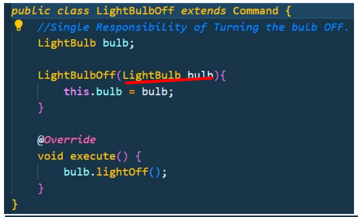


```java
public class Remote {
    Command btn1;
    Command btn2;
    Command btn3;
    Command btn4;

    void button1Click() {
        btn1.execute();
    }

    void button2Click() {
        btn2.execute();
    }

    void button3Click() {
        btn3.execute();
    }

    void button4Click() {
        btn4.execute();
    }
}
```

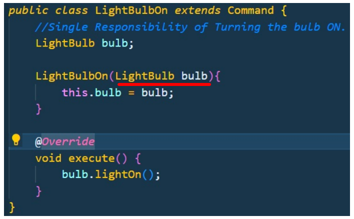


```java
public abstract class Command {
    abstract void execute();
}
```

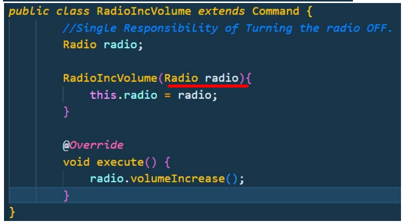
Commands →



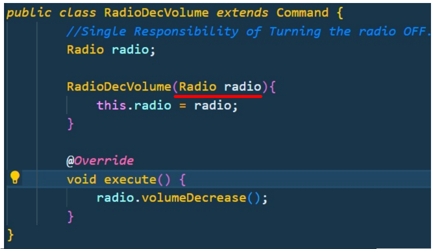
```java
public class LightBulbOff extends Command {
    //Single Responsibility of Turning the bulb OFF.
    LightBulb bulb;

    LightBulbOff(LightBulb bulb){
        this.bulb = bulb;
    }

    @Override
    void execute() {
        bulb.lightOff();
    }
}
```



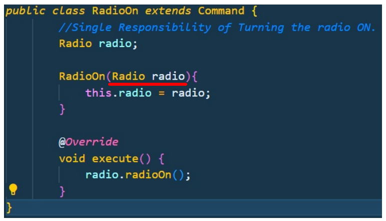
```java
public class LightBulbOn extends Command {
    //Single Responsibility of Turning the bulb ON.
    LightBulb bulb;

    LightBulbOn(LightBulb bulb){
        this.bulb = bulb;
    }

    @Override
    void execute() {
        bulb.lightOn();
    }
}
```



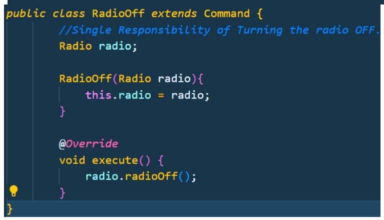
```java
public class RadioIncVolume extends Command {
        //Single Responsibility of Turning the radio OFF.
        Radio radio;

        RadioIncVolume(Radio radio){
            this.radio = radio;
        }

        @Override
        void execute() {
            radio.volumeIncrease();
        }
}
```



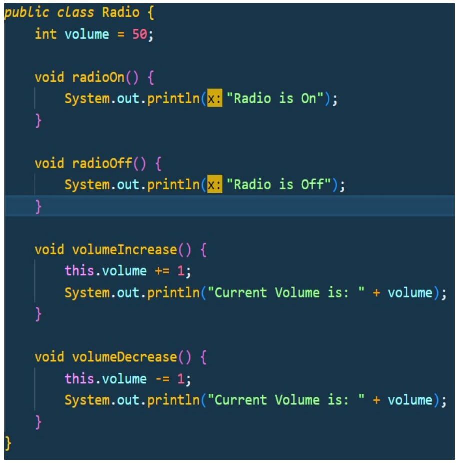
```java
public class RadioDecVolume extends Command {
        //Single Responsibility of Turning the radio OFF.
        Radio radio;

        RadioDecVolume(Radio radio){
            this.radio = radio;
        }

        @Override
        void execute() {
            radio.volumeDecrease();
        }
}
```



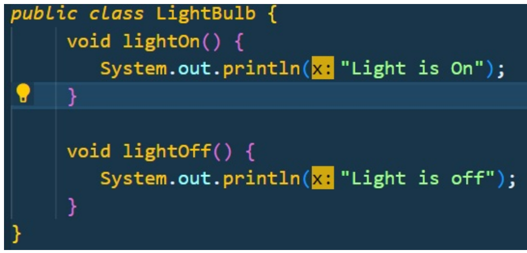
```java
public class RadioOn extends Command {
        //Single Responsibility of Turning the radio ON.
        Radio radio;

        RadioOn(Radio radio){
            this.radio = radio;
        }

        @Override
        void execute() {
            radio.radioOn();
        }
}
```



```java
public class RadioOff extends Command {
        //Single Responsibility of Turning the radio OFF.
        Radio radio;

        RadioOff(Radio radio){
            this.radio = radio;
        }

        @Override
        void execute() {
            radio.radioOff();
        }
}
```



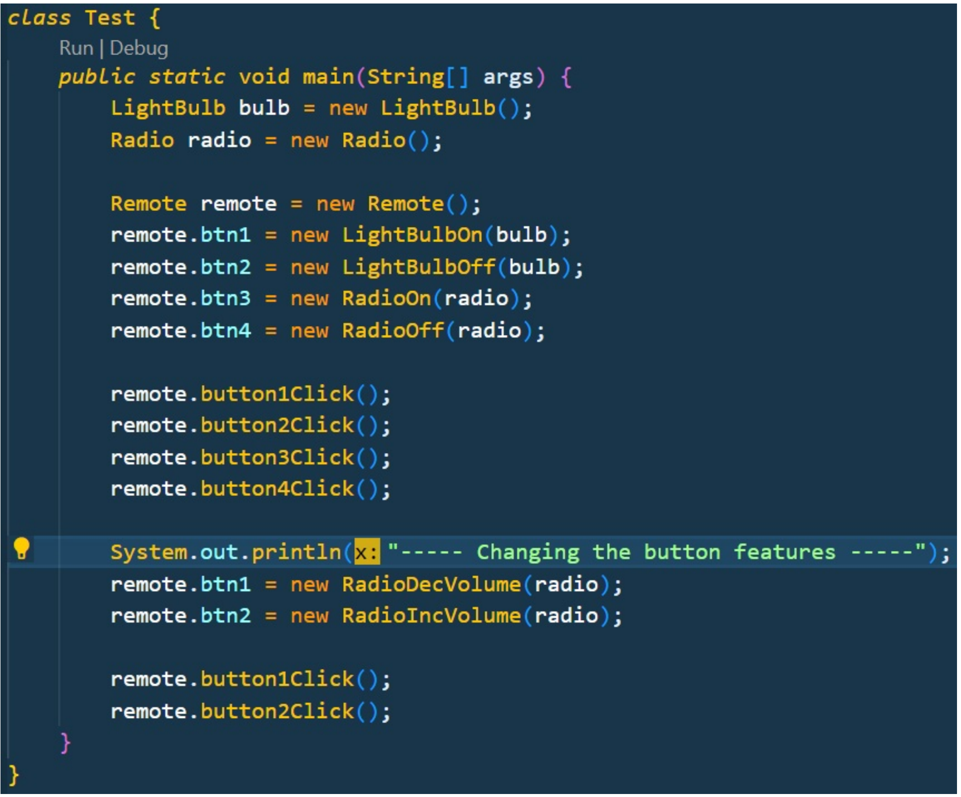
```java
public class Radio {
    int volume = 50;

    void radioOn() {
        System.out.println(x: "Radio is On");
    }

    void radioOff() {
        System.out.println(x: "Radio is Off");
    }

    void volumeIncrease() {
        this.volume += 1;
        System.out.println("Current Volume is: " + volume);
    }

    void volumeDecrease() {
        this.volume -= 1;
        System.out.println("Current Volume is: " + volume);
    }
}
```



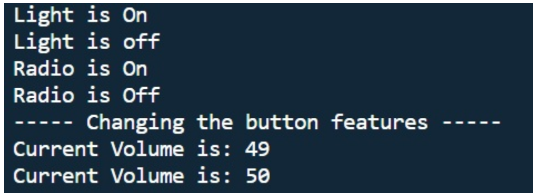
```java
public class LightBulb {
    void lightOn() {
        System.out.println(x: "Light is On");
    }

    void lightOff() {
        System.out.println(x: "Light is off");
    }
}
```

Remote →

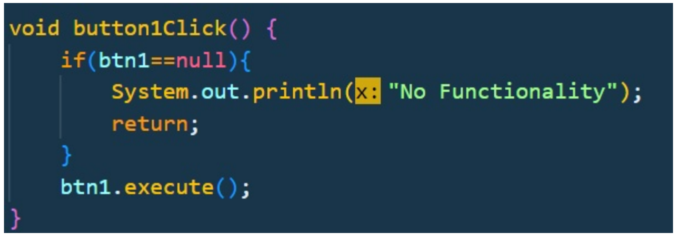
→ Reciever



```java
class Test {
    Run | Debug
    public static void main(String[] args) {
        LightBulb bulb = new LightBulb();
        Radio radio = new Radio();

        Remote remote = new Remote();
        remote.btn1 = new LightBulbOn(bulb);
        remote.btn2 = new LightBulbOff(bulb);
        remote.btn3 = new RadioOn(radio);
        remote.btn4 = new RadioOff(radio);

        remote.button1Click();
        remote.button2Click();
        remote.button3Click();
        remote.button4Click();

        System.out.println(x: "----- Changing the button features -----");
        remote.btn1 = new RadioDecVolume(radio);
        remote.btn2 = new RadioIncVolume(radio);

        remote.button1Click();
        remote.button2Click();
    }
}
```



```
Light is On
Light is off
Radio is On
Radio is Off
----- Changing the button features -----
Current Volume is: 49
Current Volume is: 50
```



```java
void button1Click() {
    if(btn1==null){
        System.out.println(x: "No Functionality");
        return;
    }
    btn1.execute();
}
```

Also add Null check for all Buttons.