```
public interface Command {
        public void execute();
}



public class LightOnCommand implements Command {
        Light light;

        public LightOnCommand(Light light) {
                this.light = light;
        }

        public void execute() {
                light.on();
        }
}




public class LightOffCommand implements Command {
        Light light;

        public LightOffCommand(Light light) {
                this.light = light;
        }

        public void execute() {
                light.off();
        }
}



public class Light {
        String location = "";

        public Light(String location) {
                this.location = location;
        }

        public void on() {
                System.out.println(location + " light is on");
        }

        public void off() {
                System.out.println(location + " light is off");
        }
}




public class StereoOnWithCDCommand implements Command {
        Stereo stereo;
```

```
        public StereoOnWithCDCommand(Stereo stereo) {
                this.stereo = stereo;
        }

        public void execute() {
                stereo.on();
                stereo.setCD();
                stereo.setVolume(11);
        }
}




public class StereoOffCommand implements Command {
        Stereo stereo;

        public StereoOffCommand(Stereo stereo) {
                this.stereo = stereo;
        }

        public void execute() {
                stereo.off();
        }
}



public class RemoteControlWithUndo {
        Command[] onCommands;
        Command[] offCommands;
        Command undoCommand;

        public RemoteControlWithUndo() {
                onCommands = new Command[7];
                offCommands = new Command[7];

                Command noCommand = new NoCommand();
                for(int i=0;i<7;i++) {
                        onCommands[i] = noCommand;
                        offCommands[i] = noCommand;
                }
                undoCommand = noCommand;
        }

        public void setCommand(int slot, Command onCommand, Command
offCommand) {
                onCommands[slot] = onCommand;
                offCommands[slot] = offCommand;
        }

        public void onButtonWasPushed(int slot) {
                onCommands[slot].execute();
                undoCommand = onCommands[slot];
        }
```

```
        public void offButtonWasPushed(int slot) {
                offCommands[slot].execute();
                undoCommand = offCommands[slot];
        }

        public void undoButtonWasPushed() {
                undoCommand.undo();
        }

        public String toString() {
                StringBuffer stringBuff = new StringBuffer();
                stringBuff.append("\n------ Remote Control -------
\n");
                for (int i = 0; i < onCommands.length; i++) {
                        stringBuff.append("[slot " + i + "] " +
onCommands[i].getClass().getName()
                                + "    " +
offCommands[i].getClass().getName() + "\n");
                }
                stringBuff.append("[undo] " +
undoCommand.getClass().getName() + "\n");
                return stringBuff.toString();
        }
}



public class NoCommand implements Command {
        public void execute() { }
        public void undo() { }
}



public class RemoteLoader {

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

                Light livingRoomLight = new Light("Living Room");
                Light kitchenLight = new Light("Kitchen");
                CeilingFan ceilingFan= new CeilingFan("Living Room");
                GarageDoor garageDoor = new GarageDoor("");
                Stereo stereo = new Stereo("Living Room");

                LightOnCommand livingRoomLightOn =
                                new LightOnCommand(livingRoomLight);
                LightOffCommand livingRoomLightOff =
                                new LightOffCommand(livingRoomLight);
                LightOnCommand kitchenLightOn =
                                new LightOnCommand(kitchenLight);
                LightOffCommand kitchenLightOff =
                                new LightOffCommand(kitchenLight);
```

```
                CeilingFanOnCommand ceilingFanOn =
                                new CeilingFanOnCommand(ceilingFan);
                CeilingFanOffCommand ceilingFanOff =
                                new CeilingFanOffCommand(ceilingFan);

                GarageDoorUpCommand garageDoorUp =
                                new GarageDoorUpCommand(garageDoor);
                GarageDoorDownCommand garageDoorDown =
                                new GarageDoorDownCommand(garageDoor);

                StereoOnWithCDCommand stereoOnWithCD =
                                new StereoOnWithCDCommand(stereo);
                StereoOffCommand  stereoOff =
                                new StereoOffCommand(stereo);

                remoteControl.setCommand(0, livingRoomLightOn,
livingRoomLightOff);
                remoteControl.setCommand(1, kitchenLightOn,
kitchenLightOff);
                remoteControl.setCommand(2, ceilingFanOn,
ceilingFanOff);
                remoteControl.setCommand(3, stereoOnWithCD,
stereoOff);

                System.out.println(remoteControl);

                remoteControl.onButtonWasPushed(0);
                remoteControl.offButtonWasPushed(0);
                remoteControl.onButtonWasPushed(1);
                remoteControl.offButtonWasPushed(1);
                remoteControl.onButtonWasPushed(2);
                remoteControl.offButtonWasPushed(2);
                remoteControl.onButtonWasPushed(3);
                remoteControl.offButtonWasPushed(3);
        }
}



public class RemoteLoader {

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

                Light livingRoomLight = new Light("Living Room");

                LightOnCommand livingRoomLightOn =
                                new LightOnCommand(livingRoomLight);
                LightOffCommand livingRoomLightOff =
                                new LightOffCommand(livingRoomLight);

                remoteControl.setCommand(0, livingRoomLightOn,
livingRoomLightOff);

                remoteControl.onButtonWasPushed(0);
                remoteControl.offButtonWasPushed(0);
```

```
                System.out.println(remoteControl);
                remoteControl.undoButtonWasPushed();
                remoteControl.offButtonWasPushed(0);
                remoteControl.onButtonWasPushed(0);
                System.out.println(remoteControl);
                remoteControl.undoButtonWasPushed();

                CeilingFan ceilingFan = new CeilingFan("Living Room");

                CeilingFanMediumCommand ceilingFanMedium =
                                new
CeilingFanMediumCommand(ceilingFan);
                CeilingFanHighCommand ceilingFanHigh =
                                new CeilingFanHighCommand(ceilingFan);
                CeilingFanOffCommand ceilingFanOff =
                                new CeilingFanOffCommand(ceilingFan);

                remoteControl.setCommand(0, ceilingFanMedium,
ceilingFanOff);
                remoteControl.setCommand(1, ceilingFanHigh,
ceilingFanOff);

                remoteControl.onButtonWasPushed(0);
                remoteControl.offButtonWasPushed(0);
                System.out.println(remoteControl);
                remoteControl.undoButtonWasPushed();

                remoteControl.onButtonWasPushed(1);
                System.out.println(remoteControl);
                remoteControl.undoButtonWasPushed();
        }
}



public class MacroCommand implements Command {
        Command[] commands;

        public MacroCommand(Command[] commands) {
                this.commands = commands;
        }

        public void execute() {
                for (int i = 0; i < commands.length; i++) {
                        commands[i].execute();
                }
        }

        //NOTE:  these commands have to be done backwards to ensure
proper undo functionality
        public void undo() {
                for (int i = commands.length -1; i >= 0; i--) {
                        commands[i].undo();
                }
        }
}
```