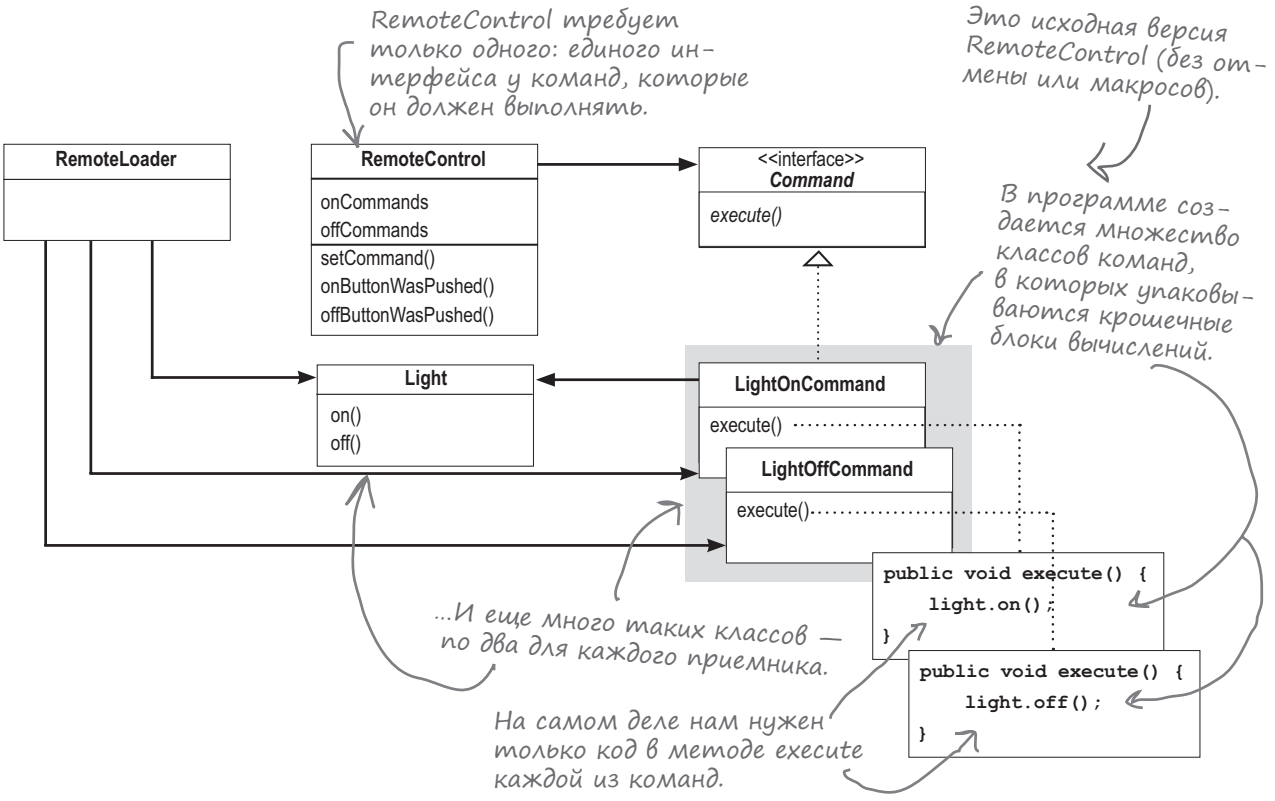
Паттерн Команда означает множество классов команд

При использовании паттерна Команда в программе появляется множество мелких классов — конкретных реализаций Command; каждый класс инкапсулирует запрос к соответствующему приемнику. В нашей реализации пульта для каждого класса приемника создаются два класса команд. Например, для приемника Light создаются классы LightOnCommand и LightOffCommand; для приемника GarageDoor — классы GarageDoorUpCommand и GarageDoorDownCommand, и так далее. Получается слишком много лишнего для создания небольших блоков вычислений, которые имеют единый интерфейс к RemoteControl:



Так ли необходимы все эти классы команд?

Команда — не более чем упакованный блок вычислений. Команды позволяют определить общий интерфейс для поведения многих разных приемников (осветительных систем, джакузи, стереосистем), каждый из которых обладает собственным набором действий.

А теперь представьте, что вы можете оставить общий интерфейс для всех команд, но вынести все вычисления из конкретных реализаций Command и использовать их напрямую. При этом вы избавляетесь от всех лишних классов, а код упрощается. Что ж, с лямба-выражениями это возможно. Давайте посмотрим, как это делается...

Упрощение кода RemoteControl с лямбда-выражениями

Если вы еще не знакомы с лямбда-выражениями (они были добавлены в Java 8), то вряд ли сразу привыкнете к ним. Вероятно, несколько ближайших страниц вам будут понятны, но если потребуется — обращайтесь к справочнику Java за информацией о синтаксисе и семантике лямбда-выражений.

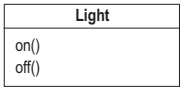
Как вы уже видели, паттерн Команда достаточно прямолинеен. Тем не менее, языка Java предоставляет удобный инструмент, который позволяет еще сильнее упростить код — а именно лямбда-выражения. Лямбда-выражение представляет собой сокращенную форму записи метода (как последовательности вычислений) точно в том месте, где она используется. Вместо того, чтобы создавать отдельный класс с методом, создавать экземпляр этого класса, а затем вызывать метод, вы просто указываете: «Вот метод, который нужно вызвать» при помощи лямбда-выражения. В нашем случае вызываться должен метод execute().

Чтобы понять, как это делается, заменим объекты LightOnCommand и LightOffCommand лямбда-выражениями. Выполните следующие действия, чтобы команды включения/выключения света реализовались лямбда-выражениями вместо объектов команд:

Шаг 1. Создание приемника

Этот шаг нисколько не изменился.

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



Шаг 2. Реализация команд пульта с использованием лямбда-выражений

Здесь происходит все самое интересное. Теперь вместо того, чтобы создавать объекты LightOnCommand и LightOffCommand для передачи remoteControl.setCommand(), мы просто передаем вместо каждого объекта лямбда-выражение с кодом из соответствующего метода execute():

Два лямбда-выражения.

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

Лямбда-выражения передаются в качестве команд методу setCommand.

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

Шаг 3. Активация кнопок

Этот шаг тоже не изменился — если не считать того, что при вызове метода onButtonWasPushed(0) команда в слоте 0 представлена объектом функции (созданным при помощи лямбда-выражения).

Когда мы вызываем execute() для команды, этот метод сопоставляется с методом, определяемым лямбда-выражением, который затем выполняется.

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

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

В элементе 0 массива onCommands хранится лямбда-выражение, переданное setCommand на шаге 2. Метод execute() сопоставляется с методом в лямбда-выражении, который затем выполняется.

```
() -> { livingRoomLight.on(); }
```

Кого вы пытаетесь надуть? У лямбда-выражения, которое передается методу setCommand, даже нет метода execute. И как будет вызываться метод в лямбда-выражении?

По волшебству, как же еще?

Ладно, мы пошутили... никакого волшебства. Мы используем лямбда-выражения для замены объектов Command, а интерфейс Command содержит всего один метод: execute(). Лямбда-выражение, которое мы используем, должно иметь совместимую сигнатуру — так оно и есть: метод execute() не получает аргументов (как и наше лямбда-выражение) и не возвращает значения (как и наше лямбда-выражение). Компилятор всем доволен.

Лямбда-выражение передается в параметре Command метода setCommand():

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

Нет аргументов... Не возвращает значение.

```
public void setCommand(int slot,
                       Command onCommand,
                       Command offCommand) {
    ...
}
```

```
() -> { livingRoomLight.on(); }
```

Нет аргументов...

Не возвращает значение.

Да, сигнатура лямбда-выражения соответствует методу Command. Подходит!

Компилятор проверяет, что интерфейс Command содержит ровно один метод, соответствующий лямбда-выражению. Так оно и есть: это метод execute().
Затем, когда для этой команды вызывается execute(), будет вызван метод из лямбда-выражения:

```
onCommands[slot].execute();
```

Интерфейс Command содержит только один метод; именно этот метод будет заменен лямбда-выражением.

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

```
() -> { livingRoomLight.on(); }
```

Запомните: если интерфейс параметра, в котором передается лямбда-выражение, содержит один (и только один!) метод, и сигнатура этого метода совместима с сигнатурой лямбда-выражения, все сработает как надо.

Ссылки на методы

Чтобы еще сильнее упростить код, можно воспользоваться ссылками на методы. Если передаваемое лямбда-выражение вызывает всего один метод, вместо лямбда-выражения можно передать ссылку на метод. Это делается так:

```
remoteControl.setCommand(0, livingRoomLight::on, livingRoomLight::off);
```

Ссылка на метод on() объекта livingRoomLight.

Ссылка на метод off() объекта livingRoomLight.

Вместо того чтобы передавать лямбда-выражение, которое вызывает метод on() объекта livingRoomLight, мы *передаем ссылку на сам метод.*

А что, если лямбда-выражение должно выполнять сразу несколько операций?

Иногда лямбда-выражения, используемые для замены объектов Command, не ограничиваются одной операцией. Давайте посмотрим, как заменить объекты stereoOnWithCDCommand и stereoOffCommand лямбда-выражениями. Далее будет приведен полный код RemoteLoader, чтобы вы поняли, как все эти идеи сочетаются друг с другом.

Объект stereoOffCommand выполняет всего одну простую команду:

```
stereo.off();
```

Для этой команды вместо лямбда-выражения можно использовать ссылку на метод `stereo::off`. Но stereoOnWithCDCommand выполняет не одну, а *три* операции:

```
stereo.on();
stereo.setCD();
stereo.setVolume(11);
```

В таком случае использовать ссылку на метод не удастся. Вместо этого придется либо записывать лямбда-выражение во встроенном виде, либо создать его отдельно, присвоить имя, а потом передать методу setCommand() объекта remoteControl по имени. Вариант с созданием отдельного лямбда-выражения и присвоением ему имени выглядит так:

```
Command stereoOnWithCD = () -> {
    stereo.on(); stereo.setCD(); stereo.setVolume(11);
};
remoteControl.setCommand(3, stereoOnWithCD, stereo::off);
```

Это лямбда-выражение выполняет три операции (как и метод execute у stereoOnWithCDCommand).

Лямбда-выражение можно передать по имени.

Обратите внимание: Command используется как тип лямбда-выражения. Лямбда-выражение сопоставляется с методом execute() интерфейса Command и с параметром Command, в котором оно передается методу setCommand().