Свои классы

Когда существующих не хватает

Содержание

1. Как написать свой класс и зачем

В предыдущей лекции мы решали некоторые задачи и остановились на последней. В которой нужно было написать аналог чат-бота. Если вы не решили сами или же не смотрели хотя бы решение, то настоятельно рекомендую хотя бы ознакомиться с условиями задачи.

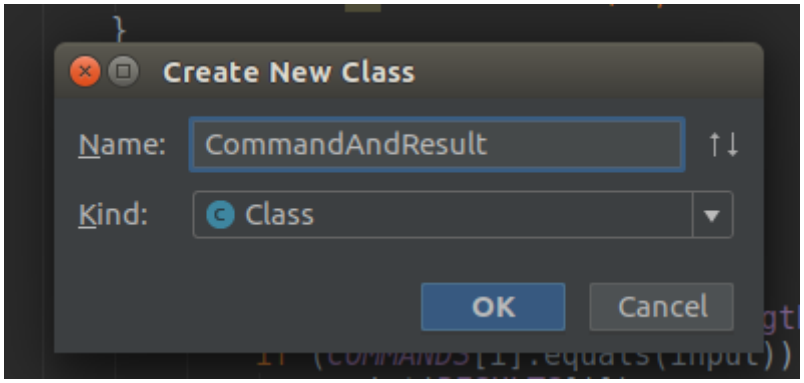
Для начала давайте подытожим на данный момент что мы изучили

Сначала мы вывели в консоль строку, после разобрались как выводить в консоль что угодно, как работать со строками и числами. Операции над числами, типы чисел и их диапазоны. Мы изучили методы, возвращаемый тип метода, как вызывать одни в других и области видимостей переменных. Константы. Изучили логические переменные и операторы, конструкции, проверки, циклы for и while, научились прерывать бесконечные циклы. Массивы данных и ввод в консоль.

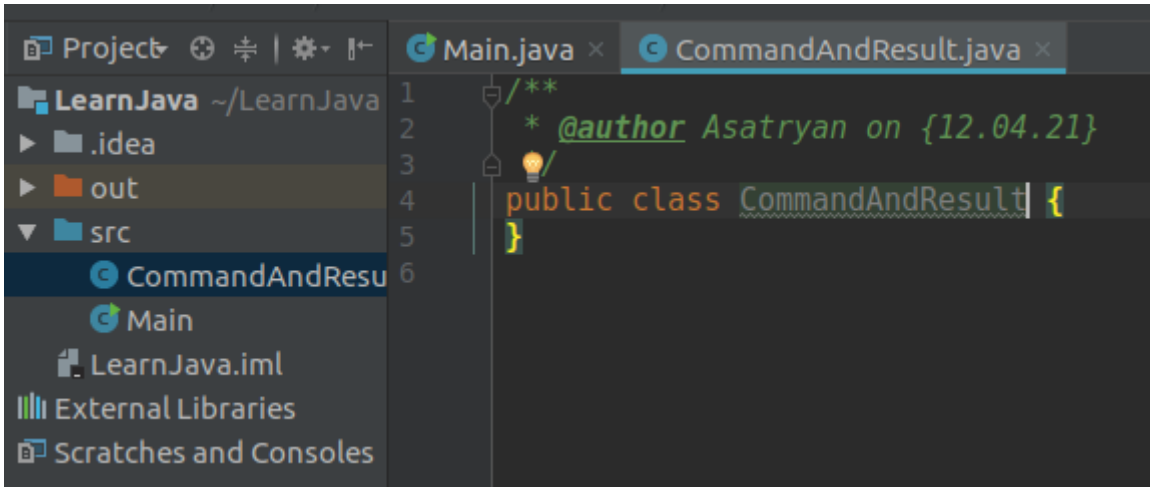
И до сих пор мы работали лишь с существующими классами и типами данных. В последней задаче у нас было 2 массива для данных. Первый для команд и второй для результата.

И давайте теперь подумаем вот над чем. Если завтра я захочу изменить список команд, то мне придется вводить изменения и в массив команд и в массив результата, так? Даже если мне просто нужно будет передвинуть например команду финиша в конец массива, чтобы в любом случае получать его вне зависимости от длины массива, то придется ввести аналогичное изменение и во второй массив.

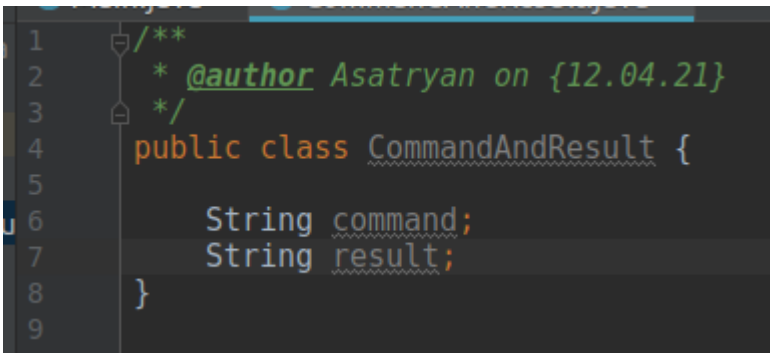
Было бы классно как-нибудь соединить эти 2 массива данных в 1. Чтобы каждой команде соответствовал какой-нибудь результат. И мы это можем сделать! Мы же создали класс Main. Значит можем создать и класс для хранения команды и результата. Для этого просто кликаем мышкой правой кнопкой на папку src и создаем new → java class.



Вводим название и жмем ок. Готово! Вы создали еще один класс!



И конечно же он пустой. Теперь, идем дальше. Мы его создавали чтобы хранить в нем и команду и результат. Так? Значит мне нужно переделать 2 массива строкового типа в массив моего класса, где будет 2 поля строкового типа. Я могу просто так взять и написать в нем их? Попробуем же!



Неожиданно, но да! Все получилось. Среда разработки выделяет серым все то, что пока не используется. Ок, мы создали класс с 2 полями (переменные класса). Теперь нам нужно использовать этот класс в нашей программе, т.е. в Main. Но как? Просто обратите внимание как мы до сих пор инстанциировали обьекты. Вспомним как мы создавали новый массив new int[] {}. И вспомним как мы инициализировали сканер строк в консоли

Scanner scanner = new Scanner(System.in);

и там и там присутствует ключевое слово new. Оно нужно для инстанциирования обьекта класса (или же массива). Попробуем же заменить 2 массива строк массивом нашего класса.

```
private static final CommandAndResult[] COMMAND_AND_RESULT = new CommandAndResult[4];

private static final String[] COMMANDS = new String[]{
        "/help", "/start", "/end", "/getLocation"
};
private static final String[] RESULTS = new String[]{
        "", "Bot started!...", "Bot ended", "location is Moscow"
};
```

Обратите внимание. Если раньше у нас был массив строк String[], то теперь у нас массив собственного класса – CommadAndResult[]. Но мы создали его пустым! Хотя и с размером 4.

Давайте попробуем сразу его заполнить данными, ровно так же как и строки. Но если строку мы создаем без ключевого слова new (хотя так тоже можно, типа String text = new String(“value”), просто в джава можно для строк не писать так, а сразу String text = “value”;), то для создания нашего обьекта нам он понадобится. Смотрим

```
private static final CommandAndResult[] COMMAND_AND_RESULT = new CommandAndResult[]{
        new CommandAndResult(), new CommandAndResult(), new CommandAndResult(), new CommandAndResult()
};

private static final String[] COMMANDS = new String[]{
        "/help", "/start", "/end", "/getLocation"
};
private static final String[] RESULTS = new String[]{
        "", "Bot started!...", "Bot ended", "location is Moscow"
};
```

Эм, ну окей. Мы создали массив и заполнили его обьектами нашего класса. Но мы же не передали значения никак. Как это сделать? Помните я раньше говорил, что можно создать массив чисел и заполнить его потом?

```
int[] array = new int[2];
array[0] = 5;
array[1] = 6;
```

Так же можно было бы сделать и с нашим классом. Но мы заполнили его пустыми обьектами. Там нет информации по команде и результату. Значит, нужно по аналогии с массивом числа заполнить его. Но как? Давайте посмотрим.

```
public static void main(String[] args) {
    COMMAND_AND_RESULT[0].command = "/help";
    COMMAND_AND_RESULT[0].result = "";
    COMMAND_AND_RESULT[1].command = "/start";
    COMMAND_AND_RESULT[1].result = "Bot started!...";
```

Мы обращаемся к каждому члену массива, который является обьектом класса нашего CommandAndResult и заполняем поля/переменные класса обращаясь к ним через точку так же как мы обращались к System.out.println. Но вы скажете – а где профит? Работы стало еще больше! Мы раньше создавали массив строк и заполняли его сразу же. А сейчас что? Создаем сначала пустые обьекты, а потом уже заполняем по полям? Слишком много работы. Давай упрощать!

Кстати, запомните. Мы создаем класс, а потом инстанциируем обьект этого класса. Т.е. создать класс можно через правый клик на пакет src New → Java class и мы имеем public class MyOwnClass. А создать обьект этого класса можно только через new MyOwnClass();

Подробнее чуть ниже.

2. Структура класса, конструктор

Итак, друзья мои. Мы написали свой класс с 2 полями строкового типа. Смогли заполнить их, хоть и получилось это у нас не так красиво. А как сделать красиво? Было бы классно во время создания обьекта класса сразу передать значения. Для этого давайте вернемся к тому, что у нас есть. На пару минут отойдем от задачи про чатбота и поэкспериментируем с классом в чистом public static void main.

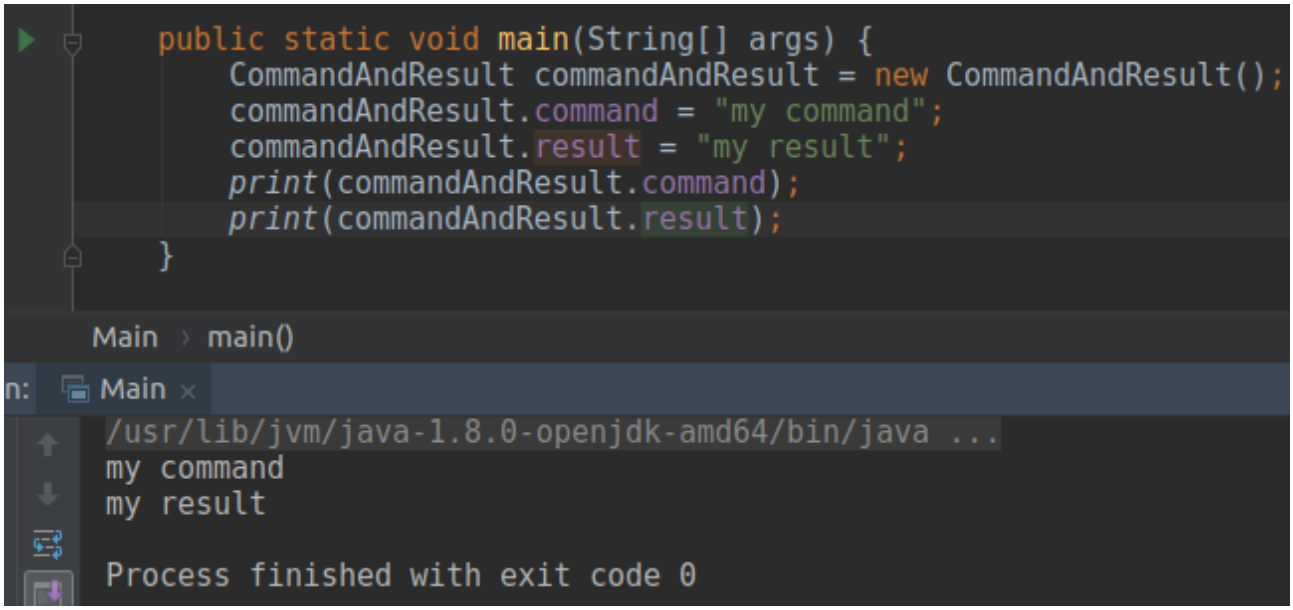
```
public static void main(String[] args) {
    new CommandAndResult();
}
```

Вот мы создали обьект класса. Ок. А как к нему обращаться? Нужно выделить в переменную. Мы знаем как – Ctrl+Alt+V.

```
public static void main(String[] args) {
    CommandAndResult commandAndResult = new CommandAndResult();
}
```

Ровно так же как и со строкой – тип переменной, имя переменной, создание переменной.

В нашем случае мы пишем сначала имя класса, после имя переменной, new имя класса и круглые скобки. О них чуть позже. Давайте инициализировать поля класса.



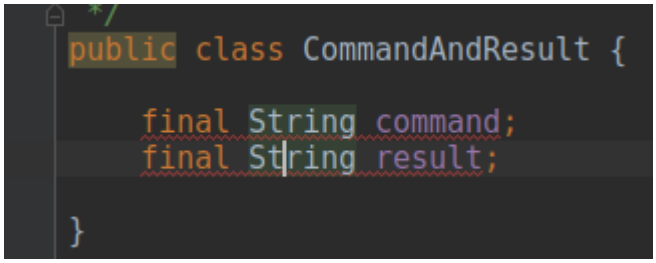
Мы в созданном обьекте класса обращаемся к полям класса и инициализируем строки. Чтобы быть уверенным в том, что мы их проинициализировали, мы можем просто вывести в консоль их. Для этого обращаемся к полям класса через точку ровно так же как и при инициализации. И здесь у вас встанет резонный вопрос. Если поля класса строкового типа, то их можно менять и после того как один раз проинициализировали, так? Вспомните про повторную инициализацию переменных. Пробуем!

Итак, как видим, после повторной инициализации поле класса поменялось. Но нам не нужно допускать такого.

ОЧЕНЬ ВАЖНО

Если ваш класс имеет какие-либо поля, то старайтесь не допускать их изменения. Обьект с другими полями класса должен быть другим обьектом.

И давайте вспомним, как мы можем запретить повторную инициализацию. Правильно! С помощью ключевого слова final перед обьявлением переменной. Попробуем?



Итак, мы обьявили переменные final и это значит что их нужно инициализировать. Как видите наша среда разработки сразу подчеркнула поля. Вы знаете что делать: ставим курсор на поле и жмем Alt+Enter. И получаем предложение : Add constructor parameters. Жмем ок что бы это ни значило. И получаем следующее.

Что же произошло? Мы получили конструктор класса, в который получаем аргументами 2 строки. Помните функции/методы? Они получали в аргумент что-то и с ними что-то делали. Так что это за штука такая – конструктор? Это штука, которая порождает обьект класса. Т.е. создает новый инстанс обьекта. И если раньше мы создавали обьект класса new CommandAndResult() с пустыми круглыми скобками, т.е. не передавали ничего, то теперь давайте посмотрим на наш мейн метод.

```java
public class CommandAndResult {

    final String command;
    final String result;

    public CommandAndResult(String command, String result) {
        this.command = command;
        this.result = result;
    }
}
```

А тем временем в нашем мейне произошло нечто ужасное

```java
public static void main(String[] args) {
    CommandAndResult commandAndResult = new CommandAndResult(command, result);
    commandAndResult.command = "my command";
    commandAndResult.result = "my result";
    print(commandAndResult.command);
    print(commandAndResult.result);
    commandAndResult.command = "new command";
    print(commandAndResult.command);
    print(commandAndResult.result);

}
```

Мы поменяли конструктор нашего класса и среда разработки нашла использование этого конструктора и поменяла в нем вызов. Т.е. теперь, если вам нужно создать обьект класса, то нужно сразу в конструкторе при инициализации передать данные. Давайте исправим этот код.

```java
public static void main(String[] args) {
    CommandAndResult commandAndResult = new CommandAndResult( command: "my command",  result: "my result");
    print(commandAndResult.command);
    print(commandAndResult.result);
```

```
Main
Main ×
/usr/lib/jvm/java-1.8.0-openjdk-amd64/bin/java ...
my command
my result

Process finished with exit code 0
```

Теперь, мы создаем обьект класса и сразу передаем аргументами ему значения. Теперь не нужно сначала создавать обьект и после уже отдельные поля инициализировать. Но что же будет если мы попробуем изменить поля класса? А то же, что и если мы попробуем повторно инициализировать переменную final типа.

Так что постарайтесь запомнить это. Если вы создаете обьект класса, то передайте ему все нужные значения в аргументы конструктора. Считайте, что конструктор класса это мейн метод этого класса, который выполняется когда вы создаете обьект. Чтобы в этом убедиться, давайте просто выведем в консоль строку изнутри конструктора. Да, это легально. Ведь конструктор ни что иное как метод, верно?

```
public static void main(String[] args) {
    CommandAndResult commandAndResult = new CommandAndResult( command: "my command",  result: "my result");
    print(commandAndResult.command);
    print(commandAndResult.result);
    commandAndResult.command = "new command";
    print(commandAndResult.command);
    print(commandAndResult.result);

}
```

```
Main › main()
Build
Information: java: Errors occurred while compiling module 'LearnJava'
Information: javac 1.8.0_282 was used to compile java sources
Information: 12.04.21 12:00 - Compilation completed with 1 error and 0 warnings in 545 ms
/home/johnny/LearnJava/src/Main.java
Error:(21, 25) java: cannot assign a value to final variable command
```

```
public class CommandAndResult {

    final String command;
    final String result;

    public CommandAndResult(String command, String result) {
        this.command = command;
        this.result = result;
        System.out.println("command: " + this.command + ", result: " +  this.result);
    }
}
```

И в мейн методе Мейн класса просто создадим обьект. Посмотрим что будет.

```
public static void main(String[] args) {
    CommandAndResult commandAndResult = new CommandAndResult( command: "my command",  result: "my result");
}
```

```
public static void main(String[] args) {
    CommandAndResult commandAndResult = new CommandAndResult( command: "my command",  result: "my result");
}
Main › main()
Main
/usr/lib/jvm/java-1.8.0-openjdk-amd64/bin/java ...
command: my command, result: my result

Process finished with exit code 0
```

Теперь мы можем видеть, что конструктор класса вызвался. Кстати говоря о конструкторе. Давайте внимательно посмотрим на код, который сгенерила наша среда.

Мы в конструктор передаем аргумент String command, и в то же время у нас поле класса с тем же именем. И мы знаем, что нельзя обьявлять более 1 переменной в той области видимости в которой мы работаем. Поэтому среда сгенерировала код

this.command = command.

Что же значит this? Это именно экземпляр этого класса. И мы ему через точку инициализируем поле. т.е. обращаемся к полю класса через точку и слово this и пишем в нее аргумент. Чтобы это понять, достатчно переименовать поля класса.

```java
public class CommandAndResult {

    final String fieldCommand;
    final String fieldResult;

    public CommandAndResult(String command, String result) {
        fieldCommand = command;
        fieldResult = result;
        System.out.println("command: " + fieldCommand + ", result: " + fieldResult);
    }
}
```

Вы вольны решать сами как называть поля класса. Но для удобства можно обозначить поля класса каким-то образом, чтобы различать от аргументов конструктора. Как видите в этом случае нам не нужно использовать this. Хотя вы так же можете написать

```java
public class CommandAndResult {

    final String fieldCommand;
    final String fieldResult;

    public CommandAndResult(String command, String result) {
        this.fieldCommand = command;
        this.fieldResult = result;
        System.out.println("command: " + fieldCommand + ", result: " + fieldResult);
    }
}
```

Вроде разобрались, но вспомним для чего мы все это затеяли. Чтобы упростить наш код программы чатбота. Так что ж, вернемся к нему!

Мы удалили массивы строк и заменили их массивом обьектов нашего класса.

Так как все команды нужно получить в цикле, то мы проходим по нему и перезаписываем элемент массива, ведь мы не можем менять поле класса. И теперь везде, где мы обращались к элементам массива команд мы обращаемся к элементам массива нашего класса и через точку получаем или команду или результат. Просто сравните старый код и новый и вы поймете что есть что. Самое главное – теперь легко добавлять в код новую пару команды и результата или же менять их местами. Вы сразу берете пару значений, а не 2 раза совершаете одно и то же действие сначала для команды а потом для результата.

Надеюсь все было понятно, если нет, то пишите в чат.

Для закпреления темы вот вам пара заданий

1. Написать класс Личности, в нем должны быть поля имени и фамилии как минимум, остальное на ваше усмотрение.

2.Написать класс для хранения некоего айди числового и строки с информацией

3.Написать класс для урока. В нем должен быть номер урока и флаг – завершен или нет.

```java
import java.util.Scanner;

public class Main {

    private static final CommandAndResult[] COMMAND_AND_RESULT = new CommandAndResult[]{
            new CommandAndResult( command: "/help",  result: ""),
            new CommandAndResult( command: "/start",  result: "Bot started!"),
            new CommandAndResult( command: "/getLocation",  result: "location is Moscow"),
            new CommandAndResult( command: "/end",  result: "Bot ended")
    };

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        String commands = "commands available for this bot \n";
        for (CommandAndResult item : COMMAND_AND_RESULT) {
            commands += item.command + "\n";
        }
        COMMAND_AND_RESULT[0] = new CommandAndResult(COMMAND_AND_RESULT[0].command, commands);

        String input;
        boolean found;
        while (true) {
            input = scanner.nextLine();
            found = false;
            for (int i = 0; i < COMMAND_AND_RESULT.length; i++) {
                if (COMMAND_AND_RESULT[i].command.equals(input)) {
                    print(COMMAND_AND_RESULT[i].result);
                    found = true;
                    break;
                }
            }
            if (COMMAND_AND_RESULT[COMMAND_AND_RESULT.length - 1].command.equals(input)) {
                break;
            }
            if (!found) {
                print("no command found for that input. Try typing " + COMMAND_AND_RESULT[0].command);
            }
        }
    }

    private static void print(Object object) {
        System.out.println(object);
    }
}
```

```
Run:  Main
/usr/lib/jvm/java-1.8.0-openjdk-amd64/bin/java ...
go
no command found for that input. Try typing /help
/help
commands available for this bot
/help
/start
/getLocation
/end

/start
Bot started!
/getLocation
location is Moscow
/end
Bot ended

Process finished with exit code 0
```