

Алексей Назаров.

Джава для взрослых

Предисловие.

Блуждая в интернетах и натываясь на вопросы типа: “Мне уже за (можете подставить любое число, но не поверите, я встречал даже 26!!!) не поздно ли изучать Java?” я хотел кричать:

- “Друг! Если ты еще можешь попадать пальцами по клавиатуре, тебе не поздно!”.

Я первый раз увидел компьютер в 22 года и мне даже дали на нем попечать! Не могу Вам описать свои ощущения, когда я сидел около этого светящегося и шелестящего вентилятором чуда. У меня закипала кровь, я точно знал, что был рожден для ИТ, но жизнь распорядилась по-другому.

Теперь, когда я почти реализовал свою мечту и люди стали спрашивать меня, как правильно это делать я понял, что каждому не ответить, - нужно слишком много времени. Я решил, что напишу книгу, в которой постараюсь ответить на все вопросы людей, желающих стать разработчиками (программистами), а особенно на языке Java – на котором я пишу сам.

Я хотел назвать эту книгу “Java для тех, кому за...”, в общем вроде как не для молодых. И задумался: молодость - это когда тебе еще не “за...” сколько? Ответ пришел сам собой: эти “за” мы себе устанавливаем сами, а объективно их нет: есть только СЕЙЧАС!

Вот и ответьте себе сами: “Не поздно учить JAVA сейчас?”

Не поздно <подставьте что угодно> сейчас? Риторический вопрос, правда? И ответ понятный.

Эту книгу я писал, представляя тебя (прости за фамильярность, дорогой читатель), как самого себя: человека средних лет, пользующегося компьютером для составления документов и отчетов и периодически испытывающего проблемы с составлением таблиц в Excel. Поэтому, возможно, где-то описания слишком подробные, через-чур утомительные. Тех, кто лучше владеет компьютером прошу набраться терпения и пролистывать такие слишком подробные описания.

Глава 1. Что нужно чтобы стать программистом.

Здравствуйте!

Меня зовут Алексей, мне 44 года и уже год я работаю программистом. Эту книгу я решил написать, чтобы помочь людям, которые хотят стать программистами, но не знают с чего начать или уже изучают Java, но не знают куда двигаться. Эта книга своего рода ментор - она будет Вас направлять в процессе обучения вплоть до Вашего первого трудоустройства.

Самая большая проблема человека начинающего любую деятельность в отсутствии опыта. Конечно, это быстро и эффективно решается если есть наставник. Но не просто наставник, а человек разбирающийся в предмете и умеющий структурировать и приоритизировать информацию.

Таких людей не много и обычно они очень заняты и хотят достаточно ощутимую оплату за свое потраченное время. Когда я изучал язык программирования Java у меня такого ментора не было.

Мне пришлось пройти путь в одиночестве и помогли мне тут 2 фактора:

1. У меня был 15-летний опыт планирования, благодаря которому я умею быстро отделять главное от второстепенного.

2. У меня был 9 летний опыт системного администрирования. Это конечно никак не связано с программированием, но иногда дает представление о том в какой стороне проблема, если программа, например, не может получить данные из базы. Если у Вас нет опыта администрирования - не переживайте. Большинство программистов его не имеют.

Далее я опишу как я стал программистом. Возможно, кому-то это будет не интересно или покажется затянутым, но привожу все это здесь исключительно для того, чтобы Вы могли убедиться в двух вещах:

1. Мечты сбываются если не бояться мечтать и двигаться к мечте.

2. Все, что я советую в этой книге, я проделал сам. А как говорится: только тот, кто помог себе - знает, как помочь другим.

1.1 Как я стал программистом.

Мой путь к программированию был очень длинным - я с детства мечтал стать программистом, но из-за проблем с математикой в школе мне говорили, что это невозможно. И я стал юристом.

Однако меня всегда тянуло к ИТ. В 1998 году я не мог себе позволить купить компьютер и купив жесткий диск вечером после работы подключил его к рабочему ПК, установил Linux Mandrake 8.0 -

это была первая ОС за исключением DOS и Windows 95, что я увидел. Непередаваемый восторг!

Через пару лет я купил собственный компьютер и установил на него несколько операционных систем. В свободное время я изучал операционные системы Linux и Unix, так как установленная на всех ПК операционная система для домохозяек вызывала у меня отторжение.

Однажды, сидя в интернете, (через модем) мы с другом увидели рекламу каких-то ИТ услуг. На баннере был изображен системный администратор в свитере и джинсах, сидящий между серверных стоек, на коленях у него лежал ноутбук, подключенный к серверу.

Я воскликнул что хочу быть таким. На что мой друг ответил - хочешь, значит будешь. Я посмеялся - где ИТ и где я - юрист, не осиливший школьную алгебру...

В свободное время я продолжал изучать операционные системы. Конечно, без постоянного доступа к интернету это происходило в основном методом тыка. Когда появилась виртуализация, я по вечерам, после работы, играл с виртуальными машинами создавая сеть, настраивал права доступа, поднимал веб-серверы и т.д.

В 2011 году после четвертого сокращения штатов моя и без того не головокружительная карьера в очередной раз рухнула, но в этот раз уже ниже плинтуса и я решил, что пора заняться любимым делом и осуществить мечту - я устроился работать системным администратором.

Ведомственная организация, в которой я стал системным администратором, имела ряд специфических требований к кандидатам по причине принадлежности к органам правопорядка, а заработная плата, учитывая специфику оставляла желать лучшего.

Так что я, как линуксоид хоть и с домашним опытом, был практически вне конкуренции - желающих идти работать сисадмином на эту вакансию не было: те, кто работал в организации не фанатели от операционных систем, а линуксоиды, приходившие с рынка труда, прослушав требования и условия работы хмыкнув удалялись в светлое будущее.

И вот я - системный администратор! Мне не верилось! Я - тот парень из рекламы что я видел 13 лет назад!

Поработав несколько лет, я пришел к выводу что пора двигаться дальше чтобы получить больше опыта и увеличить зарплату и тут меня ждал сюрприз: большинство компаний не мечтали нанять 38-летнего админа: не успевал я отослать резюме, как через несколько минут получал отказ.

Однажды мне пришел отказ ровно через 1 минуту после моего отклика на вакансию. Я разыскал в сети менеджера по персоналу (HR), которая мне прислала отказ, и вежливо поинтересовался в чем причина отказа. В приватной беседе чата одной из социальных сетей она пояснила, что ее начальник не понимает, как он будет руководить людьми старше себя. Так, что поиск первой моей работы в условиях равной конкуренции затянулся на полтора года.

Очень кстати мне помогла книга одного парня из Москвы. К сожалению, сейчас я уже не вспомню ни ее названия, ни фамилии автора. Он был успешный бизнесмен, в прошлом системный администратор и описывал как правильно развиваться в ИТ как системный администратор. Как повышать свою стоимость на рынке и избегать распространенных ошибок (особенно не отращивать свитер, грязную кружку и чувство собственной важности, как делали классические администраторы 90-х). Книга стоила тогда целых

полторы тысячи, что было ощутимо для моего бюджета. Я раздумывал неделю и решился. Эта книга сильно продвинула меня.

Тем не менее, как я уже писал выше, в том числе в силу возраста, сменить работу мне удалось только после двухмесячных вечерних курсов английского, по результатам которых я прошел конкурс в русско-американскую компанию инженером технической поддержки второй линии.

Чтобы Вы понимали это большой шаг назад по карьерной лестнице - обычно из техподдержки переходят в системные администраторы или (что значительно реже) в программисты.

В этой компании, кроме технического уровня, требовался еще достаточный для общения английский и хорошее здоровье: четыре смены в неделю по 12 часов и 2 из них - ночные. Старше меня в компании был только еще один инженер - все остальные ребята от 18 до 30 лет. Это был физически и психологически сложный период, однако благодаря этому моя карьера в следующих компаниях как системного администратора пошла в гору. Дальнейший период моей трудовой деятельности я опущу - там уже не было таких примечательных рывков и дауншифтов: просто движение вверх.

Примерно в 2018 году я понял, что больше не хочу быть системным администратором - новизна пропала. Конечно, появился ряд новых технологий, но я понял, что мне скучно изучать системы их принципы работы и параметры настроек. Я сам хочу создавать, творить. Так я задумался чтобы всерьез, во второй раз, сменить свой род деятельности и стать программистом.

Я тогда работал в одном банке ИТ-инженером. Мне было поручено задание: при помощи языка Python связать две системы, одна из них - это система заявок Jira, а вторая - для развертывания

ПО Jenkins. На тот момент все, что я знал о Python, что это язык программирования, который имеет низкий порог вхождения.

Проект занял у меня 2 недели, безусловно мой код был ужасен и кошмарен, и надеюсь, что его никто не видел, но система регулярно отплевывалась заявками и все были довольны. А мне очень понравилось сидеть в IDE - среде разработки и творить. Я решил изучить Python и стать программистом. И безусловно, попался бы в ловушку, если бы не совет из книги для системных администраторов, что я упоминал выше, - постоянно следить за рынком труда. Я решил изучить рынок труда - насколько разработчики Python востребованы в Новосибирске. Оказалось, что практически ни насколько (напоминаю год был 2018, удаленка отсутствовала - пандемия еще не пришла).

Это был неприятный сюрприз, который заставил меня задуматься какой язык выбрать. Так я пришел к выводу что надо изучать Java. (О выборе языка я расскажу далее во 2 главе).

Я стал покупать один известный онлайн курс и выполнять задания. На этом ресурсе я потратил полгода. Именно потратил. Мне казалось, что однотипных заданий слишком много, а лекции не способствуют скорому продвижению. В общем я бросил курсы на 8 уровне (из 40). И стал изучать Java самостоятельно по книгам. И как же я удивился, когда через три недели учебы по книге стал изучать многопоточность. На упомянутых курсах это был если не ошибаюсь 20 уровень.

Позже я пришел к выводу, что большинство курсов сделаны таким образом, чтобы человек дольше учился - дольше платил. Поэтому там слишком глубокая проработка основных вещей и много второстепенных, без которых на первом этапе можно смело обойтись.

1.2 Что будет в этой книге.

Все, что будет в этой книге подчинено только одной цели - как можно быстрее сделать из новичка программиста уровня “джуниор”, чтобы начинать искать работу. Это книга - гайд как учить Java, чтобы стать джуниор-разработчиком.

1.3. Чего НЕ будет в этой книге.

Здесь не будет никакой воды, рассуждений и избыточной информации.

Здесь не будет глубокого изучения какого-либо инструмента\технологии.

Эта книга не заменит Вам книги по Java. Объясняю еще раз: это книга-ментор, которая будет вести Вас и направлять, поясняя, что и в какой степени изучать в Java и не только.

При этом, конечно, всегда нужно иметь ввиду, что у каждого человека разная скорость освоения материала. Не могу гарантировать что у Вас получится быстрее в два раза чем у меня, но точно уверен, что если идти этим путем - можно стать Java-разработчиком. По крайней мере я им стал.

1.4. Для кого эта книга

Эта книга для тех, кто решил попробовать себя в программировании, но пока не знает какой язык выбрать и с чего

начать, равно как и для тех, кто уже начал изучать Java, но не знает куда двигаться. Когда я искал для себя мотивацию и инструменты для изучения Java, где-то в интернетах наткнулся на комментарий человека, который учился на курсах и закончив их попытался написать проект, а в итоге как он сам выразился «получился набор классов и что с этим делать непонятно».

Со мной такого не случилось и надеюсь, если Вы будете придерживаться указанных рекомендаций, не случится и с Вами.

1.5. Начальные требования

Для того, чтобы начать без боли погружаться в программирование нужны 2 вещи:

1. Уметь печатать неглядя, десятью пальцами. Если у Вас нет этого навыка найдите в интернете онлайн тренажер и за две-три недели каждодневных тренировок у Вас все получится. Конечно, можно и без этого, но будет сложновато.

2. Владение английским на уровне хотя бы чтения обычных текстов. Если этого нет можно, конечно, продолжать, но учить английский все равно придется и без знания языка будет сложнее понимать термины, аббревиатуры и т.д. Не говоря о том, что часто нужно будет читать техническую литературу, поскольку все что на русском, как правило уже не актуально т.к. перевод занимает не менее года после выхода оригинала.

Также следует иметь ввиду, что учиться придется постоянно. Не получится выучить язык программирования и просто писать код. Язык весьма обширен и досконально выучить его нереально, к тому же он постоянно развивается, выходят новые версии. Кроме языка

нужно будет изучать еще много инструментов (приложений). Так что работа в ИТ - это постоянное обучение, особенно для программиста.

Глава 2. Выбор языка программирования

Языков программирования очень много и у каждого языка есть своя область применения. Какой-то общей классификации не существует, тем не менее один из разграничивающих параметров - это уровень языка.

Уровень языка - это степень, в которой семантика языка учитывает особенности мышления человека, нежели машины — то есть уровень языка тем «ниже», чем он «ближе к машине», и тем «выше», чем он «ближе к человеку».

Когда я только думал о том, чтобы стать программистом я не имел никакого представления о языках программирования. Слышал только, что одни языки учить легче, другие - труднее. Вот соответственно чем ниже уровень языка, тем учить его труднее, но тем быстрее в среднем будет работать написанная на нем программа и меньше расходовать системные ресурсы.

Еще тут нужно учитывать, что не всегда этот параметр достаточно объективен: например, если говорить об одном из самых низкоуровневых языков программирования - Ассемблере, то он действительно очень сложен, судите сами.

Это листинг программы, которая выводит на монитор сообщение "Hello, world!".

```
-----Hello, world!---- Assembler:  
  
SECTION .data
```

```
msg    db "Hello, world!",0xa
```

```
len    equ $ - msg
```

```
SECTION .text
```

```
global _start
```

```
_start:
```

```
mov eax, 4
```

```
mov ebx, 1
```

```
mov ecx, msg
```

```
mov edx, len
```

```
int 0x80
```

```
mov eax, 1
```

```
mov ebx, 0
```

```
int 0x80
```

А вот аналогичная программа на Java:

-----Hello, world!---- Java:

```
public static void main(String[] args) {  
    System.out.println("Hello world!");  
}
```

C# приводить не буду - синтаксис похож на Java и области применения сходны: это и веб разработка и десктоп приложения. Тем не менее на рынке труда Java более востребована. Вот результат запроса на одном из самых известных поисковиков вакансий:

10865 вакансий «java»

6153 вакансии «C#»

Теперь таже самая программа на Python:

-----Hello, world!---- Python:

```
print("Hello world!")
```

Всего 1 строка.

Однако Питон более высокоуровневый чем Java, что обеспечивает более низкий порог вхождения, но более узкую область применения:

- относительно недавно стал применяться в веб-разработке.
- Используется в data science: машинное обучение, анализ данных и визуализация.

Ну и конечно программа написанная на Python в большинстве случаев будет выполняться дольше, чем написанная на Java.

А вот, к чему я все это вел, такая же программа на языке GO, который позиционируется как высокоуровневый (как Python), но значительно быстрее:

-----Hello, world!---- Go:

```
import "fmt"

func main() {
    fmt.Println("Hello world!")
}
```

Я рассматривал GO как кандидата но, когда открыл что-то посложнее понял, что Java читается легче.

Вы, конечно, можете составить свое мнение и посмотреть примеры программ в интернете. На такой маленькой программе сложно показать особенности языков, но на более длинном листинге лично для меня GO совсем не прост.

Кто-то может спросить почему я не упомянул JavaScript. Могу ответить одно - мне не понравился его синтаксис. И насколько я понимаю у него более узкая сфера применения чем у Java. Все это конечно вкусовщина - каждому свое. Посмотрите примеры на этом языке, попробуйте, возможно - это Ваш язык. И конечно сначала посмотрите насколько язык, который Вы выбираете, востребован на рынке труда.

2.1. Плюсы Java:

Несмотря на растущую популярность Go и Python, Java остается одним из самых востребованных языков уже более десяти лет.

У Java возможно самая широкая сфера применения. Java используется для:

- веб-разработки - на нем написаны сайты такие как Амазон, РЖД, Одноклассники и т.д;
- мобильной разработки - Java основным языком разработки на Android;
- десктопной разработки - даже есть реализации графических пользовательских интерфейсов операционных систем Linux, Solaris;
- создания серверной логики — бэкэнд большинства крупных сайтов, порталов, магазинов и т. д. написан на Java;
- создания распределенных систем - например Hadoop: набор утилит, библиотек и фреймворк для разработки и выполнения распределённых программ, работающих на кластерах из сотен и тысяч узлов.
- создания баз данных (база H2 написана на Java);

- переносимость. Java можно запустить даже на роботе-пылесосе, если туда установить JVM - виртуальную машину джава в которой исполняется код. Поэтому операционная система совершенно не важна.

2.2. Минусы Java.

Конечно, есть и минусы. Из наиболее значимых:

Java создавался как замена C++ поэтому код похож, но более "громоздкий": один из минусов Java - это многословность, что делает его более читабельным, но загромождает код.

Еще к минусам относят медленность программ, однако это только в сравнении с еще более низкоуровневыми языками такими как C и C++. Но мы помним обратную сторону более «быстрых» языков программирования - сложность изучения. Кроме того, на рынке эти языки востребованы значительно меньше. Они очень "нишевые" на них в основном пишутся операционные системы, системные и десктопные приложения и т.д.

Так что в сравнении с большинством востребованных на рынке труда языков программирования язык Java:

1. В целом быстрее;
2. По востребованности превосходит всех за исключением Python;
3. По широте применения ему практически нет равных.

Глава 3. Среда разработки

В этой главе мы поговорим об IDE — Integrated Development Environment - среде разработки. Вы часто будете встречать такое сокращение - IDE.

Что такое среда разработки (IDE) - это приложение, которое включает в себя:

Редактор кода.

Компилятор.

Сборщик.

Отладчик.

Это рабочий инструмент программиста. В некоторых источниках Вы можете встретить мнение, что начинать нужно программируя в блокноте. Вероятно, это правильно, если Вам 13 лет и до трудоустройства еще далеко. Но раз Вы читаете эту книгу - значит, время ограничено.

Учитесь сразу работать в IDE - это значительно ускоряет обучение, так как там есть подсветка синтаксиса (ключевые слова выделяются цветом), есть подсказки - среда Вам подскажет варианты завершения кода и конечно сразу подсветит синтаксические ошибки.

Сред разработки существует великое множество - примерно около трех десятков. На данный момент лидируют три:

1. IntelliJ IDEA

Де факто - стандарт индустрии. В большинстве компаний используют именно ее. Существует куча плагинов на все нужды и потрясающая документация. Есть платная и свободная версии. Далее в этом разделе мы установим Idea и рекомендую пользоваться именно ей. В большей части обучающего видео Вы увидите именно эту среду разработки.

2. Eclipse

Тоже очень часто используется разработчиками, и во многих видео Вы можете увидеть Эклипс. Бесплатна.

3. NetBeans

Бесплатна. Менее распространена чем предыдущие.

Конечно это “в среднем по больнице” - наверняка где-то есть разработчики которые используют другие IDE.

3.1. Установка IDE IntelliJ Idea.

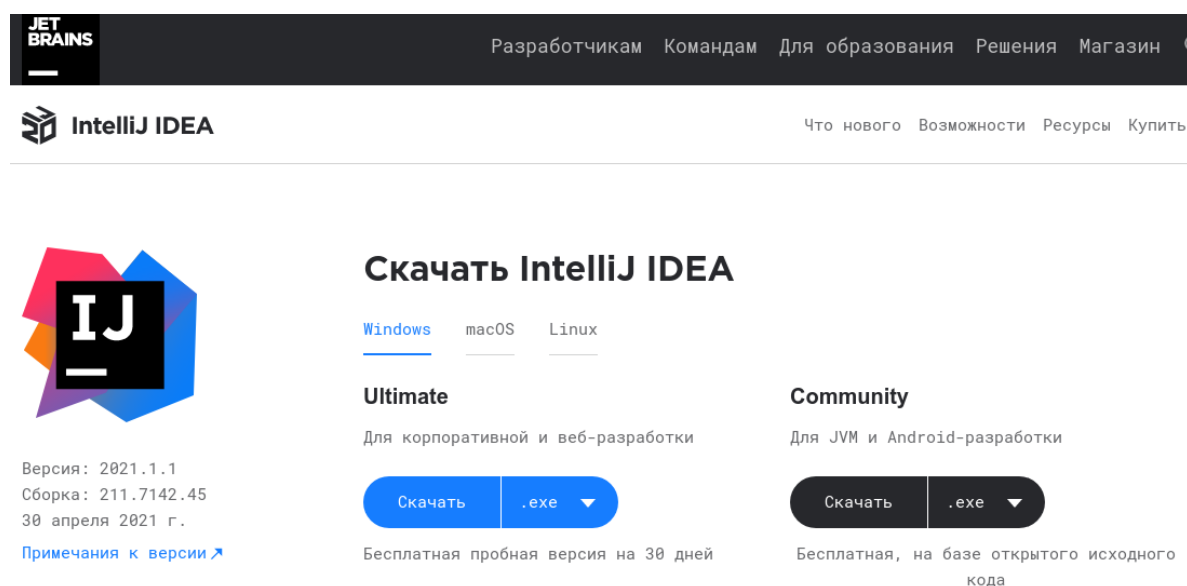
Несмотря на то, что установка достаточно тривиальна опишу процесс полностью, возможно для кого-то установка программ на ПК не повседневное занятие.

3.1.1. Скачиваем дистрибутив.

Идем на сайт IntelliJ Idea:

<https://www.jetbrains.com/ru-ru/idea/download/#section=windows>

3.1.2. Выбираем бесплатную Community версию (черная кнопка на картинке). Ее пока достаточно. Запускаем скачанный файл.



JET BRAINS

Разработчикам Командам Для образования Решения Магазин

IntelliJ IDEA

Что нового Возможности Ресурсы Купить

Скачать IntelliJ IDEA

Windows macOS Linux

Ultimate
Для корпоративной и веб-разработки

Скачать .exe

Бесплатная пробная версия на 30 дней

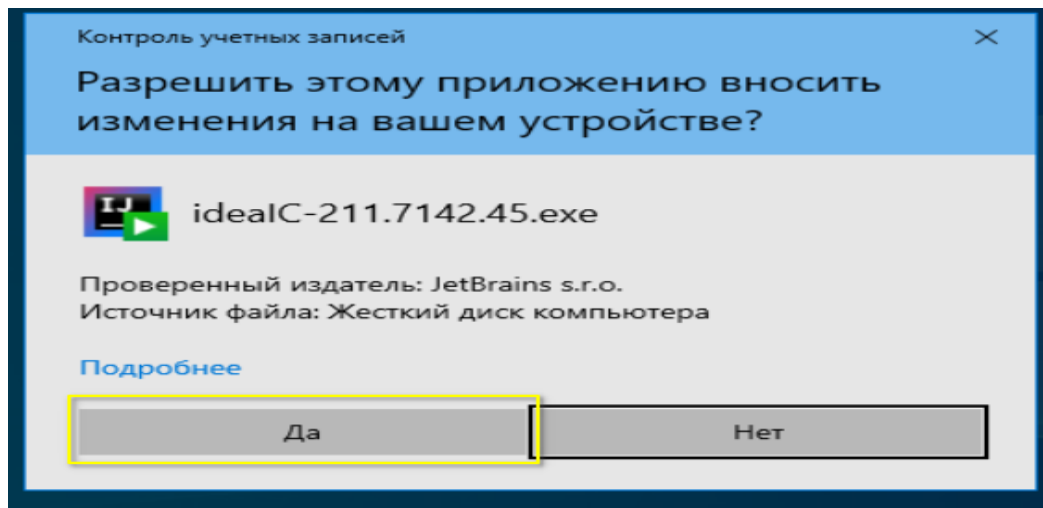
Community
Для JVM и Android-разработки

Скачать .exe

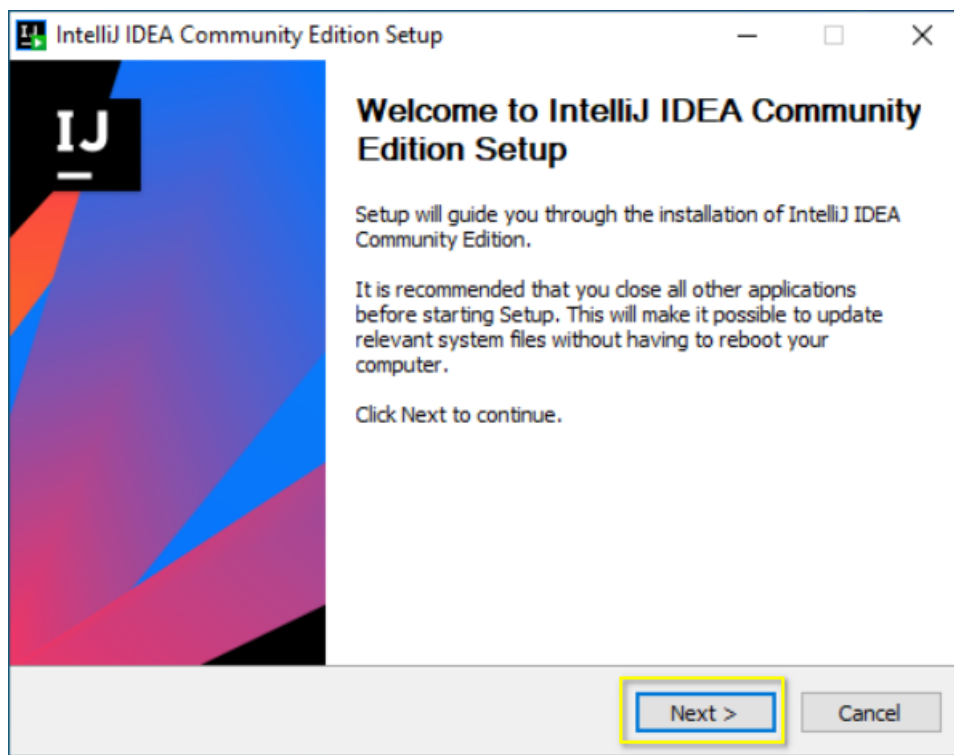
Бесплатная, на базе открытого исходного кода

Версия: 2021.1.1
Сборка: 211.7142.45
30 апреля 2021 г.
[Примечания к версии](#)

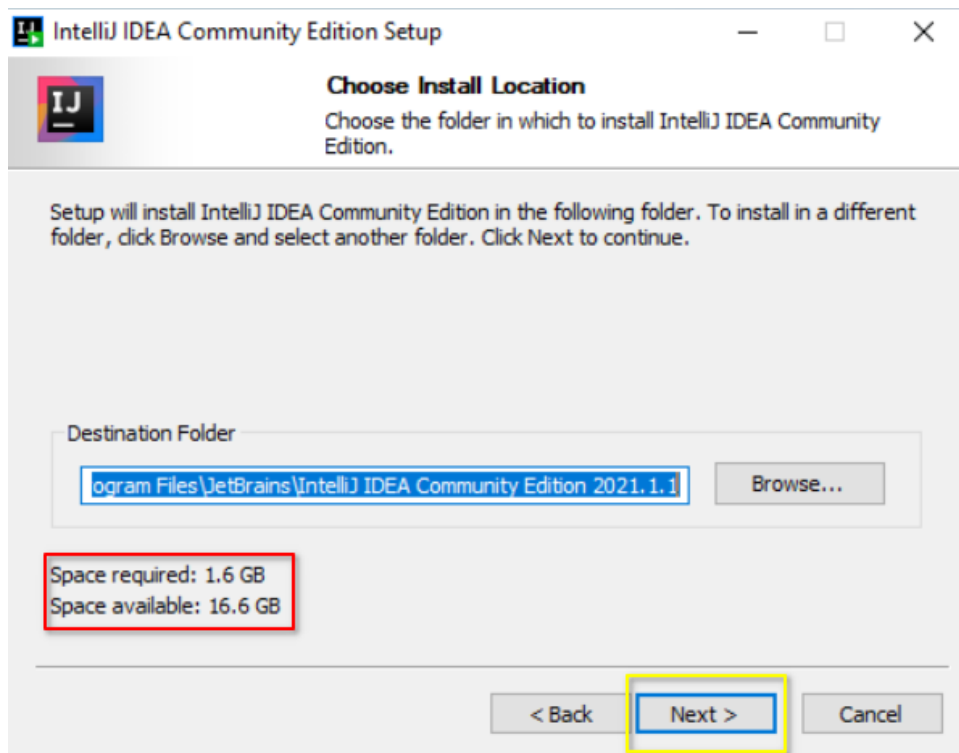
3.1.3 Нажимаем кнопку “ДА”.



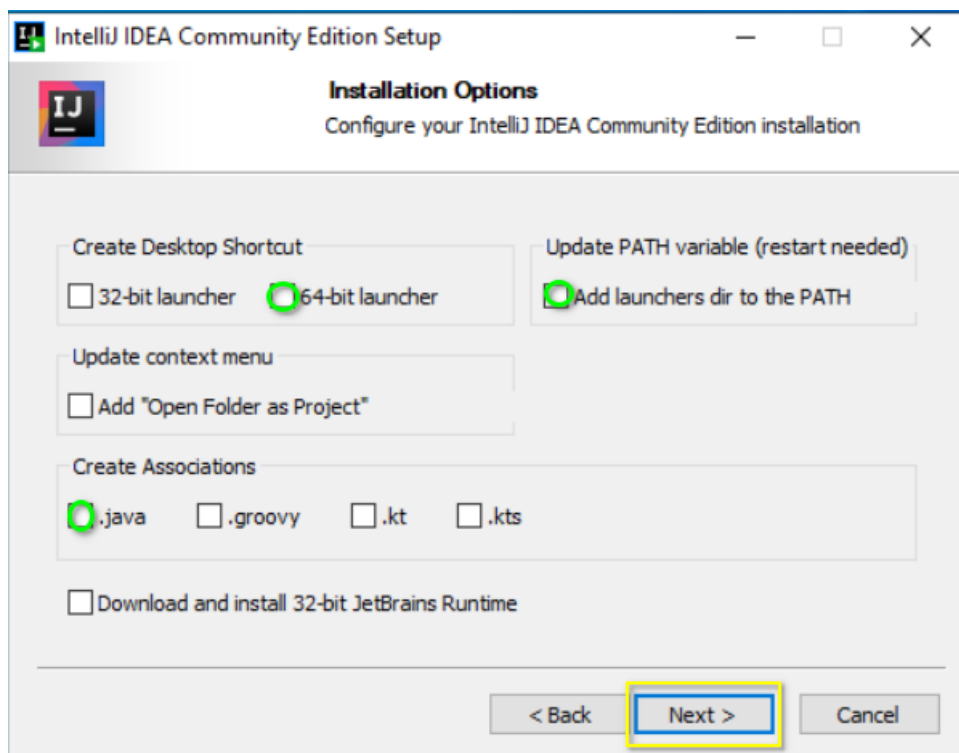
3.1.4. Нажимаем кнопку "Next".



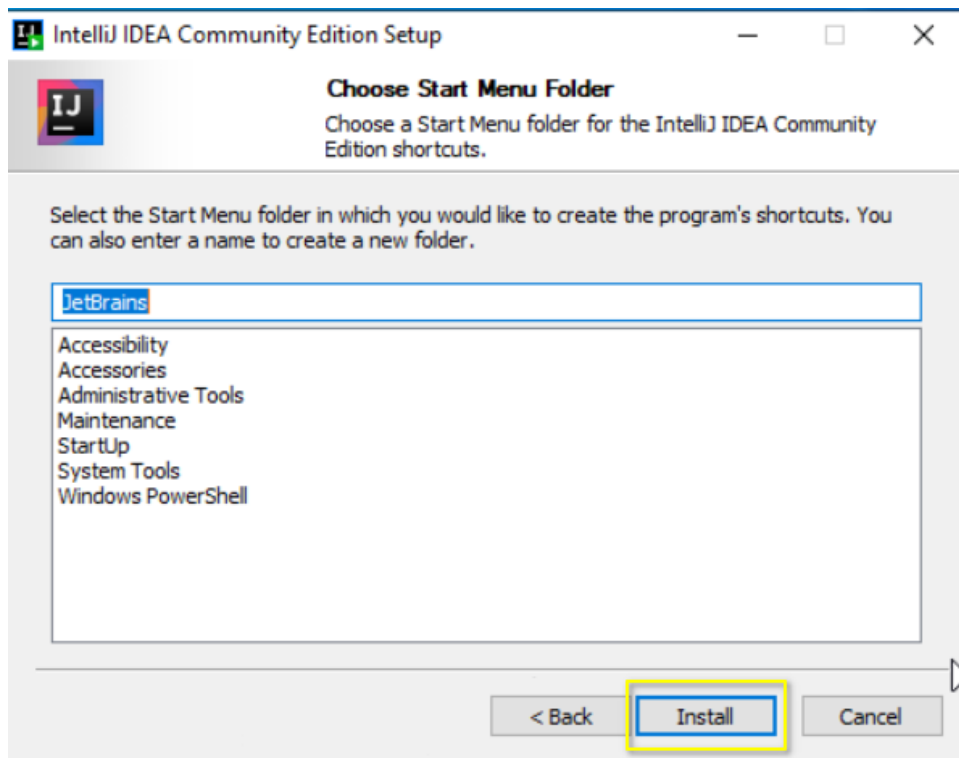
3.1.5. Здесь перед тем как нажать "Next" убедитесь, что верхняя цифра (на картинке выделено красным прямоугольником) меньше нижней - то есть на диске достаточно места. Жмем "Next".



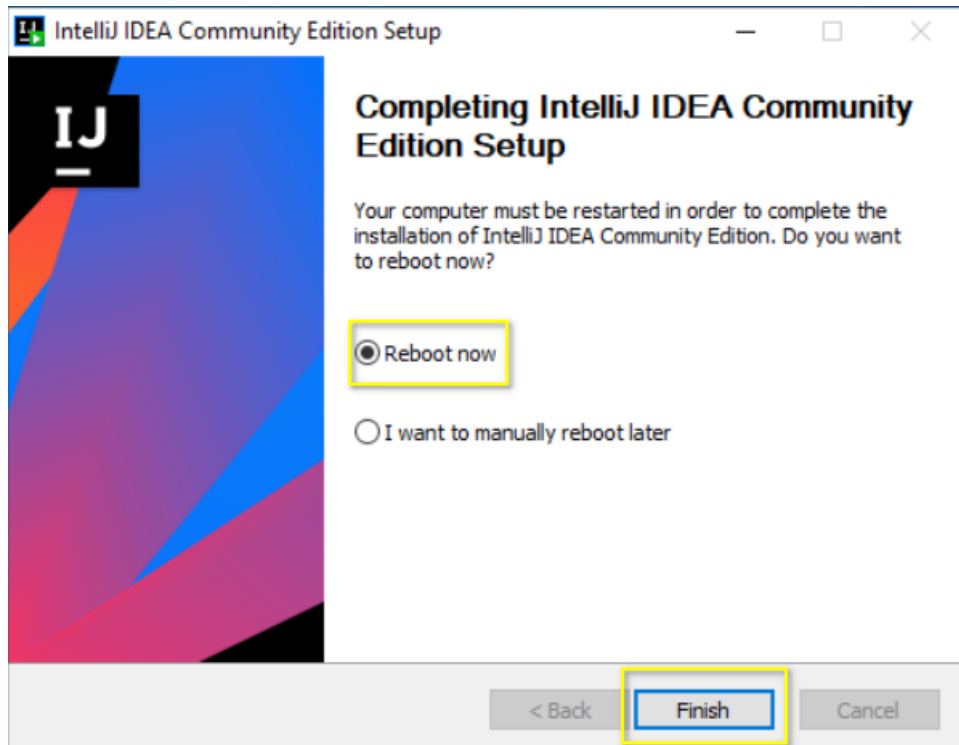
3.1.6. Проставляем галочки там, где на картинке зеленые кружки и ждем “Next”.



3.1.7. На этом этапе просто ждем “Install” и начнется установка IDE.



3.1.8. Установка завершена. Ставим отметку "Reboot now" и жмем "Finish". Компьютер будет перезагружен.



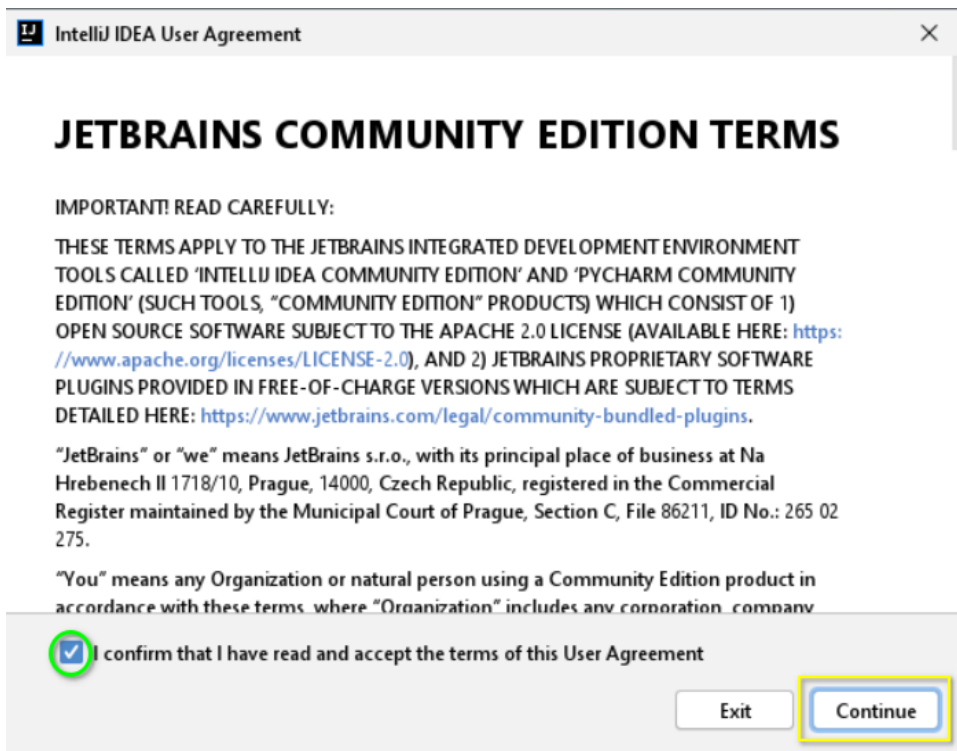
На рабочем столе появится иконка:



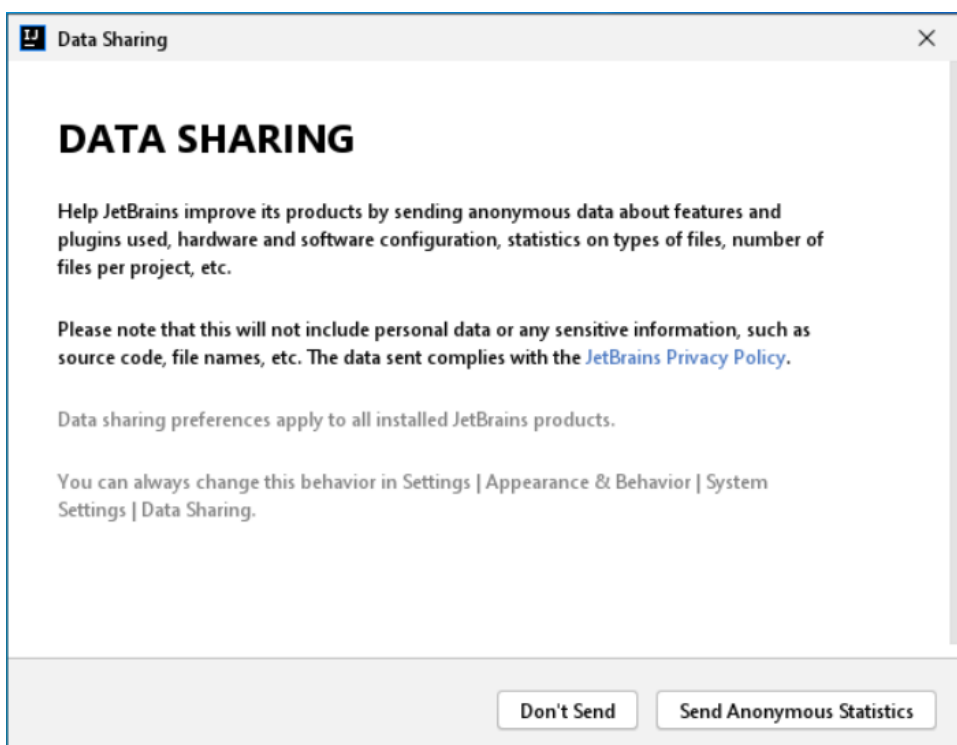
3.2. Настройка первого проекта

3.2.1. Давайте запустим IDE. (Кликаем по иконке приложения).

3.2.2. Ставим галочку, что мы согласны с лицензионным соглашением и жмем "Continue":



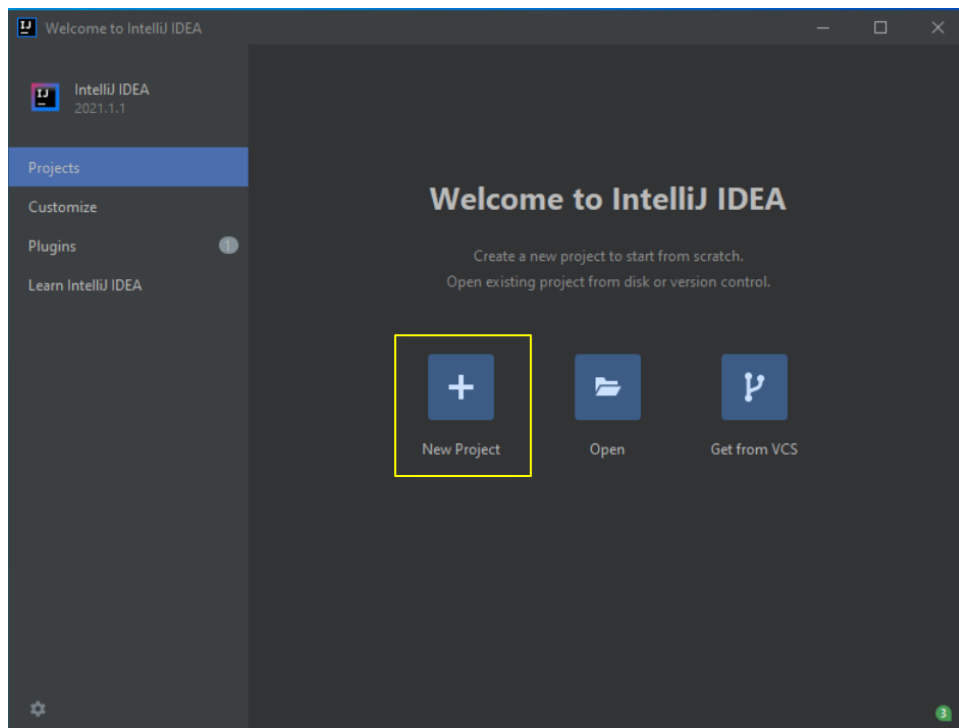
3.2.3. На этом экране нас спрашивают согласны ли мы отсылать анонимную статистику, чтобы помочь делать продукт лучше. Выбор ответа на Ваше усмотрение.



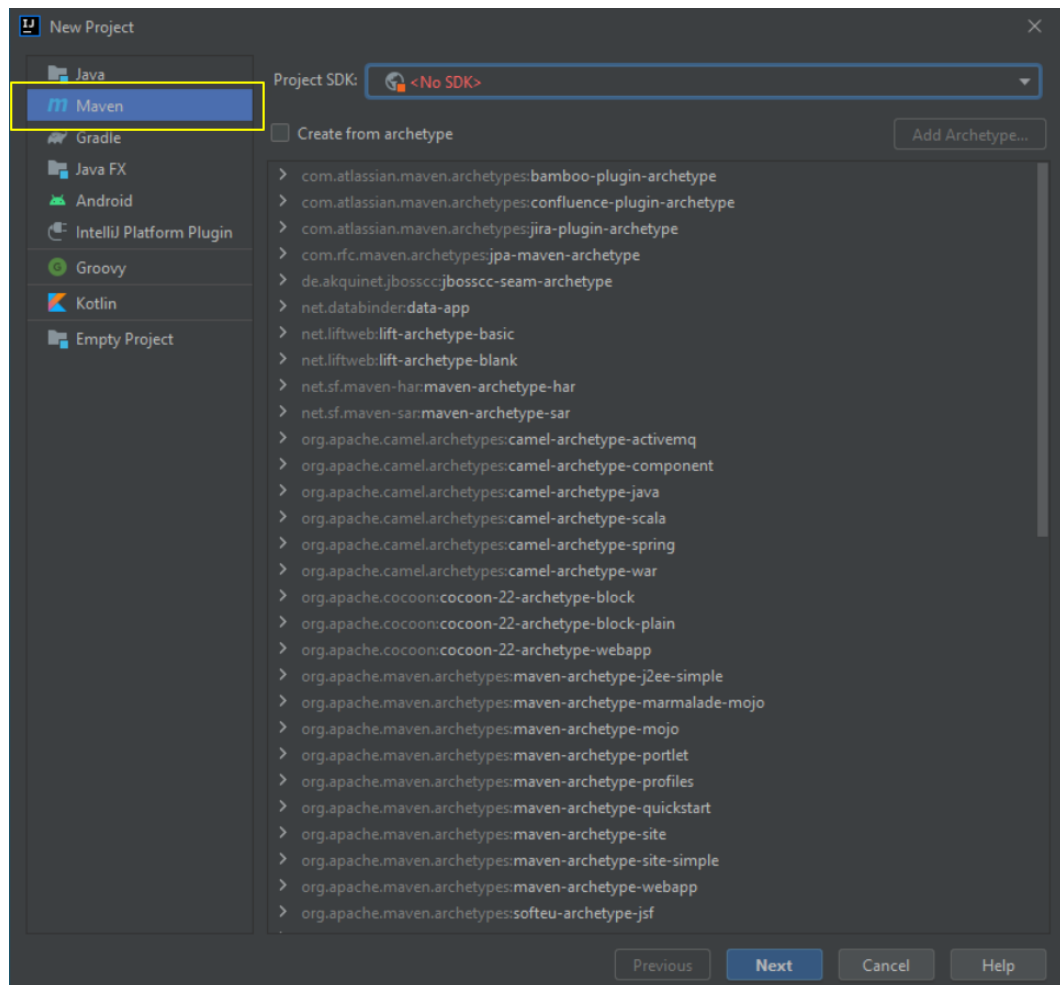
3.2.4. Этот экран говорит о том, что IDE IntelliJ Idea загружается.



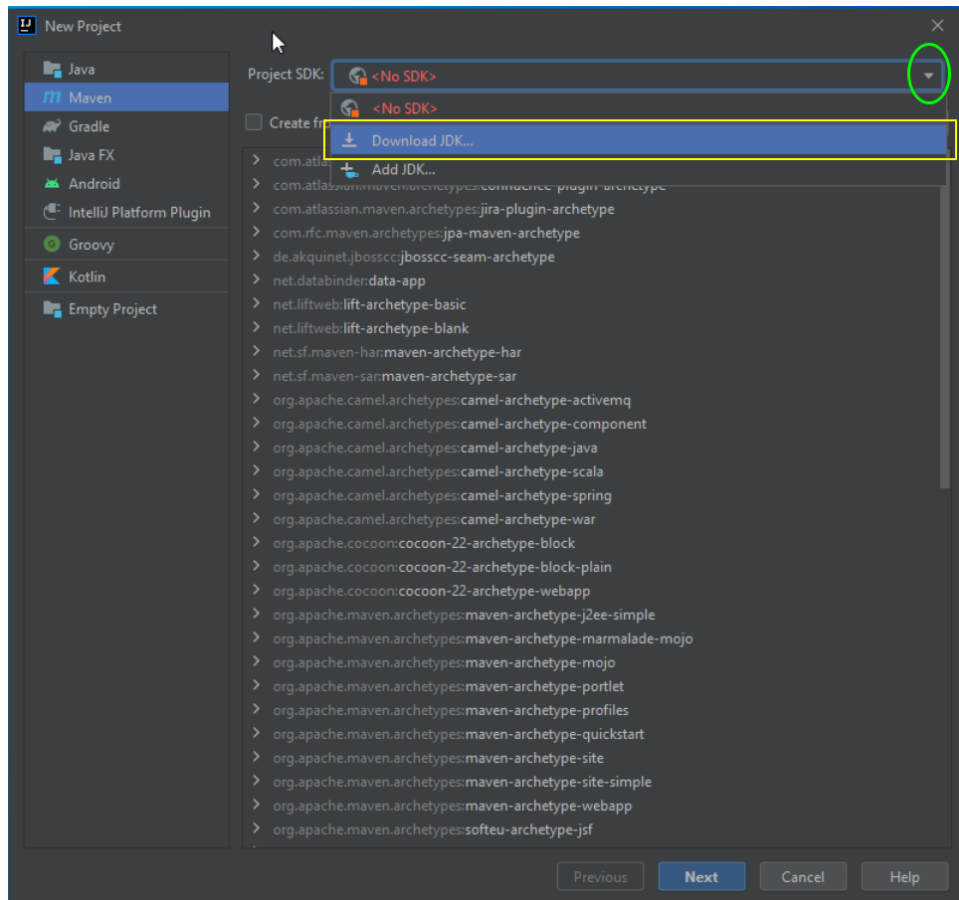
3.2.5. Выбираем New Project - создаем новый проект.



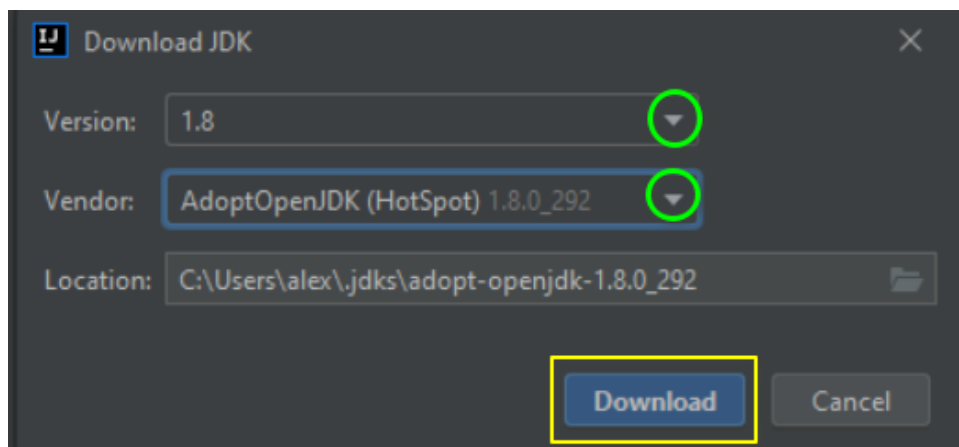
3.2.6. Выбираем Maven и ждем когда он подтянет архетипы - Вам это подскажет курсор: он перестанет крутиться, а в средней части приложения появится список архетипов как на картинке внизу:



3.2.7. Теперь нам нужно установить JDK - это и есть сама Java. Кликните мышкой там, где на картинке зеленый овал и в выпавшем меню выберите Download JDK...



3.2.8. Далее в выпадающих списках (жмите галочки чтобы они раскрывались) нужно выбрать:



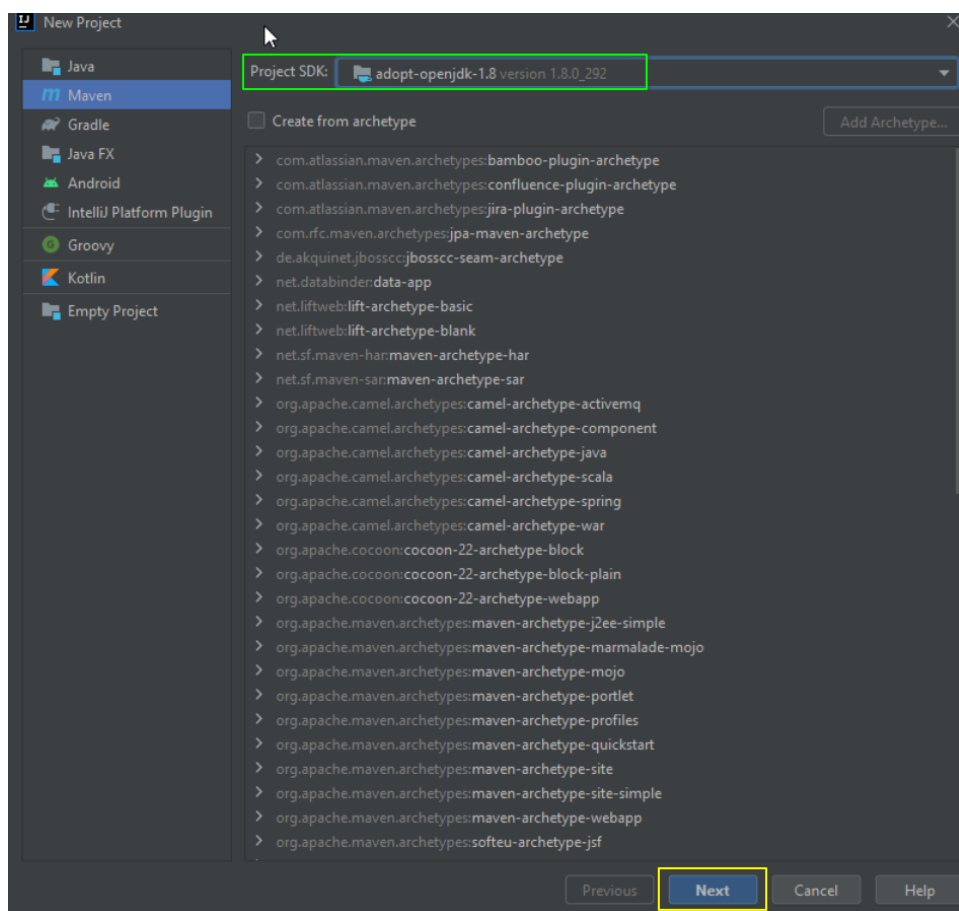
Версия языка 1.8 (О версиях JDK я расскажу дальше, пока установим эту)

Производитель: AdoptOpenJDK(HotSpot)

Все как на картинке. Локацию менять не нужно. Жмем Download и начнется загрузка дистрибутива Java.

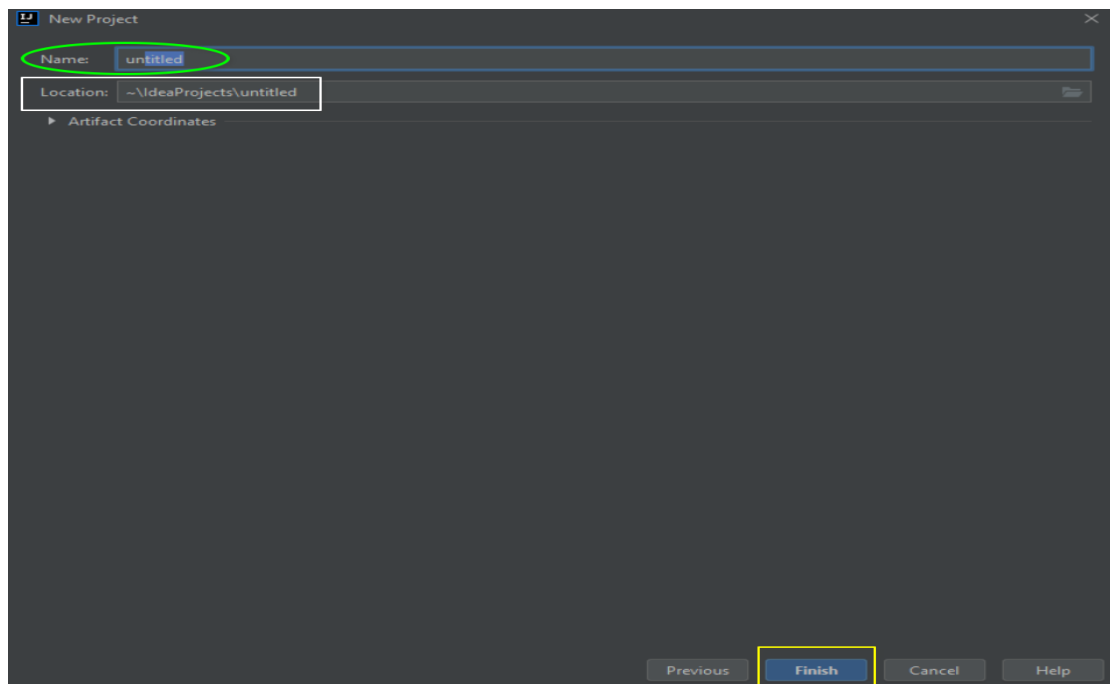
Кстати, установив одну версию JDK Вы можете затем установить и другую и даже все сразу, что доступны. В каждом проекте можно будет переключать версии. Но пока достаточно одной - чтобы не путаться.

3.2.9. Итак, видим, что JDK 1.8 у нас установился - жмем "Next".

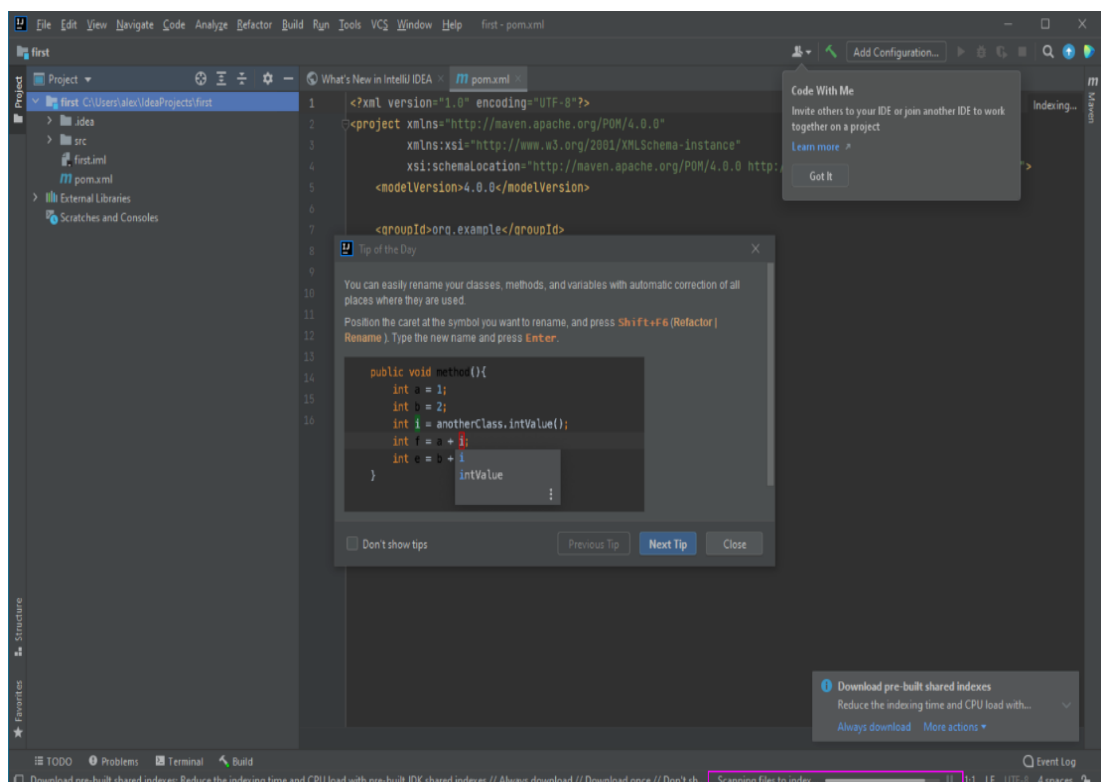


3.2.10. Теперь нужно задать имя проекта - там где зеленый эллипс. Например first. Используйте только латиницу! В меню Location указано будущее расположение проекта - ничего менять не

нужно. Жмем Finish. И ждем, когда среда разработки сформирует проект.



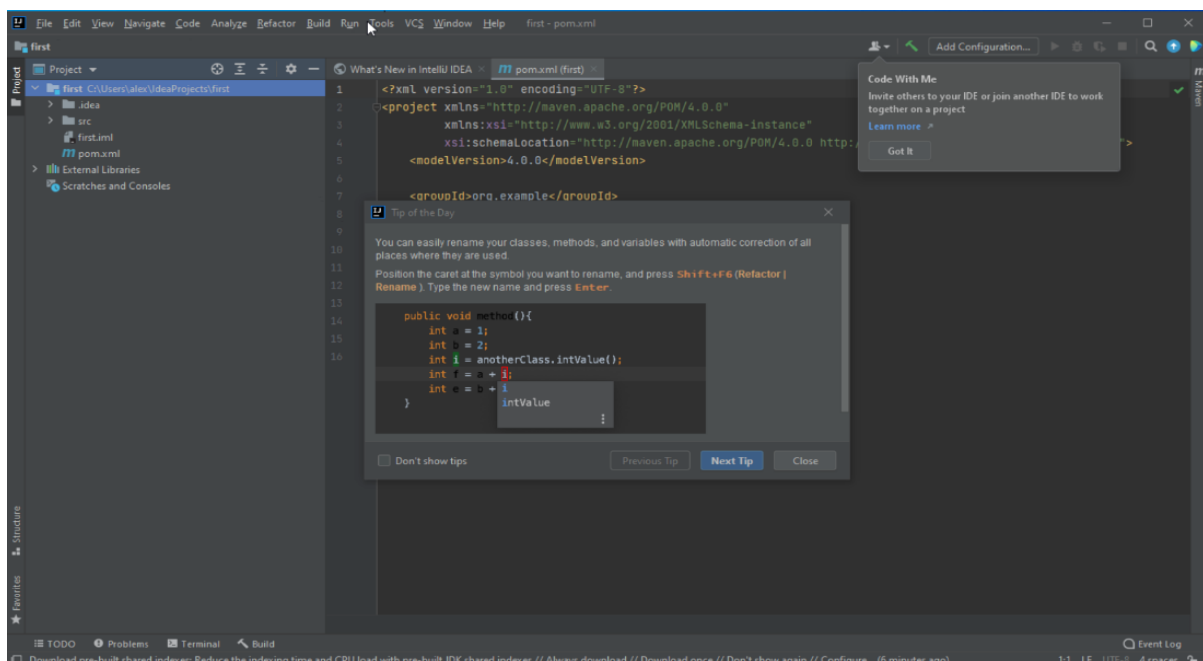
3.2.11. Обратите внимание: внизу (выделено фиолетовым прямоугольником) есть статус бар:



На картинке сейчас написано Scanning files to index... - идет индексация файлов. Затем будет индексация JDK, затем будут подтянуты и установлены плагины, обновления и т.д. Нужно

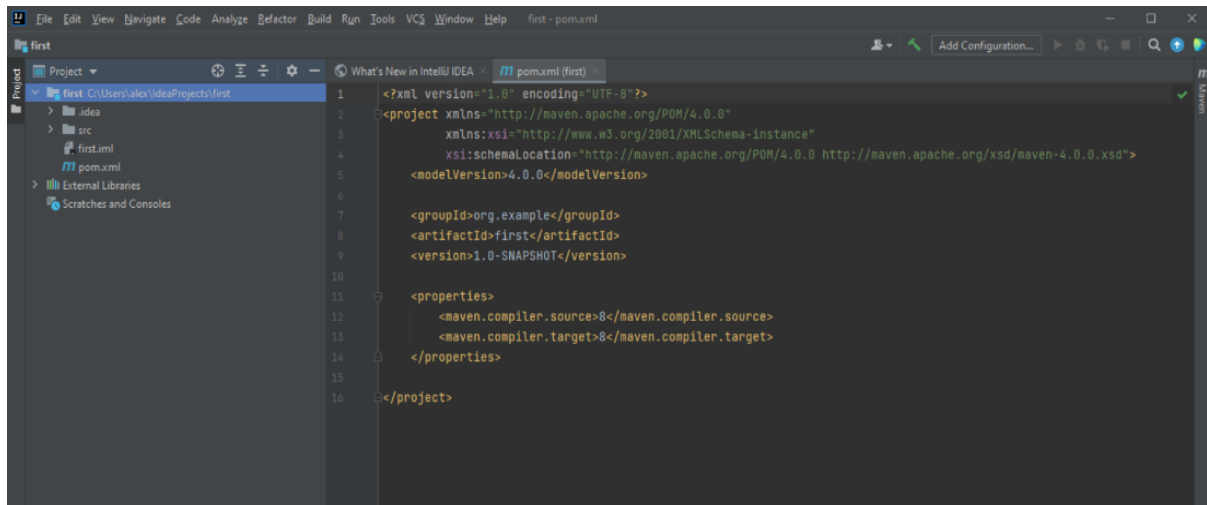
дождаться завершения. Даже когда Вы заново откроете проект IDE как правило будет снова индексировать файлы. Обращайте внимание на эту область - если там появился статус выполнения какой-либо операции - лучше дождаться завершения и только затем начинать работу. Я первое время не замечал этих процессов и сильно раздражался почему вдруг IDE тормозит, а то и не отвечает.

3.2.12. Здесь мы видим - что статус бара внизу нет, значит все okay, можно начинать работу. Верхнее окошко говорит нам что появилась новая опция совместной работы в IDE просто жмем "Got It". Среднее окошко выдает нам совет дня - при каждом запуске IDE будет выдаваться новый совет. Если Вы этого не хотите поставьте галочку "Don't show tips". В любом случае жмем "Close".



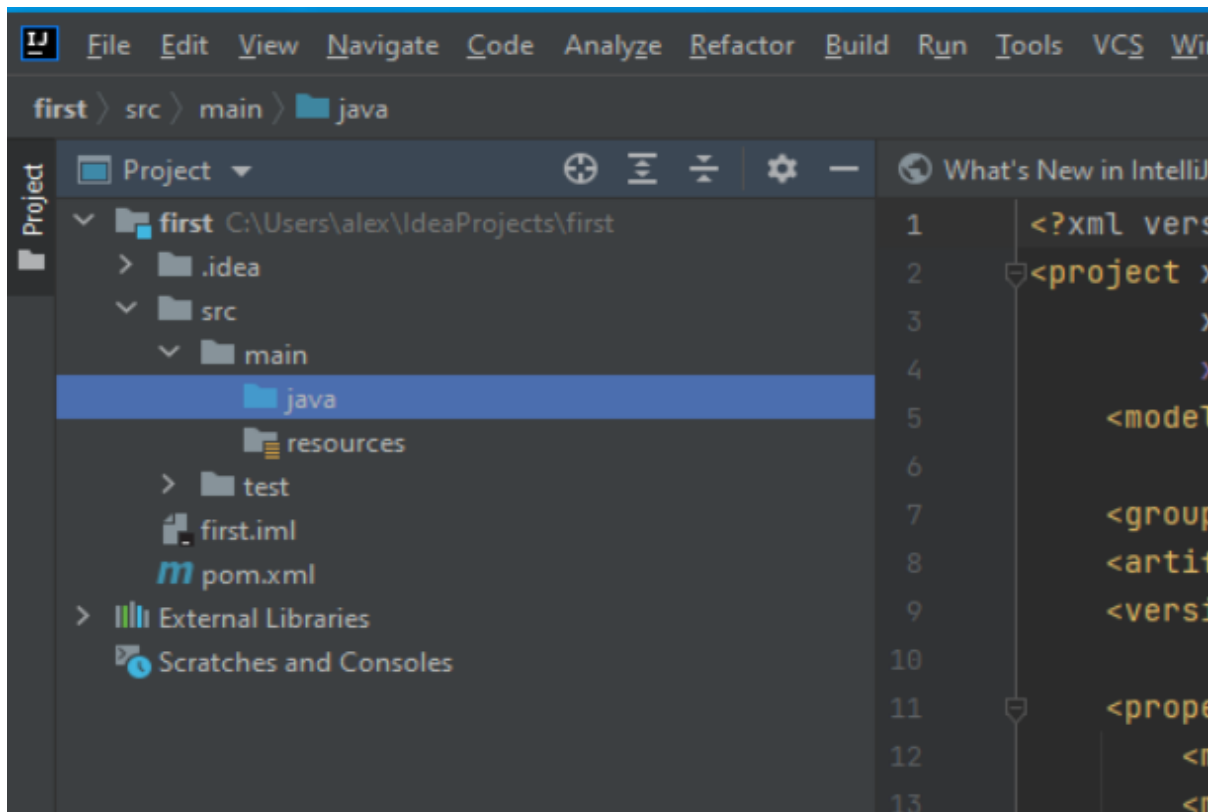
3.2.13. Итак мы позакрывали все окна и вот видим наш проект. В редакторе открыт файл pom.xml - это главный конфигурационный файл проекта. Пока просто запомните это. Слева, где синяя полоска

(на самом деле это курсор), структура проекта - это файлы и директории, которые физически находятся на диске.

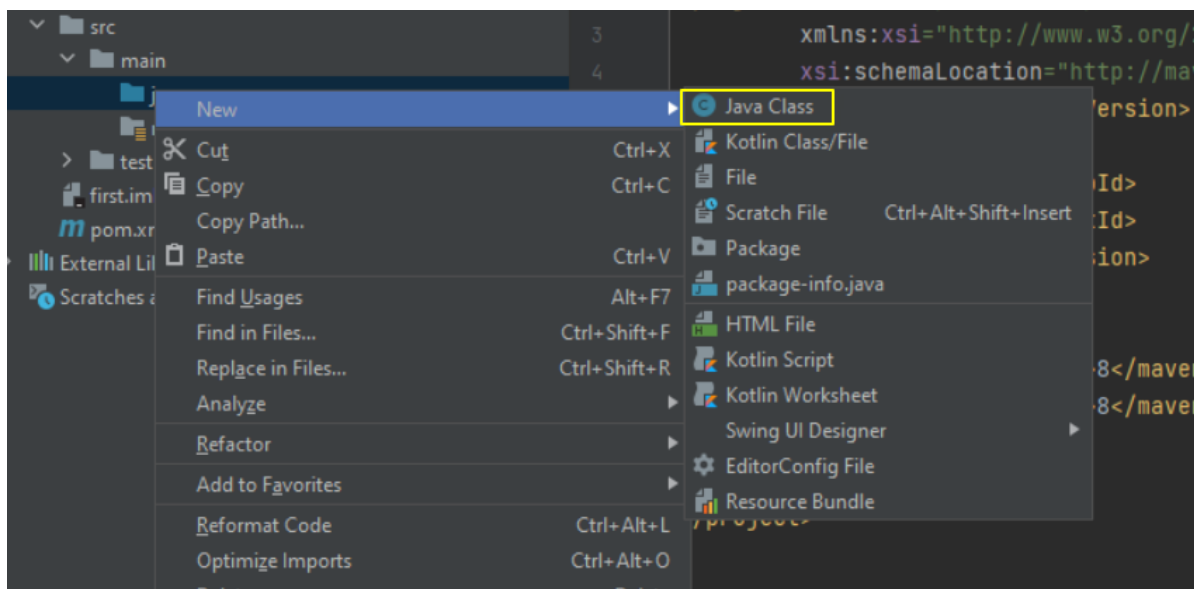


3.3. Пишем первое приложение.

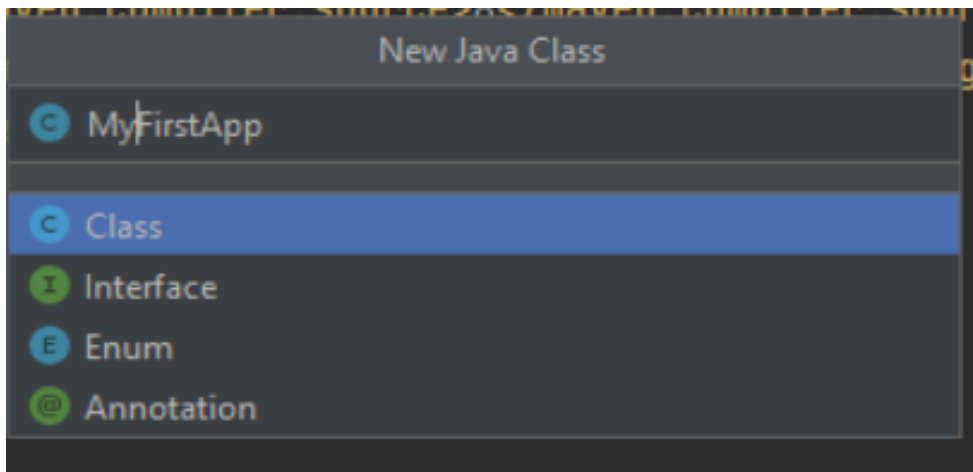
3.3.1. Давайте напишем первое наше приложение. Нажмите на галочки напротив директорий `src`, `main` и поставьте курсор (синюю полосу) на директорию `java`. Вот так:



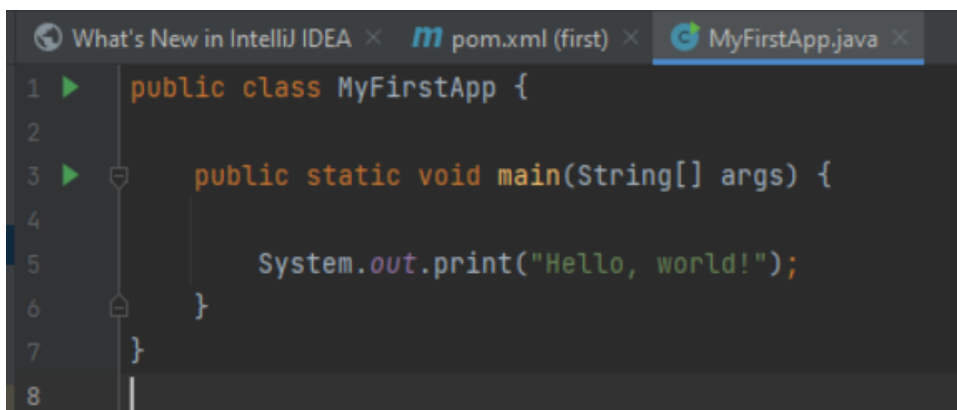
3.3.2. Далее нажмите правой кнопкой мыши на директории java и выберите New > Java Class:



3.3.3. Далее введем имя класса: MyFirstApp

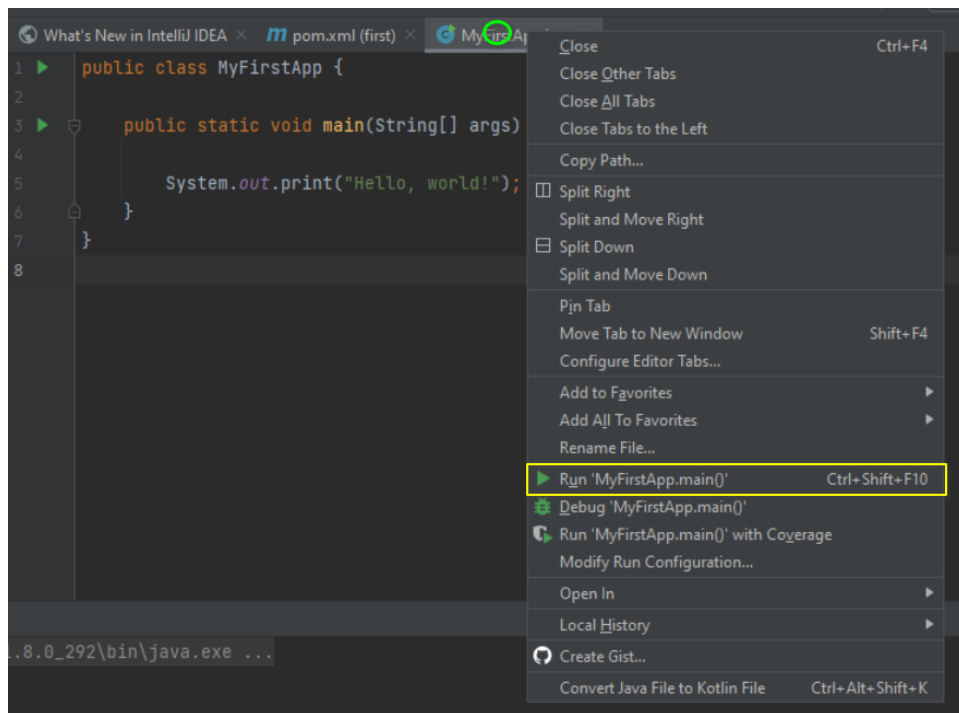


3.3.4. Жмем Enter и у нас слева в директории Java появляется файл MyFirstApp, а в редакторе этот файл автоматически откроется. Напишем в этом файле первое наше приложение:

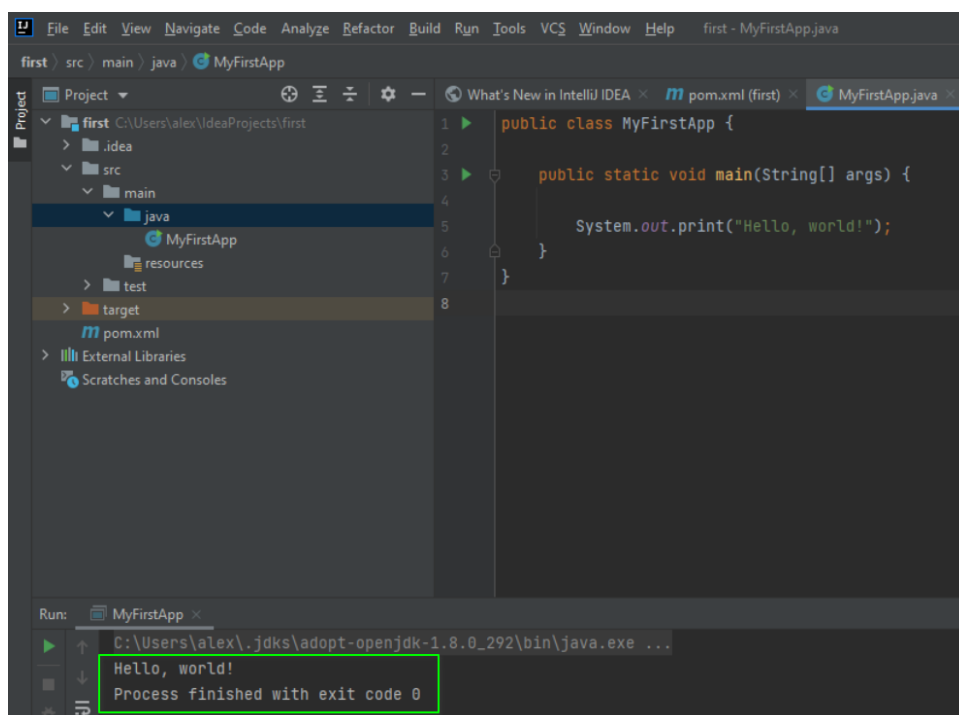


Не огорчайтесь, если пока ничего не понятно - все придет со временем. Пока просто повторяйте то, что видите.

3.3.5. Запускаем наше приложение. Для этого правой кнопкой мыши нажимаем на таб открытого файла (там, где на картинке зеленый кружок) и выбираем **Run MyFirstApp.main()** тут уже кликаем левой кнопкой:

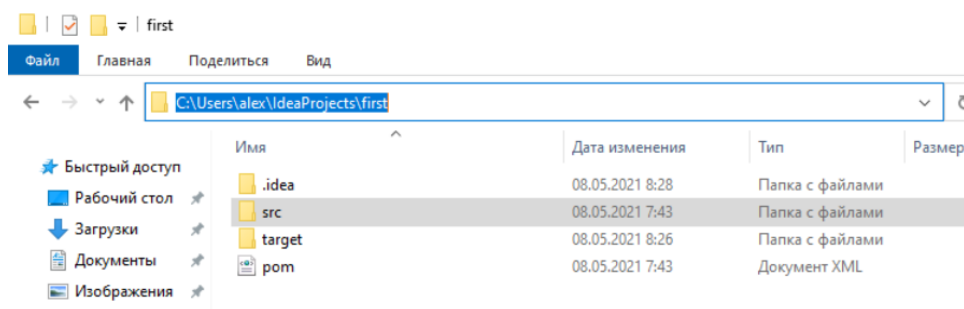


3.3.6. Справа внизу Вы увидите, как появится статус бар (как на шаге 3.2.11.) - IDE собирает и запускает приложение. Когда приложение запустится внизу откроется консоль вывода, в которой Вы увидите результат выполнения Вашего приложения:



На картинке зеленым прямоугольником обозначен вывод приложения. Мы вывели строку "Hello, world!". Следующая строка - служебное сообщение, которое говорит о том, что процесс завершен успешно - код 0. В случае неуспеха там будет сообщение об ошибке и код 1.

3.3.7. Все Ваши проекты будут храниться в Вашей домашней директории в каталоге IdeaProjects. Давайте убедимся в этом - зайдите в свою домашнюю директорию, перейдите в каталог IdeaProjects и там Вы увидите папку first - это и есть проект что мы создали. Внутри обычные файлы, которые можно открывать блокнотом:



Теперь Вы готовы к следующему шагу - приступить к изучению Java!

Глава 4. Подружитесь с приложениями.

Если Вы с информационными ресурсами (сайтами, программами) «на вы», нужно этот момент исправить. Дело в том, что Вам в работе часто будет нужно устанавливать какие-то программы и в этом никто помогать не будет. Хороший программист должен уметь установить на свой ПК любую операционную систему

и настроить ее под себя. Установить какую-нибудь систему заявок и настроить в ней порядок обработки этих заявок. Установить систему управления базами данных на свой компьютер, создать базу и таблицы, наполнить базу данными.

Все это, я пишу для того, чтобы Вы понимали: если, Вам сложно кликать по сайтам, не можете найти как зарегистрироваться на сайте, как изменить настройки любой программы на компьютере, не можете и не любите работать с почтой и тому подобное Вам придется это исправить. Иначе в “айти” не войти. И у программиста, хоть он и является высокооплачиваемым специалистом, слуг нет - разбираться с программами и сайтами ему приходится самостоятельно.

Если вдруг Вы осознали, что для Вас это проблема, сделайте следующее:

1. Заведите аккаунт в соцсетях. Желательно в нескольких (Вконтакте, Facebook и т.д.) Максимально заполните профили, найдите знакомых, вступите в переписку, если хотите. В общем изучите полностью функционал. И удалите свои аккаунты: вам теперь некогда. Ну или хотя бы не висите там чаще чем раз в неделю.

2. Обязательно изучите Телеграмм и Watsapp - все функции, поменяйте статусы, попереписывайтесь с семьей или с другом, смените фон и прочие настройки. Эти два инструмента Вам точно понадобятся. Несмотря на то, что во многих компаниях есть свои мессенджеры, всегда начальство предпочитает эти инструменты причем тут разделение такое: если Ваш босс будет из европейской части России (от Белгорода до Челябинска, возможно Омска) то скорее всего это будет Телеграмм, от Новосибирска до Японского моря - Watsapp. Ну конечно возможны исключения.

3. Для Вас не должно быть проблемы создать любой файл в Windows, сохранить его в любое другое место, переименовать, удалить. Заpackовать файлы в архив *.zip, *.rar и распаковать их из архива в любую директорию.

4. Обязательно минимально разобраться в Word и Excel, рабочая документация будет в этих форматах.

Что нужно уметь:

- редактировать файлы этих форматов,
- вставлять, удалять строки,
- сохранить файл с другим именем,
- сохранить в формате pdf если Вы не хотите это все распечатывать-подписывать-сканировать как требуют кадровые работники которые будут присылать Вам документы на подпись.

5. Обязательно нужно разобраться в программах SKYPE и ZOOM. Установите, заведите аккаунты, настройте, пообщайтесь с кем-нибудь при помощи этих инструментов. Разберитесь с настройкой камеры и звука - это очень важно! В период пандемии - это основные инструменты, посредством которых проводят собеседования.

6. Заведите почту на Yandex и/или Google. Создайте дополнительные папки для писем, настройте фильтры, чтобы письма от одного адресата попадали в одну папку, от другого в другую. Разберитесь как удалять эти правила-фильтры. Изучите как все тоже самое делать в MS Outlook. Если у Вас его нет - установите. Вообще фильтрация почты это весьма важно в ИТ, так как почта это основной инструмент деловой переписки. Писем будет поступать достаточно много и, если не настроить фильтрацию, можно что-то пропустить. И достаточно часто приходится искать в почте какие-то

письма с нужной информацией. Так что поиск в Outlook тоже очень важный навык.

Глава 5. Что нужно знать о версиях JDK

Итак, прежде чем Вы начнете изучать Java Вам необходимо ориентироваться в версиях языка. В настоящее время последняя версия - 16. Выпущена буквально несколько месяцев назад.

Обратите внимание я пишу последняя, а не актуальная. Мы с Вами установили версию 1.8 или по новой нумерации просто 8. Дело в том, что после версии 1.4 нотация нумерации сменилась и следующая версия 1.5 стала называться просто 5. Кроме этого сменился релизный цикл - если раньше до 9 версии обновление выходило раз в 3 года, то сейчас выходит каждые полгода, но это не значит, что все сразу перескакивают на новую версию: в разработке в основном пользуются так называемыми LTS версиями - с длительным сроком поддержки.

Итак, какие версии актуальны и какие нужно учить? На данный момент большинство компаний использует версию 8, которую мы установили. Некоторые уже переходят на 11, в сентябре 2021 ожидается выход 17 версии, которая будет LTS. При этом значительное количество старых приложений остается в продуктовой среде и их тоже надо поддерживать - вносить изменения так, что во многих компаниях Вам прямо скажут, что есть легаси на 7 и 6 версиях. Есть еще один нюанс который нужно понимать: после 8 версии изменения уже не такие значительные. 8 была самой прорывной - именно поэтому она всех устраивает и только из-за политики Оракл пошла гонка релизов. То есть изучив 8 Вы затем быстро можете догнать до нужной версии 11 или 17 и при этом еще более-менее будете понимать 7 и 6 - хотя там конечно кода

больше и меньше “синтаксического сахара” (то есть когда вы пишете строчку когда, а под капотом выполняется два листа). В старых версиях придется самому написать два листа.

Ниже приведена таблица нумерации версий Java:

Версия	Дата выпуска
JDK-Beta	1995
JDK-1.0	январь 1996
JDK-1.1	февраль 1997
J2SE-1.2	декабрь 1998
J2SE-1.3	май 2000
J2SE-1.4	февраль 2002
J2SE-5.0	сентябрь 2004
Java-SE-6	декабрь 2006
Java-SE-7	июль 2011
Java-SE-8-(LTS)	март 2014
Java-SE-9	сентябрь 2017
Java-SE-10	март 2018
Java-SE-11-(LTS)	сентябрь 2018
Java-SE-12	март 2019
Java-SE-13	сентябрь 2019
Java-SE-14	март 2020
Java-SE-15	Сентябрь 2020
Java-SE-16	Март 2021
Java-SE-17-(LTS)	Сентябрь 2021

Еще в Java сохранена обратная совместимость. Что это значит? Если Вы возьмете новую версию JDK, например 16, и попыдаете запустить на ней приложение, которое написано на 8 версии, то оно успешно запустится и будет работать.

Однако в случае работы над исходным кодом, то есть когда будете писать приложение, вполне возможно, что возникнут проблемы. Например, у меня стояла JDK 11, и когда я пытался работать с кодом Java 8 версии - сыпались ошибки на стадии компиляции, пришлось установить 8 версию JDK. То есть версия языка и версия JDK если не совпадают, то при разработке могут быть проблемы. Но не обязательно. До этого я год писал проект на синтаксисе 8 версии используя JDK 11.

Глава 5. Начинаем изучать Java

Книга, с которой я рекомендую начать называется «Head First Java» Берта Бейтса и Кэти Сьерра. Несмотря на то, что написана она по Java 5 версии, при работе на JDK 8 проблем не возникнет - проверено.

Почему я советую эту книгу: она быстро введет Вас в язык. Да, многое будет непонятно по началу, но изложение материала очень хорошее, здесь нет общего подхода для всех книг по программированию: длительного и нудного изучения темы за темой - от примитивов и конструкций языка и только через треть книги предлагается начать писать код и т.д. В Head First Java все объясняется по ходу. То есть Вы начнете сразу писать код.

Многие ругают книгу за то, что мол она поверхностная, но это я считаю просто люди не поняли смысла книги - она быстро вводит в язык. Такие люди говорят, что книги Шилдта лучше. Лично мне не зашло, там именно академическое и достаточно сухое изложение. Впрочем, это все вкусовщина, может Вам больше понравится книга Шилдта, я могу советовать только то, что попробовал сам. Итак, берите Head First Java и вперед!

Можно параллельно изучать какие-либо курсы, например на <https://stepik.org/> они бесплатные, также можете заглядывать сюда <http://proglang.su/java> - это самоучитель джава. Когда Вы запомните синтаксис языка, то можно полностью погружаться в книги.

Несколько рекомендаций:

- Если Ваш английский позволяет - читайте на нем. Я читал на русском.

- Книгу нужно отработать: обязательно постараться выполнить и разобрать все упражнения. Очень важно не просто проходить

главы, а действительно понять материал. Мне приходилось несколько раз возвращаться из-за того, что в погоне за скоростью я “прошел” главу, но не усвоил материал. В общем не гоните черезчур, но и не затягивайте - каждый день должен быть прогресс. Если, конечно, Вы хотите скорее сменить работу.

- Ежедневные хотя бы часовые занятия значительно лучше, чем многочасовые раз в неделю. Ну и конечно нужно выделять в выходные хотя бы 2-3 часа.

- 12 и 13 главы можно смело пропускать. Там рассказывается о том, как создавать графический интерфейс для настольных приложений. Таких вакансий на рынке можно пересчитать по пальцам одной руки. SWING совершенно не востребован и никогда на собеседовании про него не спросят. В конце концов можно сделать веб-интерфейс и использовать хоть локально, хоть с сервера.

- Главу 18 тоже можете пропустить. Описанное там понадобится Вам не скоро, а может никогда.

У меня изучение этой книги заняло около двух месяцев по несколько часов в день. Когда отработаете Java Head First можно двигаться дальше.

Глава 6. Дополнительные инструменты

В повседневной деятельности разработчик использует не только среду разработки, но и другие инструменты.

Одним из них является система контроля версий - VCS. Как обычно в ИТ - подобных инструментов множество. Наиболее распространены 2:

Первый - это GIT, фактически стандарт индустрии. Второй - SVN. SVN мне приходилось встречать только в двух маленьких компаниях, где он использовался на собственных серверах. GIT используется практически во всех крупных компаниях. Такие веб-ресурсы как GitHub, Bitbucket, GitLab и им подобные под капотом содержат именно GIT. Так что на собеседованиях если и будут вопросы по системам контроля версий - только про GIT. Мы с Вами немного изучим GIT.

Чтобы начать пользоваться этим инструментом, Вам совершенно не нужно погружаться в его устройство, изучать достаточно объемный tutorial и т.д. Это все Вы сможете сделать позже - когда Вам действительно понадобятся сложные манипуляции с кодом. Это будет когда Вы отработаете в компании программистом хотя бы год. По крайней мере джуну это точно не ставят в требования.

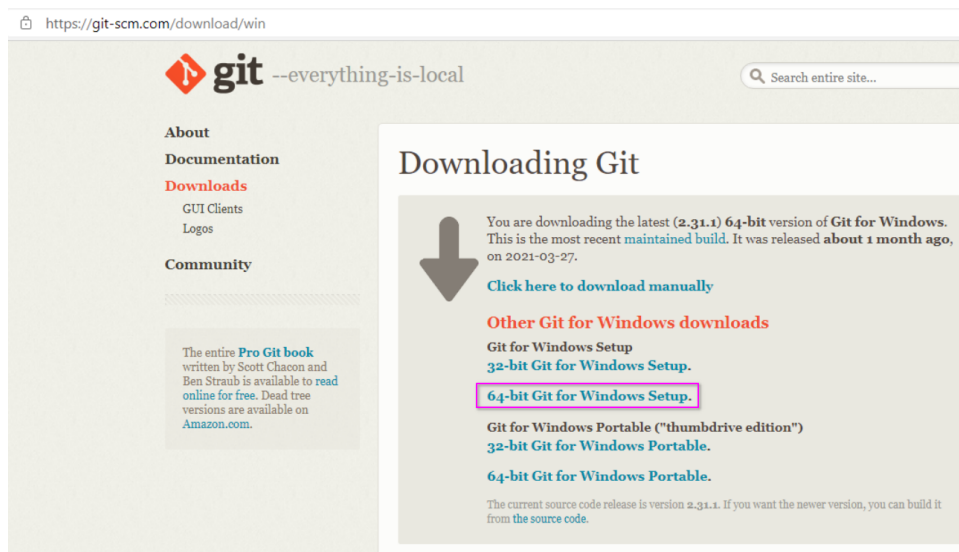
На сегодня Вам нужно знать следующее:

Система контроля версий (далее для краткости я буду писать GIT, но Вы помните - что их достаточно много, тем не менее функционал практически одинаков) позволяет иметь одновременно несколько версий файла (кода), а также откатываться к предыдущим версиям файла вплоть до его первого изменения.

GIT позволяет команде разработчиков одновременно работать над одним и тем же кодом.

Перейдем к практическому использованию GIT (для общего сведения можете конечно копнуть глубже и загуглить, однако предлагаю пока не заморачиваться и оставить в голове место для Java). Реально нам на сегодня для ведения разработки понадобится всего 5 команд из GIT.

Давайте скачаем дистрибутив приложения и установим его. Для этого переходим на сайт <http://git-scm.com/download/win> и нажимаем пункт выделенный фиолетовым прямоугольником:

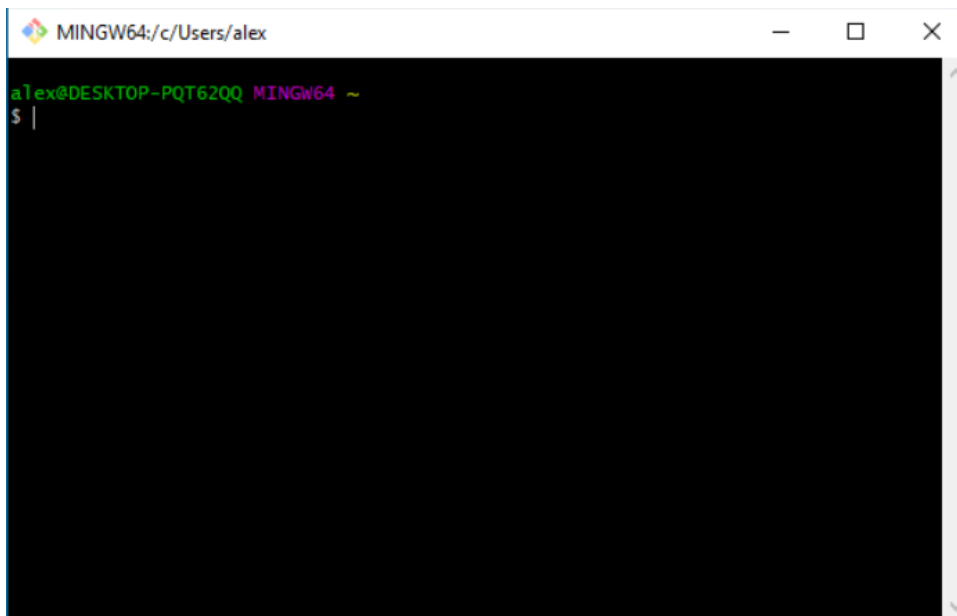


Запускаем скачанный файл. Скриншоты установки приводить не буду - там все тривиально: жмите NEXT ничего не меняя пока установка не закончится. По завершении нажмите Finish.

Теперь в меню Пуск у вас есть в "Недавно добавленные" три значка:

1. GIT GUI - это версия с графическим интерфейсом;
2. GIT CMD – Windows консоль;
3. GIT Bash - линуксовая консольная версия в которой мы и будем работать. Поскольку я в Windows мало что понимаю. Вы можете затем самостоятельно изучить графическую версию, однако не рекомендую этого делать: консоль будет всегда, даже если Вы как сейчас будете работать в Windows. А вот графика нет - есть компании, где принципиально разработчикам выдают ПК с Linux или MacOS и вот там графика в GIT под большим вопросом.

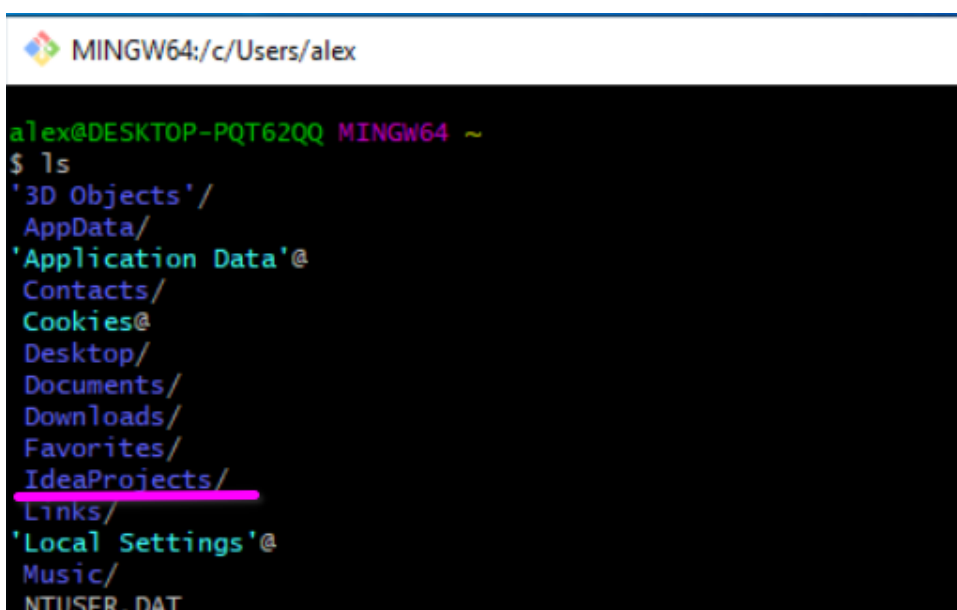
Итак, запускаем Git Bash и любуемся на черную консоль. Шучу - по-настоящему черная она в Linux))).



```
MINGW64:/c/Users/alex
alex@DESKTOP-PQT62QQ MINGW64 ~
$ |
```

Не пугайтесь, сейчас все станет яснее! Правда придется чуть-чуть изучить Bash - это командная оболочка, используется в большинстве Linux по умолчанию.

Взгляните еще раз на картинку, там Вы видите 2 строки. В первой строке для Вас важно, что написано после фиолетовых букв. Сейчас там тильда “~” что означает что Вы находитесь в домашнем каталоге. Давайте посмотрим, что есть в нем. Наберите на клавиатуре ls - это команда которая выполняет listing текущего каталога, то есть отображает все файлы и директории находящиеся в нем:



```
MINGW64:/c/Users/alex
alex@DESKTOP-PQT62QQ MINGW64 ~
$ ls
'3D Objects'/
AppData/
'Application Data'@
Contacts/
Cookies@
Desktop/
Documents/
Downloads/
Favorites/
IdeaProjects/
Links/
'Local Settings'@
Music/
NTUSER.DAT
```

Смотрите! Там есть наш каталог проектов - IdeaProjects. Я подчеркнул фиолетовым цветом. Давайте перейдем в него. Это делается командой cd сокращение от change directory.

Вводим cd Idea и нажимаем клавишу Табуляции (Tab) Bash сам дополнит имя каталога и слэш (разделитель пути). Будет выглядеть так:

```
$ cd IdeaProjects/
```

Теперь нажимаем Enter. Обратите внимание: в верхней строке после тильды "~" появилось имя веденного каталога:

```
alex@DESKTOP-PQT62QQ MINGW64 ~/IdeaProjects  
$ |
```

Это означает что мы сейчас находимся в этом каталоге. Посмотрим, что в нем находится. Вызовем уже известную Вам команду ls (listing):

```
$ ls
```

И нажмем Enter. У меня там проект first:

```
alex@DESKTOP-PQT62QQ MINGW64 ~/IdeaProjects  
$ ls  
first/
```

Перейдем в директорию известной Вам командой cd (change directory):

\$ cd first (вы можете перейти в любой другой каталог, где у Вас сейчас есть проект) Жмем Enter.

Обратите внимание что адресная строка снова сменилась:

```
alex@DESKTOP-PQT62QQ MINGW64 ~/IdeaProjects/first  
$ |
```

Мы в каталоге first. Вы можете перейти в любой другой каталог. Смысл тот же.

Теперь приступаем к начальному изучению GIT.

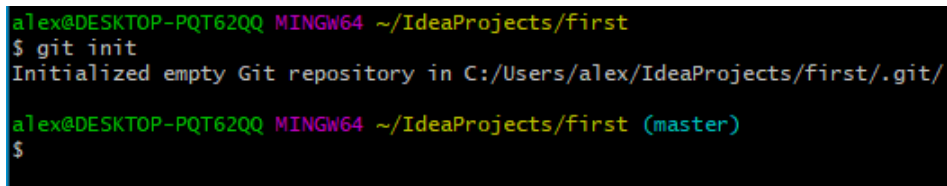
Выполните команду `cd ~` - это переместит Вас в домашний каталог, так как текущий каталог нам еще понадобится. Перейдите по ссылке <https://githowto.com/ru> нажмите там кнопку "Поехали" и пройдите там уроки с 1 по 9.

Прошли? Отлично! Теперь давайте снова войдем в каталог first:

```
$ cd ~/IdeaProjects/first
```

Чтобы создать репозиторий нужно выполнить команду `git init` находясь в нужном каталоге. Мы уже находимся в нем так что выполняем:

```
$git init
```

A screenshot of a terminal window with a black background and green text. The prompt is 'alex@DESKTOP-PQT62QQ MINGW64 ~/IdeaProjects/first'. The command '\$ git init' is entered, followed by the output 'Initialized empty Git repository in C:/Users/alex/IdeaProjects/first/.git/'. The prompt then changes to 'alex@DESKTOP-PQT62QQ MINGW64 ~/IdeaProjects/first (master)' and '\$' is entered again.

```
alex@DESKTOP-PQT62QQ MINGW64 ~/IdeaProjects/first
$ git init
Initialized empty Git repository in C:/Users/alex/IdeaProjects/first/.git/
alex@DESKTOP-PQT62QQ MINGW64 ~/IdeaProjects/first (master)
$
```

Отлично. Как Вы можете видеть теперь после имени текущей директории у нас появилась надпись голубого цвета (master) - это означает что мы в гите находимся на основной ветке. Пока это просто для информации. Мы все будем делать в одной ветке. Работу с ветками предлагаю осваивать уже при подготовке к собеседованиям.

Теперь проиндексируйте файлы: `$git add .`

И закоммитьте: `$git commit -am "FirstCommit"`

Глава 7. Запиливаем репу!

Как Вы поняли Git сохраняет изменения в директории first/.git. Однако хранить информацию на ПК это не слишком хороший навык - в компаниях за это ругают, особенно если вдруг ПК выходит из строя и проделанная работа погибла вместе с компьютером.

Для хранения информации нужно использовать специализированные сервисы. Давайте и мы начнем хранить наш проект правильно. Перейдите по ссылке <https://github.com/> нажмите там sign up и создайте аккаунт. Затем перейдите в свой почтовый ящик, там будет письмо от GitHub - нажмите в нем кнопку Verify email address.

Снова вернитесь в GitHub. В правом верхнем углу нажмите плюсики и выберите New repository. Введите название и внизу нажмите зеленую кнопку Create repository.

Перед Вами будет список гитовых команд нам нужно скопировать эту:

```
«git remote add origin  
https://github.com/вашеИмя/имяРепозитория.git»
```

и вставить ее в консоль Git. Жмем Enter. Это мы указали гиту, что этот репозиторий будет храниться в этой удаленной ветке на GitHub. Теперь нам нужно залить туда сами файлы. Выполняем команду:

```
$git push
```

Гит выдаст нам сообщение что удаленной ветки не существует. И нужно выполнить команду:

```
$git push - -set-upstream origin master
```

Теперь в GitHub кликните по имени своего репозитория и вы увидите список своих файлов. Обратите внимание также что в списке присутствуют каталоги .idea и target - это нехорошо. Эти каталоги никому не нужны в репозитории, их синхронизация занимает лишнее время, и выдает человека что он не умеет пользоваться

гитом. Давайте сообщим гиту что служебные каталоги не нужно отслеживать. Делается это при помощи файла `.gitignore`. В левом верхнем углу GitHub есть строка поиска давайте в ней напишем `gitignore`. Если прямо сейчас нажать Enter, то GitHub будет искать в наших репозиториях. А у нас этого файла нет. Нужно искать по всему GitHub. Нажмите на кнопку, которая находится в этой же строке поиска (в правой ее части) и в выпадающем меню выберите All GitHub. Теперь мы видим список найденных репозиторий. Самый верхний `github/gitignore`. Заходим в него и найдем там файл `Java.gitignore`. Скопируем содержимое.

Откройте свой проект в IDE. Правой кнопкой кликните на головной директории проекта > New > File и введите имя файла: `.gitignore` (имя файла должно начинаться с точки). Теперь откройте этот файл и вставьте содержимое, что мы скопировали в GitHub. Теперь давайте добавим в этот файл (в любое его место) наши директории `.idea` и `target`.

В GIT индексируем: `$git add .`

Коммитим: `$git commit -am "gitignoreAdded"`

Заливаем на GitHub: `$git push`

Идем на GitHub, обновим страницу и... файл `.gitignore` приехал, а директории `.idea` и `target` никуда не делись! А дело в том, что GIT не применил наш файл, а взял данные из своего кэша.

Выполняем в GIT команды:

`$git rm -rf - -cached .` (удалить кэш всех файлов. Точка в конце команды - это все файлы)

Снова индексируем: `$git add .`

Коммитим `$git commit - -am "removeGarbage"`

Заливаем на GitHub: `$git push`

Идем в GitHub, обновляем и любимся! У нас теперь репозиторий выглядит почти как у профессионального разработчика! Почему почти - потому, что нет директории test - где хранятся файлы с юнит тестами. Но тестирование нужно изучать после Java, а то будет каша.

Теперь чтобы почувствовать себя настоящим разработчиком выполним трюк: для чего все это и нужно было - удалим репозиторий на компьютере и скачаем его с GitHub.

Закройте IDE чтобы можно было удалить файлы.

Закройте GIT.

Зайдите в директорию проектов IdeaProjects и удалите директорию с проектом.

Зайдите в GitHub, нажмите на зеленую кнопку Code и в выпавшем меню скопируйте адрес своего проекта. Скопировать можно нажав на кнопку справа, в виде блокнотика.

Откройте GIT Bash.

Перейдите в директорию проектов: `$cd ~/IdeaProjects`

`$git clone <здесь вставьте адрес своего проекта, который скопировали на шаге 4>`

Теперь в Windows откройте директорию проектов IdeaProjects и о, чудо! Директория с проектом на месте!!!

Запустите IDE и откройте свой проект. Дождитесь завершения индексации файлов (помните, статус бар в правом нижнем углу).

Теперь Вы можете продолжить работу с проектом как ни в чем не бывало.

Поздравляю! Вы на шаг ближе к мечте! Теперь не забывайте после внесения изменений в проект пушить его в GitHub. Это очень важно:

Во-первых, вы приучаетесь работать профессионально. Во-вторых, когда Вы начнете пилить свой демо-проект для собеседований, там должно быть значительное количество коммитов - многие работодатели обращают на это внимание, это говорит о том, что человек умеет пользоваться гитом.

Правильно выполнять коммит на каждое изменение. Новички часто вносят много изменений и делают один коммит. Так делать не нужно, так как в случае если была допущена ошибка и на каждое изменение был бы коммит, то можно было бы просто откатить коммит с ошибкой, а так придется ковыряться в коде. Кроме того, при проведении код-ревью очень неудобно, когда в одном коммите значительное количество кода. Так что приучайте себя пользоваться гитом постоянно.

Кстати, в GitHub можно создавать сколько угодно репозиториев. Чтобы потом в них не путаться в каждый репозиторий нужно добавлять файл README.md - в котором описывается что это за репозиторий. Создайте файл README.md в IDE и напишите в него, например, "Мой первый репозиторий!". Проиндексируйте, закомитьте и запустите. Зайдите на GitHub, откройте свой репозиторий и ниже списка файлов Вы увидите содержимое файла:

src/main/java	first	1 hour ago
.gitignore	asdf	1 hour ago
README.md	Update README.md	7 minutes ago
pom.xml	first	1 hour ago

README.md

first

Это мой первый репозиторий!

Глава 8. Продолжаем изучение Java

Теперь можно продолжить изучение Java. Предлагаю это делать с помощью одного из бестселлеров Кея Хорстманна - двухтомника "Java Библиотека профессионала" десятое издание - оно как раз по 8 версии языка.

Рекомендации по прочтению:

- Если Ваш английский позволяет - читайте на нем. Я читал на русском.

- Старайтесь понять и запомнить, но сильно не залипайте на одном месте. Когда начнете писать код будете заглядывать если что забудете.

- Ежедневные хотя бы часовые занятия значительно лучше, чем многочасовые раз в неделю. Ну и конечно нужно выделять в выходные хотя бы 2-3 часа.

- Главы с 10 по 13 первого тома читать не нужно.

- Второй том достаточно прочитать по 6 главу включительно.

- Обязательно пишите код - просто читать недостаточно.

- Постоянно пытайтесь писать свой проект.

На мой взгляд лучше всего читать следующим образом:

1 том главы с 1 по 9 включительно;

затем 2 том 5 главу "Работа с базами данных" и переходите к 10 главе этой книги, чтобы писать демоприложение.

Затем "добиваете" 1 том - прочитываете 11 главу. И конечно все пробуете в коде.

Далее 2 том 1 – 4, 6 главы.

Глава 9. SQL и базы данных.

Как разработчик Вы обязаны уметь работать с базами данных. И Ваше демо приложение должно читать и писать с\в базу данных.

Баз данных огромное множество, все не перечислить. Наиболее распространены Oracle, MySQL, PostgreSQL и H2.

Oracle достаточно сложно устанавливать и настраивать и с насюда не взять - нужно изучить документацию.

MySQL последнее время теряет позиции и в общем-то не слишком котируется из-за относительного малой функциональности.

H2 написана на Java и работает в оперативной памяти. Очень удобно для тестирования каких-либо фишек приложений, не требует установки.

PostgreSQL - широко используется, бесплатна и по возможностям близка к Оракулу, в связи с чем многие компании даже с Оракла переходят на нее по финансовым соображениям. Поэтому изучать стоит PostgreSQL.

Еще один нюанс - все перечисленные базы используют SQL - structured query language — «язык структурированных запросов», но у каждой базы свой диалект, поэтому синтаксис незначительно, но отличается. И соответственно если запрос составленный для одной базы попытаться выполнить в другой - получим ошибку.

Вы можете скачать дистрибутив любой системы управления базами данных и установить, однако кроме потерянного времени никакого профита Вам это не даст - в реале никто не использует базы данных на Windows, разве что в учебных целях. Установка на Linux многих баз проходит штатно, за исключением Oracle, который не будет устанавливаться если ядро ОС предварительно не подготовлено. Так что, если Вы на Linux ставьте PostgreSQL, те кто на Windows, чтобы не возиться с установкой сервера баз данных, перейдите по ссылке <https://sourceforge.net/projects/pgsqlportable/> нажмите зеленую кнопку Download и скачайте портабельную версию. Распакуйте zip архив. Поместите директорию в удобное для

Вас место, например в каталог проектов IdeaProjects. Главное не размещайте его на рабочем столе - возможны глюки.

9.1. Старт базы данных

Заходим в директорию скачанного дистрибутива Postgres и видим там пакетные файлы Start, Stop, Restart. Тут думаю все понятно. А вот PgAdmin4 - это приложение для администрирования и разработки. Запускаем Start.

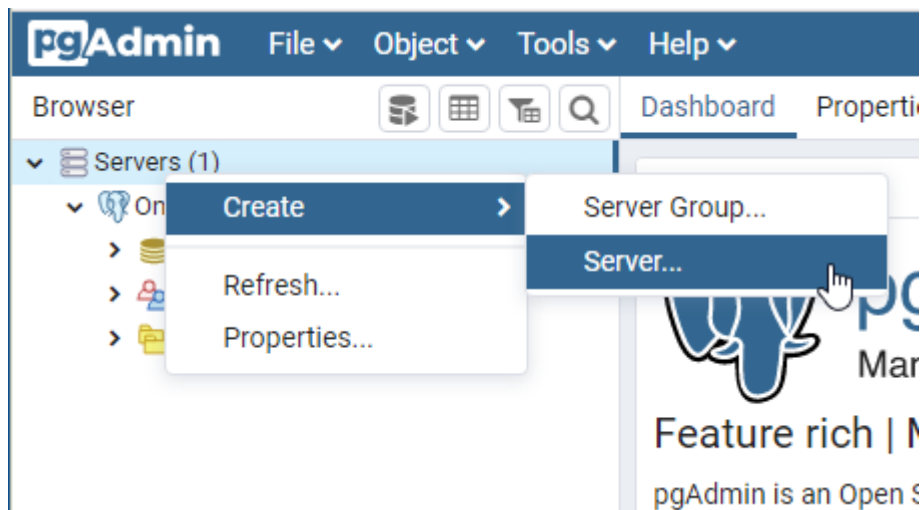
Если вдруг получили сообщение об ошибке из-за отсутствия файла vcruntime140.dll, - зайдите на <https://support.microsoft.com/ru-ru/help/2977003/the-latest-supported-visual-c-downloads> В разделе Visual Studio 2015, 2017 и 2019 загрузите файлы:

vc_redist.x64.exe

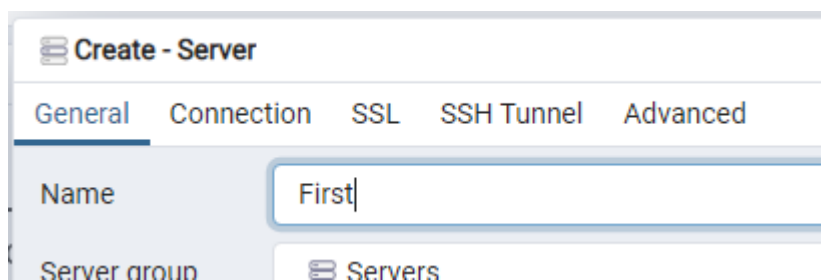
vc_redist.x86.exe

Запустите загруженные файлы и установите необходимые компоненты. Если при запуске одного из двух файлов Вам сообщат, что компоненты уже установлены, перейдите к установке второго. И снова запустите Start.

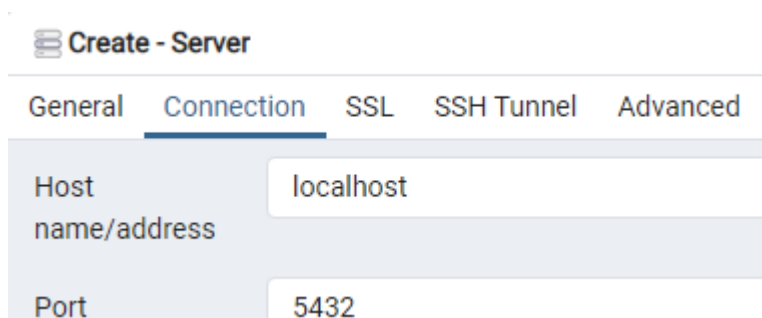
Теперь запустите PgAdmin4. Вводим пароль 123. Создаем сервер:



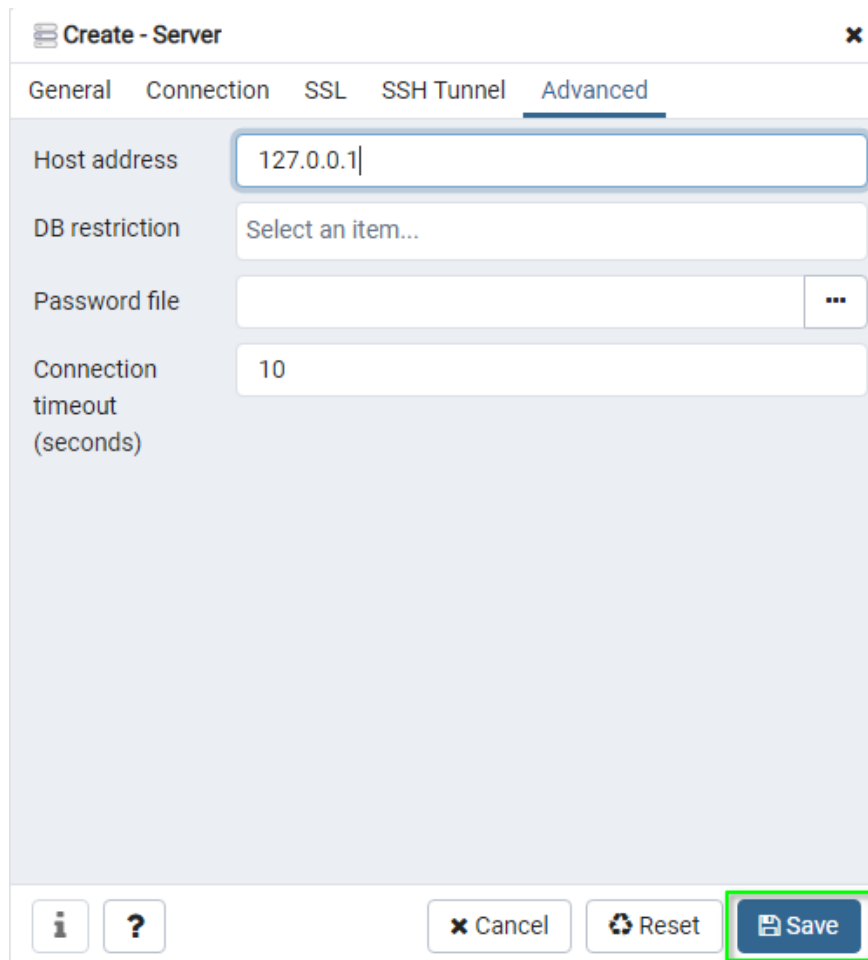
В строку Name вводим имя сервера:



В меню Connection в строке Host name вводим localhost:



В меню Advanced > Host address вводим 127.0.0.1 и жмем Save:

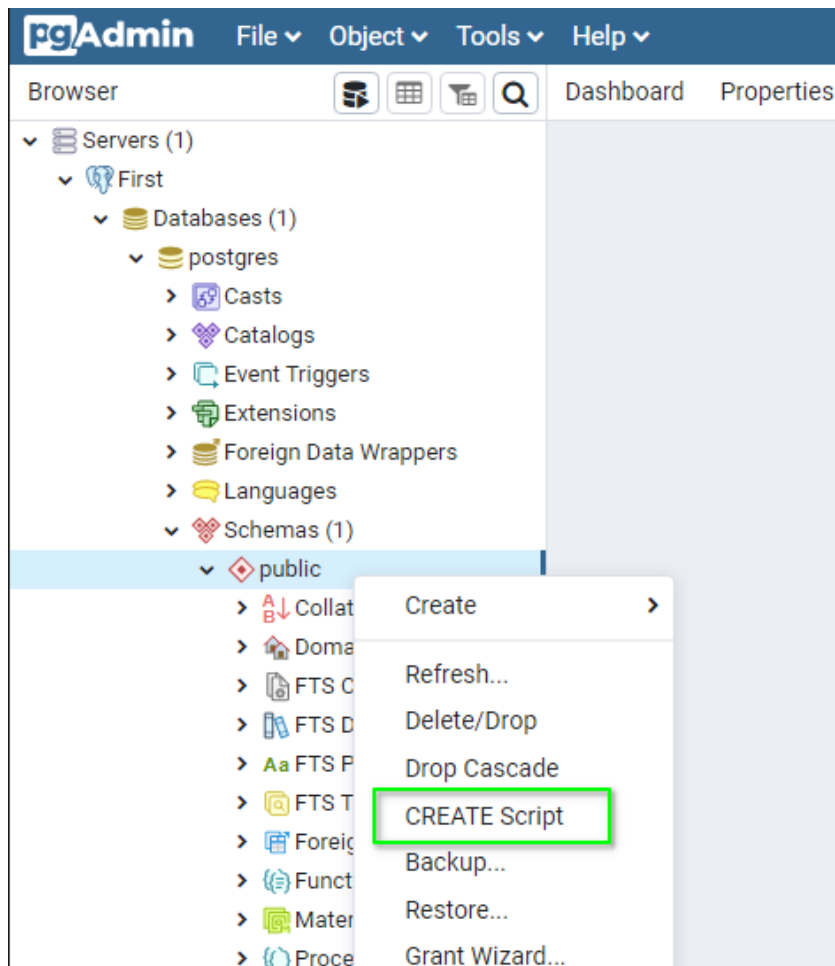


The screenshot shows a 'Create - Server' dialog box with the 'Advanced' tab selected. The 'Host address' field contains '127.0.0.1'. The 'DB restriction' field shows 'Select an item...'. The 'Password file' field is empty with a dropdown arrow. The 'Connection timeout (seconds)' field contains '10'. At the bottom, there are buttons for 'Cancel', 'Reset', and 'Save'. The 'Save' button is highlighted with a green border.

Field	Value
Host address	127.0.0.1
DB restriction	Select an item...
Password file	
Connection timeout (seconds)	10

9.2. Создаем таблицу

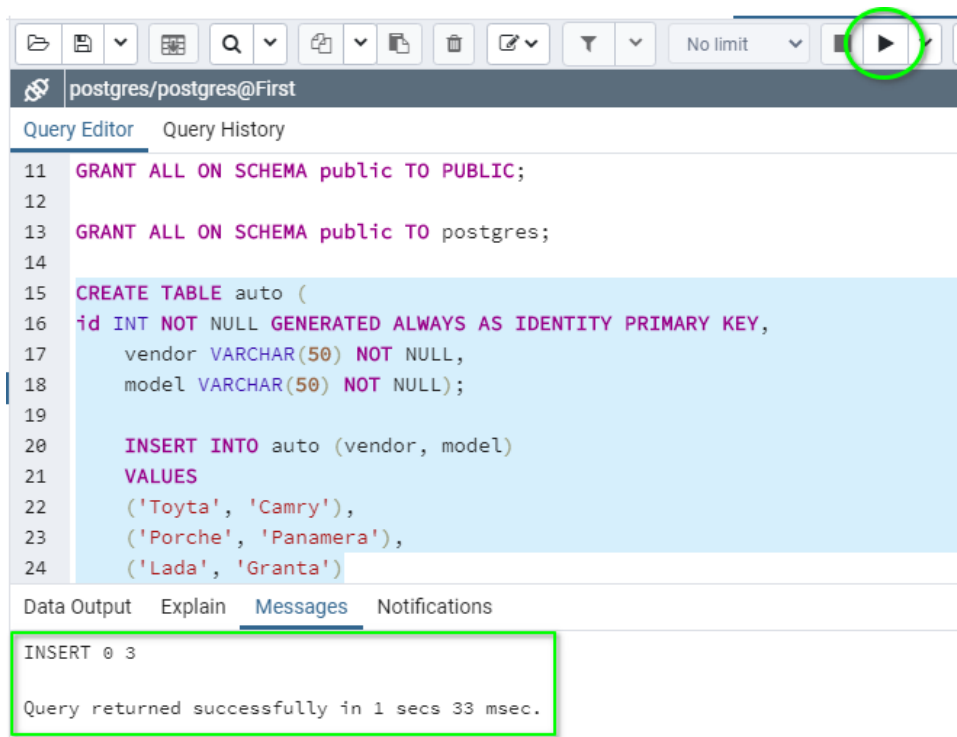
Теперь создаем скрипт. Идем в First > Databases > postgres > public – нажимаем правой кнопкой мыши и выбираем CREATE Script:



Откроется Query Editor. В самый низ пишем:

```
CREATE TABLE auto (  
id INT NOT NULL GENERATED ALWAYS AS IDENTITY PRIMARY  
KEY,  
vendor VARCHAR(50) NOT NULL,  
model VARCHAR(50) NOT NULL  
);  
INSERT INTO auto (vendor, model)  
VALUES  
(‘Toyota’, ‘Camry’),  
(‘Porche’, ‘Panamera’),  
(‘Lada’, ‘Granta’)
```

Вот так:



Затем выделите этот текст, чтобы он стал синеньким как на картинке и нажмите кнопку наверху (Выделена зеленым кружком). Вывод (я отметил зеленым прямоугольником) говорит о том, что команды успешно выполнились. Если Вы получили другой вывод - значит, где-то закралась ошибка, проверьте все строки.

Проверим, что у нас есть такая таблица. Выполним запрос:

SELECT * FROM auto;

Также выделим и нажмем на кнопку:

A screenshot of a PostgreSQL query editor interface. The query editor shows a SQL script with line numbers 24 to 26. The script includes an INSERT statement and a SELECT * FROM auto; statement. The 'Data Output' tab is selected, displaying a table with 3 rows and 3 columns: id, vendor, and model. The table contains the following data: (1, 'Toyta', 'Camry'), (2, 'Porsche', 'Panamera'), and (3, 'Lada', 'Granta').

	id [PK] integer	vendor character varying (50)	model character varying (50)
1	1	Toyta	Camry
2	2	Porsche	Panamera
3	3	Lada	Granta

Вот, собственно, мы видим нашу созданную таблицу.

Теперь пройдите этот курс <https://learndb.ru/>

Достаточно первых трех тем: "Введение", "Отсечение и сортировка", "Соединения".

SQL достаточно простой язык, но объем все равно не маленький. До какой степени его учить? Я так скажу: до такой, чтобы удобно было пилить демо приложения и ответить на собеседовании на вопросы. Чтобы работать с базой достаточно знать следующее:

1. Как создать\удалить базу
2. Как создать\удалить таблицу
3. Как наполнить таблицу данными (INSERT)
4. Как выполнить запрос. (SELECT) Вот здесь есть куча параметров, их как раз изучать не обязательно пока. Достаточно будет знать как работает JOIN (каждый вид).
5. Что такое первичный ключ, внешний ключ.
6. Нормализация таблиц: для чего она и какие виды бывают (все помнить не обязательно. Первые трех за глаза).

Далее можно пользоваться справочником, например таким: <https://unetway.com/tutorials/sql>

Ну и конечно, лучше всего пользоваться документацией от производителя: <https://www.postgresqltutorial.com/>

Если Вам сложно читать по-английски - начинайте подтягивать язык, в том числе с помощью чтения таких ресурсов. Без английского в ИТ невозможно. На всякий случай русская документация здесь <https://postgrespro.ru/education>

Не пытайтесь изучить SQL досконально, объем немалый но, самое главное, если Вы не будете каждый день пользоваться SQL он

быстро улетучится. Говорю это, как и все остальное здесь, на основании собственного опыта. Главное понимать принцип и знать ответы на перечисленные выше вопросы. Все остальное можно посмотреть в документации.

Если Вы еще не прочитали главу 5 второго тома Кея Хорстманна “Java Библиотека профессионала”, то сейчас самое время. Когда прочтете по раздел 5.4.4 включительно можно двигаться дальше, а остальное дочитать потом. Если Вы уже все прочитали, то есть смысл пролистать 5 главу и просмотреть по диагонали чтобы освежить в памяти детали.

Глава 10. Пишем приложение для коннекта с базой данных.

Убедитесь, что PostgreSQL стартован. Открываем IDE, создаем проект (кто не помнит, как это делать посмотрите в первой части книги, мы уже создавали проект first), в новом проекте создаем класс TestDb и пишем:


```

public static void main(String[] args) {
    TestDb s = new TestDb();
    s.makeQuery();
}

private void makeQuery() {
    try {
        Class.forName("org.postgresql.Driver");
        String url = "jdbc:postgresql://localhost:5432/postgres";
        String login = "postgres";
        String password = "123";
        Connection con = DriverManager.getConnection(url, login, password);

        Statement stm = con.createStatement();
        ResultSet rs = stm.executeQuery("sql: \"SELECT * FROM auto\"");
        while (rs.next()) {
            String str = rs.getString("columnLabel: \"vendor\"") + ":" + rs.getString("columnIndex: 3");
            System.out.println(str);
        }
        con.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

10.1. Загрузка драйвера JDBC

Почти все готово. Осталось загрузить драйвер JDBC чтобы наше приложение “говорило” на языке PostgreSQL.

Идем в репозиторий мавена: <https://mvnrepository.com/>

В строке поиска САЙТА набираем: jdbc и жмем Enter. Далее просматриваем поисковую выдачу - ищем “PostgreSQL...”:


mvnrepository.com/search?q=jdbc


Maven BB GitHub fenglish.ru USATV Techrocks.ru


5


4


Microsoft JDBC Driver for SQL Server.
Last Release on Apr 29, 2021

6. **Hive JDBC**
 org.apache.hive » hive-jdbc
Hive JDBC
Last Release on Aug 27, 2019

7. **JDBC**
 org.clojure » java.jdbc
JDBC
Last Release on Feb 2, 2021

8. **Postgis JDBC Driver**
 net.postgis » postgis-jdbc
PostGIS adds support for geographic objects to the PostgreSQL
Last Release on Mar 30, 2020

9. **PostgreSQL JDBC Driver**
 org.postgresql » postgresql
PostgreSQL JDBC Driver Postgresql
Last Release on Apr 22, 2021

10. **Play JDBC**


Жмем прямо на черную надпись "PostgreSQL JDBC Driver" и проваливаемся внутрь:

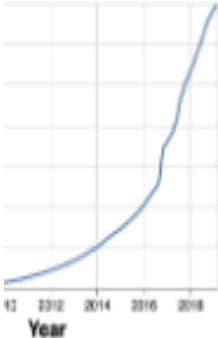
mvnrepository.com/artifact/org.postgresql/postgresql

ujl Maven BB GitHub fenglish.ru USATV Te

POSITORY

Search for groups, a

ifacts (20.5M)



Year

ategories

ted

orks

etrics

aries

ie Parsers

mentations

iting

Home » org.postgresql » postgresql

PostgreSQL JDBC Driver

PostgreSQL JDBC Driver Postgresql

License	BSD 2-clause
Categories	PostgreSQL Drivers
Tags	database postgresql driver
Used By	2,678 artifacts

Central (137) Atlassian 3rd-P Old (1) Redhat GA (

42.2.20

42.2.20.jre7

42.2.20.jre6

Жмем на последнюю версию (она всегда вверху списка любого репозитория, на картинке я подсветил зеленым). И копируем содержимое (достаточно щелкнуть мышкой и уже скопировано, удобно!):



PostgreSQL JDBC Driver » 42.2.20

PostgreSQL JDBC Driver PostgreSQL

License	BSD 2-clause
Categories	PostgreSQL Drivers
Organization	PostgreSQL Global Development Group
HomePage	https://jdbc.postgresql.org
Date	(Apr 22, 2021)
Files	jar (981 KB) View All
Repositories	Central
Used By	2,678 artifacts

[Maven](#) [Gradle](#) [SBT](#) [Ivy](#) [Grape](#) [Leiningen](#) [Buildr](#)

```
<!-- https://mvnrepository.com/artifact/org.postgresql/postgresql -->
<dependency>
  <groupId>org.postgresql</groupId>
  <artifactId>postgresql</artifactId>
  <version>42.2.20</version>
</dependency>
```

☒ Include comment with link to declaration

Copied to clipboard!

Теперь в файле конфигурации проекта pom.xml или в простонародье “помник” пишем:

```
<dependencies>
</dependencies>
```

И между этими тегами вставляем скопированное. Должно получиться вот так:

```
<dependencies>
<!-- https://mvnrepository.com/artifact/org.postgresql/postgresql -->
<dependency>
  <groupId>org.postgresql</groupId>
  <artifactId>postgresql</artifactId>
  <version>42.2.20</version>
</dependency>
</dependencies>
```

На следующей картинке этот блок выделен зеленым прямоугольником:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>org.example</groupId>
  <artifactId>untitled</artifactId>
  <version>1.0-SNAPSHOT</version>

  <properties>
    <maven.compiler.source>8</maven.compiler.source>
    <maven.compiler.target>8</maven.compiler.target>
  </properties>

  <dependencies>
    <dependency>
      <groupId>org.postgresql</groupId>
      <artifactId>postgresql</artifactId>
      <version>42.2.20</version>
    </dependency>
  </dependencies>

</project>
```

Обратите внимание на фиолетовую стрелочку. После любого изменения файла появляется такой значок, который говорит о том, что изменения еще не применены. Нажмем на этот значок и мавен подтянет нужные зависимости или наоборот удалит если Вы что-то удалили.

В нашем случае Maven скачает драйвер JDBC. Дождемся завершения процесса, а затем запустим наше приложение (как мы запускали ранее) и насладимся результатом:

```
StartApp x
C:\Users\alex\.jdk\adopt-openjdk-1.8.0_291\bin\java -Dmaven.home=C:\Users\alex\.m2\maven -jar C:\Users\alex\.m2\maven\bin\maven-idea.jar
Toyota:Camry
Porsche:Panamera
Lada:Granta

Process finished with exit code 0
```

Здесь хотел заострить Ваше внимание на двух моментах:

1. Обратите внимание на строчку:

```
String str = rs.getString("vendor") + ":" + rs.getString(3);
```

Как видите в первом случае мы получаем значение поля по имени столбца: **"vendor"**, а во втором случае - по его порядковому номеру: **3** (нумерация столбцов начинается с единицы).

Измените строку следующим образом:

```
String str = rs.getString(2) + ":" + rs.getString("model");
```

И посмотрите на результат запуска. Попробуйте выполнить другие запросы, например, попробуйте получить имя производителя модели Самгу и т.д.

2. Весь приведенный Выше код, для Вас сейчас должен быть абсолютно ясен: каждая строка и каждый используемый аргумент. Перечитайте код и убедитесь, что Вы точно все понимаете. Если есть непонятные места - перечитайте раздел 5 Хорстманна или хорошо прогуглите вопрос. Не оставляйте белых пятен в знаниях.

Глава 11. Пишем демо приложение CRUD REST API

Наше приложение будет работать с базой данных H2 и позволять управлять записями вида Имя, Фамилия, возраст, через REST API.

Что такое REST и SOAP довольно частый вопрос на собеседованиях:

REST (Representational State Transfer — «передача состояния представления») — архитектурный стиль взаимодействия компонентов распределённого приложения в сети. REST представляет собой согласованный набор ограничений, учитываемых при проектировании распределённой гипермедиа-системы. В определённых случаях (интернет-магазины, поисковые системы, прочие системы, основанные на данных) это приводит к повышению производительности и упрощению архитектуры. Вызов удалённой процедуры представляет собой обычный HTTP-запрос (GET, POST, PUT и т.д., если Вы не знакомы с протоколом HTTP самое время его изучить например почитайте здесь <https://ru.wikipedia.org/wiki/HTTP>);

такой запрос называют «REST-запрос»). Необходимые данные передаются в качестве параметров запроса.

Для веб-служб, построенных с учётом REST, применяют термин «RESTful». В отличие от веб-сервисов (веб-служб) на основе SOAP, не существует «официального» стандарта для RESTful веб-API. Так как **REST является архитектурным стилем**, в то время как **SOAP является протоколом**.

SOAP (Simple Object Access Protocol — простой протокол доступа к объектам) — протокол обмена структурированными сообщениями в распределённой вычислительной среде. Первоначально SOAP предназначался в основном для реализации удалённого вызова процедур (RPC). Сейчас протокол используется для обмена произвольными сообщениями в формате XML, а не только для вызова процедур.

SOAP может использоваться с любым протоколом прикладного уровня: SMTP, FTP, HTTP, HTTPS и др. Однако его взаимодействие с каждым из этих протоколов имеет свои особенности, которые должны быть определены отдельно. Чаще всего SOAP используется поверх HTTP.

Итак, создаем новый проект на Spring (точнее говоря на Spring Boot, но на данный момент это не столь важно) Сделать это можно 2 способами:

Первый - идем на сайт <https://start.spring.io/> там выбираем нужные параметры, например как на картинке:

start.spring.io

Maven BB GitHub fenglish.ru USATV Techrocks.ru JPA Spring Boot pepper.ru

spring initializr

Project

☒ Maven Project ☐ Gradle Project

Language

☒ Java ☐ Kotlin ☐ Groovy

Spring Boot

☐ 2.5.0 (SNAPSHOT) ☐ 2.5.0 (RC1) ☐ 2.4.6 (SNAPSHOT) ☒ 2.4.5

☐ 2.3.11 (SNAPSHOT) ☐ 2.3.10

Project Metadata

Group

Artifact

Name

Description

Package name

Packaging ☒ Jar ☐ War

Java ☐ 16 ☐ 11 ☒ 8

GENERATE CTRL + G EXPLORE

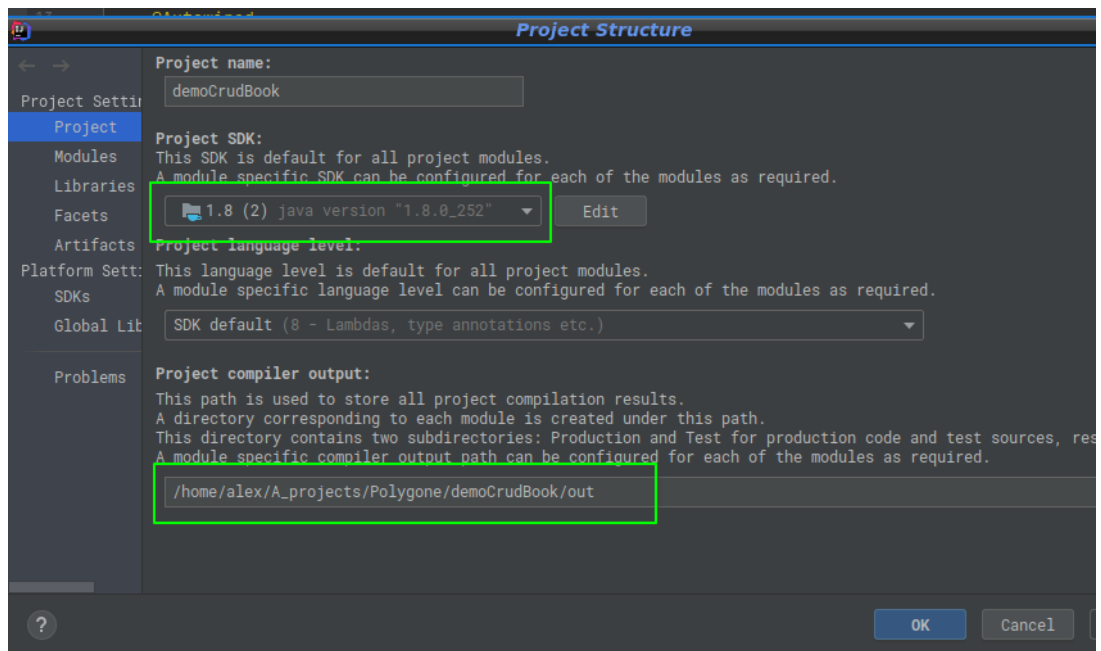
И жмем кнопку внизу “GENERATE” автоматически скачается архив с проектом demo.zip.

Распаковываем архив, открываем проект в IDE: File > Open и находим директорию с проектом.

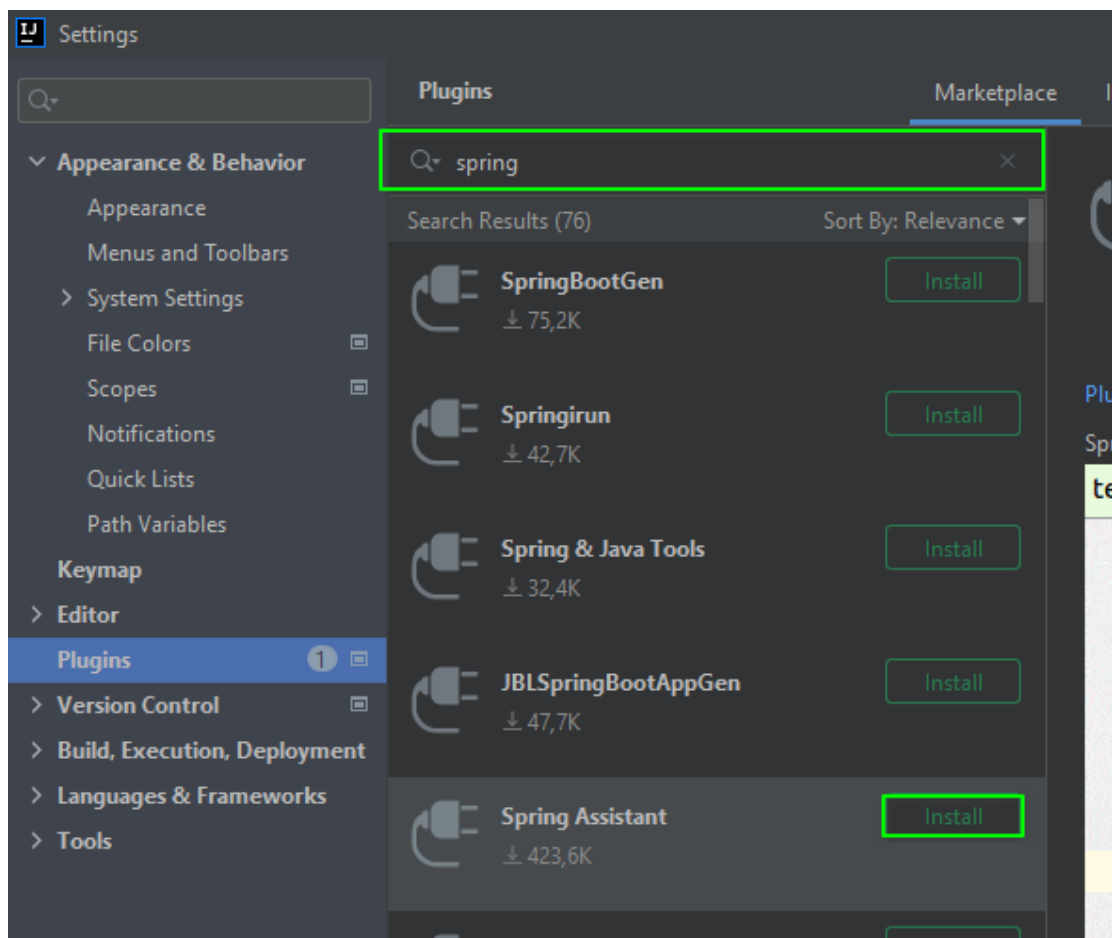
Далее важно убедиться в настройках проекта (File > Project Structure):

1. Что указан JDK 1.8
2. Указана директория для компиляции если нет, выбирайте C:\\TEMP.

На следующем скрине эти два параметра я выделил зелеными прямоугольниками:



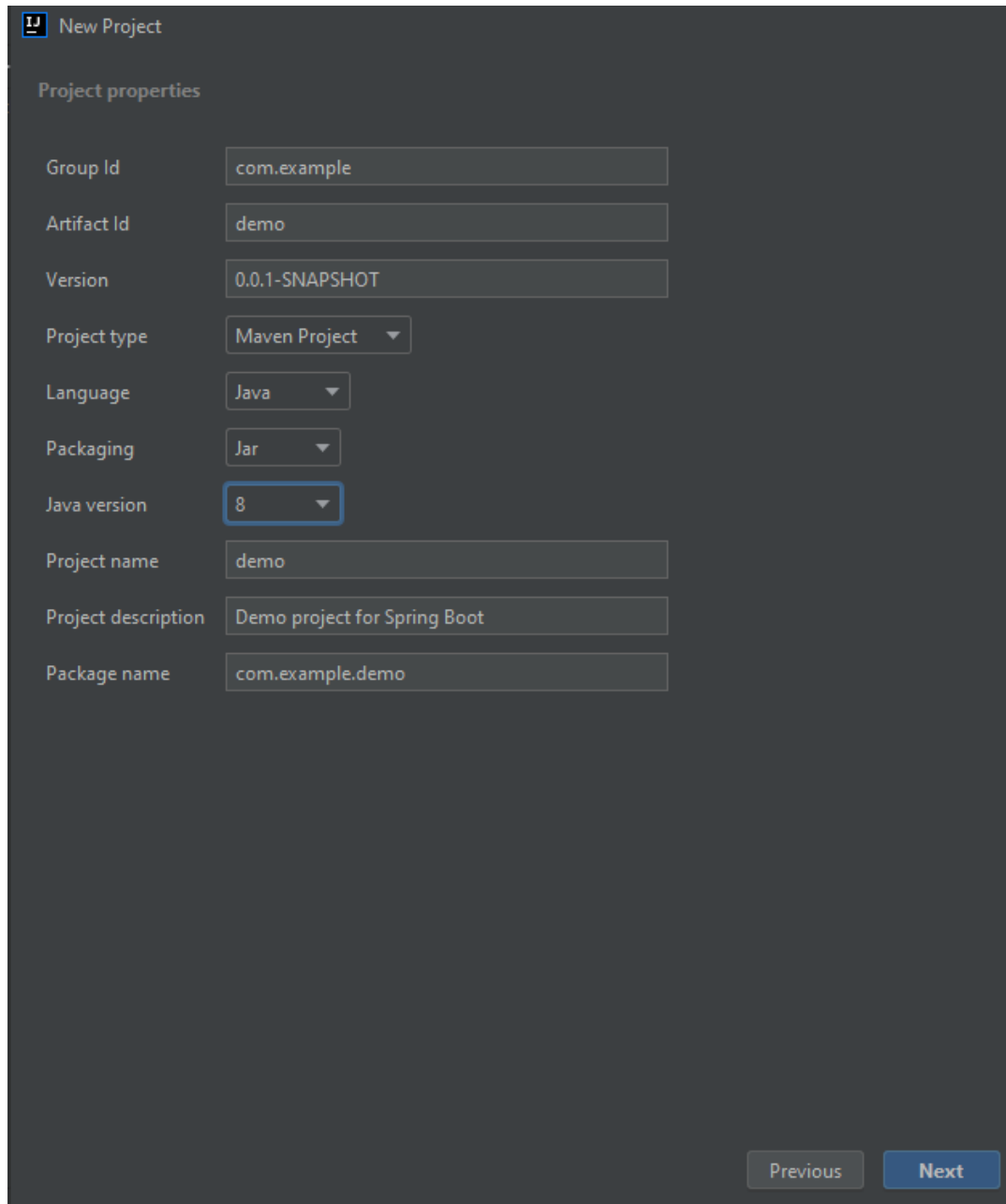
Второй вариант создания проекта на Spring: создать проект в IDE, но сначала нужно установить плагин: File > Settings > Plugins. В строке поиска пишем Spring, находим Spring Assistant и жмем кнопку Install:



По завершении установки перезапускаем IDE.

Теперь создаем проект. File > New Project > Spring Assistant жмем Next.

Далее устанавливаем параметры. Из критичных параметров тут - версия Java должна быть 8:



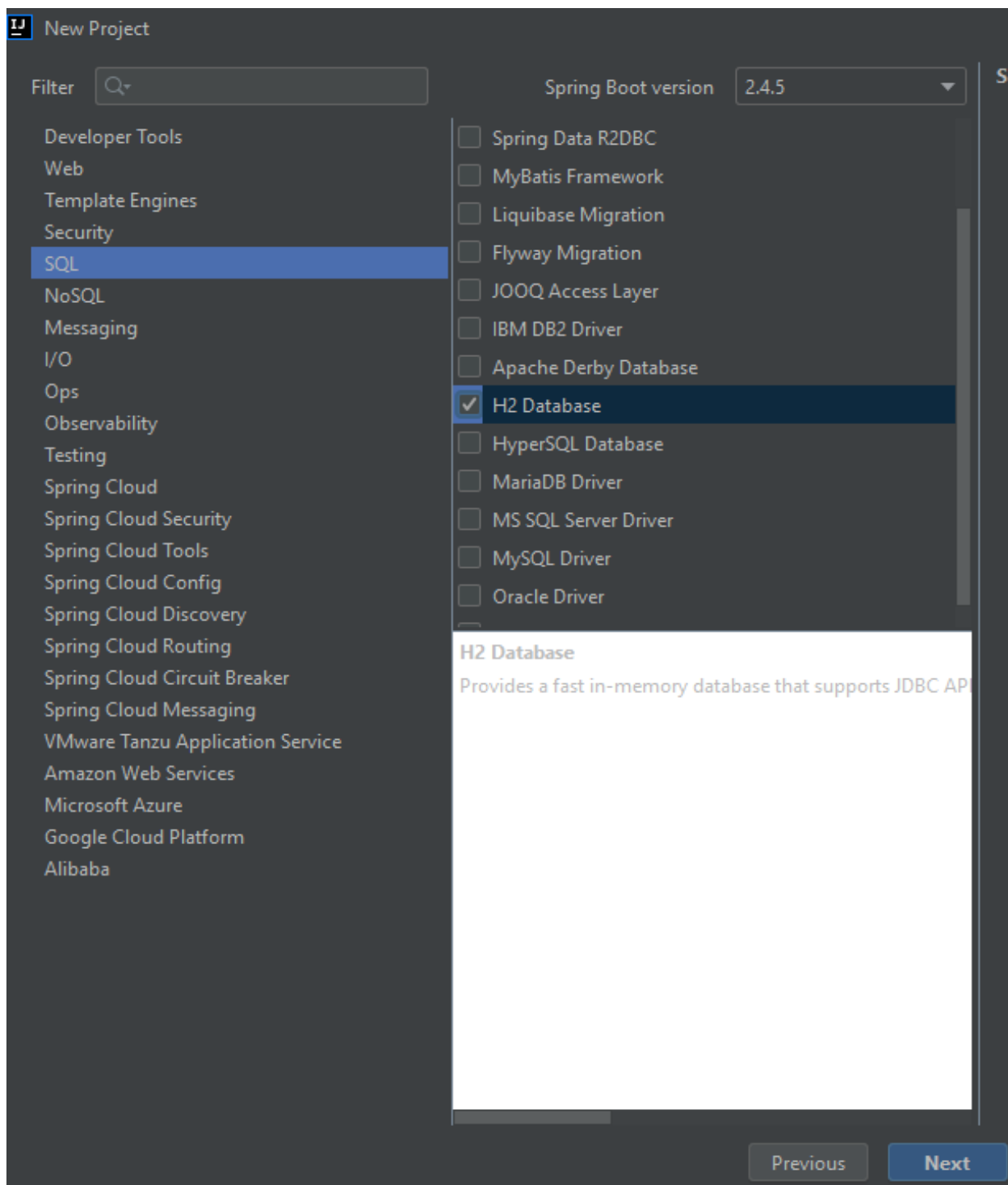
The screenshot shows the 'New Project' dialog box in an IDE. The title bar says 'New Project'. Below it, the section 'Project properties' contains several fields:

- Group Id: com.example
- Artifact Id: demo
- Version: 0.0.1-SNAPSHOT
- Project type: Maven Project (dropdown)
- Language: Java (dropdown)
- Packaging: Jar (dropdown)
- Java version: 8 (dropdown, highlighted with a blue box)
- Project name: demo
- Project description: Demo project for Spring Boot
- Package name: com.example.demo

At the bottom right, there are two buttons: 'Previous' and 'Next'.

Жмем Next.

Нажмите SQL > H2 Database:



Далее жмем кнопки Next до завершения создания проекта.

Теперь откроем файл настроек проекта pom.xml и увидим там следующее:

```
Application.java × pom.xml (demo) ×
<version>0.0.1-SNAPSHOT</version>
<name>demo</name>
<description>Demo project for Spring Boot</description>
<properties>
  <java.version>1.8</java.version>
</properties>
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter</artifactId>
  </dependency>

  <dependency>
    <groupId>com.h2database</groupId>
    <artifactId>h2</artifactId>
    <scope>runtime</scope>
  </dependency>

  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
  </dependency>
</dependencies>
```

Теперь, как Вы поняли, когда мы в графическом интерфейсе при создании проекта установили галочку “H2 Database” никакой магии не произошло, просто в помник добавлены эти строки. То есть можно в принципе никакие зависимости не добавлять, а скопипастить их потом в процессе разработки.

Теперь, каким бы способом Вы ни создали проект, продолжаем добавлять зависимости в помник:

1. Это даст возможность приложению стартовать с поддержкой веб:

```
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-web</artifactId>
</dependency>
```

2. Поддержка персистентности (сохранения данных в базе, в данном случае):

```
<dependency>  
<groupId>org.springframework.boot</groupId>  
<artifactId>spring-boot-starter-data-jpa</artifactId>  
</dependency>
```

3. Дев тулы дают различные инструменты для отладки. В частности, в этом проекте мы будем пользоваться Web-консолью базы H2:

```
<dependency>  
<groupId>org.springframework.boot</groupId>  
<artifactId>spring-boot-devtools</artifactId>  
<scope>runtime</scope>  
<optional>true</optional>  
</dependency>
```

4. Все нижеследующее позволит нам комфортно работать с базой:

```
<dependency>  
<groupId>org.springframework.data</groupId>  
<artifactId>spring-data-commons</artifactId>  
<version>2.4.8</version>  
</dependency>
```

```
<dependency>  
<groupId>org.hibernate</groupId>  
<artifactId>hibernate-core</artifactId>  
<version>5.4.30.Final</version>  
</dependency>
```

```
<dependency>  
<groupId>org.hibernate.validator</groupId>  
<artifactId>hibernate-validator</artifactId>  
<version>6.0.17.Final</version>  
</dependency>
```

```
<dependency>
<groupId>org.hibernate</groupId>
<artifactId>hibernate-entitymanager</artifactId>
<scope>runtime</scope>
</dependency>
```

5. Ну и непосредственно поддержка REST API:

```
<dependency>
<groupId>org.springframework.data</groupId>
<artifactId>spring-data-rest-core</artifactId>
<version>3.4.3</version>
</dependency>
```

Теперь в файл настроек приложения (мы помним, что pom.xml - это файл настроек ПРОЕКТА), а нам нужен файл настроек ПРИЛОЖЕНИЯ – applications.properties. Находится он в SRC > MAIN > RESOURCES. Добавим в applications.properties настройки для БД H2 (Первая строка - без пробелов!!!):

```
spring.datasource.url=jdbc:h2:mem:db;DB_CLOSE_DELAY=-
1;DB_CLOSE_ON_EXIT=FALSE
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
```

Как обычно выглядит CRUD приложение:

1. Класс, описывающий сущность, являющуюся таблицей базы данных. Обычно его именуют как entity или model, например, Entity.java или по имени сущности которая там описана: Person.java, User.java, Product.java и так далее.

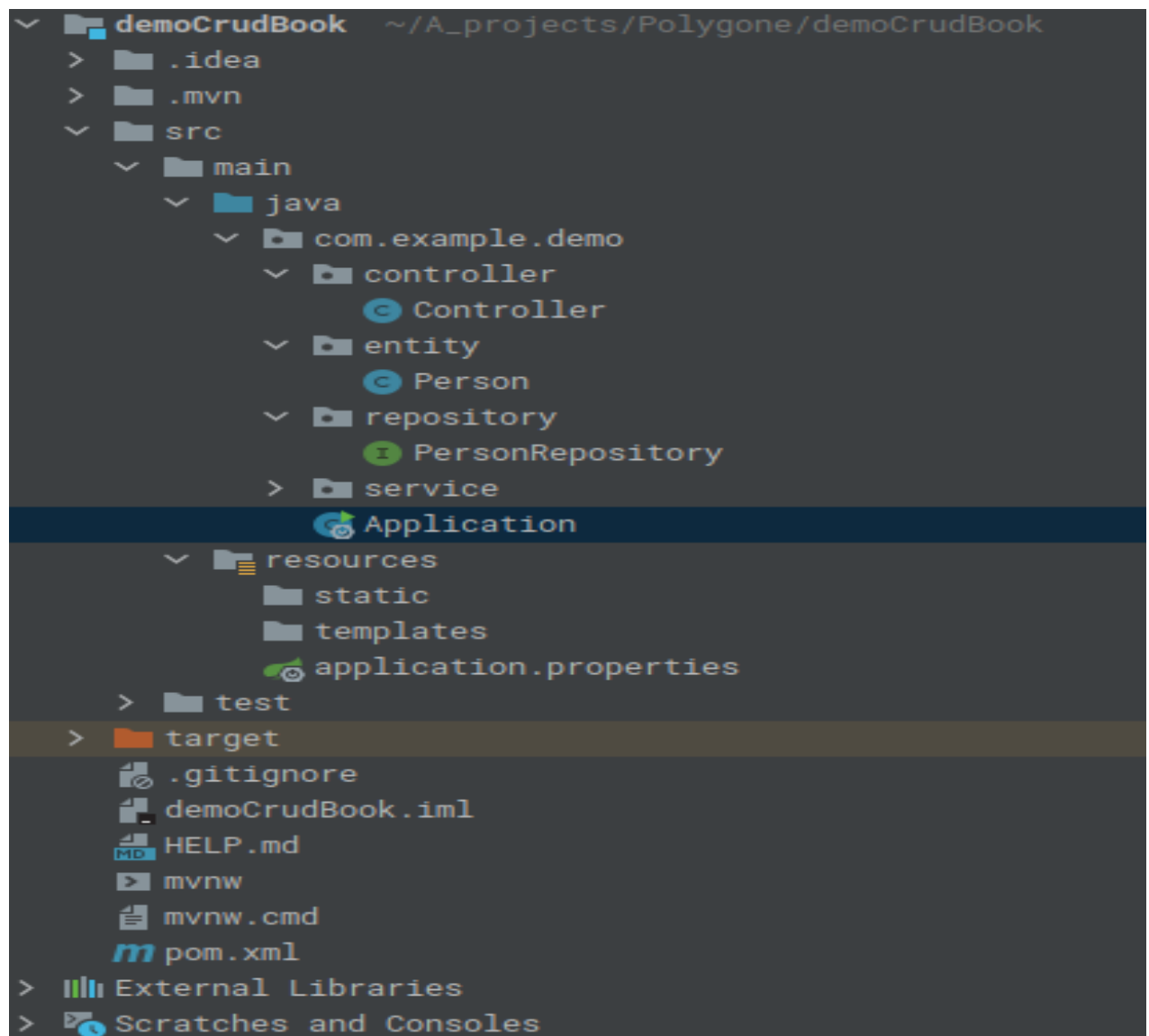
2. Интерфейс, описывающий запросы к базе – его называют <имяСущности>Repository например: PersonRepository.java

3. Класс, реализующий этот интерфейс – service: PersonService.java

4. Контроллер, регламентирующий работу с API – Controller, RESTController и т.д.

И естественно, все классы, кроме Application.java должны лежать в своих пакетах.

Структура нашего приложения будет такой:



1. Создаем пакет entity, а в нем класс Person:

@Entity

public class Person {

@Id

@GeneratedValue

private long id;

private String name;

private String surname;

private int age;

public Person() {}

public Person(long id, String name, String surname, int age) {

this.id = id;

this.name = name;

this.surname = surname;

this.age = age;

}

public long getId() {

return id;

}

public void setId(long id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getSurname() {

return surname;

}


```
public void setSurname(String surname) {  
    this.surname = surname;  
}
```

```
public int getAge() {  
    return age;  
}
```

```
public void setAge(int age) {  
    this.age = age;  
}
```

Все что будет подсвечено красным, разрешаем с помощью Alt + Enter: выбираем Add Maven Dependency и там, близкое по смыслу: если импортируете лишнее страшного ничего не случится, просто дистрибутив будет толще.

2. Создаем пакет repository и в нем интерфейс PersonRepository:

```
@RepositoryRestResource  
public interface PersonRepository extends JpaRepository<Person,  
Long> {  
  
    List<Person> findAll();  
}
```

Создаем пакет service и в нем класс PersonService:

```
@Service  
public class PersonService {  
  
    @Autowired  
    private PersonRepository personRepository;
```

```
public List<Person> findAll() {  
    return personRepository.findAll();  
}
```

```
public Person save(Person person) {  
    return personRepository.save(person);  
}
```

```
public void delete(long id) {  
    personRepository.deleteById(id);  
}  
}
```

3. Создаем пакет controller и в нем класс Controller:

@RestController

```
public class Controller {
```

@Autowired

```
private PersonService personService;
```

@PostMapping("/addPerson")

```
public Person addPerson(@RequestBody Person person) {  
    return personService.save(person);  
}
```

@GetMapping("/persons")

```
public List<Person> findAll() {  
    return personService.findAll();  
}
```

@DeleteMapping("/delete/{id}")

```
public void delete(@PathVariable long id) {  
    personService.delete(id);  
}
```

Теперь наше приложение готово. Запускаем - ждем на класс Application выбираем Run и ждем загрузки приложения:

[illegible]

Что мы тут видим:

- зеленым прямоугольничком выделен номер порта на котором стартовало наше приложение, по умолчанию 8080, в настройках приложения можно задать другой порт;
- фиолетовая колонка не содержит Еггог - значит, все в порядке.

Сейчас у нас база пуста, соответственно увидеть ничего не получится. Нужно заполнить базу. Устанавливаем расширение для браузера Chrom - <https://reqbin.com/> подобных приложений тоже масса я установил это, вполне достаточно. Открываем его и пишем:



1 – Это адрес нашего приложения `http://localhost:8080/addPerson`. Надеюсь тут Вы понимаете, что локалхост это ваш ПК так как у нас нет ДНС (если эти понятия Вам не знакомы – можно не заморачиваться, джуну это не нужно, просто запомните `localhost` – это доменное имя Вашего ПК, по которому можно обратиться к Вашему компьютеру только с Вашего компьютера т.е. локально), `8080` это порт, что мы видели в выводе стартованного приложения, а `/addPerson` ни что иное, как часть адреса, что мы указывали в классе `Controller`.

2 - выбрать тип запроса `POST`. Чтобы понимать в чем разница между типами запросов почитайте спецификацию протокола `http`. Просто вбейте в гугл “протокол `http`” - источников тьма.

3 - Переключитесь на вкладку `Content`.

4 – И введите, то что вы видите на скрине. Это `json` файл. Приложения общаются между собой сообщениями в виде `json` или `xml` файлов, ну Вы теперь-то это знаете.

5 - Нажмите кнопку `Send` и Ваш запрос уйдет в приложение.

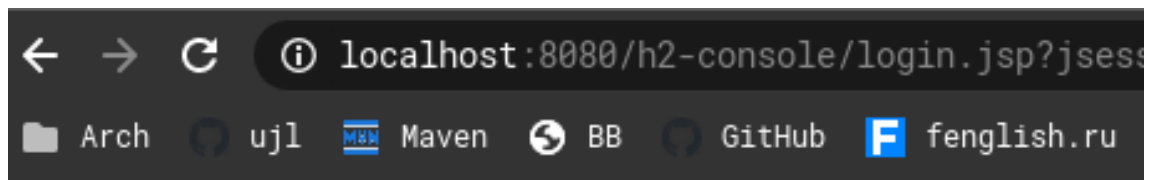
6 - Вы видите результат: `Status 200` говорит о том, что запрос выполнен (это опять спецификация протокола `http`) А ниже вы видите ответ что записи присвоен идентификатор `1` ну и так далее...

Давайте убедимся в этом.

Помните, в классе `Controller` мы также указывали ссылку `/persons` на которой должны выводиться все записи.

Перейдем в браузере по адресу `http://localhost:8080/persons` и увидим нашу запись: `[{"id":1,"name":"John","surname":"Smith","age":35}]`

Теперь давайте заглянем в нашу базу, ее веб-консоль здесь:
`http://localhost:8080/h2-console:`



English ▼ [Preferences](#) [Tools](#) [Help](#)

Login

Saved Settings: Generic H2 (Embedded) ▼

Setting Name: Generic H2 (Embedded) Save Remove

Driver Class: org.h2.Driver

JDBC URL: jdbc:h2:mem:db

User Name: sa

Password:

Connect Test Connection

Нажмем Connect,

затем кликнем по таблице PERSON (слева) и нажмем Run Selected (правее вверху):

jdbc:h2:mem:db

PERSON

INFORMATION_SCHEMA

Sequences

Users

H2 1.4.200 (2019-10-14)

Run

Run Selected

Auto complete

Clear

SQ

SELECT * FROM PERSON

SELECT * FROM PERSON;

ID	AGE	NAME	SURNAME
1	35	John	Smith

(1 row, 5 ms)

Edit

Видим результат выполнения запроса. На данный момент нам это не нужно - мы можем смотреть из браузера все записи как делали в предыдущем шаге. Но когда вы пишете приложение и вдруг база остается пустой - эта консоль очень помогает: вы сразу можете видеть создалась ли для начала таблица в базе если да, то можно посмотреть есть ли в таблице данные. И если их нет, значит дело в коде – попробуйте выполнить Insert в таблицу из этой консоли. Если данные записались в базу: вы их увидели через SELECT здесь же в консоли - дело точно в коде.

Теперь давайте проверим работает ли удаление. Заходим снова в Online REST & SOAP API Testing Tool:



1 - вводим адрес, по которому приложение удаляет, и как помните, оно там ждет id записи которую нужно удалить.

2 - запрос DELETE

3 - жмем кнопку

4 - видим статус 200 значит все ок. Ответа естественно никакого не будет.

Ну и как проверить удалилась ли запись Вы уже знаете.

Итак, наше приложение может Create, Read, Delete. Теперь, чтобы оно было CRUD нам нужно добавить Update и конечно поиск хотя бы по основным полям.

А) Добавляем в интерфейс PersonRepository метод:

Person findPersonById(Long id);

Б) Теперь реализацию метода Update в классе PersonService:

public Person update(Person person) {

```

    Person    existingPerson    =
    personRepository.findById(person.getId());
    existingPerson.setName(person.getName());
    existingPerson.setSurname(person.getSurname());
    existingPerson.setAge(person.getAge());
    return personRepository.save(existingPerson);
}

```

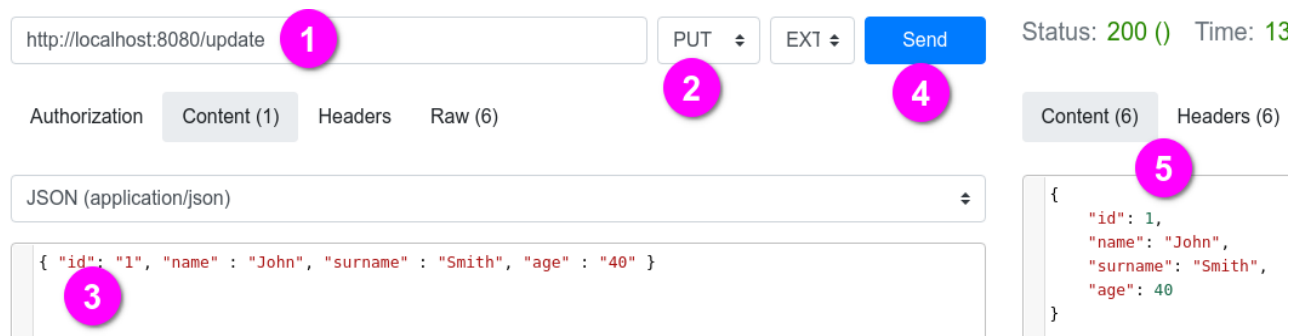
В) И, наконец, добавляем в Controller:

```

@PutMapping("/update")
public Person update(@RequestBody Person person) {
    return personService.update(person);
}

```

Проверяем, что работает. Перезапускаем приложение. Как вы помните, H2 база которая по умолчанию работает в оперативной памяти компьютера - она пустая сейчас. Поэтому сначала в Online REST & SOAP API Testing Tool создадим запись, как мы делали ранее. А теперь попробуем ее проапдейтить:



- 1 - путь у нас теперь update
 - 2 - тип запроса PUT
 - 3 – поскольку мы правим существующую запись, добавляем ее ключ "id": "1" и меняем любой параметр, я накинул Смиту годков до 40.
 - 4 - жмем Send.
 - 5 - любимся результатом.
- Ну давайте еще добавим, например, поиск по имени:

А) Добавляем в интерфейс PersonRepository:

```
Person findPersonByName(String name);
```

Б) Теперь реализацию в классе PersonService:

```
public Person findPersonByName(String name) {  
    return personRepository.findPersonByName(name);  
}
```

В) И, наконец, добавляем в Controller:

```
@GetMapping("/persons/{name}")  
public Person findPersonByName(@PathVariable String name) {  
    return personService.findPersonByName(name);  
}
```

Проверяем. Протокол подберите самостоятельно. Для закрепления темы добавьте поиск по фамилии и возрасту.

Код этого проекта находится здесь:

https://github.com/alexeynovosibirsk/Book_CRUD

11.1. Переключаем базу данных.

Чтобы проект можно было использовать в качестве демо нужно переключить базу данных на Postgres, поскольку как Вы помните – это наиболее востребованный сервер баз данных. Делается это так:

А) Идем в pom.xml и удаляем (или комментируем) зависимость. Можно выделить и нажать на клавиатуре Ctrl + / H2:

```
<dependency>  
<groupId>com.h2database</groupId>
```



```
<artifactId>h2</artifactId>  
<scope>runtime</scope>  
</dependency>
```

Б) Идем в репозиторий мавена ищем там JDBC драйвер PostgreSQL и вставляем в помник. Мы это уже делали в прошлой главе – 1.1. Загрузка драйвера JDBC.

В) Идем в файл конфигурации приложения - application.properties и комментируем все содержимое. (выделить и нажать на клавиатуре Ctrl + /), а добавляем в файл настройки PostgreSQL:

```
spring.datasource.url=jdbc:postgresql://localhost:5432/postgres  
spring.datasource.username=postgres  
spring.datasource.password=123  
spring.datasource.driver-class-name=org.postgresql.Driver  
spring.jpa.database=postgresql  
spring.jpa.database-  
platform=org.hibernate.dialect.PostgreSQL10Dialect
```

Проанализируйте, какие настройки были, и какие Вы добавили сейчас.

В отличие от H2 остальные серверы баз данных будут возвращать Вам ошибку (500), если вы попытаетесь выполнить запрос не создав таблицу. Поэтому стартуем базу PostgreSQL как мы это делали во второй части этой книги. Старт базы данных, затем открываем pgAdmin4 и (также как во второй части книги) создаем таблицу:

```
CREATE TABLE person (  
  Id BIGSERIAL PRIMARY KEY,  
  name VARCHAR(100) NOT NULL,  
  surname VARCHAR(100) NOT NULL,  
  age VARCHAR(100) NOT NULL)
```

Выделяем, выполняем. (Это тоже описано во второй части).

Не обязательный шаг, но в данном случае его придется сделать:

Идем в класс Person и к аннотации **@GeneratedValue** добавляем **(strategy = GenerationType.IDENTITY)** то есть теперь аннотация выглядит так:

@GeneratedValue(strategy = GenerationType.IDENTITY)

Эта аннотация указывает базе предоставлять id самостоятельно, однако в каждом сервере баз данных на этот счет свои настройки и несколько стратегий, я однажды потерял кучу времени на этом. Прогуглите этот вопрос - чтобы не наступать на такие грабли.

Теперь приложение будет работать с PostgreSQL. Чтобы в этом убедиться, запустите приложения и поэкспериментируйте в Online REST & SOAP API Testing Tool как мы делали это с базой H2.

Глава 12 Демо CRUD с Web-интерфейсом

Создавать проект мы будем с использованием фреймворка Vaadin, который нам обеспечит графический интерфейс. Это самый легкий в плане освоения фреймворк. Он правда иногда бывает капризным в установке, но это компенсируется его возможностями.

12.1. Создаем приложение

Открываем IDE и создаем новый проект и выбираем Spring Assistant, далее:

- Spring Boot version 2.3.12(SNAPSHOT),
- Spring Boot DevTools, SQL > H2 Database.

Жмем далее, финиш и ждем пока проект подтянет все зависимости.

1. Добавляем необходимые зависимости. Добавляем в pom.xml версию Vaadin (третья строка, все остальное там уже есть):

```
<properties>  
<java.version>1.8</java.version>  
<vaadin.version>14.6.1</vaadin.version>
```

</properties>

Это мы сделали для того, чтобы не указывать каждый раз версию фреймворка. Кстати подобное можно делать для любых зависимостей. далее после тега **<dependencies>**:

```
<dependency>
<groupId>com.vaadin</groupId>
<artifactId>vaadin-spring-boot-starter</artifactId>
</dependency>
```

```
<dependency>
<groupId>org.vaadin.crudui</groupId>
<artifactId>crudui</artifactId>
<version>4.4.1</version>
</dependency>
```

Сразу за закрывающим тегом **</dependencies>** вставляем:

```
<dependencyManagement>
<dependencies>
<dependency>
<groupId>com.vaadin</groupId>
<artifactId>vaadin-bom</artifactId>
<version>${vaadin.version}</version>
<type>pom</type>
<scope>import</scope>
</dependency>
</dependencies>
</dependencyManagement>
```

Теперь в раздел **<repositories>**:

```
<repository>
<id>vaadin-addons</id>
<url>https://maven.vaadin.com/vaadin-addons</url>
```

</repository>

Нажимаем обновить (маленький синий значок справа-вверху).

Наше приложение будет аналогично предыдущему, за исключением того, что у нас не будет REST API, а будет графический интерфейс, точнее говоря веб-интерфейс (web-UI).

2. Создаем класс Person. Обратите внимание на класс нужно повесить аннотацию **@Entity** которая скажет спрингу, что данный класс представляет сущность в базе данных:

@Entity

public class Person {

@Id

@GeneratedValue

private Long id;

private String firstName;

private String lastName;

public Person() {}

public Person(Long id, String firstName, String lastName) {

this.id = id;

this.firstName = firstName;

this.lastName = lastName;

}

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

```
public String getFirstName() {  
    return firstName;  
}
```

```
public void setFirstName(String firstName) {  
    this.firstName = firstName;  
}
```

```
public String getLastName() {  
    return lastName;  
}
```

```
public void setLastName(String lastName) {  
    this.lastName = lastName;  
}
```

Обратите внимание, что в классе создаются 2 конструктора: 1 - по умолчанию (без параметров) и второй со всеми параметрами. Чтобы это не писать руками можно воспользоваться кодогенерацией. Для этого, установите курсор на пустую строку где должен будет появиться код и в меню выберите Code > Generate, в следующем меню выберите конструктор и обратите внимание, что чуть ниже можно выделять поля класса, чтобы включить их в качестве параметров конструктора. Вообще здесь стоит немного остановиться и изучить какие варианты кодогенерации предоставляет IDE.

Также (в меню кодогенерации) создайте гетеры и сетеры.

3. Создаем интерфейс PersonRepository:

```
@Repository  
public interface PersonRepository extends JpaRepository<Person,  
Long> {  
}
```

4. Создаем класс PersonService (реализацию интерфейса PersonRepository):

@Service

```
public class PersonService implements CrudListener<Person> {
```

```
    private final PersonRepository personRepository;
```

```
    public PersonService(PersonRepository personRepository) {  
        this.personRepository = personRepository;  
    }
```

@Override

```
    public Collection<Person> findAll() {  
        return personRepository.findAll();  
    }
```

@Override

```
    public Person add(Person person) {  
        return personRepository.save(person);  
    }
```

@Override

```
    public Person update(Person person) {  
        return personRepository.save(person);  
    }
```

@Override

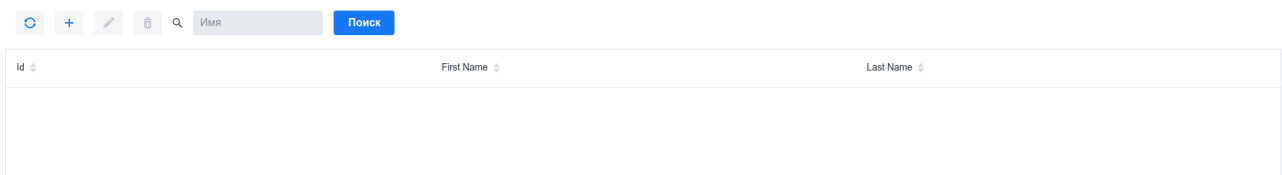
```
    public void delete(Person person) {  
        personRepository.delete(person);  
    }  
}
```

Не забываем при помощи Alt+Enter разрезолвивать то, что подсвечивается красным.

5. Создаем класс PersonView - нашу вьюшку которая нам будет показывать графику:

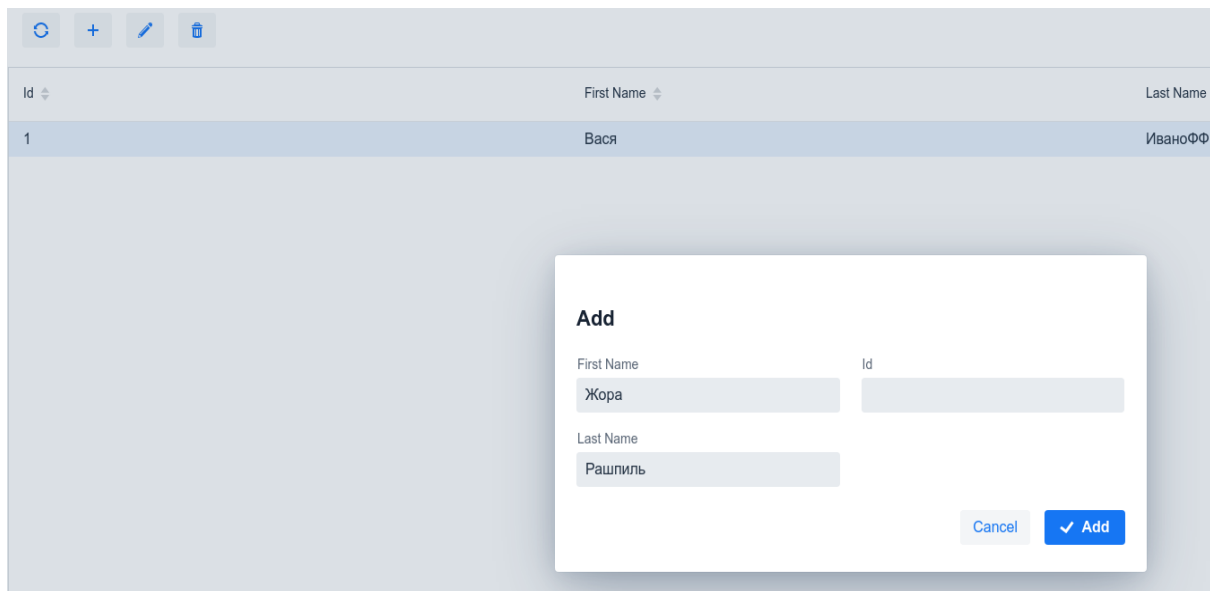
```
@Route("")  
public class PersonView extends VerticalLayout {  
    public PersonView(PersonService personService) {  
        GridCrud<Person> crud = new GridCrud<>(Person.class,  
personService);  
        add(crud);  
        setSizeFull();  
    }  
}
```

Обновите и запустите приложение. Будут предупреждения красным шрифтом от прт - игнорируйте. Идем по адресу приложения <http://localhost:8080> и любуемся:



The screenshot shows a web application interface. At the top, there is a search bar with a magnifying glass icon and a blue button labeled "Поиск". Below the search bar is a table with three columns: "Id", "First Name", and "Last Name". The table is currently empty.

Понятно, что база пустая - нужно создать записи, воспользуйтесь кнопкой +. Id можно не указывать, но даже если Вы его укажете, база проигнорирует это поле и сгенерирует следующий id.



Вообще, изучите все кнопки интерфейса, заведите несколько записей, попробуйте редактировать и т.д. Уже похоже на базу данных как ее представляют пользователи, правда?

Не хватает только фильтров поиска. Давайте создадим один, остальные Вы сможете сделать самостоятельно.

12.2. Создание фильтра поиска

1. Добавляем метод в PersonRepository:

```
List<Person> findAllByFirstName(String firstName);
```

2. Добавляем реализацию метода в класс PersonService:

```
public List<Person> findAllByFirstName(String firstName) {  
    return personRepository.findAllByFirstName(firstName);  
}
```

3. Ну и теперь нам нужно дергать этот метод из UI. Идем в класс PersonView, создаем фильтр (поле куда будем вводить имя):

```
TextField textFieldName = new TextField();  
textFieldName.setClearButtonVisible(true);
```



```
textFieldName.setPlaceholder("Имя");
```

4. Далее нам нужно сделать кнопку чтобы применять поиск:

```
Button buttonFind = new Button("Поиск");  
buttonFind.addClickShortcut(Key.ENTER);  
buttonFind.addThemeVariants(ButtonVariant.LUMO_PRIMARY);
```

Ну и заодно, во второй строке мы говорим, что нажатие на клавиатуре клавиши Enter равнозначно нажатию кнопки. Третья строчка просто делают кнопку более яркой.

Пока у нас только визуальные элементы, реально как Вы знаете в джава все делается методами. И у кнопки есть такой метод называется ClickListener.

4.1. Реализуем этот метод:

```
buttonFind.addClickListener(e -> {  
crud.getGrid()  
.setItems(personService.findAllByFirstName(textFieldName.getValue(  
)));  
});
```

Как видите сам метод ничего не делает, он вызывает в третьей строке наш метод, что мы реализовывали на шаге 1 и 2, а в качестве параметра ему передается значение, которое содержит фильтр `textFieldName.getValue()`

4.2. Ну и конечно, теперь нужно сказать Vaadin что мы хотим видеть добавленные элементы:

```
crud.getCrudLayout().addFilterComponents(textFieldName,  
buttonFind);
```

После внесения всех изменений класс `PersonView` должен выглядеть так:

```

@Route("")
public class PersonView extends VerticalLayout {

    public PersonView(PersonService personService) {

        GridCrud<Person> crud = new GridCrud<>(Person.class,
personService);
        crud.getGrid().setColumns("id", "firstName", "lastName");

        TextField textFieldName = new TextField();
        textFieldName.setClearButtonVisible(true);
        textFieldName.setPlaceholder("Имя");

        Button buttonFind = new Button("Поиск");
        buttonFind.addClickShortcut(Key.ENTER);
        buttonFind.addThemeVariants(ButtonVariant.LUMO_PRIMARY);

        buttonFind.addClickListener(e -> {
            crud.getGrid().setItems(personService.findAllByFirstName(textFieldN
ame.getValue()));
        });

        add(crud);
        crud.getCrudLayout().addFilterComponents(textFieldName,
buttonFind);
        setSizeFull();
    }
}

```

Давайте убедимся, что все работает. Запустим приложение, откроем
нужный линк в браузере и создадим 3 записи, например

1. One One,
2. Two One,

3. Two Two:

Поиск

Id	First Name	Last Name
1	One	One
2	Two	One
3	One	Two

Теперь введем в фильтре имя Two и нажмем кнопку:

Two

Поиск

Id	First Name	Last Name
2	Two	One

Ура, все работает. Чтобы окончательно убедиться попробуйте задавать другие значения для поиска. Ну и конечно попробуйте создать фильтр поиска по фамилии.

Полный код приложения здесь

https://github.com/alexeynovosibirsk/Book_CRUDUI

Теперь нужно причесать наше приложение. Во-первых, все классы, кроме основного, запускающего нужно распахать по одноименным пакетам.

Во-вторых, как мы уже раньше делали, переключите приложение на сервер баз данных PostgreSQL, не забудьте добавить в нем соответствующую таблицу, а в классе сущности - стратегию генерации первичных ключей.

Хорошо бы добавить еще фильтр поиска по фамилии, если Вы не сделали этого ранее. Вообще, конечно, лучше бы создать таблицу поинтереснее, например, сотрудников, с годом рождения, зарплатой и/или стажем или таблицу с товарами их количеством, ценой и т.п. и реализовать поиск по всем этим полям.

Также нужно реализовать аутентификацию. Как это сделать можете посмотреть здесь <https://www.youtube.com/watch?v=pteip-kZm4M>

И затем соединить два проекта. Кстати, слишком залипать в Vaadin не стоит - вакансий на рынке не очень много, Angular и прочее менее удобные, но более старые фреймворки его обгоняют по востребованности. Я показал Вам Vaadin потому, что тут можно по-быстрому, если Вам требуется приложение с UI.

Глава 13. Как получить опыт и как написать пет-проект.

Теперь Вы можете попробовать получить опыт, который можно будет указать в резюме как рабочий. Для этого попробуйте поучаствовать в опенсорс проекте. Как это делать можно прочитать, например, здесь:

<https://habr.com/ru/post/347066/>

<https://javarush.ru/groups/posts/2507-kachaem-skillih-open-source-proektih-na-github-dlja-nachinajushikh-dzhavistov>

<https://tproger.ru/articles/open-source-for-beginners/>

Это конечно не обязательно, но, если получится - даст Вам опыт и будет что написать в резюме. Работодатели говорят, что это круто. Я лично этого не делал просто по причине нехватки времени - я пилил свой пэт-проект и готовился к собеседованиям. Так случилось, что прошел собеседование и устроился на работу - необходимость участия в опенсорсном проекте временно отпала.

А свой пэт-проект я не бросил: до сих пор в него коммичу.

14.1. Как написать свой пэт-проект

Чтобы писать свой пэт-проект нужно придумать его, подумайте над тем, что Вам интересно. Если Вы хотите стать программистом, значит Вы хотите писать программы. И Вы наверняка были недовольны программами, которыми пользовались: Вам хотелось сделать их удобнее. Вот и настало время это сделать.

Я сначала побаловался написанием блокнота для записочек, который бы открывался в браузере и сохранял записочки в определенную директорию. Затем я решил попробовать написать радио-плеер. Я люблю слушать музыку и мне хотелось, чтобы в плеере были только мои любимые жанры и любимые радиостанции. Еще мне хотелось, чтобы плеер при переключении станций и запуске приложения разговаривал голосом робота. Я все это реализовал. Правда голос робота пришлось выкинуть - очень уж неудобно было, когда его голос перебивал музыку. Теперь плеер только здоровается, когда загружается и говорит что урл недоступен если там нет аудиопотока.

Если Вам интересно можете посмотреть его здесь:
https://github.com/alexeynovosibirsk/PetProject_RadioPlayer

Так, что нужно подумать, что за приложение Вам хотелось бы реализовать - оно должно Вас увлекать, ведь не зря же это называется пэт-проект. Если совсем нет идей - возьмите любое приложение и повторите его функционал или часть функционала, возможно в процессе реализации Вы увлечетесь или поймете, что хотите делать что-то другое. Главное не комплексовать - Ваших знаний уже достаточно.

Нужно сделать декомпозицию задачи или как говорят в тайм-менеджменте "разрезать слона на стейки". То есть идею распланировать по частям на простые задачи. Например, я делал так:

1. Попробовать в Java воспроизвести потоковый аудиострим. Когда я за это брался я понятия не имел как это делать. Впрочем, как и делать все другое.
2. Затем, прочитать из файла адрес аудиострима и воспроизвести его.
3. Добавить валидацию - проверку, что считанное из файла действительно веб-ссылка.
4. Игнорировать пустые строки в файле плейлиста.
5. Реализовать остановку воспроизведения.
6. Реализовать веб-интерфейс, через который можно останавливать воспроизведение.
7. Реализовать запуск воспроизведения и вывести соответствующую кнопку в пользовательском интерфейсе.

8. Реализовать переключение между плейлистами.

Ну и так далее. Думаю, примерно алгоритм Вам ясен: сначала реализуем главную функцию, для чего создаем приложение, а затем шаг за шагом добавляем функциональность и плюшки. И особенно важно здесь не забывать пушить в Git после того, как новый функционал заработал. Потому, что в процессе адского гугления забываешь, где что поменял, а оно раз - и не работает! А если Вы были молодец и запушили после удачной реализованной фичи, Вы просто выкидываете рабочий каталог - делаете пулл из репы и вуаля! Все снова работает!

Самое главное, это понимать, что Вы хотите добиться следующей фичей и гуглить. Поначалу ответы будут не слишком релевантные. Но чем больше будет расти Ваша экспертиза в Java, тем грамотней будут Ваши запросы и соответственно точнее поисковые выдачи.

В общем не отчаивайтесь и пробуйте что-нибудь реализовать, ведь программировать это интересно! Вы об этом мечтали и теперь это умеете! И чем больше Вы будете это делать - тем лучше у Вас это будет получаться!

А как написать какой-то функционал на Java, кроме просто гуглежа можно посмотреть на этих ресурсах:

<https://www.baeldung.com/>

<https://mkyong.com/>

<http://tutorials.jenkov.com/>

<https://stackoverflow.com/>

И теперь Вы готовы приступить к поиску работы.