ПАВЕЛ АЛЕКСАНДРОВИЧ
ЗАБЕЛИН

JAVA 2021: ЛЕГКИЙ СТАРТ

Павел Забелин

JAVA 2021: лёгкий старт

«Издательские решения»

Забелин П. А.

JAVA 2021: лёгкий старт / П. А. Забелин — «Издательские решения»,

ISBN 978-5-00-515483-5

Главная цель этой книги — показать читателю, что программирование на Java, гораздо более проще, чем принято об этом думать. Как известно «хочешь лучше понять сам — расскажи об этом другому», что я и попытался сделать на страницах этой книге в меру своих сил и времени. Эта книга как раз вам поможет обрести базовые знания программирования и языка программирования Java, и избавит вас от проблем с пониманием основ программирования.

ISBN 978-5-00-515483-5

© Забелин П. А.
© Издательские решения

Содержание

JAVA 2021: лёгкий старт

Павел Александрович Забелин

Посвящается моей музе Кицунэ ЗЛА

© Павел Александрович Забелин, 2020

ISBN 978-5-0051-5483-5
Создано в интеллектуальной издательской системе Ridero

Введение

Главная цель этой книги – показать читателю, что программирование на Java, гораздо более проще, чем принято об этом думать. Имея за плечами опыт программирования больше 15 лет, я относительно недавно увлекся программированием на Java и эта книга – неоконченная «сжатая» история самообучения. Как известно «хочешь лучше понять сам – расскажи об этом другому», что я и попытался сделать на страницах этой книге в меру своих сил и времени. Я прочитал несколько книг и прошел несколько курсов в интернете: в университете Skillbox, Udemy, Stepik, что и вам советую. Но прежде чем купить какие-либо курсы и начать их проходить, я рекомендую прочесть эту книгу: зачастую курсы грешат провалами в теории и скачками сложности преподаваемого материала, да и сложно определить начальный уровень подготовки студента. Эта книга как раз вам поможет обрести базовые знания программирования и языка программирования Java, и избавит вас от проблем с пониманием основ программирования.

Для кого эта книга?

Эта книга для любого, кто хочет научиться программировать. Программирование только лишь окутано завесой чего-то очень сложного. На своем пути я видел людей абсолютно различных профессий (мало относящихся к компьютерам), которые успешно освоили программирование. Программирование – это очень широкая область деятельности, которая позволяет проявить разнообразные способности и умения. К тому же «побочные эффекты» профессионального программирования, такие как возможность работы без привязки к месту жительства и достойная оплата труда, которая позволит проживать практически в любом уголке планеты, еще больше мотивируют попробовать погрузиться в мир IT. Не говоря уже о том, что человеческая цивилизация чем дальше, тем больше уходит в «цифровые миры» и возможность быть не только пользователем, но и создателем программ – это очень интересно.

Все, что вам потребуется для успешного прочтения этой книги и продуктивного усваивания материала, это: в первую очередь желание и намерение не сдаваться перед трудностями. Второе: компьютер с операционной системой Windows, в 2020 году вам подойдет компьютер из любой ценовой категории – Java без проблем работает на любых процессорах семейства Intel, ну и конечно лучше если у вашего компьютера будет хотя бы 4 гигабайта оперативной памяти. Конечно же вы можете использовать компьютер на базе ОС Linux или MacOS, только вам придется самостоятельно установить среду разработки JetBrains IntelliJ IDEA (это будет единственное отличие).

От вас не требуется каких-то особых познаний в информатике или программировании, все необходимое для того, чтобы начать программировать я и так вам расскажу в этой книге.

> Да и вам не нужно знать высшую математику – это ответ на вопрос из топа страшных вопросов «что должен знать программист ДО того как станет писать код». То есть достаточно знать математику в пределах простейших математических операций и умения раскрыть скобки, ну и решать уравнения с одной переменной.

Большим плюсом будет умение читать и понимать английский язык, так исторически сложилось, что все в интернете найти ответ по программированию легче всего на английском языке. Плюс, также все новые и интересные книги и статьи публикуются на английском языке. Обратите на это внимание. При устройстве на работу знание английского также будет плюсом.

Почему Java?

Небольшой обзор текущего положения дел в языках программирования. Существует всемирный рейтинг языков программирования TIOBE (https://www.tiobe.com/tiobe-index/):

Dec 2019	Dec 2018	Change	Programming Language	Ratings	Change
1	1		Java	17.253%	+1.32%
2	2		C	16.086%	+1.80%
3	3		Python	10.308%	+1.93%
4	4		C++	6.196%	-1.37%
5	6	^	C#	4.801%	+1.35%
6	5	v	Visual Basic .NET	4.743%	-2.38%
7	7		JavaScript	2.090%	-0.97%
8	8		PHP	2.048%	-0.39%
9	9		SQL	1.843%	-0.34%
10	14	^^	Swift	1.490%	+0.27%
11	17	^^	Ruby	1.314%	+0.21%
12	11	v	Delphi/Object Pascal	1.280%	-0.12%
13	10	v	Objective-C	1.204%	-0.27%
14	12	v	Assembly language	1.067%	-0.30%
15	15		Go	0.995%	-0.19%
16	16		R	0.995%	-0.12%
17	13	vv	MATLAB	0.986%	-0.30%
18	25	^^	D	0.930%	+0.42%
19	19		Visual Basic	0.929%	-0.05%
20	18	v	Perl	0.899%	-0.11%

И здесь мы видим, практически неизменную пятерку лидеров: Java, C, Python, C++, C#. Но этот рейтинг имеет такое же отношение к реальности, как и прогноз погоды на неделю. Сейчас я расскажу про языки из топ-10, чтобы вы смогли попробовать оценить свои представления и желания в области программирования.

Стоит сразу сказать, что языки в рейтинге, с местами 10+ не очень подходят для начального обучения, ну кроме Ruby. Но они также востребованы, и лучше, как минимум на них посмотреть – может вам понравится. А может быть столкнетесь с ними, и жизнь заставит вас выучить какой-либо из них.

10-е место Swift. Детище Apple, замена ObjectiveC для платформы MacOS\iOS (равно как Kotlin, замена Java на Android). Если вы хотите связать свою профессиональную деятельность с корпорацией Apple, то стоит начать его учить и не отвлекаться ни на что более. Он простой для изучения и даже есть курсы для детей. К тому же, на сегодняшний день специалистов со знанием его не так много.

9-е место SQL. Этот язык знать **обязательно,** потому что это язык «общения» с базами данных, сейчас ни одно приложение или сайт не может существовать без баз данных. Но учить его как первый язык смысла не имеет.

8-место PHP. «Домашний проект» датского программиста Расмуса Лердорфа, переросший в самый востребованный язык программирования сайтов в интернете. 80% сайтов используют PHP. Но это может быть и минусом – на PHP вы только сможете писать серверную часть (известны попытки писать приложения на нем, но это только в качестве экспериментов). PHP достаточно прост в изучении, и что может быть для некоторых решающим фактором, специалисты очень востребованы в многочисленных веб-студиях. Т.е. для цели: быстро изучить и пойти «зарабатывать, чтобы на жизнь хватало» – это язык номер 1.

7-е место JavaScript. Очень долгая история у этого языка программирования, можно сказать, что он появился вместе с интернетом (когда интернет стал доступным для массового использования). Но только в последние несколько лет он стал суперпопулярным. Для этого есть несколько причин: он стал удобным для написания больших проектов, кроме написания простых скриптов «чтобы появлялось красивое окошко» теперь на нем можно писать практически все – клиентские приложения для iOS, MacOS, Android, Windows (фреймворк Electron), серверные приложения (фреймворк NodeJS). Он очень подходит для людей, которым нужна «движуха»: идеален для написания проектов на хакатонах, новые фреймворки (библиотеки программного кода, очень облегчающие жизнь программиста и делающие очень много черной работы) появляются каждый год – с ним не бывает скучно! Специалисты очень востребованы, можно сказать что «через 20 лет будет только JavaScript».

6-место VisualBasic.NET. Скажу честно: я не знаю почему он не только в топ-10, но и вообще почему он здесь. Единственное могу предположить, что до сих пор на нем пишут макросы для MS Office, ну и может быть в Америке есть много приложений которые до сих пор требуют поддержки и обновления.

5-е место C#. Основной язык программирования проприетарной (закрытой) платформы Microsoft. NET. Великолепный язык, постоянно развивается и будет развиваться, пока в него инвестирует деньги Microsoft. На нем можно писать все что угодно, только с одним ограничением: это все может работать там, где установлена Windows или работает виртуальная машина. NET и в отличии от Java, диапазон гораздо уже (пока что). С мобильными приложениями дела обстоят совсем печально – есть проект Xamarin, который использовать никто не советует. Но также есть игровой движок Unity, в котором используется C#, но это только для игр на большом количестве платформ. C# особенно любим на территории СНГ, т.к. любима Windows. Если вы хотите спокойно развиваться как программист, иметь поддержку крупной корпорации, никогда задаваться вопросом стабильной зарплаты – C# очень хорошо подходит для этого.

4-е место C++. Самый крутой язык программирования и также круто сложен. Писать можно все и подо все. Особенно востребован там, где требуется скорость исполнения кода: игры, мобильные игры, сервера. Если вы знаете C++, то выучить что-либо еще перестает быть проблемой. Еще раз повторю: самый сложный язык из нормальных (да-да есть еще и *ненормальные*, например, Brainfuck) и не сильно подходит как первый язык для изучения программирования, хотя в ВУЗах учебные программы могут начинаться именно с него.

3-е место Python. **Самый простой и лучший язык для изучения программирования**. Но, есть одно «НО»: на нем либо писать сайты (Flask\Django), либо заниматься научными исследованиями в области искусственного интеллекта, big data, для которых нужны очень математические мозги в первую очередь. Стоит учесть, что простота его изучения довольно опасная вещь – студент в процессе обучения не видит, ЧТО реально стоит за многочисленными библиотеками (а они уже написаны на C и там все сложно).

2-е место C. Честно скажу не знаю почему, наверное, проектов на нем больше чем на C ++. Он быстрее C++, но в нем приходиться писать более сложно.

1-е место Java. Но это не точно :). Просто языки в первой тройке любят меняться местами. Но раз книга про Java… Если вы прочитали и запомнили характеристики предыдущих языков, то можете сказать: а почему Java? Она ведь и не самая быстрая, и не самая про-

стая в изучении, и в ней нет постоянных изменений и улучшений как в других языках. Дело в том, что Java это не только язык программирования, а это целая платформа (да C# – это тоже платформа, только меньше по охвату), которая позволяет вам писать программы для практически любых устройств: начиная от кофемолок, заканчивая огромными дата центрами. Java использует большие корпорации, например Yandex и Google. Она надежна и безопасна, на ней пишут большие корпоративные системы. С одной стороны код написанный на Java выглядит многословным, но это же позволяет избежать критических и трудно заметных ошибок. Существуют тысячи библиотек с открытым кодом, которые можно использовать в своих проектах. На Java вы можете писать приложения для Android, и поверьте это легче чем писать под iOS (на Objective C). И самое главное: это лучший язык для того, чтобы научиться писать правильные объектно-ориентированный код, с использованием хороших паттернов программирования. Хотите научиться писать красивый и понятный другим людям код (а это очень важное умение) – пишите его на Java. Огромным преимуществом Java является то, что этому языку много лет и за все эти годы в интернете накопилась огромная база знаний и ответов на многие проблемы, с которыми сталкиваются программисты на Java. И поэтому зачастую проще вбить запрос в Yandex и получить готовое решение, которое будет легче адаптировать к своей программе, чем «изобретать велосипед»

Что все-таки выбрать?

Здесь я позволю себе посмотреть на проблему выбора языка программирования с точки зрения требований рынка. И рынок диктует свои жесткие требования, и они растут из года в год. Первое десятилетие 21 века было «золотым веком» для того чтобы выбрать язык программирования по-вкусу: достаточно было выучить *только его*, и вы уже могли идти и трудоустраиваться. В 2020 году все намного сложнее – от вас будут требовать гораздо больших знаний даже на позиции джуниора. Вот что вам потребуется знать «в довесок» и чем вы будете заниматься, если выберете какой-либо из языков топ-10.

Swift. Вы будете писать приложения под iPhone\iPad. В основном это будут приложения с многочисленными окошками. Периодически от вас будут требовать написать что-либо на Objective C, потому что Swift еще слишком молод и нестабилен, чтобы были проекты только на нем.

PHP. Вы будете все время писать сайты, и это в лучшем случае. А скорее всего вы будете настраивать WordPress\Drupal\Joomla под требования заказчиков сайтов. Да есть маленькая вероятность, что вы найдете работу где надо будет писать сервер на PHP (т.е. не сайт, а, например, систему для мультиплеерных игр), но обычно для этого используются другие технологии. А также вам придется (жизнь заставит) выучить HTML, CSS, JavaScript – сайты без них не могут существовать.

JavaScript. Тут есть несколько вариантов. Первый, самый распространённый: вы доучиваете еще HTML, CSS и идете писать сайты, веб-приложений, мобильные SPA (single-page application). Второй вариант: вы используете знание языка JavaScript для написание серверной части на NodeJS – и сейчас это очень востребовано, потому что это гораздо легче, чем писать сервер на C++/C#/Java. Третий вариант: вы доучиваете еще HTML, CSS и идете писать игры – Adobe Flash (основная технология для написания игр для браузеров) уже умер, – конкурентов нет. За последний год сформировался новый тренд (пятый путь): вы учите **основы** HTML CSS JavaScript, потом изучаете в деталях фреймворк ReactJS и вуаля! – вы реакт-программист, – это такой мутант, порожденный трендом и спросом (но сейчас это самый! Простой способ войти в IT с достойной зарплатой – спрос на таких программистов просто зашкаливает).

C#. Вы не будете скорей всего писать на нем программы для Windows. С 99% вероятностью вы будете использовать технологию ASP.NET и писать сайты, да вам придется выучить HTML, CSS и JavaScript (на уровне основ). Второй вариант: писать игры на Unity. Третий вариант: вы все-же будете писать десктопные приложения, например, дописывать новые функции Skype.

C++\C. Вы сможете писать ВСЕ. Из очень востребованного сейчас: сервера, мобильные игры, системы управления дронами и автомобилями, системы безопасности и наблюдения.

Python. Тут все просто: идете в написание серверной части для сайтов, а так как никто не любит содержать много разработчиков без надобности – доучиваете HTML, CSS, JavaScript. Второй вариант, аналитик данных – дорога в банки, обычно там это самое востребованное. Третий вариант: вы не программист, а научный деятель.

Java. Разработка серверов и поддержка существующих систем в банках и корпорациях, которые вложили миллионы долларов в программные комплексы и хотят продолжать их развивать. Второй вариант – это разработка Android приложений. И здесь вам тоже есть что выбирать: стартапы, компании среднего размера, корпорации.

Про зарплаты

Мы же все живые люди и хотим применять наши знания и получать при этом не только интеллектуальное удовольствие, но и материальное вознаграждение. Если не сильно вдаваться в детали, в мире IT разработчиков принято относить к нескольким категориям компетентности.

Junior (джун, малыш) – разработчик, который выучил технологию в **теории**, но опыт коммерческой разработки у него равен нулю. Его основная задача – это не погоня за зарплатой, а за опытом, чтобы скорей перейти в следующий статус.

Middle (середняк) – разработчик с опытом, таких большинство. Выполнение поставленных задач – его зона ответственности.

Senior (сеньор) – настоящий профессионал, он знает технологию в нюансах, он сталкивался с огромным количеством «черной магии в коде» и у него есть необходимые «заклинания» чтобы эту магию рассеять. К нему прислушиваются, его советов спрашивают.

Architector (архитектор) – это вершина карьеры разработчика, дальше только управление проектами и человеческим ресурсом. Архитектор строит системы с учетом требований заказчика. Это как академик в науке.

Так вот, про зарплаты. Они конечно же разнятся в зависимости от:

– Заказчика – зарубежный заказчик платит больше и это «больше» может быть больше в разы.

– Востребованности – чем меньше специалистов, тем дороже (хотя может быть нет специалистов, потому что это уже никому не нужно)

– Технологии – заказчики очень падки на тренды, хотите получать больше – следите за трендами.

Если говорить о зарплатах, как о «средней температуре по больнице»:

Джун – это вилка зарплат от 400—500 долларов до 800—1000 долларов в месяц;

Миддл – это от 1500 до 3000 долларов в месяц;

Сеньор – это соответственно от 3000 до 5000—6000 долларов в месяц.

Еще раз, это очень неточные цифры. Бывают и Java миддлы, которые живут в глубинке и получают 800 долларов, бывают JavaScript миддлы, которые получают 7000 долларов. А бывают и разработчики-комбайны, которые «колбасят» и на 10к долларов в месяц. А есть PHP разработчики, которые за адаптивный шаблон на WordPress получили 2 000 000 долларов. Вы всегда можете отыскать актуальные цифры и для стран СНГ, и для Европы, и для Америки – это открытая и интересная информация. Также стоит знать, что независимо от выбранной технологии, на уровне «сеньор» зарплаты примерно равны +\– сотня баксов роли не играет.

Этим сфера IT и хороша – здесь ВСЕ зависит только от ваших знаний, умений, стараний и терпения. А дальше выбор за вами.

Часть I

Глава 0. Мы программируем
Мы программируем железо

В наше время мы уже настолько окружены техникой, которая способна выполнять разнообразные программы, что уже воспринимаем это как должное. Что там компьютеры… телефоны, телевизоры, пылесосы (!), видеокамеры, автомобили, даже лифты – все они уже исполняют программы. Но для того чтобы писать хорошие программы, надо хотя бы примерно понимать, как устроен компьютер. В мире программистов компьютеры называются «железом» (хотя сейчас там гораздо больше пластика, стекла и кремния). У некоторых были уроки информатики, но, наверное, как это обычно происходит, они проходили за компьютерными играми :)

Что-же такое компьютер? Компьютер – это устройство, в которое можно ввести какие-либо данные, эти данные могут быть обработаны программой, и в конечном итоге будет выведен результат. В нашей реальности мобильный телефон (это тоже компьютер), в котором запущена программа (Instagram), в который вы вводите данные (ваши нажимы и касания пальцев, когда вы скролите ленту), и в итоге видим результат – новые посты в ленте.

Компьютер – это сложное устройство, он состоит из нескольких модулей, которые нужны для разных задач. В первую очередь самый заметный модуль (устройство вывода) – монитор\дисплей, через него мы получаем всю визуальную информацию. Устройства ввода, по меньшей мере два: клавиатура и мышь (или, например, тачпад). А еще устройство ввода – это цифровой сканер или VR-шлем (и он же устройство вывода). Далее непосредственно сам компьютер, обычно мы видим его как системный блок или нижнюю часть ноутбука.

Самые главные части современного компьютера: процессор (или несколько процессоров), оперативная память, твердый диск, видеокарта, звуковая карта. Процессор – это «мозг» компьютера, но в отличие от человеческого мозга, он не думает, а исполняет команды. Оперативная память – это куда загружаются программы, чтобы выполнять их в текущий момент. Оперативная память не сохраняет данные после выключения питания. Твердый диск – хранит разнообразные данные и программы. Данные на нем не будут потеряны после выключения питания. Чтобы понять разницу между оперативной памятью и твердым диском, вспомните себя, когда вы считаете сколько денег потратили: вы в голове перемножаете количество товаров на их цену (оперативная память), а потом записываете промежуточный результат на бумажку (твердый диск). Манипуляция мыслями занимает гораздо меньше времени, чем запись и потом поиск информации на бумажке – также и с компьютером: он быстро работает с оперативной памятью и гораздо медленнее с твердым диском. Видеокарта занимается ускоренным просчетом данных для последующей отрисовки, как мы все знаем это особенно критично для видеоигр :) Звуковая карта преобразует цифровые данные в электрические импульсы, которые «понимают» аудиоколонки.

Программист на Java в 99% случаев взаимодействует (и иногда сталкивается с нюансами) с дисплеем, клавиатурой, мышкой, процессором, оперативной памятью и диском, а также с другими компьютерами (обычно серверами) которые присылают данные и ждут ответов через интернет.

Мы программируем и пишем программы

Существует огромная пропасть между представлением данных для человека и для компьютера. Для компьютера данные представляются в виде электрических импульсов – есть ток, нет тока. Человек не может читать электричество, и он придумал различные абстракции. И поэтому наличие\отсутствие электрического тока представляется двумя цифрами 0 и 1, это уж позволяет использовать бинарную математику, это уже позволяет путем ввода нулей и единиц управлять процессором и соответственно писать программы. Но такие программы не читабельны – мы привыкли общаться словами и предложениями. И поэтому возникли разнообразные **языки** программирования, использовать которые для написания программ намного лучше, чем писать единицы и нолики. Сначала возникли языки программирования только чтобы просто программировать компьютеры, позже люди стали создавать языки программирования для работы в разных областях науки и бизнеса. Например, Fortran – для математиков и их формул, Cobol – для экономистов и биржевых операций, Assembler – для низкоуровневого программирования устройств, BASIC – для обучения школьников, Go – для программирования распределённых систем, PHP – для упрощенного написания веб-сайтов. Некоторые языки были созданы как следующий эволюционный шаг: Delphi – эволюция Pascal, Java – эволюция C, C# – эволюция C++ и Java.

Но Java создавали не только для того, чтобы сделать улучшенную версию языка C. Когда компьютер превратился из «монстра научных институтов» в персональный компьютер, стали появляться, одна за другой, операционный системы; возникло большое количество производителей компьютерного железа, которые выпускали кучу всяких устройств, программировать которые обычному программисту, без погружения в глубины документации и драйверов (управляющих железом микропрограмм) было невозможно. И вот в светлый ум доктора информатики Джеймса Гослинга пришла идея создать такую технологию, которая позволила бы программисту писать такой код, который мог бы запускаться на **любом железе.** И его идея была им воплощена в язык программирования Java и виртуальную машину Java.

В чем суть? Виртуальная машина Java это программный компьютер внутри компьютера обычного на базе операционной системы Windows\MacOS\Unix\…. Виртуальная машина Java (Java Virtual Machine – JVM) специально создается для каждой операционной системы. И JVM знает, как «общаться» с операционной системой и через нее со всеми устройствами, которые могут быть подключены к компьютеру. В чем выгода для программиста? – программист пишет свой код только один раз (!) и ему в своем коде не надо знать на каком компьютере, с какой операционной системой его код будет запускаться. Это огромная экономия времени и денег. Именно поэтому Java код очень надежный и безопасный, он с одной стороны может ограничивать программиста, когда тот хочет «пошалить», а с другой стороны не дает программисту написать код, который может что-то сломать.

Стоит заметить, что концепция «виртуальных машин» получила дальнейшее развитие в других языках программирования, главным образом из-за безопасности. Например, виртуальная машина JavaScript, она встроена в браузер, дает определенные возможности для работы с документом в браузере, для обращения к серверу, но полностью запрещает произвольное считывание данных пользователя с диска. Очень похожая на JVM виртуальная машина Microsoft. NET.

Глава 1. Первая программа
Что такое программирование

Программирование – это процесс написания команд, которые потом будет выполнять компьютер. Очень важно понимать, что компьютер **не умеет думать**. Все, что компьютер **делает: он исполняет команды**. У программиста может складываться впечатление, что происходит какое-то «колдунство» и компьютер вытворяет самолично с программой, что захочет. Но на самом деле, это будет только означать, что программист не учел каких-то особенностей функционирования программы или библиотек, которые он использует, или нюансов как работает внешний источник данных, к которому он обращается. Чем профессиональней программист – тем меньше «неожиданных чудес» можно ожидать от написанного им кода.

Программирование на Java состоит из нескольких этапов:

– Написание программы на языке Java в редакторе

– Компилирование программы в байт-код (код понятный виртуальной машине Java) с помощью программы-компилятора

– Исправление ошибок компиляции (compilation errors), если такие произошли в процессе компиляции

– Запуск программы в виртуальной машине Java.

– Исправление ошибок выполнения (runtime errors), если видим, что «что-то пошло не так»

– Повторение пунктов 2—5 пока мы не получили работающую по нашему замыслу программу.

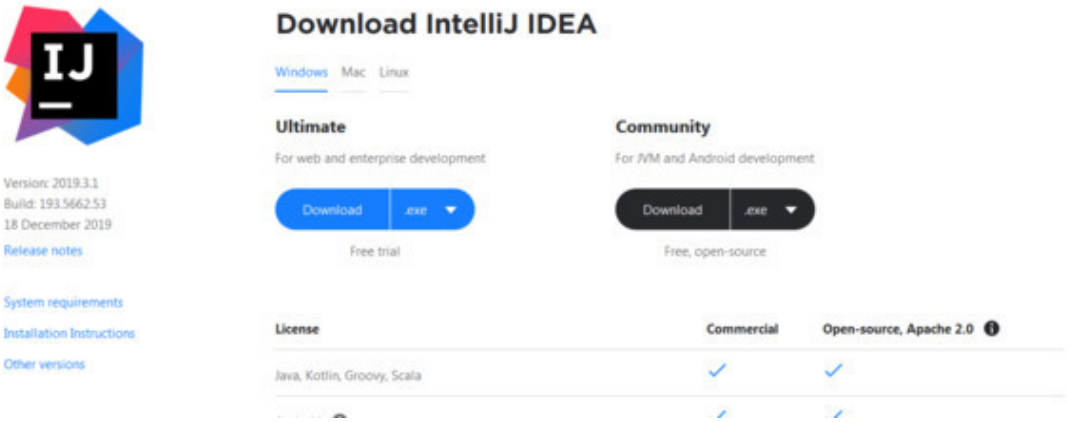
Можно писать код в одном из текстовых редакторов (Notepad, Notepad++, Atom, Sublime) и потом дальше через командную строку запускать компилятор, а потом запускать программу. Но все это громоздко и неудобно, именно поэтому программисты написали специальные программы, в которых можно делать полный цикл разработки программы гораздо проще и удобнее. Такие программы называются IDE (Integrated Development Environment) – интегрированная среда разработки, в ней происходит и написание программы, и компиляция, и выявление ошибок, и запуск программы. К тому же, большинство из них еще и подсказывают разработчику, что и в каком случае можно использовать и где он возможно уже совершает ошибку.

В мире Java-программирования есть несколько популярных IDE: IntelliJ IDEA, Eclipse, NetBeans. NetBeans самая редко используемая IDE на текущее время. Eclipse – это **бесплатная** IDE, с тысячами полезных плагинов, облегчающая жизнь разработчика. Поэтому, вполне возможно, что в крупной компании, в которую вы придете работать, будут использовать именно Eclipse. И это стоит учитывать, потому что на самом деле вы захотите пользоваться только одной IDE: IntelliJ IDEA – лучшая и самая удобная IDE на текущий момент для написания программ на Java.

Настраиваемся на программирование. Устанавливаем IDE

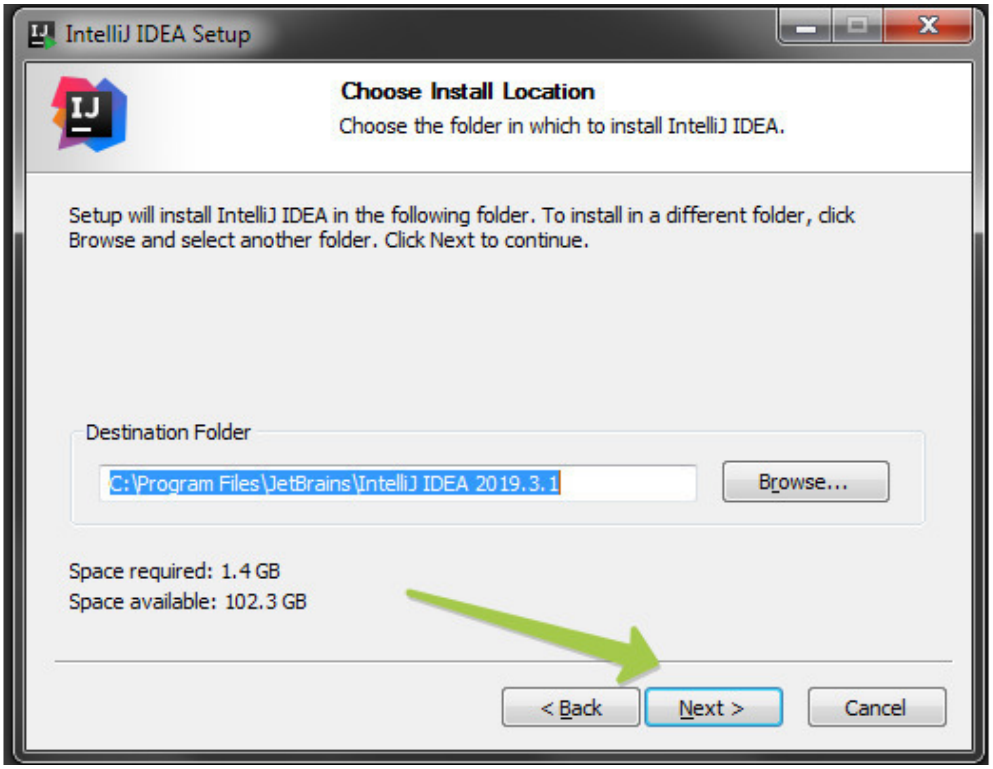
Так как предполагается, что мы изучаем с нуля, то мы не будем заморачиваться с установкой виртуальной машиной Java и полного пакета для разработчика Java SDK. Достаточно будет скачать текущую версию IntelliJ IDEA, в которой есть все что нам будет нужно для старта.

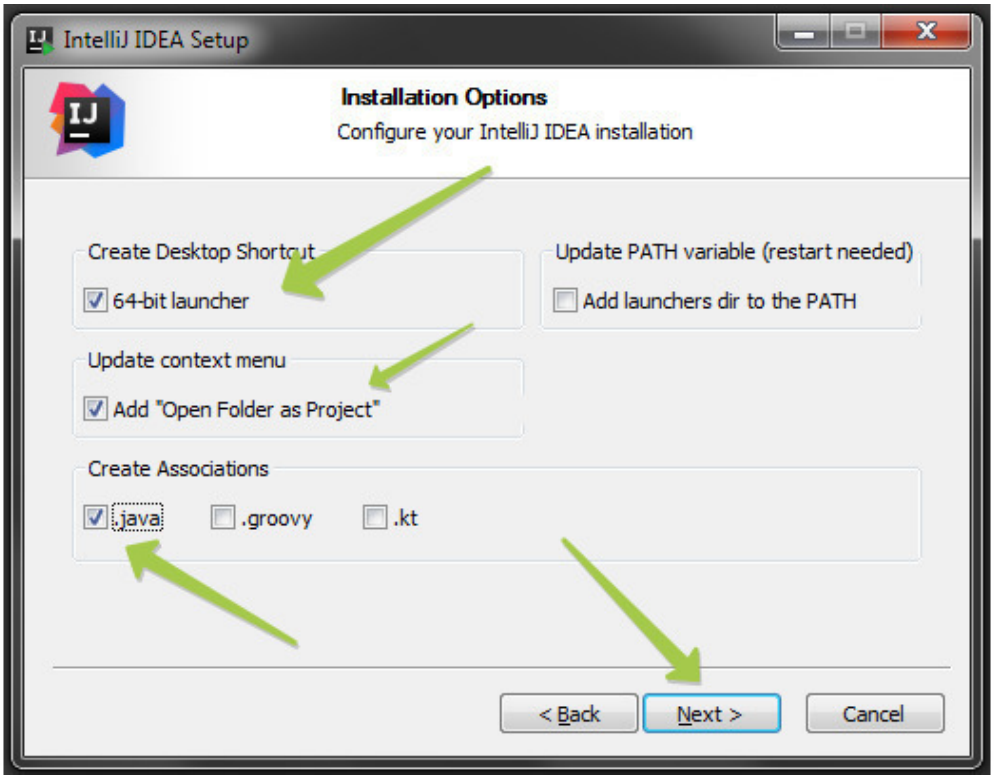
Заходим на сайт https://www.jetbrains.com/idea/download/

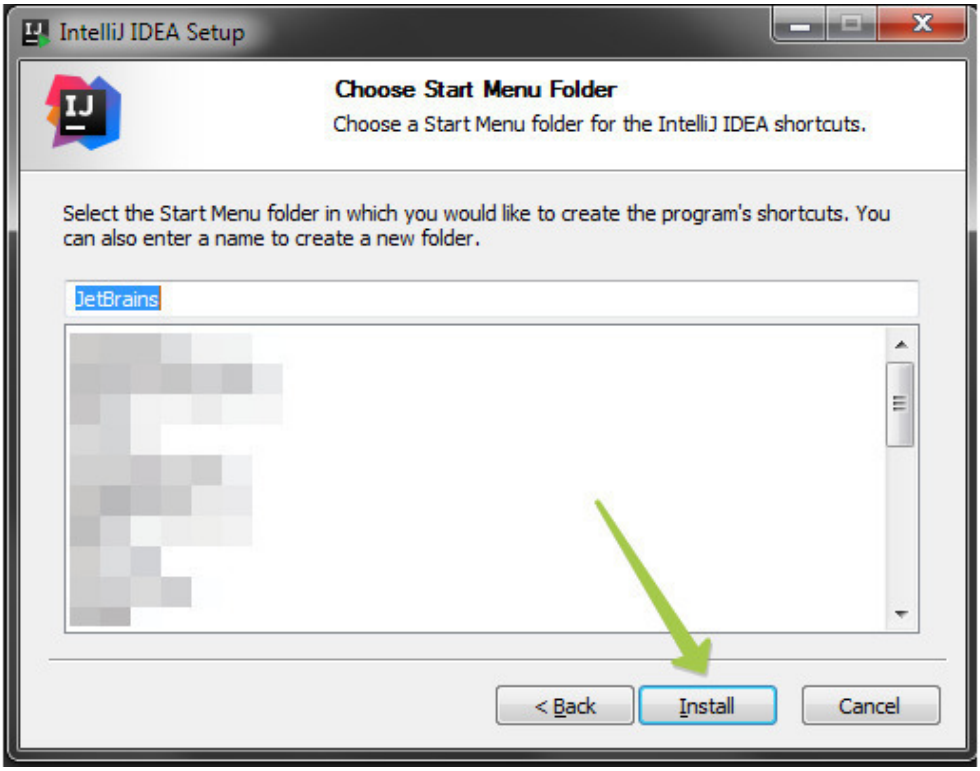


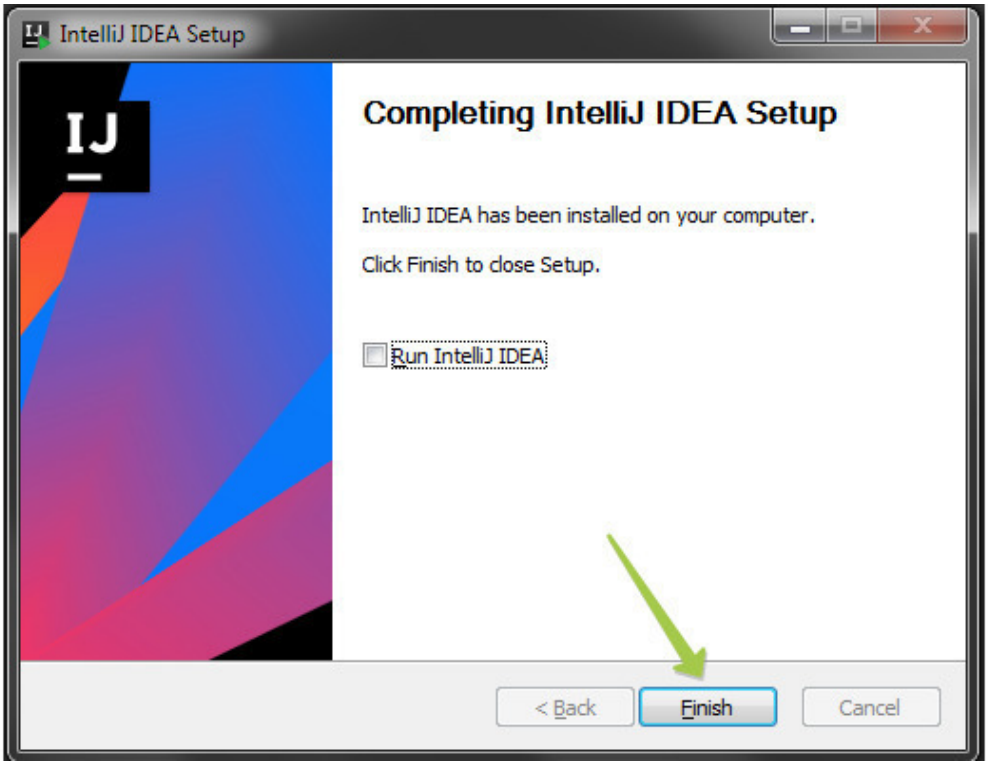
выбираем версию для скачивания: Ultimate (т.е. полную, но платную, хотя есть пробный период) или Community (тоже достаточную для наших целей)

Скачиваем и запускаем. Проходим через мастер установки, соглашаясь со всем что предлагают:


IntelliJ IDEA Setup
Choose Install Location
Choose the folder in which to install IntelliJ IDEA.
Setup will install IntelliJ IDEA in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.
Destination Folder
C:\Program Files\JetBrains\IntelliJ IDEA 2019.3.1
Browse...
Space required: 1.4 GB
Space available: 102.3 GB
< Back
Next >
Cancel

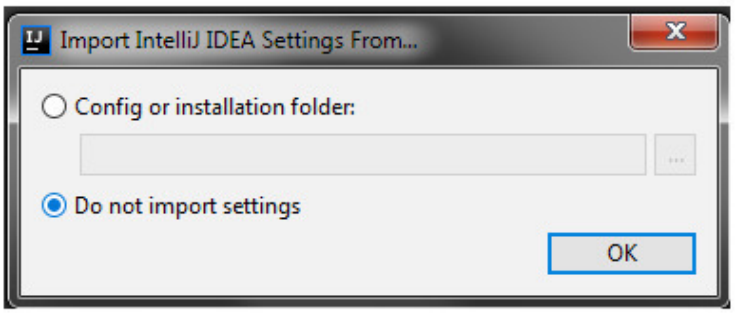

IntelliJ IDEA Setup
Installation Options
Configure your IntelliJ IDEA installation
Create Desktop Shortcut
64-bit launcher
Update PATH variable (restart needed)
Add launchers dir to the PATH
Update context menu
Add "Open Folder as Project"
Create Associations
.java
.groovy
.kt
< Back
Next >
Cancel



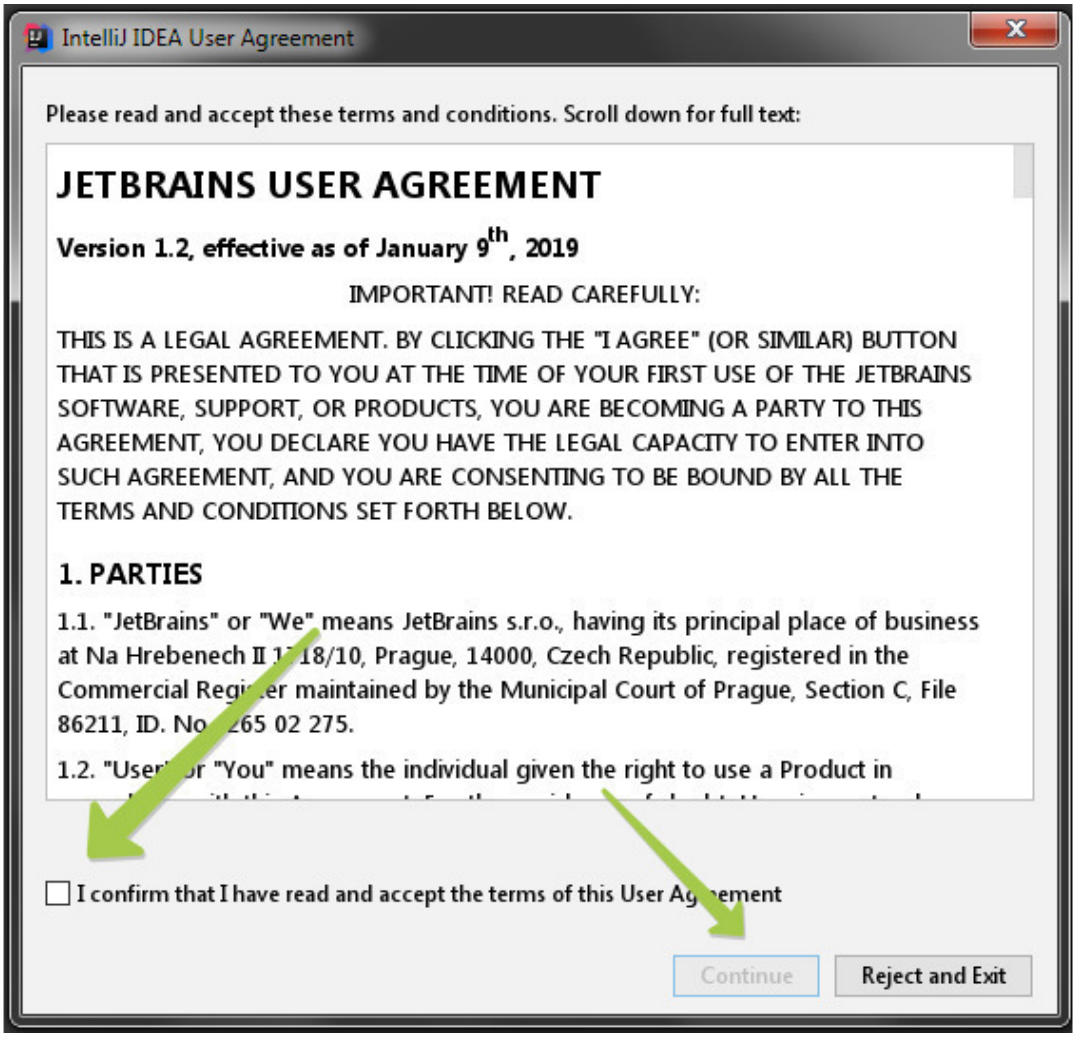


На Рабочем столе появится иконка приложения:

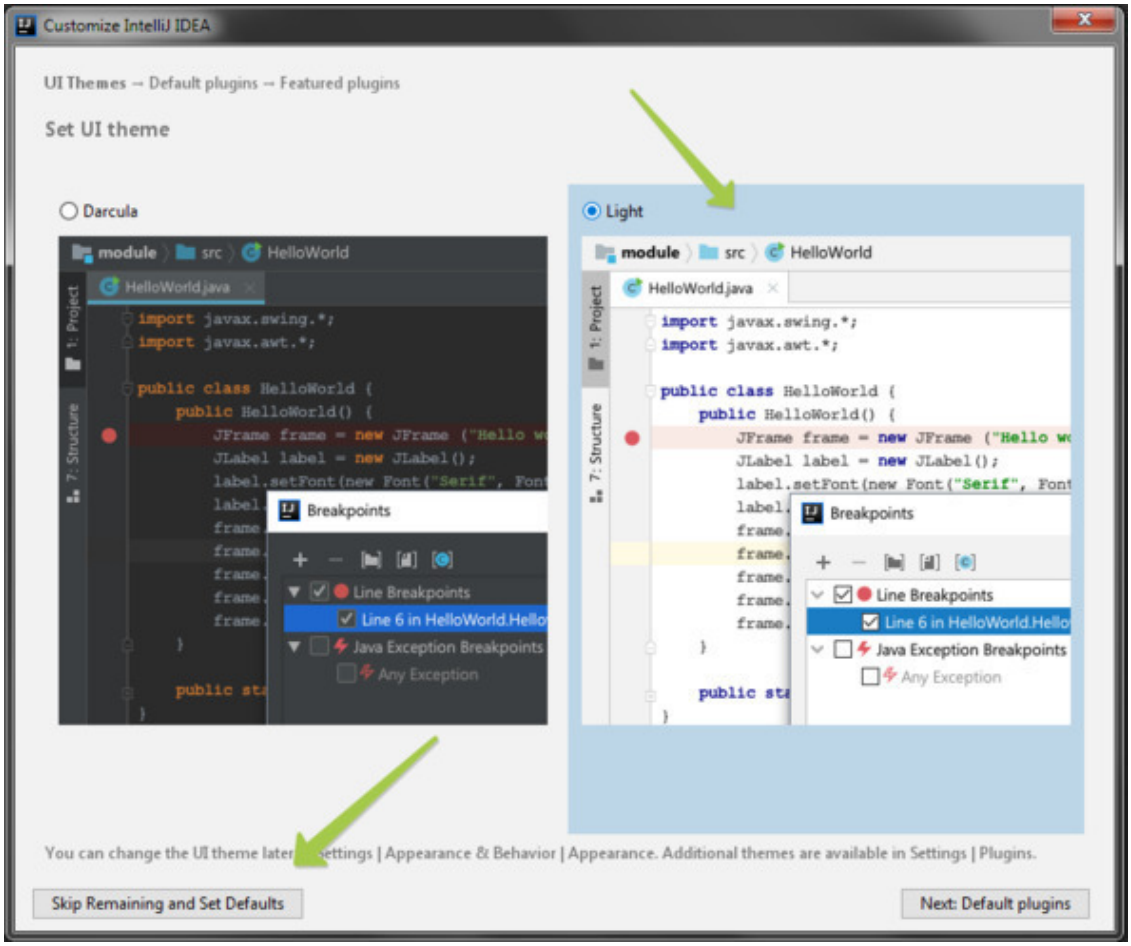
Дважды кликаем для запуска IntelliJ IDEA. И снова проходим еще через несколько экранов настройки (это будет только единожды):



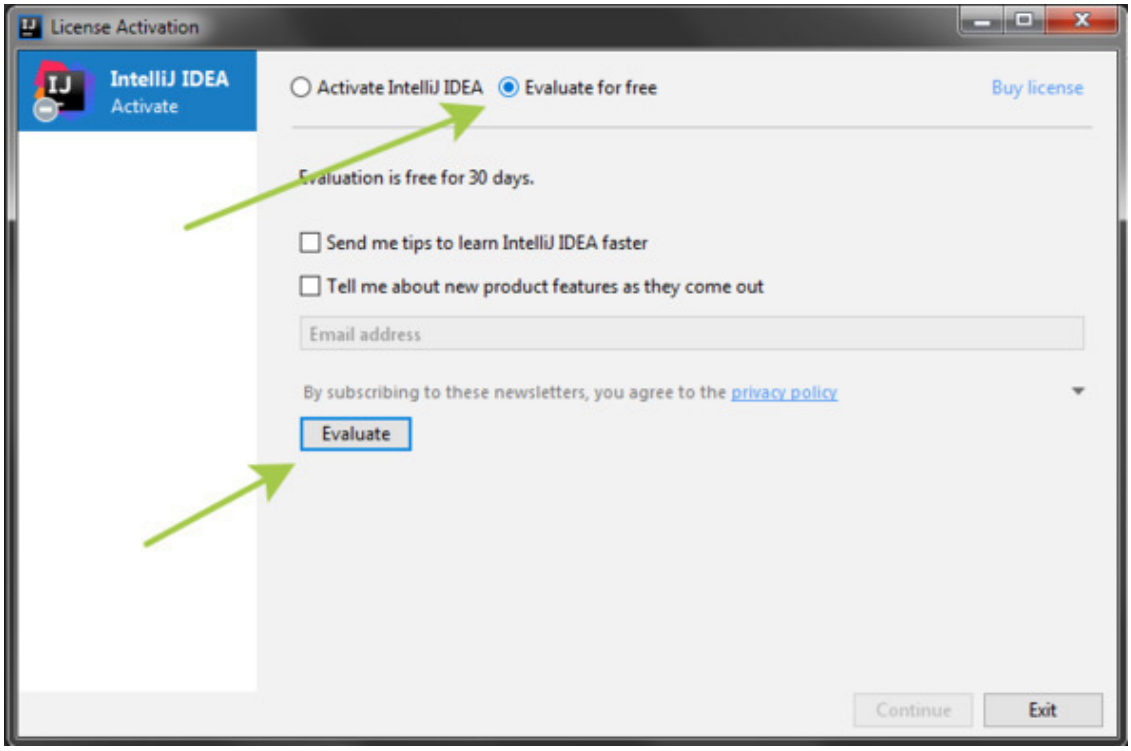
не импортируем никакие настройки



Соглашаемся…

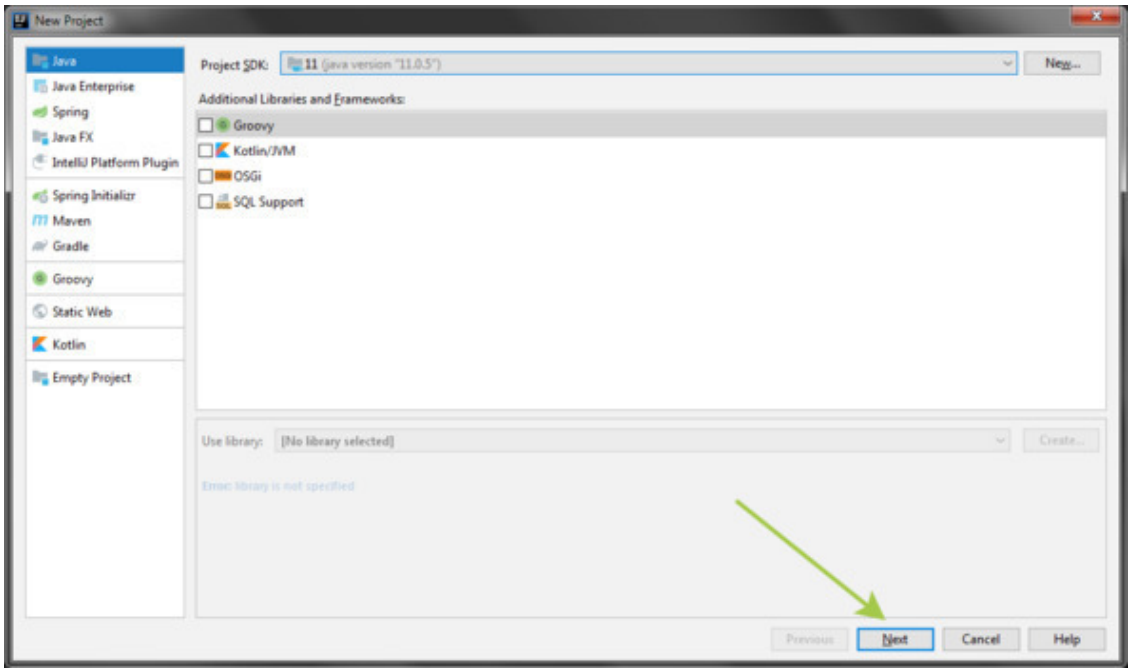


Выбираем светлую тему (это всегда можно изменить в настройках)

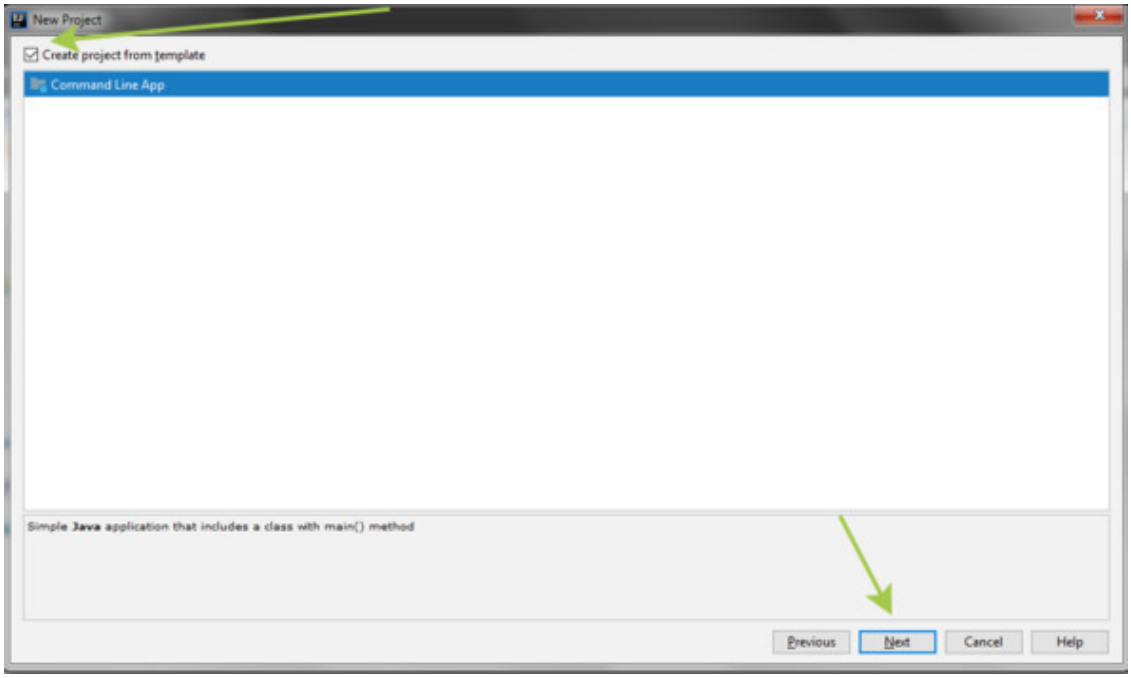


Я скачал версию Ultimate, поэтому выбираю пробный период

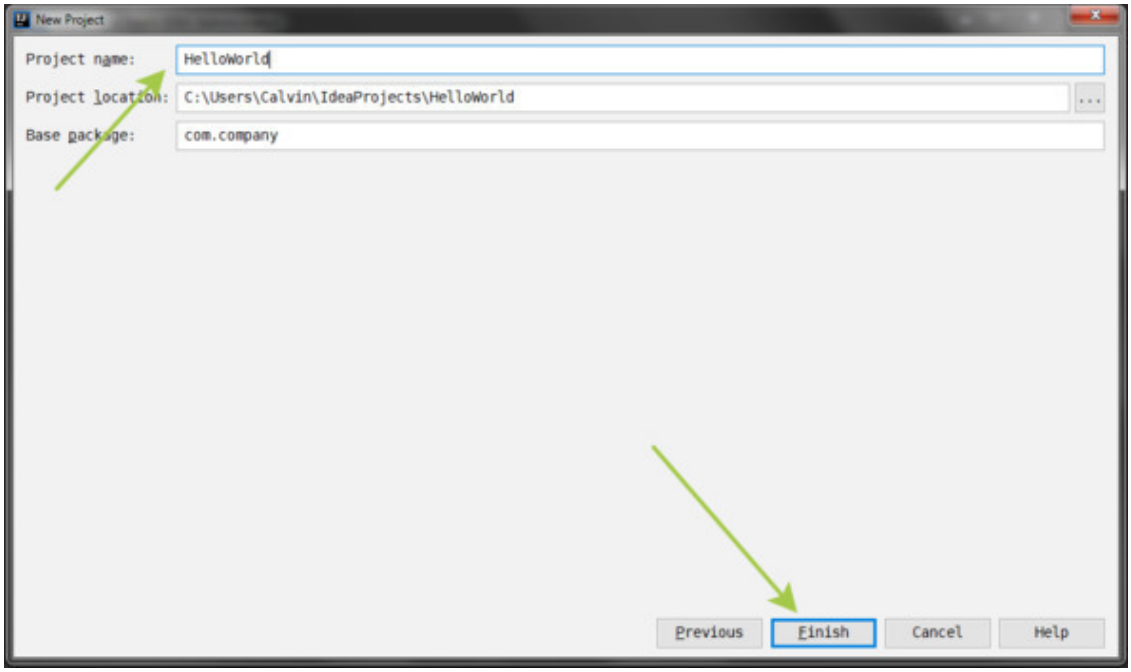
И вот финальный экран, где нужно кликнуть на Create New Project.



Слева должно быть выбрано Java, в центре вверху Project SDK: 11 (это версия Java, идущая вместе с IntelliJ IDEA), нажимаем Next.



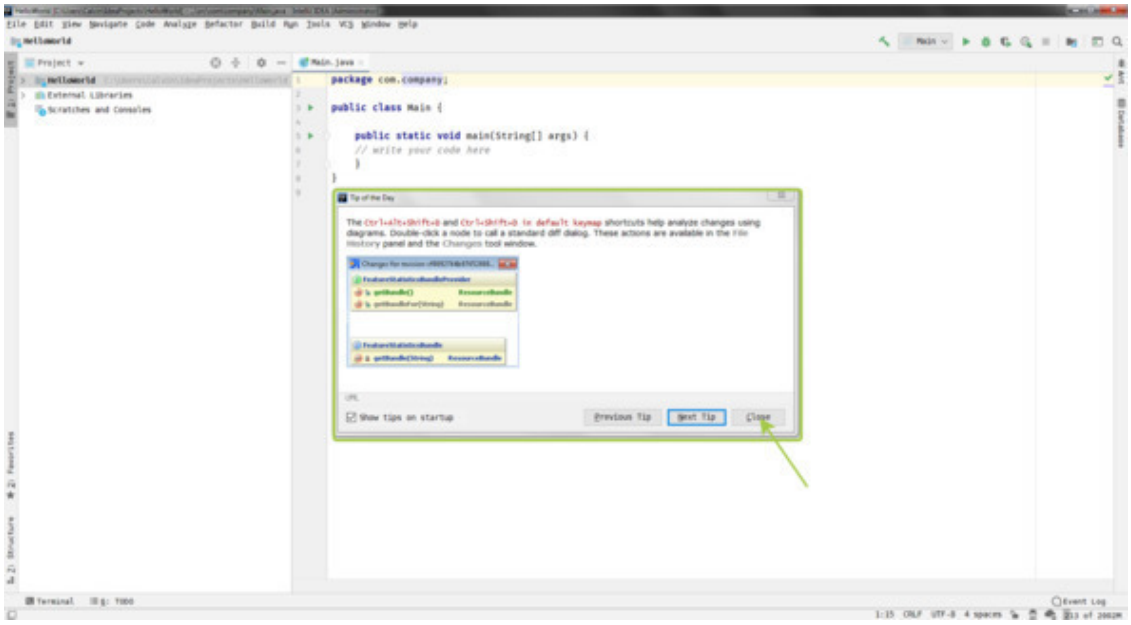
Ставим галочку Create project from template, выбираем Command Line App и кликаем Next.



В поле Project name вводим HelloWorld (это название нашего проекта) и нажимаем Finish.

Никогда не используйте русских букв (кириллицы) в названиях проектов, классов и т. д.

IDEA немного «подумав» откроет нам основное окно, в котором мы будем разрабатывать наши программы, и также откроется поверх маленькое окошко «Tip of the Day» – просто закройте его кликнув **Close.**

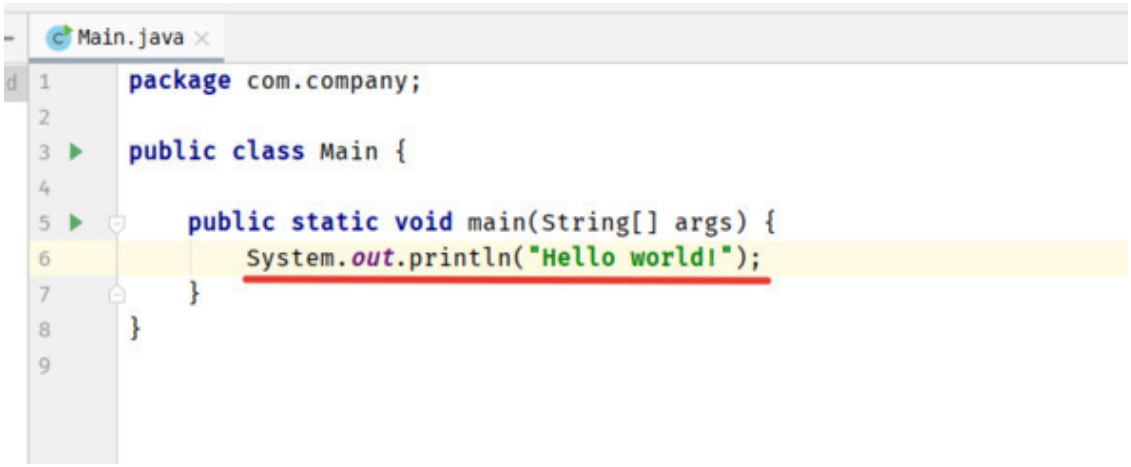


Теперь вам нужно написать свою первую строку кода, вместо надписи:

// write your code here.

Напишите там:

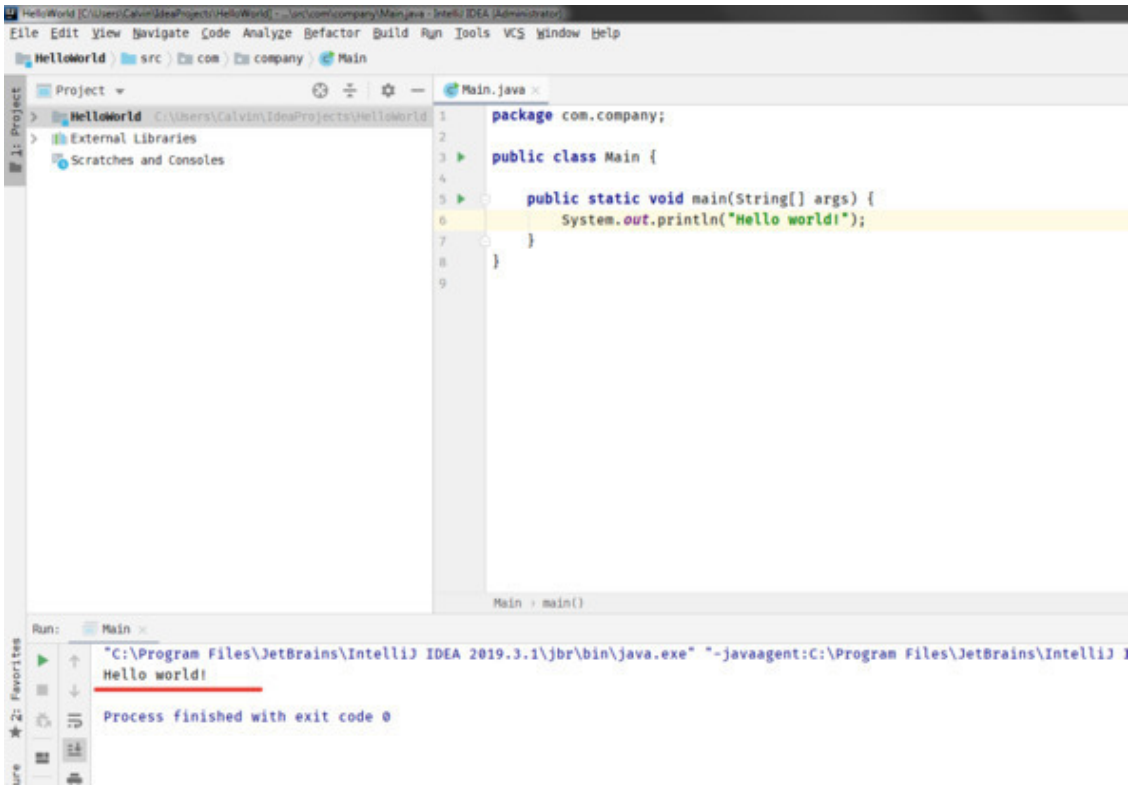
System.out.println («Hello world!»);



И запустите вашу первую программу нажатием на зеленый треугольник:

В итоге ваша программы выведет, в специальной панели IDEA:

Hello world!



То, что написано ниже: «Process finished with exit code 0», означает, что программа завершилась без каких-либо явных ошибок. Поздравляю всех, кто дошел до этого последнего шага и успешно запустил первую программу!

Если же у вас что-то не получилось с установкой IntelliJ IDEA или с созданием программы – не переживайте и не волнуйтесь. Вы можете поступить как настоящий программист: вбейте в Google\Yandex «как установить IntelliJ IDEA», «создание Hello world в IntelliJ IDEA» и уже на первой странице результатов вашего поиска вы обязательно найдете статью, в которой пошагово вам покажут, как это сделать. Так же вы можете эти запросы вбить в YouTube – в 2020 году уже существует достаточное количество видеороликов с подобной информацией.

Не бойтесь пользоваться поисковиками, сейчас информации более чем достаточно, более того в книгах не найти ответы на все свои вопросы.

Глава 2. Данные

Программирование – это процесс написания команд, которые потом будет выполнять компьютер, чтобы манипулировать данными. Основная причина, почему компьютеры стали очень распространенными устройствами – они умеют очень быстро обрабатывать огромные объемы данных. Любая задача для программиста – это задача про управление и преобразования данных, чтобы получить результат – другие данные. Соответственно данные надо где-то хранить: исходные данные, чтобы их прочитать и начать использовать, результаты – для сохранения промежуточных результатов (например, при очень сложных расчетах) или для финальных значений.

Данные, используемые программой, хранятся в памяти компьютера. Такая память называется оперативной памятью (и чем ее больше, тем быстрее работает компьютер, тем больше вкладок можно открыть в браузере :)). Память состоит из маленьких ячеек памяти. Каждая ячейка хранит одно значение 0 или 1, и называется **бит.** Принято считать, что 8 бит – это **1 байт.** 1024 байта – это **1 килобайт.** 1024 килобайт – это **1 мегабайт.** 1024 мегабайт – это **1 гигабайт.** Ну а потом уже следуют тера-, пета-, экса-…. байты.

Программисту, в процессе работы, приходится использовать большие и маленькие числа, которые могут занимать разное количество память. Для того, чтобы программа работала быстрее и старалась не использовать больше памяти, чем необходимо, программист использует **типы данных**.

Типы данных

Использование типов данных – это добровольное ограничение использования памяти. Например, программист знает, что в расчетах будет использоваться значение не больше 77, поэтому он использует тип данных **short**, для хранения этого значений. Для больших значений – соответственно другие типы данных. Вообще в Java существует 8 примитивных типов данных:

boolean – хранит один бит информации;

byte – хранит один байт информации, и соответственно целочисленные значения от -128 до 127 включительно;

short – хранит два байта информации, и соответственно целочисленные значение от -32768 до 32767 включительно;

int – хранит четыре байта информации, и соответственно целочисленные значение от -2147483648 до 2147483647 включительно;

long – хранит восемь байт информации, и соответственно целочисленные значения от –9223372036854775808 (-2 в степени 63) до 9223372036854775807 (2 в степени 63 минус 1) включительно;

float – хранит четыре байта информации, и соответственно значения с плавающей точкой от $1.4x10^{-45}$ до $3.4x10^{38}$;

double – хранит восемь байт информации, и соответственно значения с плавающей точкой от $4.9x10^{-324}$ до $1.7x10^{308}$;

char – хранит 16 бит (2 байта), и соответственно символ в формате UTF.

То есть, когда нужно сохранить число 77 в памяти, и программист определил, что надо использовать для этого тип данных **short,** виртуальная машина Java выделит из общей памяти два байта.

В основном используют int и float, т.к. они покрывают большинство нужд.

Остается только один вопрос: как программист узнает *где расположены* эти два байта и *как к ним обратиться*. Именно для этого существуют переменные.

Переменные

Переменная – это **указатель** на область памяти определенного типа. Вы можете думать обо всем этом как о большом складе. Вы приходите на склад (вся свободная память компьютера) и говорите «мне нужно содержимое коробки с таким-то именем» (переменная) и заведующий склада (виртуальная машина Java) дает вам это содержимое. Также вы можете не только взять, но и положить туда то, что вам необходимо. Но если коробка маленькая, а вы пытаетесь засунуть туда большое содержимое – заведующий склада не даст вам это сделать и скажет, что вы ошибаетесь.

Несколько примеров как это выглядит в коде:

```
int box1;
int box2 = 70;
box1 = 50;
int box3 = box1 + box2;
System.out.println (box3);
```

В первой строке мы определяем переменную **box1** типа **int**.

Во второй строке мы определяем переменную **box2** типа **int**, при этом мы сразу кладем туда (или как говорят программисты «присваиваем») значение **100**.

В третьей строке мы присваиваем переменной **box1** значение **50** (программисты еще говорят «переменная box1 проинициализирована значением 50»). Если мы этого не сделаем, то получим ошибку на этапе компиляции нашей программы: компилятор скажет, что нельзя в программе использовать переменные, у которых нет значений.

В четвертой строке мы складываем содержимое **box1** и **box2** и присваиваем новой переменной **box3** тоже типа **int**.

В пятой строке выводим на экран (или как еще говорят «распечатываем на экране») значение переменной **box3** (120).

Как долго живут переменные?

Чем больше и сложнее программа, тем больше различных данных придется хранить в различных переменных. Причем зачастую нам нужны переменные только на какое-то время совершения какой-то операции или нескольких операций. Стоит ли хранить даже такие «временные переменные»? – конечно же нет. И как раз для этого была придумана «область видимости переменной», которое определяет, как долго переменная «будет жить», т.е. будет доступна для использования. На практике область видимости определяется фигурными скобками {} – переменная объявленная и проинициализированная внутри фигурных скобок «умирает» как только поток выполнения программы выйдет за эти скобки. Отсюда следует несколько областей видимости:

Конец ознакомительного фрагмента.

Текст предоставлен ООО «Литрес».

Прочитайте эту книгу целиком, купив полную легальную версию на Литрес.

Безопасно оплатить книгу можно банковской картой Visa, MasterCard, Maestro, со счета мобильного телефона, с платежного терминала, в салоне МТС или Связной, через PayPal, WebMoney, Яндекс.Деньги, QIWI Кошелек, бонусными картами или другим удобным Вам способом.