Настройка рабочего окружения

Данное руководство предназначено для пользователей Windows. Пользователям других операционных систем необходимо будет разбираться в установках самостоятельно.

К сожалению, разные версии Windows могут иметь различия в интерфейсе, поэтому не всё в руководстве может соответствовать реальным обстоятельствам.

1 Установка JDK

Для работы с Java необходимо установить JDK (Java Development Kit). На учебных компьютерах установлен JDK версии 8. Если вы планируете сдавать задания на них, то не имеет смысла устанавливать JDK более высокой версии.

Стоит отметить, что следующей версией с долгосрочной поддержкой после JDK 8 является JDK 11.

При установке обязательно запомните, в какую папку была произведена установка.

JDK можно скачать с официального сайта Oracle:

<https://www.oracle.com/technetwork/java/javase/downloads/index.html>

Бесплатное использование Oracle JDK предназначено для некоммерческих целей. С более подробными условиями соглашения можно ознакомиться на сайте. Для выполнения учебных заданий Oracle JDK подходит.

Бесплатный аналог – OpenJDK:

<https://openjdk.java.net/install/>

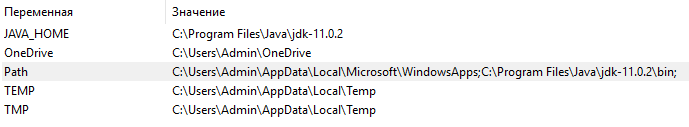
Рекомендуется использовать первый вариант.

После установки JDK необходимо будет задать переменные среды. Для этого надо перейти в меню по навигации: Компьютер -> Правый клик -> Свойства -> Дополнительные параметры системы -> Вкладка «Дополнительно» -> Переменные среды. Здесь должны быть расположены переменные для пользователя и для всей системы. Вы сами определяете, куда хотите добавить переменные.

Для начала надо добавить переменную JAVA\_HOME, значением который должен являться путь до директории установки JDK.

Теперь необходимо добавить путь до папки bin в переменную Path. Но ни в коем случае не надо удалять элементы, которые уже есть в Path. Необходимо добавить ещё один путь через точку с запятой. Пример:

%JAVA\_HOME%\bin



Перезапустите консоль (если она уже была запущена): Ctrl + R -> “cmd” –> Ок.

Наберите команду:

java -version

Должна отобразиться версия той Java, которая была установлена чуть ранее. Наберите команду:

javac -version

Должна отобразиться версия компилятора соответствующего JDK.

Если в том или ином случае указывается другая версия, то на вашем компьютере есть как минимум ещё одна Java. Перейдите в переменные среды и пробегитесь по переменным Path пользователя и системы, чтобы выяснить, упоминается ли среди их значений Java. Если есть – удалите эти значения (из переменной Path, остальное в ней оставьте неизменным!), предварительно сделав резервную копию в какой-либо файл.

Повторите попытку проверки версий, не забыв перезапустить консоль. Если ситуация не исправилась, рекомендуется обратиться к преподавателю.

После этого рекомендуется проверить работу JDK. Создайте где-нибудь файл с названием MyClass.java (откройте блокнот, затем используйте «Сохранить как…» и укажите название MyClass.java; если сначала создать текстовый файл, то к имени автоматически будет добавлено .txt – получится MyClass.java.txt, хотя этого может быть и не видно).

Добавьте в файл следующие строчки и сохраните:

public class MyClass {

public static void main(String[] args) {

System.out.println("Hello, world!");

}

}

Перейдите в консоли в директорию, где лежит этот файл. Используйте команду “cd”, чтобы это сделать:

cd <directory>

Если вам нужно перейти на другой диск, введите только название диска:

D:

Скомпилируйте созданный файл, используя компилятор javac:

javac MyClass.java

Если ошибок нет, то консоль просто вернёт управление. Обратите внимание, что должен был появиться файл MyClass.class, содержащий байт-код скомпилированного класса. Запустите исполнение файла с помощью команды java:

java MyClass

Должна появиться надпись «Hello, world!».

Обратите внимание, что для команды «javac» разрешение файла нужно указывать, а для команды «java» – нет.

Если что-то не получилось сделать – обратитесь к преподавателю.

2 Установка IDE

Для существенного упрощения разработки используются продвинутые текстовые редакторы и/или интегрированные среды разработки (IDE – Integrated Development Environment). В данном курсе будет использоваться IDE от JetBrains: IntelliJ IDEA.

Скачать её можно на сайте разработчика:

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

Представлено две версии: 1) Продвинутая, Ultimate Edition, с бесплатным ознакомительным периодом; 2) Community Edition, бесплатная свободно распространяемая. Для задач обучения вполне достаточно второй, поэтому будем использовать её.

Установщик может предложить настроить некоторые свойства на выбор, но можно оставить всё как есть.

После установки IDE предложит выбрать тему: тёмную или светлую. Выбирайте на свой вкус. Примеры в курсе будут демонстрировать на белой теме.

Все остальные настройки рекомендуется пока оставить как есть.

После окна предварительных настроек появится приветственное окно (с предложением создать новый проект).

В целом на этом можно было бы остановиться и начать создавать новый проект, поскольку Java настроена. Однако, существуют вспомогательные средства, которые позволяют упростить процесс разработки и сборки (в особенности – разработки больших приложений). В нашем случае это будут система контроля версий и система сборки.

3 Создание репозитория в VCS

VCS (Version Control System) – система контроля версий. Изучение этой системы не входит в рамки данного курса, но её минимальные знания и возможности смогут существенно облегчить выполнение заданий.

Система контроля версий – программное обеспечение для облегчения работы с изменяющейся информацией. Она позволяет вносить изменения в систему, но при этом хранит каждую версию изменений, так что в любой момент можно откатиться. С ней гораздо проще отслеживать изменения кода и находить, когда «всё сломалось». Более того, система позволяет работать над проектом сразу многим людям и поддерживать версии для каждого из них.

Будем использовать в качестве такой системы git:

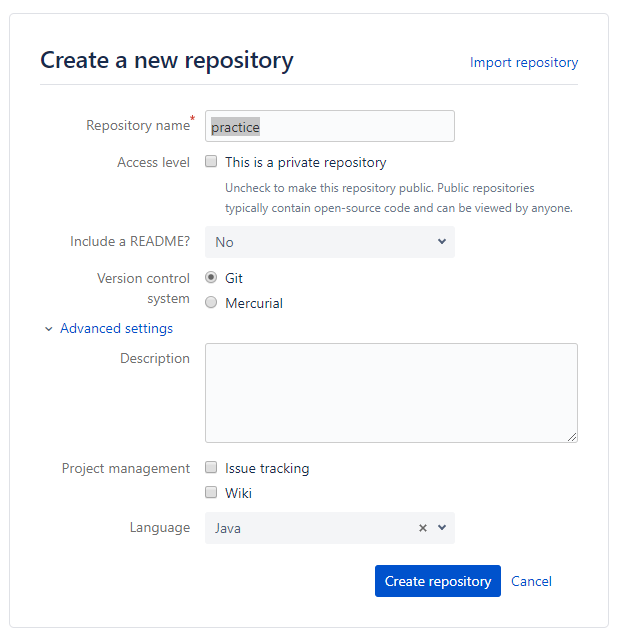
<https://git-scm.com/download/win>

При установке можно оставить опции так, как они выбраны по умолчанию.

Для хранения проекта используется специально хранилище – репозиторий. Примерами сервисов, предоставляющих репозитории, являются GitHub и Bitbucket. Создайте себе аккаунт в одном из них. Далее в примерах будет рассматриваться Bitbucket:

<https://bitbucket.org/product/>

Создайте репозиторий:



Имя репозитория соответствует вашему учебному проекту. Для самостоятельного изучения подойдёт название “sandbox” (песочница), “practice” (практика) или что-то похожее.

Уровень доступа (access level) – обязательно открытый, ни в коем случае не приватный. Ссылку на созданный репозиторий нужно будет отправить преподавателю для контроля за выполнением заданий. README не требуется. VCS – Git. Language – Java.

Имейте в виду, что у вас в течение семестра будет не один проект, поэтому название выбирайте вдумчиво, чтобы его можно было отличить от других ваших проектов.

Откройте IDE на приветственном окне. Нажмите Ctrl + Alt + S, открыв настройки среды, и запомните это сочетание клавиш. Перейдите на Version Control -> Git и в верхнем пункте укажите путь до git.exe. Примените настройки и вернитесь к окну приветствия. Выберите «Check out from Version Control».

В поле URL нужно добавить адрес вашего репозитория. Мы будем использовать протокол HTTPS. Откройте в браузере ваш репозиторий, найдите предложенную ссылку на него («https://\*\*\*@bitbucket.org/\*\*\*/\*\*\*.git) и перенесите её в окно среды разработки.

Выберите, где будет размещён проект на вашем компьютере (и запомните это расположение).

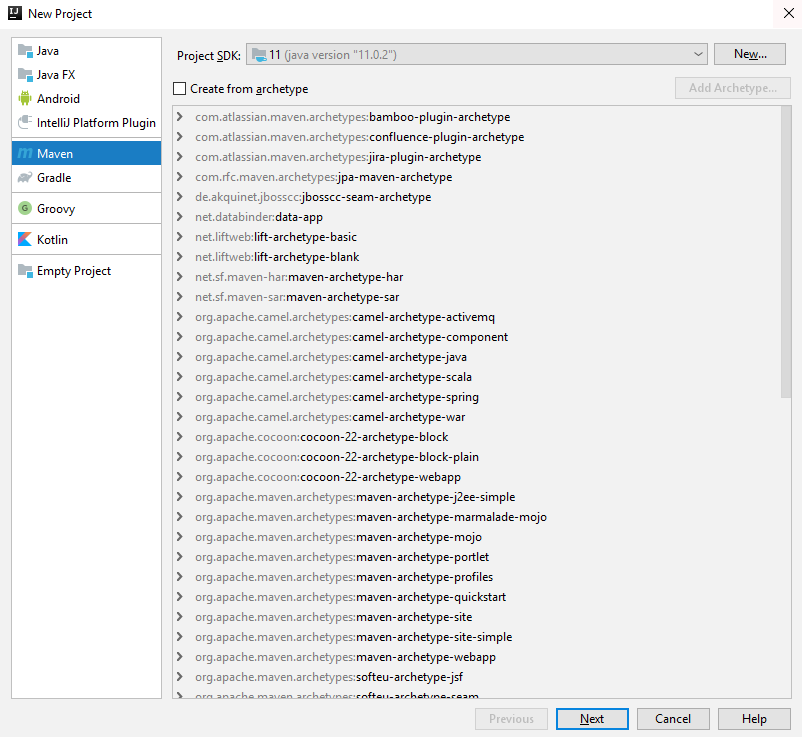
Нажмите “Clone”. Откажитесь от создания нового проекта. Если всё получилось, то вы только что клонировали свой пока ещё пустой репозиторий себе на компьютер с помощью команды “check out”.

Откройте расположение вашего проекта. Видно, что внутри лежит папка «.git» (если включено отображение скрытых папок).

4 Создание проекта

Пришло время создать проект. Вернитесь к IDE и нажмите “Create New Project”. Откроется меню, где в левой колонке нужно выбрать тип проекта. Но не нужно выбирать Java! Оставьте свой выбор на maven, потому что эта система сборки поможет в дальнейшем сократить значительное время на импорт сторонних библиотек. Про неё поговорим чуть позже.

После выбора в правой части выберите SDK (Software Development Kit) – тот JDK, который вы установили на первом этапе. Если его нет в списке, потребуется добавить. «Next».



В следующем окне вам будет предложено выбрать GroupId, ArtifactId и Version. В рамках данного курса мы не будем изучать, что это и как это выбирать. Пока укажите эти параметры по следующему шаблону:

GroupId: ru.ssau.tk.\_NAME\_.\_PROJECT\_

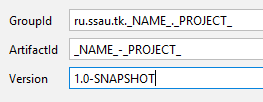
ArtefactId: \_NAME\_-\_PROJECT\_

Version: 1.0-SNAPSHOT.

Здесь «ru.ssau.tk» – перевёрнутый адрес сайта кафедры технической кибернетики.

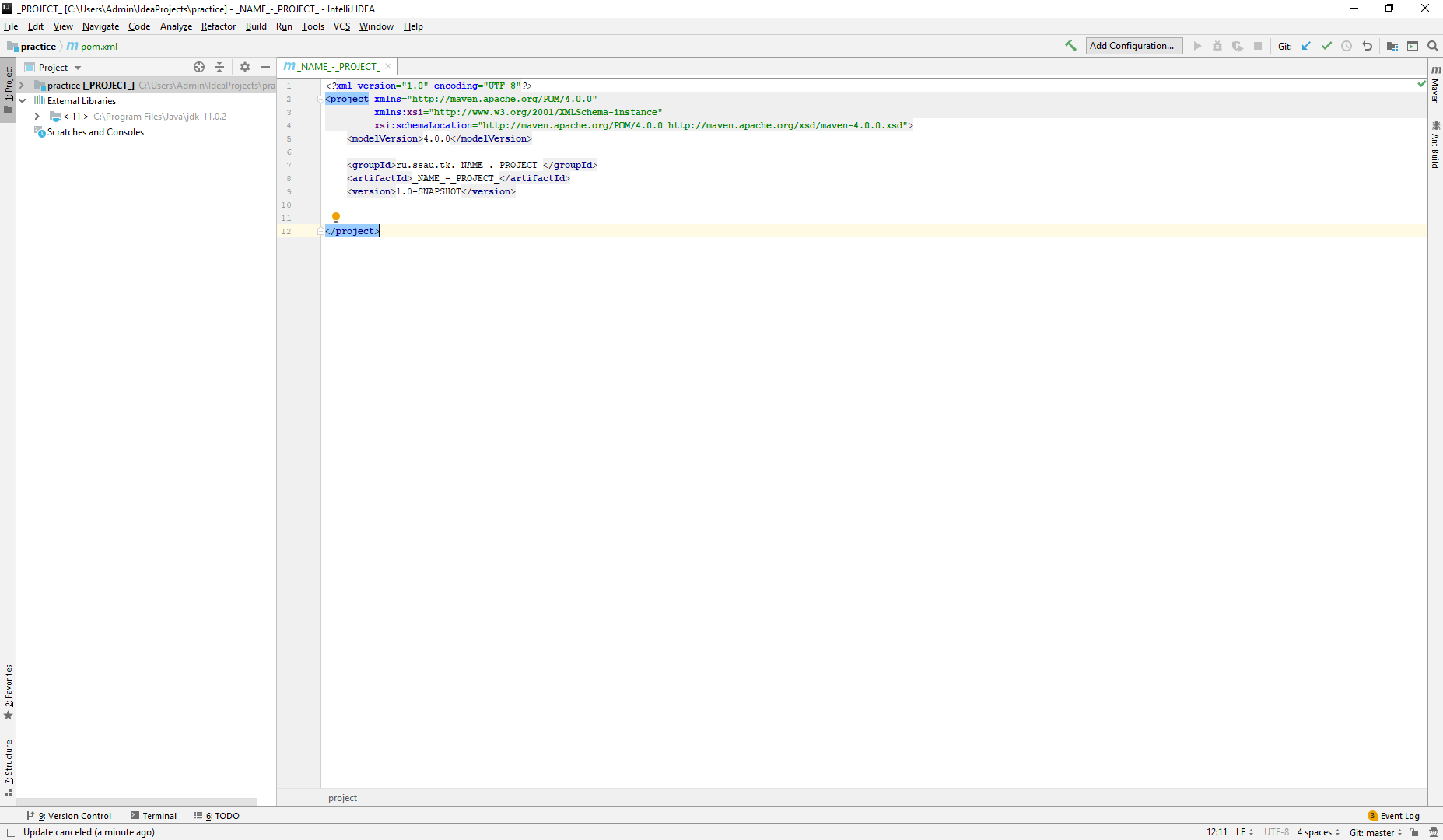
\_NAME\_ - одно слово - «имя» маленькими буквами (без подчёркиваний). На самом деле здесь вовсе не должно быть имени, а должно быть подразделение, но поскольку проект учебный, вы можете придумать название своего собственного подразделения. Придумывайте так, чтобы никто до этого больше не догадался и «имя» было уникальным.

\_PROJECT\_ - то, как вы назвали проект в репозитории, маленькими буквами (без подчёркиваний).



Нажмите «Next». В качестве названия проекта опять же рекомендуется выбрать то, каким вы назвали репозиторий (и тоже маленькими буквами). А теперь нужно выбрать локацию: выберите именно ту директорию, в которую вы клонировали проект. «Finish».

Откроется проект и, возможно, ещё какие-то другие окна, которые пока что можно закрыть (окно подсказок, окно с вопросом от git и т.п.). Останется окно, похожее на это:



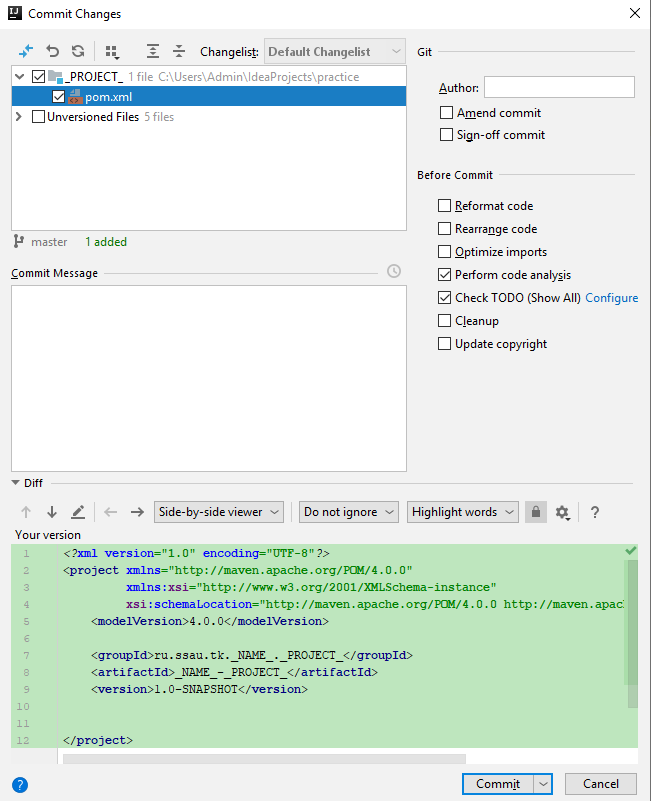
Итак, проект создан. Откройте директорию с проектом и увидите, что там появились новые папки и файлы.

Теперь требуется разместить его в репозитории в VCS. Но тут возникает вопрос: а какие файлы нужно туда записывать? Понятное дело, что файлы с исходным кодом однозначно должны туда попасть.

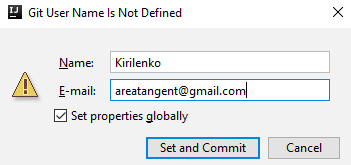
А файлы, содержащие байт-код классов, там абсолютно не нужны – они будут постоянно меняться и с каждый версией всё больше захламлять репозиторий, расходуя его память и создавая много ненужной и путающей информации в системе. Тем более файлы байт-кода всегда можно собрать из исходников.

Будем разбираться с этим в процессе разработки.

Давайте теперь произведём «коммит» изменений в наш репозиторий – т.е. запишем туда что-нибудь. Для этого следует нажать (и запомнить) сочетание клавиш Ctrl + K. Откроется новое окно:



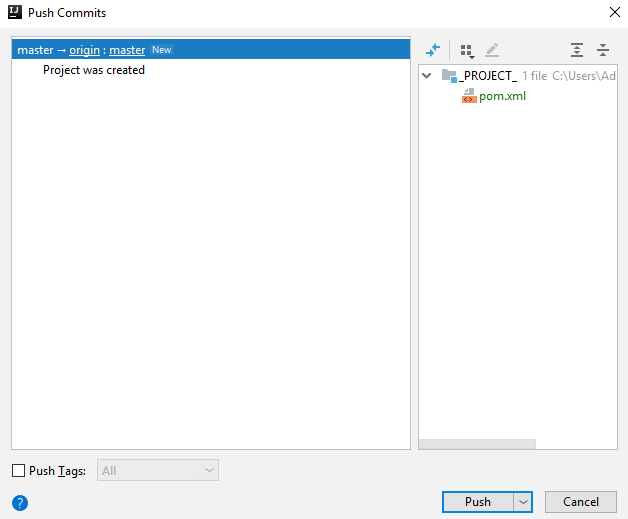
Отметьте галочкой только один файл – «pom.xml» – добавьте комментарий (Commit Message), например, «Создание проекта» и нажмите «Commit». Появится окно, предлагающее установить ваше имя и почту. Настоятельно рекомендуется в качестве имени использовать вашу фамилию.



Если всё сделано верно, то коммит будет произведён, а название файла «pom.xml» в панели навигации (слева, в директории проекта) сменит свой цвет с зелёного на чёрный.

Перейдите в браузере на репозиторий и посмотрите историю коммитов («Commits»). Скорее всего, там ничего не появилось. Дело в том, что данный коммит был создан в локальном проекте, и никуда не отправлялся. Только вы его видите. Для того, чтобы отправить все коммиты в общий репозиторий, необходимо произвести «Push». Для этого используется сочетание клавиш Ctrl + Shift + K.

Появится похожее диалоговое окно:



Здесь отражаются локальные коммиты с комментариями. Нажмите «Push». Появится окно, предлагающее ввести ваш логин и пароль от аккаунта сервиса репозиториев (например, bitbucket). Настоятельно не рекомендуется ставить галочку “Remember me”, если вы работаете за чужим компьютером!

После того, как появится сообщение об успешном «Пуше», вновь загляните в репозиторий и посмотрите историю коммитов.

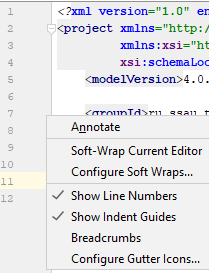
Перейдите на вкладку исходников «Source», где будет виден единственный файл – «pom.xml». Ура! Ваш проект теперь в интернете и доступен всем желающим.

Стоит отметить, что в git есть такое понятие, как ветка (branch). И сейчас работа ведётся в ветке master. Вообще разработка в master’е считается плохим тоном, но поскольку изучение VCS не входит в рамки курса и проекты учебные, то это допускается.

Важное №1: сочетание клавиш «Ctrl + T» позволяет совершить «Pull», т.е. забрать изменения из репозитория, если их кто-то туда успел запушить. Особенно актуально при командной разработке.

Важное №2: если вы пытаетесь получить изменения из репозитория, может возникнуть конфликт изменений – когда два человека меняли один и тот же файл. В этом случае конфликт нужно разрешить. Для этого надо кликнуть правой кнопкой, например, в любой файл в IDE и нажать Git -> Resolve Conflicts (если окна конфликтов не появилось). В окне конфликтов выбрать файл и нажать Merge. Появится окно с тремя колонками кода: слева – ваши изменения, справа – изменения с сервера, а по центру – результат «слияния». Нужно пройтись по всему файлу, найти куски кода, подвергшиеся изменениям (они выделяются белым, синим, зелёным и красным) и корректно «слить» их, а после нажать кнопку Apply и закоммитить изменения после исправления всех файлов.

Важное №3: откройте в IDE файл «pom.xml» (слева на закладке «Project»). Наведите на серую полосу слева от кода и нажмите «Annotate»:



Откроется весьма важная информация о том, кто и когда изменял какую строчку кода последним.

5 Конфигурирование системы сборки

Система автоматической сборки позволяет автоматизировать многие задачи, которые возникают в процессе разработки. В частности, система сборки может сама заниматься компиляцией всех файлов вместо того, чтобы это делал программист (см. пункт 1, где компилировать приходилось руками).

В рамках данного курса изучение систем автоматизации сборки не предусмотрено, поэтому требуется лишь настроить рабочую конфигурацию и забыть про неё (почти).

В созданной проекте в качестве системы сборки используется maven (он уже встроен в IDE). Информация для сборки проекта находится в файле pom.xml. Откройте этот файл в IDE.

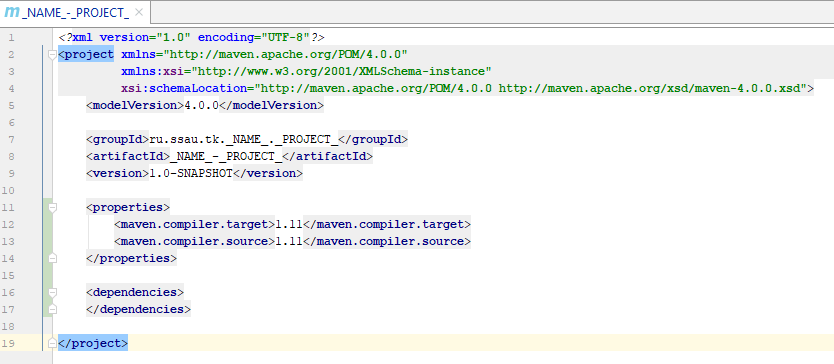
Видно, что в нём уже заданы groupId, artifactId и version, а также есть другие параметры, значения которых мы изучать не будем.

Сейчас требуется после них добавить следующие строчки:

<**properties**>  
 <**maven.compiler.target**>1.11</**maven.compiler.target**>  
 <**maven.compiler.source**>1.11</**maven.compiler.source**>  
</**properties**>  
  
<**dependencies**>  
</**dependencies**>

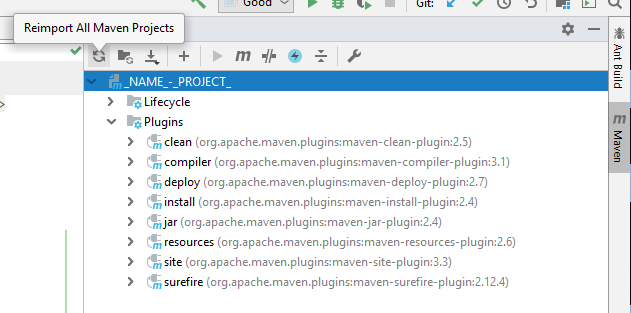
В примере выбрала версия для Java 11. Чтобы указать версию Java 8, необходимо проставить вместо 1.11 – 1.8.

Должно получиться похоже на то, что приведено на рисунке:



Обратите внимание, что добавление новых строк привело к тому, что слева появился зелёный столбик. Название файла «pom.xml» поменяло чёрный цвет на синий.

Всегда после изменения в файле «pom.xml» требуется сделать «реимпорт» проекта. Для этого найдите справа в IDE колонку с пунктом «maven», откройте её и нажмите на символ реимпорта:

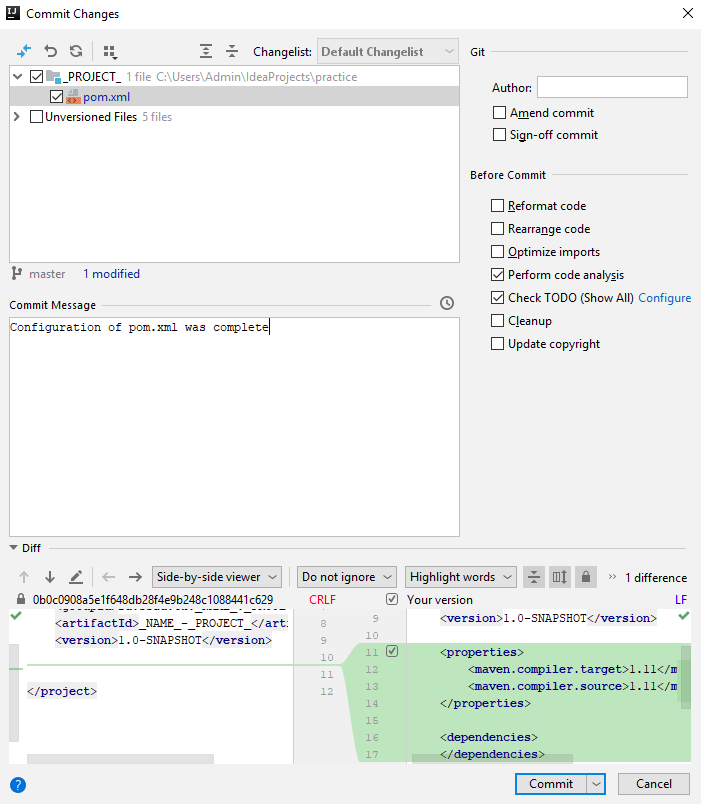


Дождитесь окончания импорта и убедитесь, что нигде не осталось текста, выделенного красным (или подчёркнутого красным). Конфигурация завершена.

6 Завершение

Теперь в проекте имеются изменения, не отражённые ни в каких коммитах (ни в локальных, ни в репозитории).

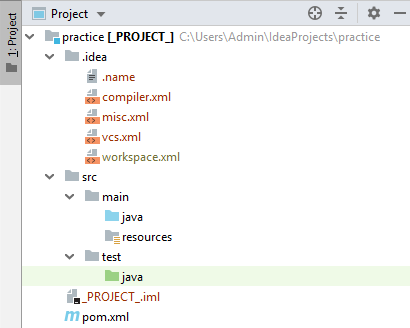
Произведите коммит и не забудьте осуществить их выгрузку в репозиторий («Пуш»).



Вновь должен быть выделен только файл pom.xml. Обратите внимание, внизу показаны версии этого файла до изменения и после. Новые строки выделены зелёным. В комментарий можно написать что-то вроде «Осуществлена конфигурация файла pom.xml».

Проверьте в браузере, что в репозитории появились новые изменения.

Изучим теперь структуру проекта:



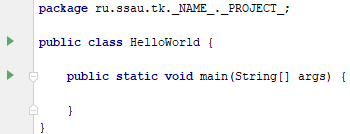
В директории есть каталог «.idea». В нём хранятся данные IDE для проекта. Они не должны попадать в репозиторий, а потому выделены специальными цветами – не зелёным, не синим и не чёрным. То же самое касается файла «\_PROJECT\_.iml».

Директория src предназначена для хранения исходного кода программы. В ней лежат каталоги main и test. Второй отведён для автоматизации процесса тестирования и пока не будет рассматриваться. Для него символ папки окрашен зелёным.

В каталоге main в свою очередь ещё два каталога – java и resources. В первом должны лежать исходные коды файлов Java. Символ папки окрашен синим. Во втором хранятся различные файлы, если они необходимы для работы программы.

Теперь пришло время для программирования. Нажмите правой кнопкой на директорию src/main/java и создайте пакет (Package) «ru.ssau.tk.\_NAME\_.\_PROJECT\_», заменив \_NAME\_ и \_PROJECT\_ на соответствующие названия. В дальнейшем не следует создавать классы и пакеты вне этого пакета.

Создайте по аналогии класс (Java Class) в этом пакете с названием HelloWorld. IDE тут же предложит добавить вам этот класс в систему контроля версий, с чем нужно согласиться. В классе тут же появятся объявление пакета и объявление самого класса. Внутри класса на новой строке наберите «psvm» и нажмите Tab. Обратите внимание на появившийся готовый метод. Рядом с методом (и объявлением класса) появился символ запуска:



Напишите в теле метода «sout» и нажмите Tab. Добавьте внутри вызова метода println() следующую строку: **"Hello, World!"** В результате файл должен иметь такое содержимое:

**package** ru.ssau.tk.\_NAME\_.\_PROJECT\_;  
  
**public class** HelloWorld {  
  
 **public static void** main(String[] args) {  
 System.***out***.println(**"Hello, World!"**);  
 }  
}

Ещё одна крайне полезная функция IDE – форматирование кода. Работая с файлом, используйте сочетание клавиш Ctrl + Alt + L. Если у вас были где-то нарушения отступов, они будут исправлены.

Запустите выполнение класса. Система сборки сама соберёт проект, создаст байт-коды файлов и запустит их. В консоли внизу должна появиться надпись «Hello, World!».

Обратите внимание, что в каталоге проекта появилась папка target. В ней и будут храниться все файлы «.class». Поскольку байт-коды классов всегда можно создать из исходников, не имеет никакого смысла хранить их в репозитории, поэтому target (и её подпапки) окрашены в красный.

Сделайте коммит (и пуш) с комментарием «Добавлен первый Java-класс» или похожим. Убедитесь, что в репозитории появились ваши изменения. Отправьте ссылку на репозиторию преподавателю для проверки.