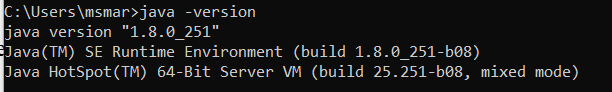
Урок 1

1. Установить Grooyy на локальной машине

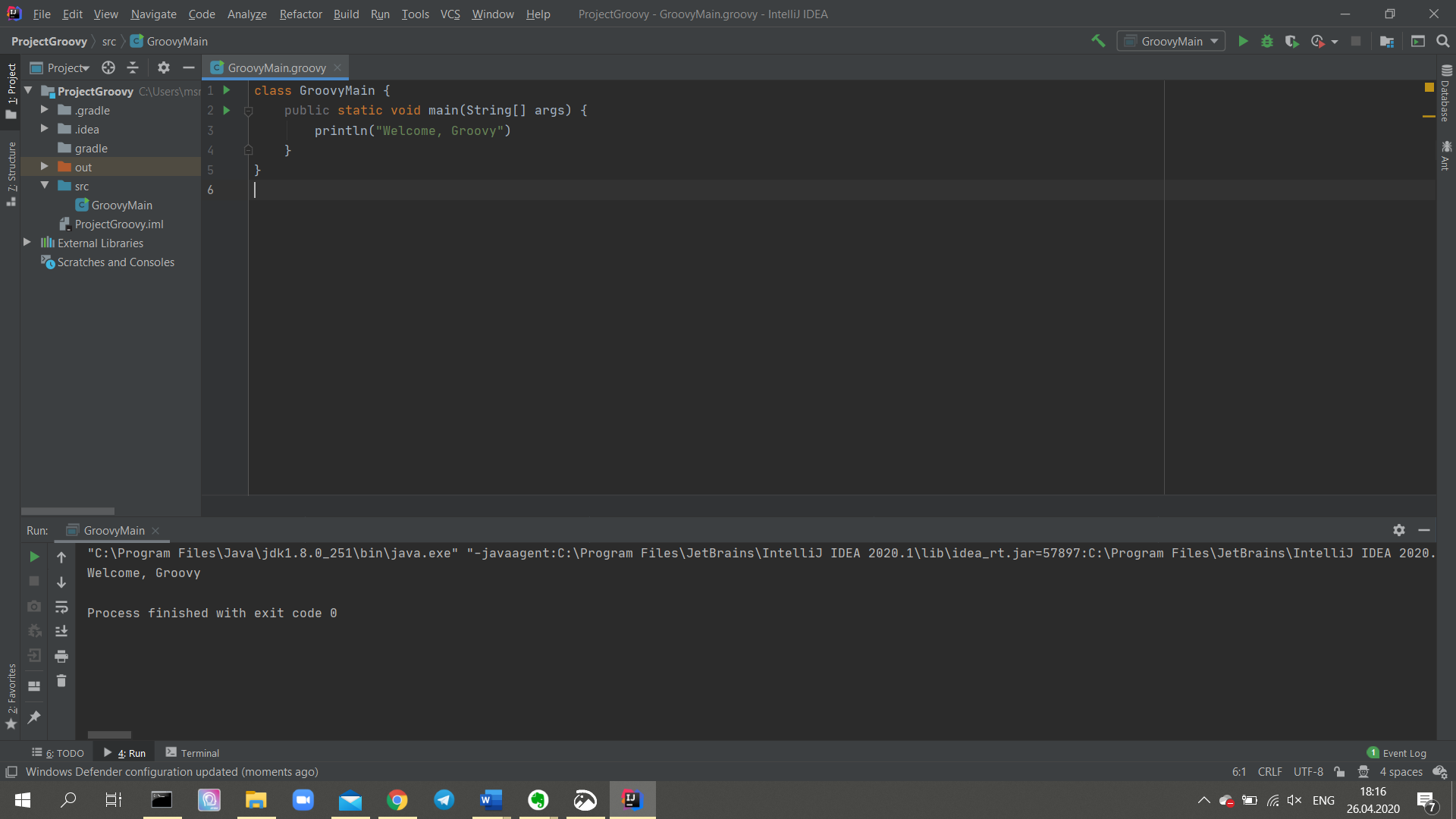
* Проверить установку JDK или JRE версии не ниже 1.7



* Запустить файл groovy-2.4.4-installer.exe и выбрать директорию для установки Groovy (далее – ДиректорияG).
* Создать переменную окружения GROOVY\_HOME = ДиректорияG
* Добавить %GROOVY\_HOME%\bin в переменную окружения PATH
* Для проверки правильности установки выполнить команду groovy -v



2. В Idea создать проект с именем ProjectGroovy для работы с Groovy



Как видно, ничего не изменилось в результате, но есть изменения в самом файле.

Попытаемся инициализировать переменные класса Person с указанием параметров по имени, передав им произвольные значения.

8. Проверка работы с конструкторами.

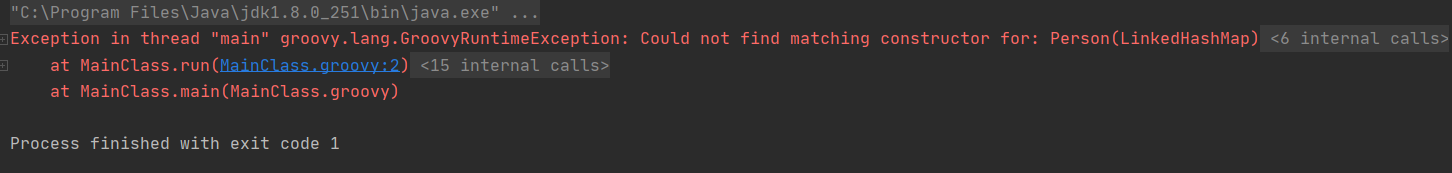
* В созданном ранее классе FirstClass создать несколько переменных уровня экземпляра класса (имена и типы выберите сами). Конструктов в этом классе создавать пока не требуется.
* Создайте класс Groovy с именем MainClass и удалите из файла этого класса весь текст.
* В файле MainClass.groovy создайте экземпляр класса FirstClass и попытайтесь инициализировать его конструктором с указанием параметров по имени, передав им произвольные значения



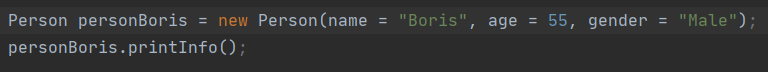


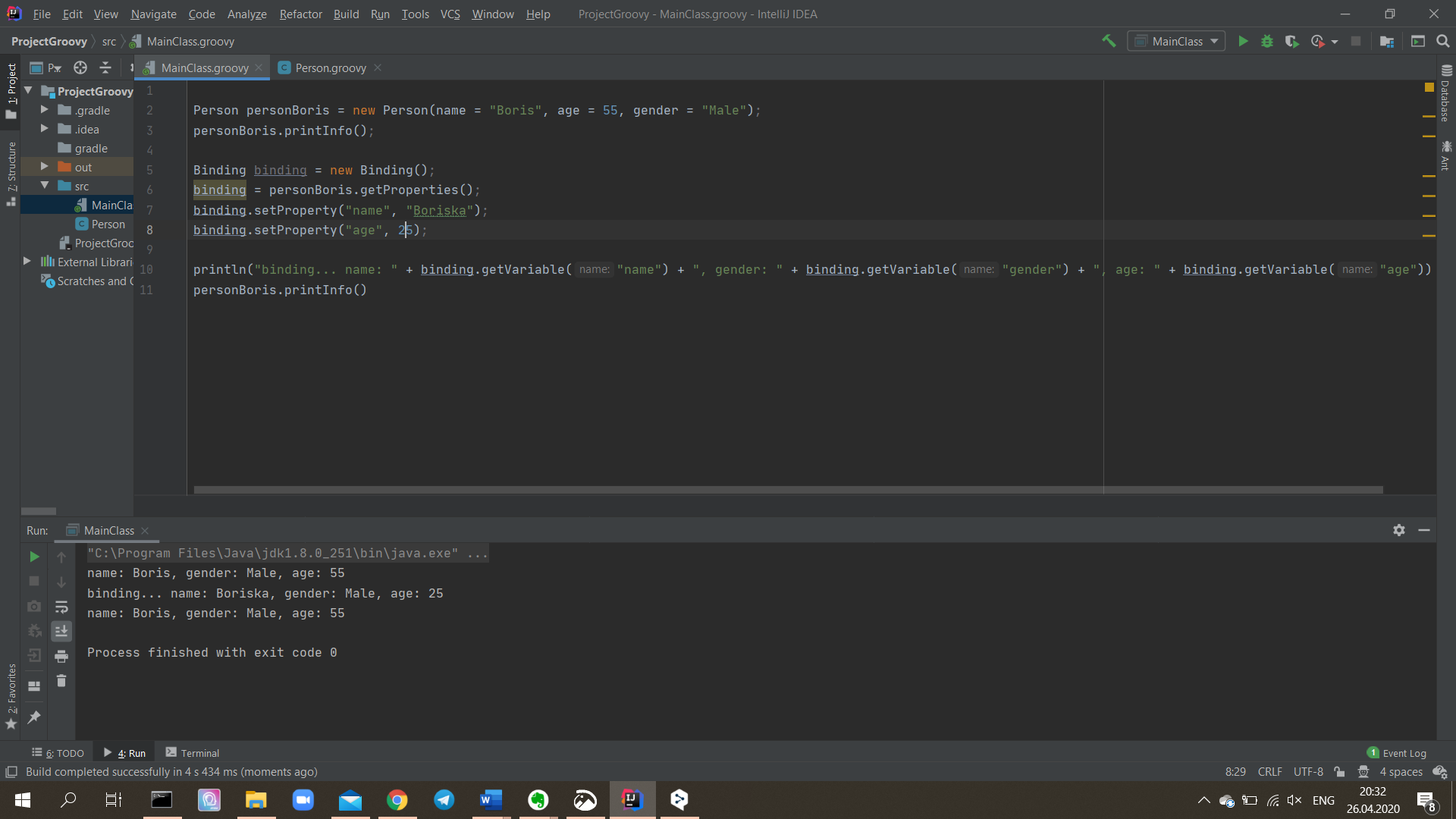
* В классе FirstClass создайте явный конструктор и попытайтесь в файле MainClass создать экземпляр класса FirstClass. Возможно ли это сделать с поименной передачей параметров в этот конструктор.

С предыдущим вариантом:



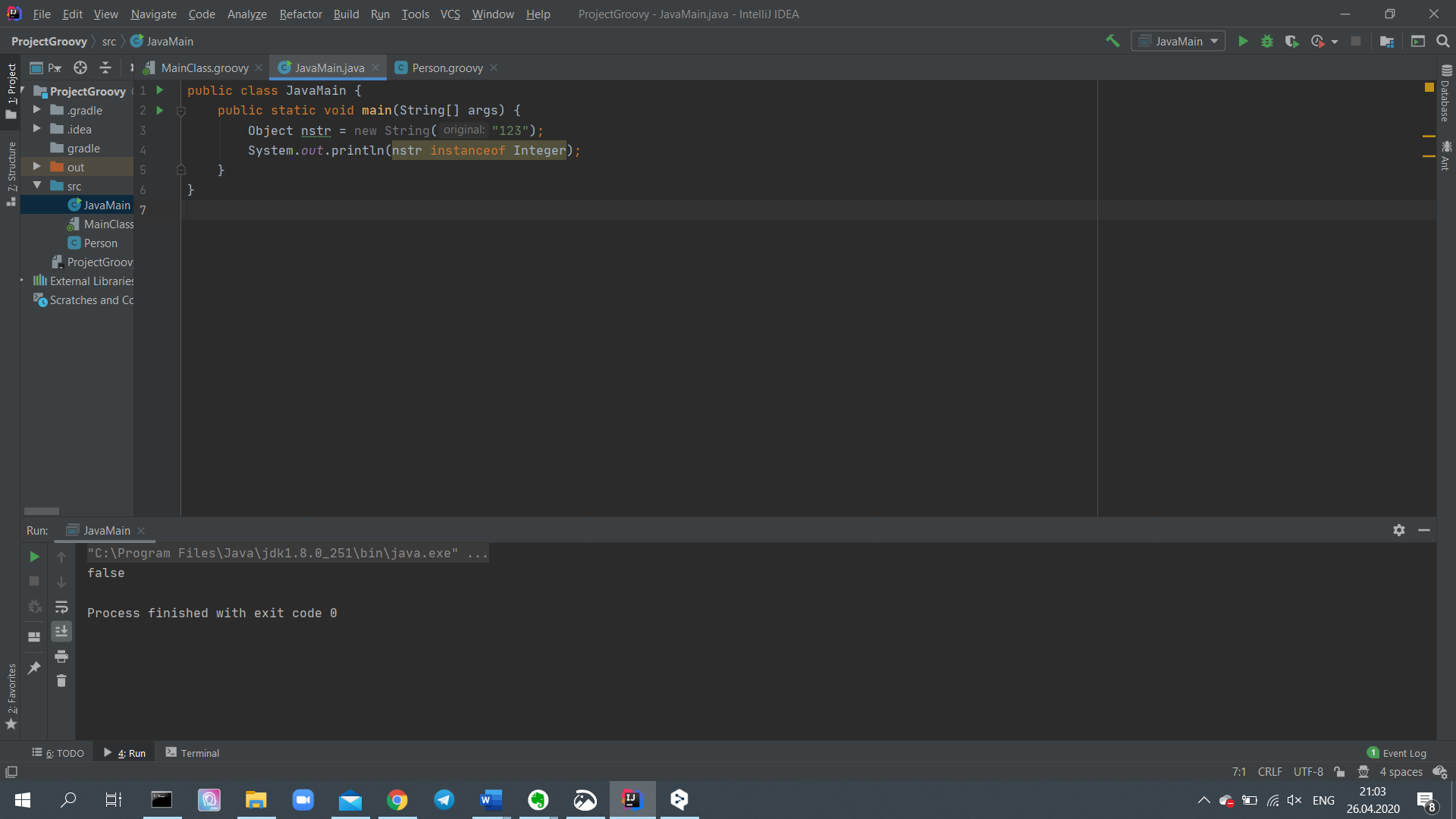
Но можно так:

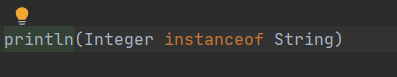
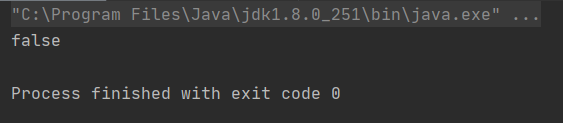


9. При помощи экземпляра класса Binding задайте значение переменных экземпляра класса FirstClass и получите это значение. Будут ли видны эти значения при помощи get- и set-методов класса FirstClass? 

Мы можем взять значение переменных экземпляра, поменять их и вывести, однако сами значения не меняются для экземпляра. При помощи set-методов возможно поменять значение переменных.

10. Проверьте при помощи оператора instanseof отношения между экземпляра класса String и классом Integer. Сделайте то же самое в Java-классе. В чем разница в операторе instanceof для Groovy и Java ?



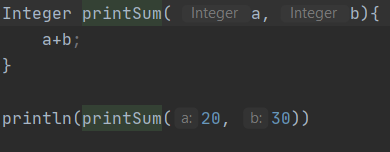
JDK не дает изначально проверить Integer instanceof String. В Java instanceof можно использовать для проверки, является ли объект экземпляром класса или же экземпляром подкласса. В Groovy же мы можем просмотреть является ли класс подклассом другого класса.

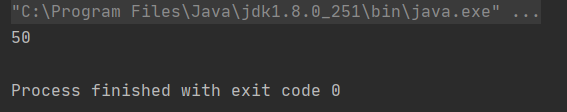
11. Проверить необходимость оператора def и явного указания типа для «сокрытия» переменных уровня экземпляра класса от метода. Сделать это в «явном» классе и «неявном».

Def == безтиповая переменная

???

12. Создайте метод с возвращаемым значением типа Integer, но без оператора return. Что надо сделать, чтобы метод выполнился без ошибок?

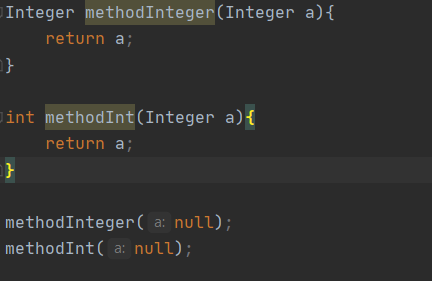




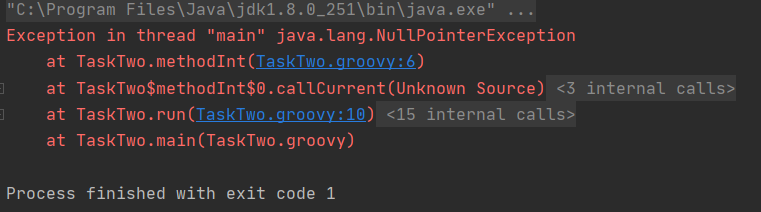
Конечная строка должна содержать результат выполнения/переменную типа Integer. Если, например, последняя строка будет содержать инициализацию строки, то возникнет ошибка GroovyCastException.

Урок 2

1. Создайте два метода, один из которых возвращает тип Integer, а второй – тип int. В обоих методах должен быть один параметр тип Integer. Попытайтесь выполнить эти методы, передав в качестве параметра значение null. Какой из методов выполнится?

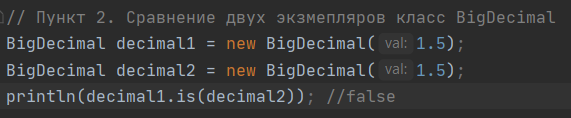


Ошибку выдает methodInt.



1. *Создайте два экземпляра класса BigDecimal при помощи оператора new. Присвойте этим переменным одинаковый значения (с использованием конструктора). Убедитесь, что эти переменные не указывают на одно и то же место в памяти. Сравните значения переменных при помощи метода equals. Какие методы можно использовать для сравнения?*

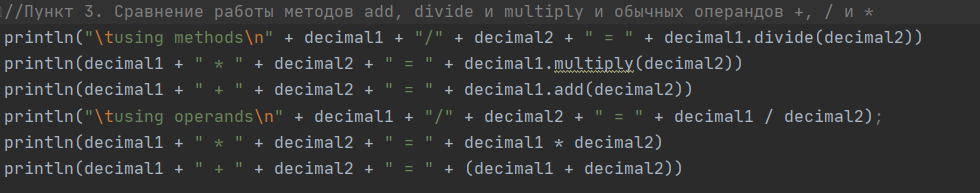
В Java == сравнивает равнозначность ссылок переменных, в Groovy же данное сравнение ведет себя по-другому – сравнивает значение переменных. Чтобы убедится, что переменные не указывают на одно и то же место в памяти мы можем использовать метод is().

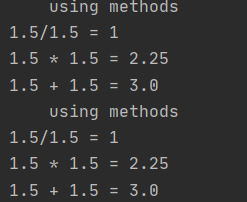


Для сравнение значений перемнных можем использовать == или equals(). Более того, IntelliJ IDEA считает == и equals() равнозначными и предлагает метод equals() заменить на оператор == для лучшей читабельности кода.



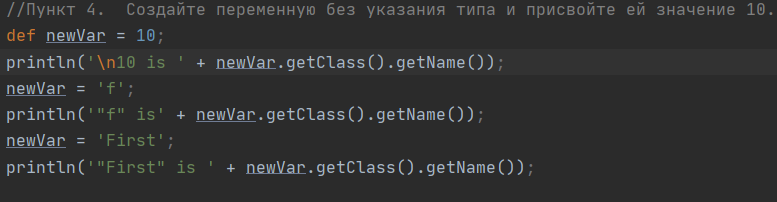
1. *С этими же переменными попробуйте использовать методы add, divide и mult. Сравните результат этих операций с использованием обычных операндов +, / и \*. Можно ли сделать то же самое в Java?*

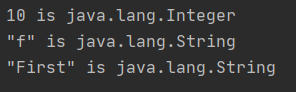
** Результат:

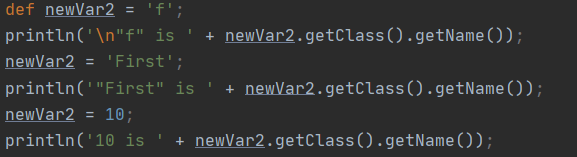
**

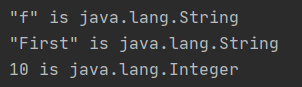
Как видно, Groovy поддерживают перегрузку использованных математических операторов +, / и \*. В Java тоже самое сделать нельзя.

1. Создайте переменную без указания типа и присвойте ей значение 10. Получится ли присвоить этой же переменной значение ‘f’ или значение ‘First’. Проверьте тип этой переменной при помощи метода getClass().getName(). Попробуйте поменять последовательность этих присвоений – сперва ‘f’, потом ‘First’, потом 10 и т.д.

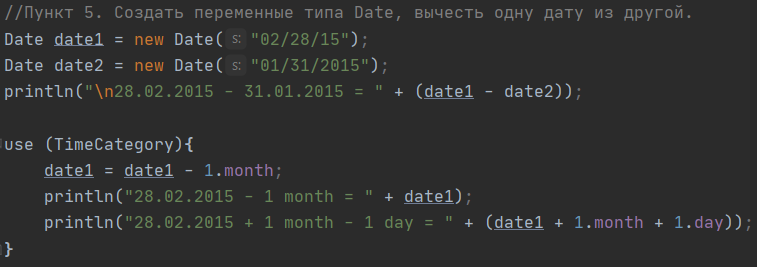


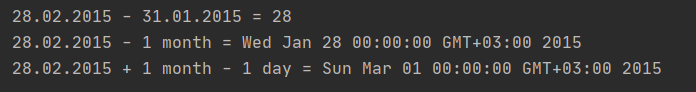


Поменяем последовательность 



1. Создайте две переменные типа Date и присвойте им значения 28 февраля 2015 года и 31 января 2015 года. Вычтите одну переменную из другой, а потом вычтите из 28 февраля 1 месяц, а затем прибавьте к результату 1 день и 1 месяц (это можно сделать двумя операциями). Что получилось?



Результат: 

TimeCategory позволяет удобно манипулировать датами.

Урок 3

Closure — это замыкание и представляет собой некую анонимную функцию со свойствами объекта.

Синтаксис замыкания: {[closureArguments->] statements}

Замыкание — это особый вид функции. Она определена в теле другой функции и создаётся каждый раз во время её выполнения. Синтаксически это выглядит как функция, находящаяся целиком в теле другой функции. При этом вложенная внутренняя функция содержит ссылки на локальные переменные внешней функции. Каждый раз при выполнении внешней функции происходит создание нового экземпляра внутренней функции, с новыми ссылками на переменные внешней функции.

Пример:

def closureFunction = {a, b ->

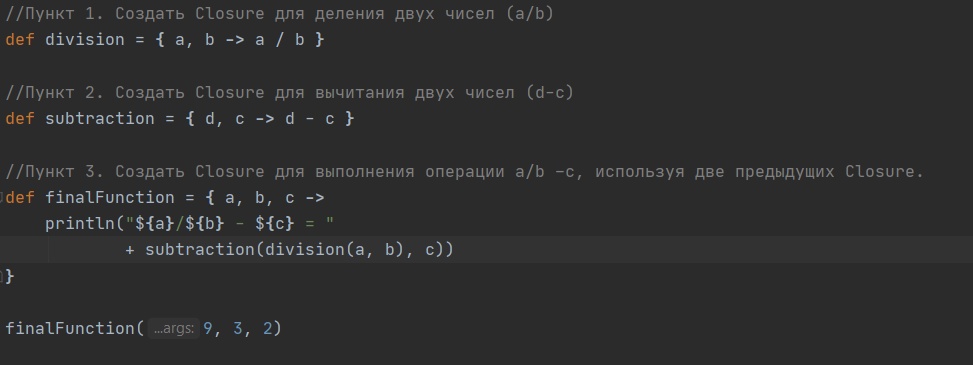
println a

println b

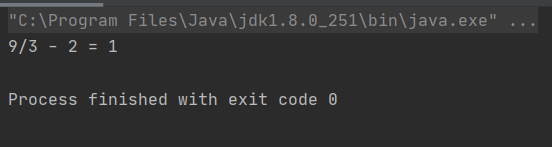
}

closureFunction(1, 2)

1. Создать Сlosure для деления двух чисел. (a/b)
2. Создать Closure для вычитания двух чисел (d-c)
3. Создать Closure для выполнения операции a/b –c, используя две предыдущих Closure.

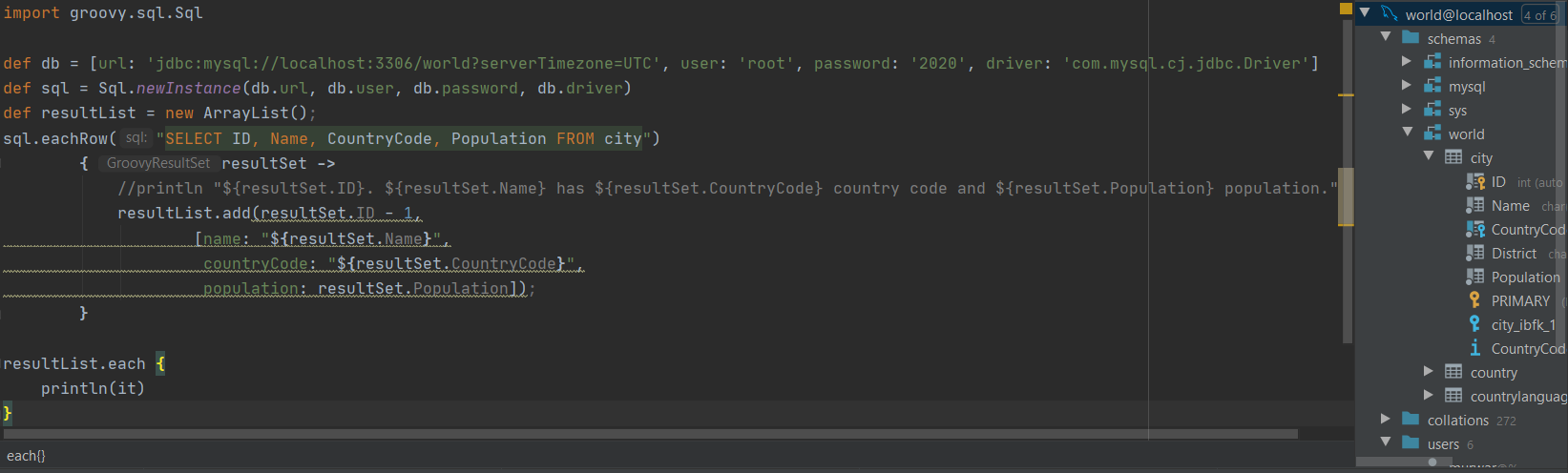


Результат:

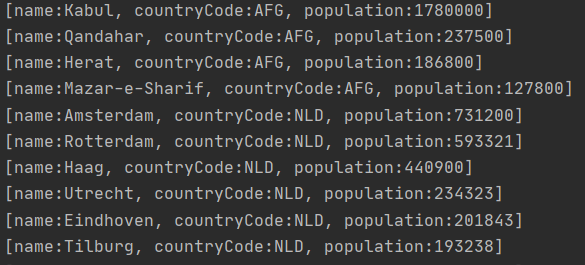


Урок 4

1. Подключиться из-под Groovy к базе данных Oracle (данные подключения высылались) под пользователем hrXX (01-12).
2. Выполнить запрос к таблице employees и сохранить результат в коллекции.
3. Проверить количество строк в коллекции.

****

**Пример вывода:**

****

**Количество записей 4079**