

## № 2 Введение в JAVA

### Задание

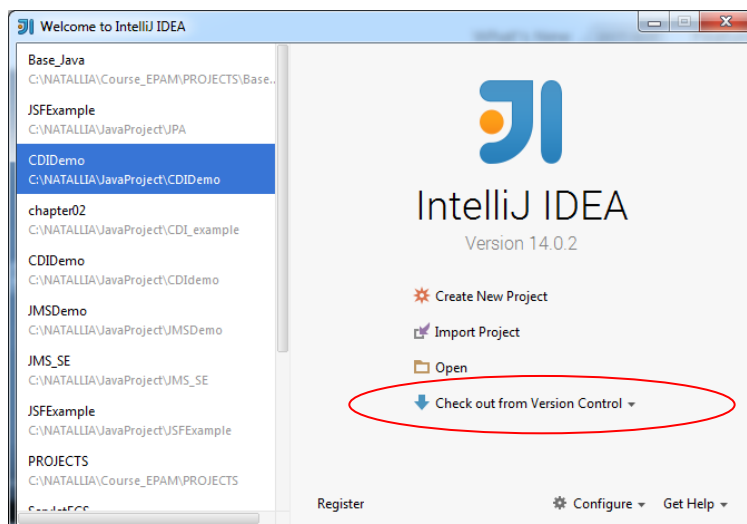
1) **Хорошо** изучите материал по темам (будьте готовы отвечать на вопросы):

- ✓ Зарезервированные слова в Java
- ✓ Зарезервированные имена методов Java
- ✓ Идентификаторы
- ✓ Литералы (целые, с плавающей точкой, логические, символьные, строчные) и константы
- ✓ Управляющие символы
- ✓ Операции и примитивные типы данных (целые типы, числа с плавающей точкой, символы и кодировки)
- ✓ Разделители
- ✓ Переменные
- ✓ Операторы
- ✓ Классы оболочки (ссылочные типы данных)
- ✓ Массивы (многомерные массивы)
- ✓ Тип String
- ✓ Консольный ввод-вывод
- ✓ Документирование кода
- ✓ Code convention

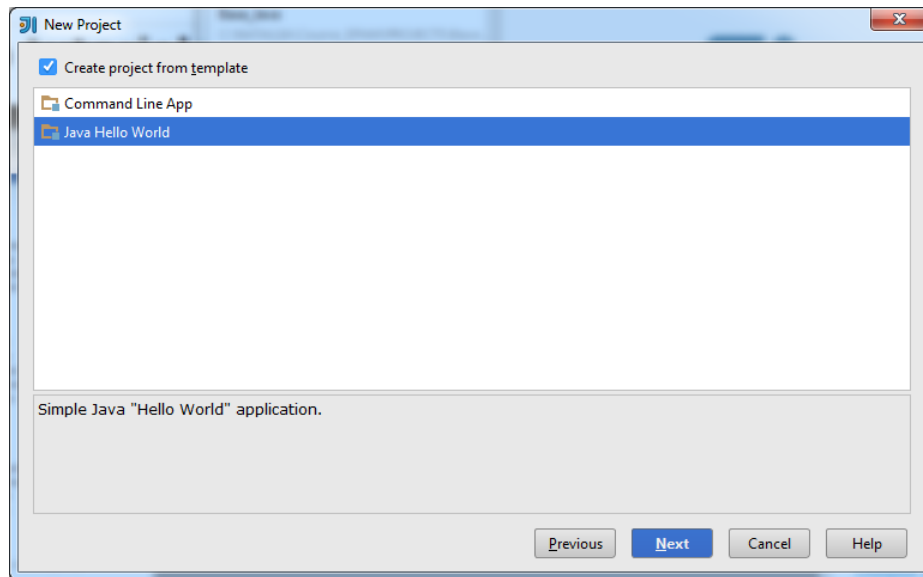
2) **Изучение структуру проекта, работу в среде**

Используйте видео : <https://www.jetbrains.com/idea/documentation/>  
или <https://www.jetbrains.com/help/idea/2016.3/general-guidelines.html>

а) Настройте связь с GitHub и репозиторием.



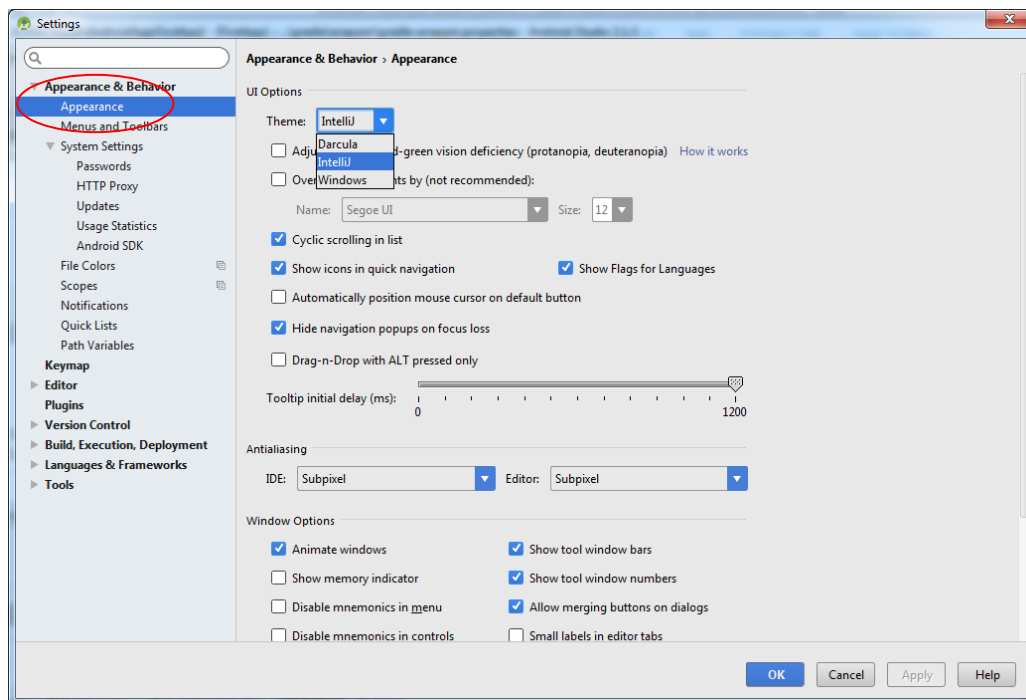
## Создайте новый проект



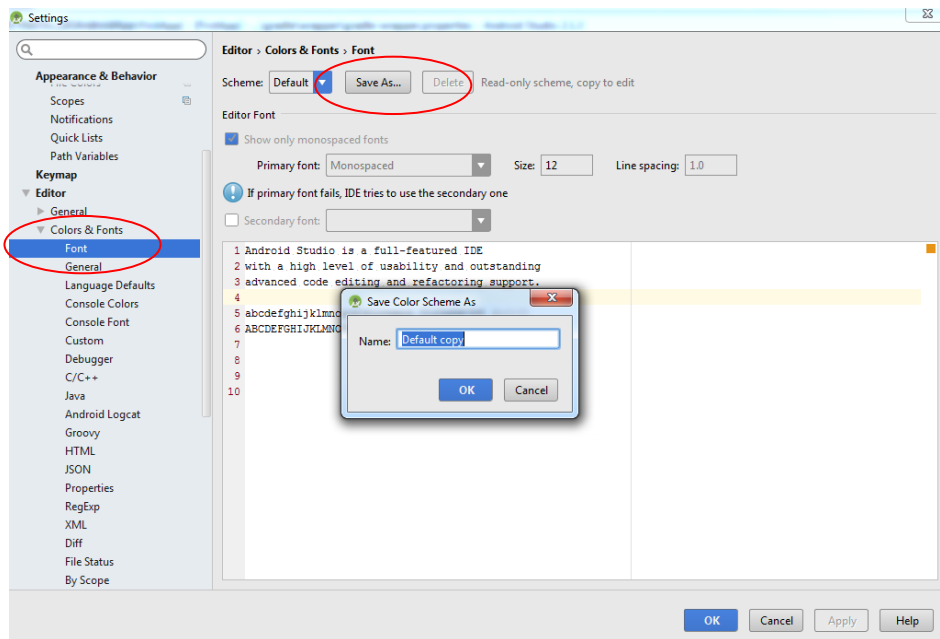
На основе созданного пустого проекта разберитесь с назначением папок.

### б) Настройте пользовательский интерфейс

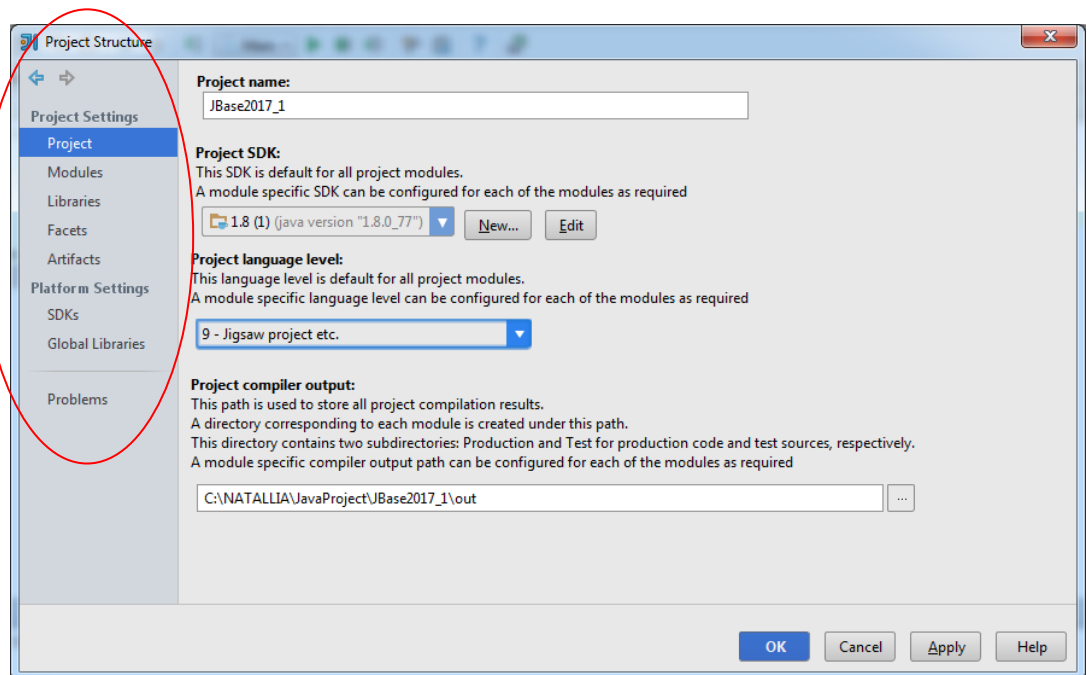
Перейдите File->Settings (ctrl-alt-s) затем Appearance



Выберите тему, можете подобрать себе цветовую схему, шрифт и размер. Editor -> Font. Предварительно нужно сохранить файл с настройками.

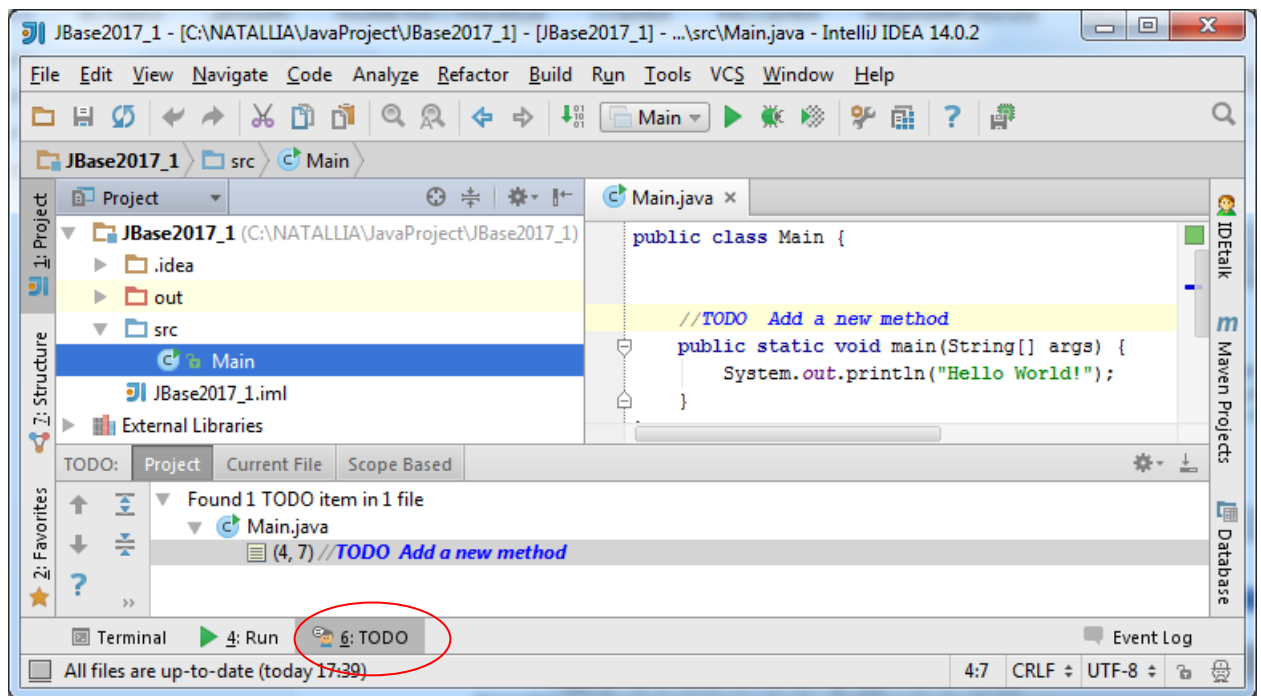


с) Изучите Структуру проекта (File→Project Structure)



Поменяйте уровень языка, разберитесь с вкладками Libraries, Artifacts и т.д.

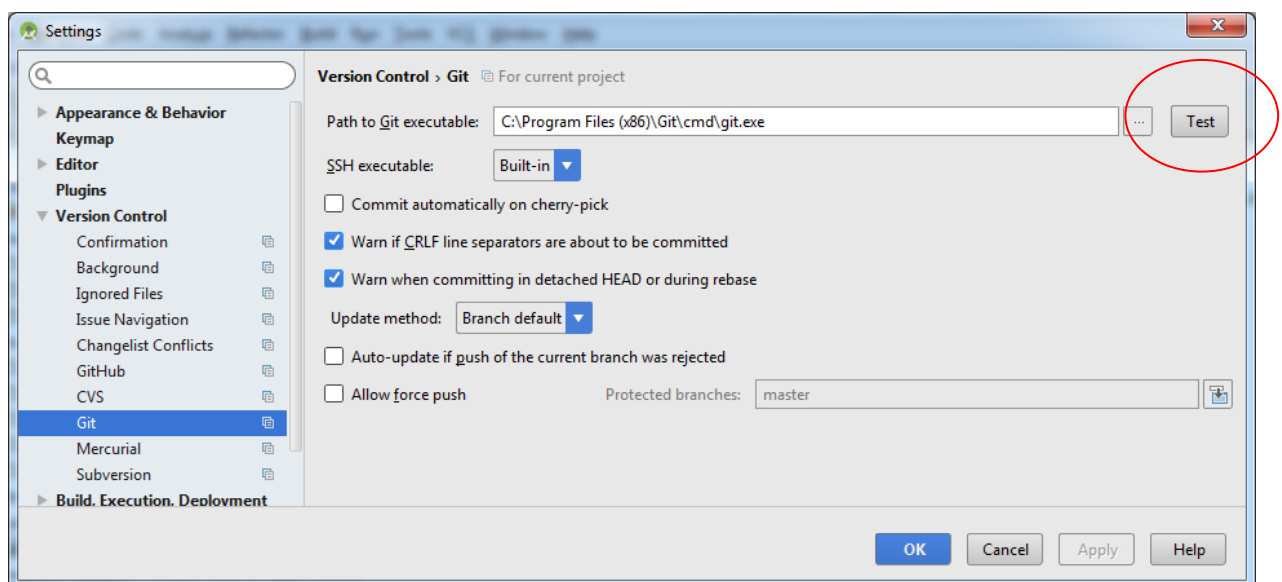
d) Добавьте в класс комментарий, который начинается с TODO.  
Найдите на нижней панели TODO. Перейдите по комментарию.



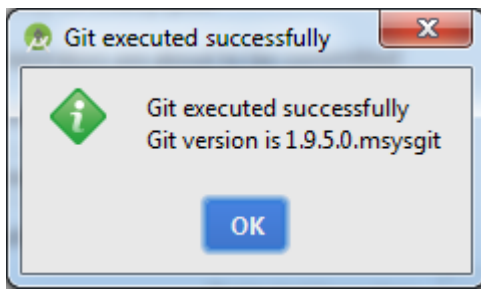
Перейдите на вкладки в нижней панели Terminal, Run. Каково их назначение?

#### е) Интеграция с Git

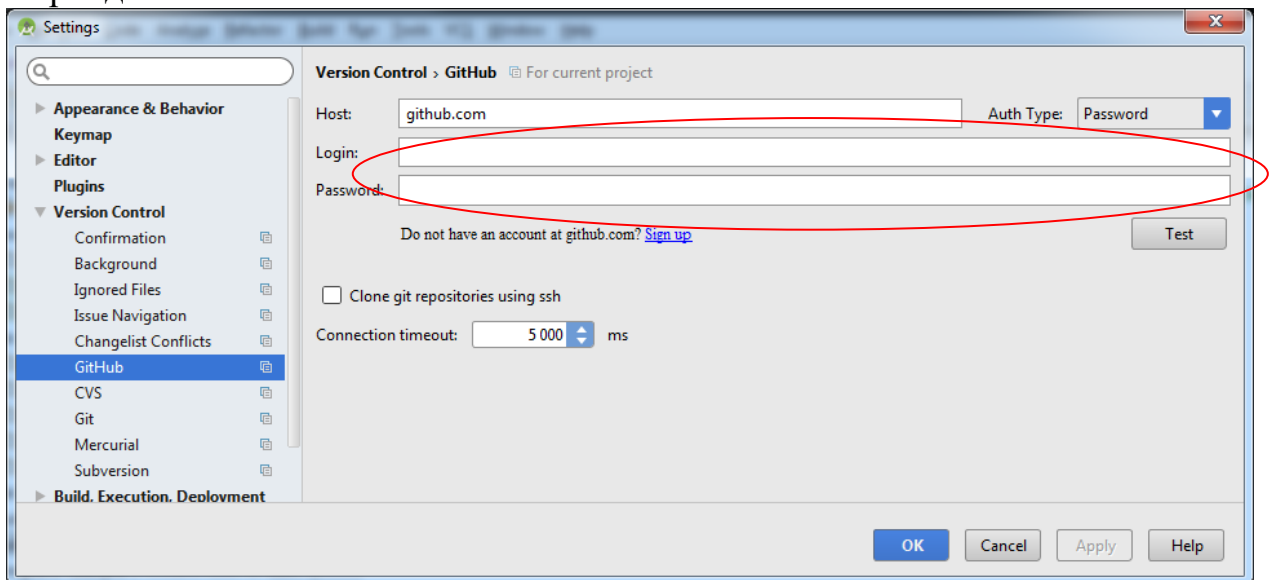
Перейдите в Settings -> Version Control -> Git



Нажмите на кнопку Test чтобы проверить что он установлен

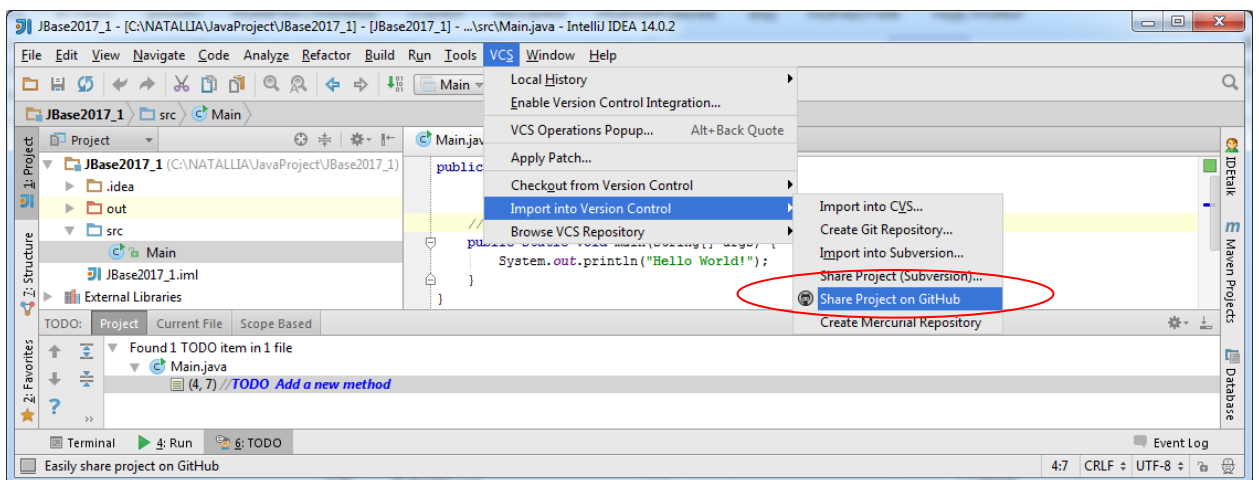


Перейдите на GitHub



Ведите логин и пароль и проверьте подключение (аккаунт вы заводили)

Выполните публикацию проекта в удаленном репозитории  
VCS -> Import into Version Control -> Share Project on GitHub

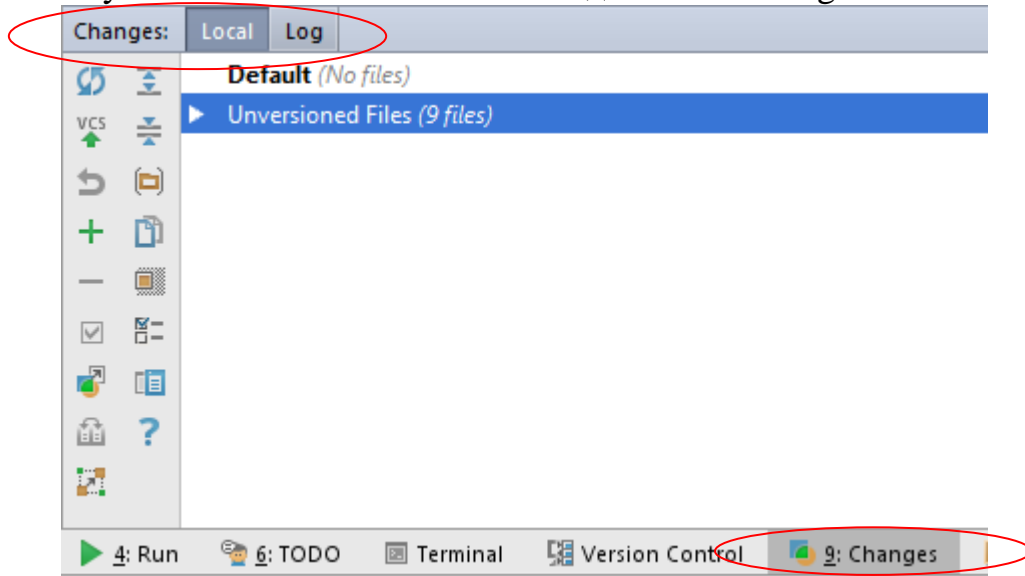


Задайте имя проекту и выполните Share.

Удалите текущий проект с диска. Запустите Idea и выберите Project from VS  
->Github

Клонируйте репозиторий и откройте его. Выполните команды commit, pull, push и т.д из меню VCS.

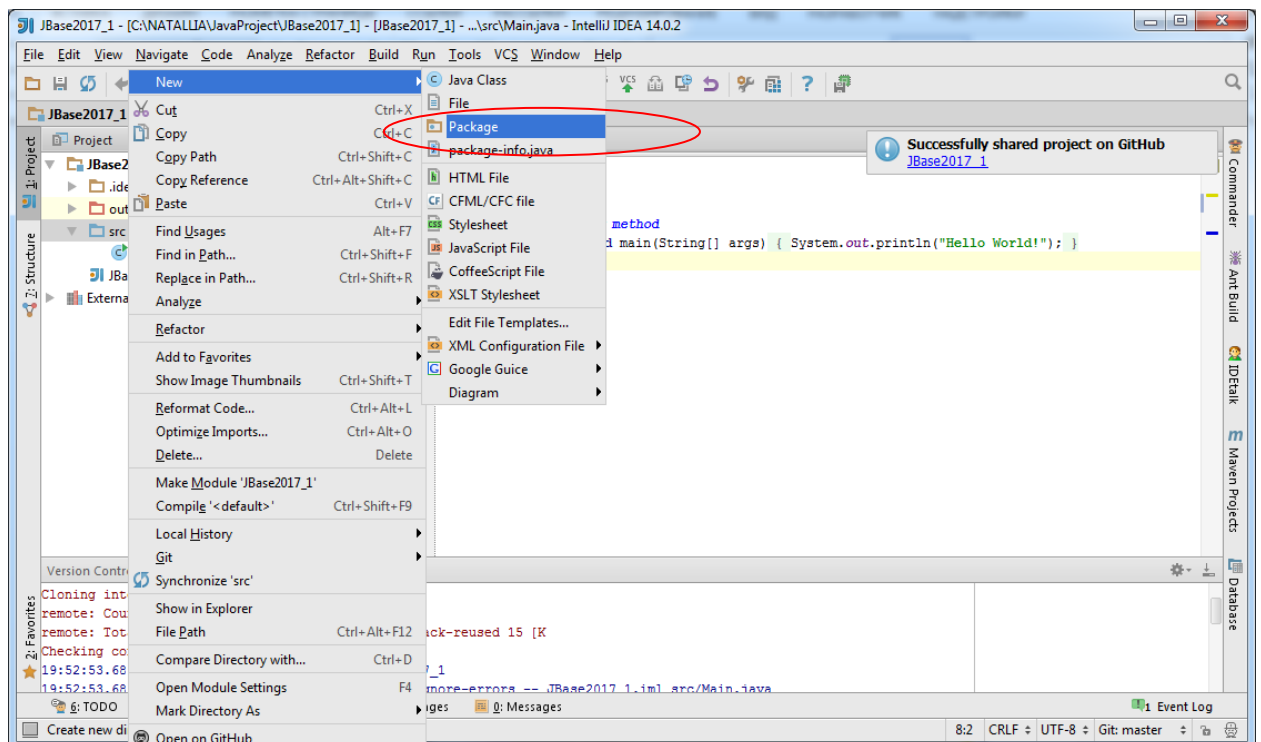
Изучите назначение и соств вкладок Local и Log



#### f) Создание пакетов и классов

Создайте новый пакет и назовите его by.belstu.it.фамилия

Создается из контекстного меню New



Затем сойздайте новый класс со следующим методом.

```

public class TextFunction {

    public String getValue()
    {return "Hello from First project";}

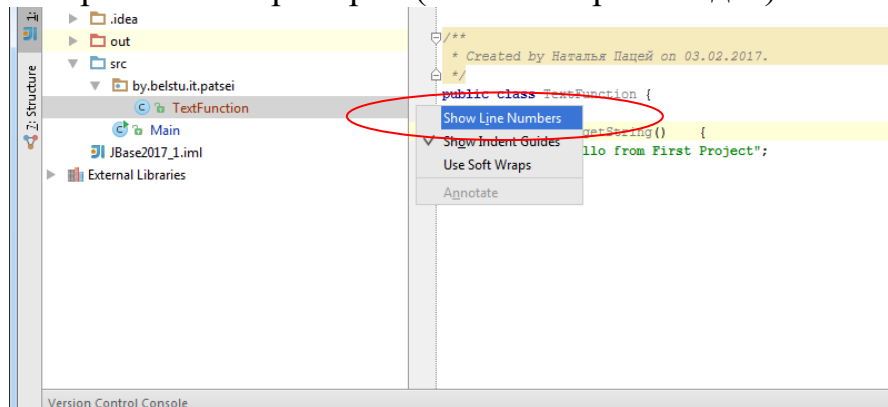
}

```

Переименуйте его – Refactor – Rename (Shift-F6).

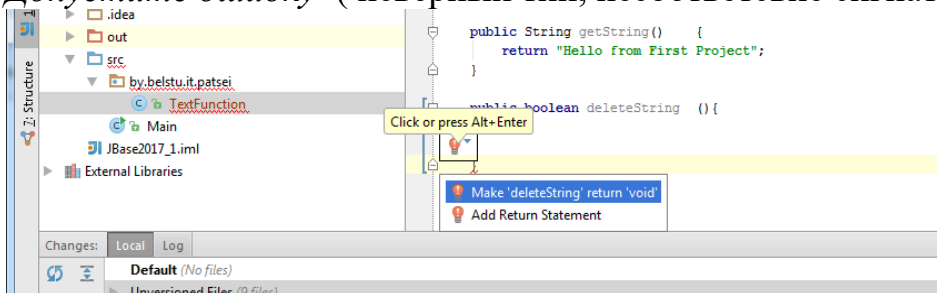
Переместите созданный класс в пакет. Обратите внимание как поменялось имя package

Отобразите номера строк ( полезно при отладке)



j) Возможности анализа кода.

Допустите ошибку ( неверный тип, несоответствие сигнатуры и т.п.)



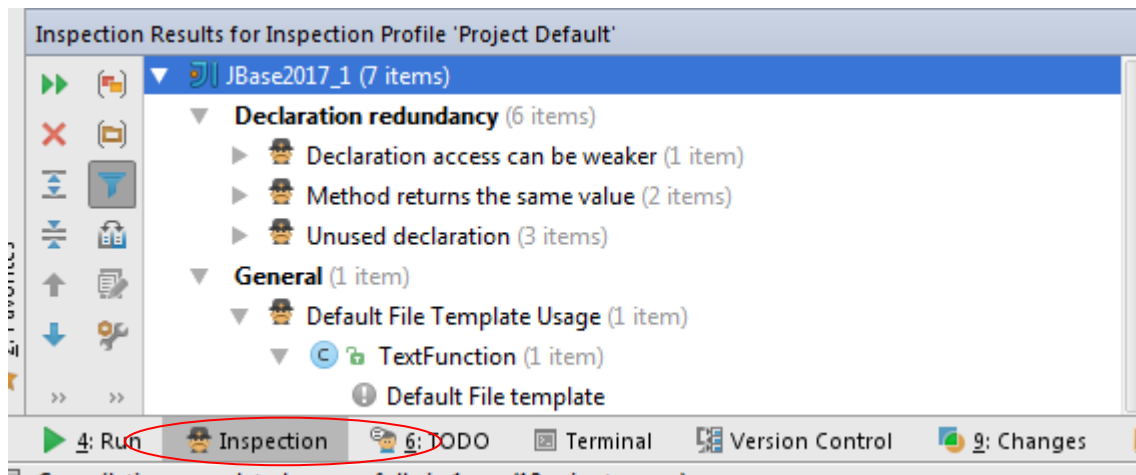
Нажмите на красную лампочку и устраните ошибки выбрав нужный вариант устранения.

Теперь создайте объект созданного класса и выполните import class для импорта класса.

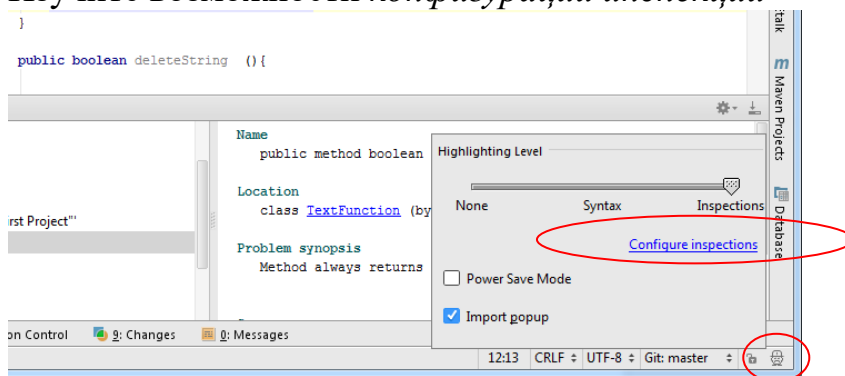
Допишите в метод любой код и сделайте *рефакторинг*: выделите написанный фрагмент в метод.

Для этого выделите код. Refactor -> Extract->Method. Задайте имя методу.

Выполните *инспекцию* написанного кода Alt+Shift+I и изучите профайл результата и поясните его результат

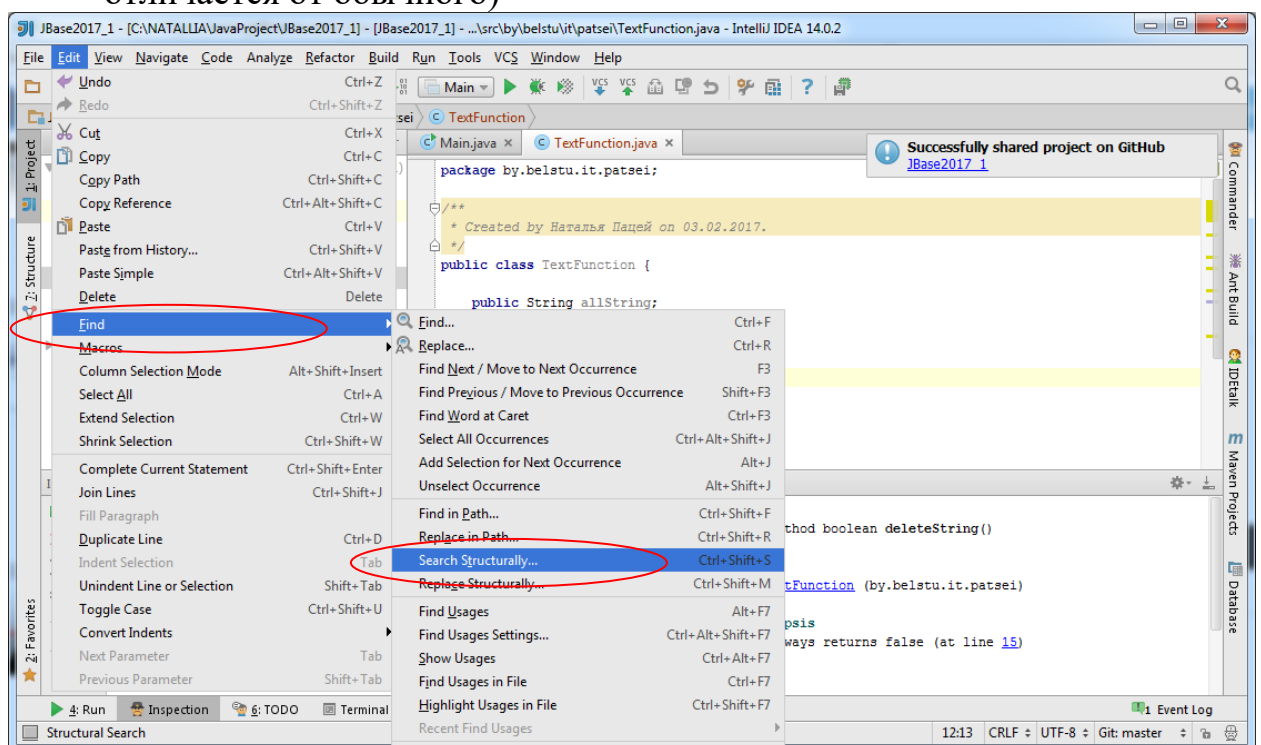


Изучите возможности *конфигурации инспекций*



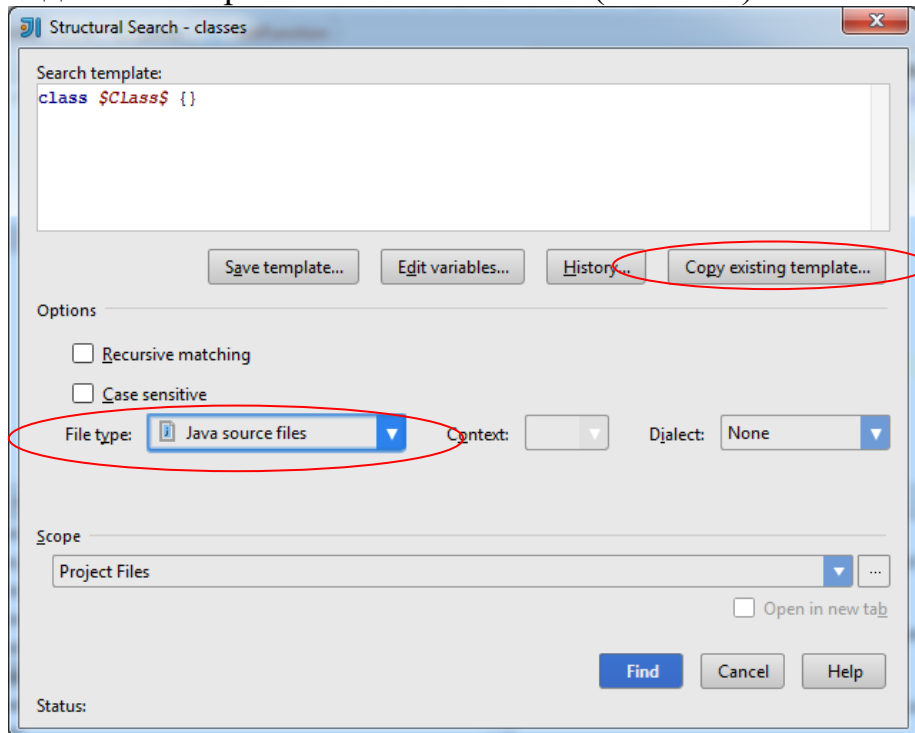
Посмотрите что еще можно выполнить в пункте Analyze.

Выполните *структурированный поиск и замену* (объясните чем он отличается от обычного)

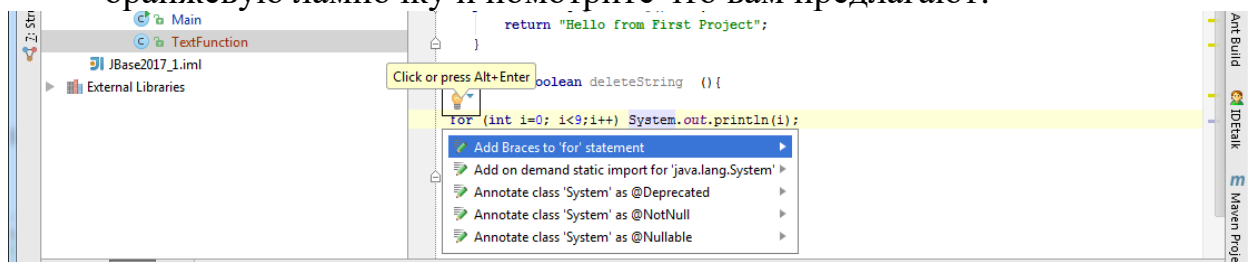




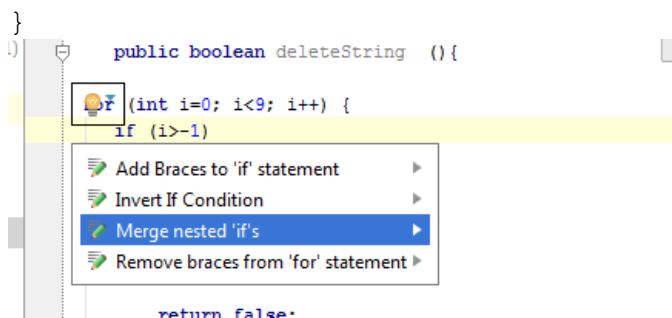
Задайте и сохраните шаблон поиска (замнены)



Изучите возможности *генерации кода*. Напишите цикл, нажмите на оранжевую лампочку и посмотрите что вам предлагают:

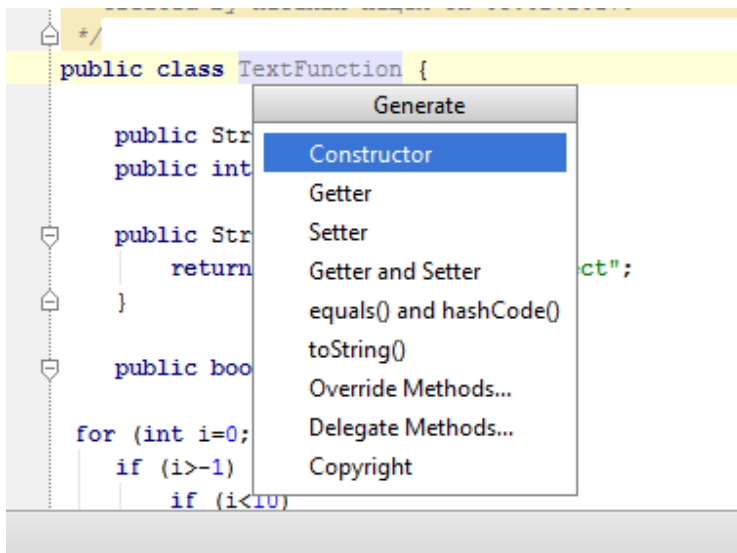


```
for (int i=0; i<9; i++) {  
    if (i>-1)  
        if (i<10)  
            System.out.println(i);  
}
```

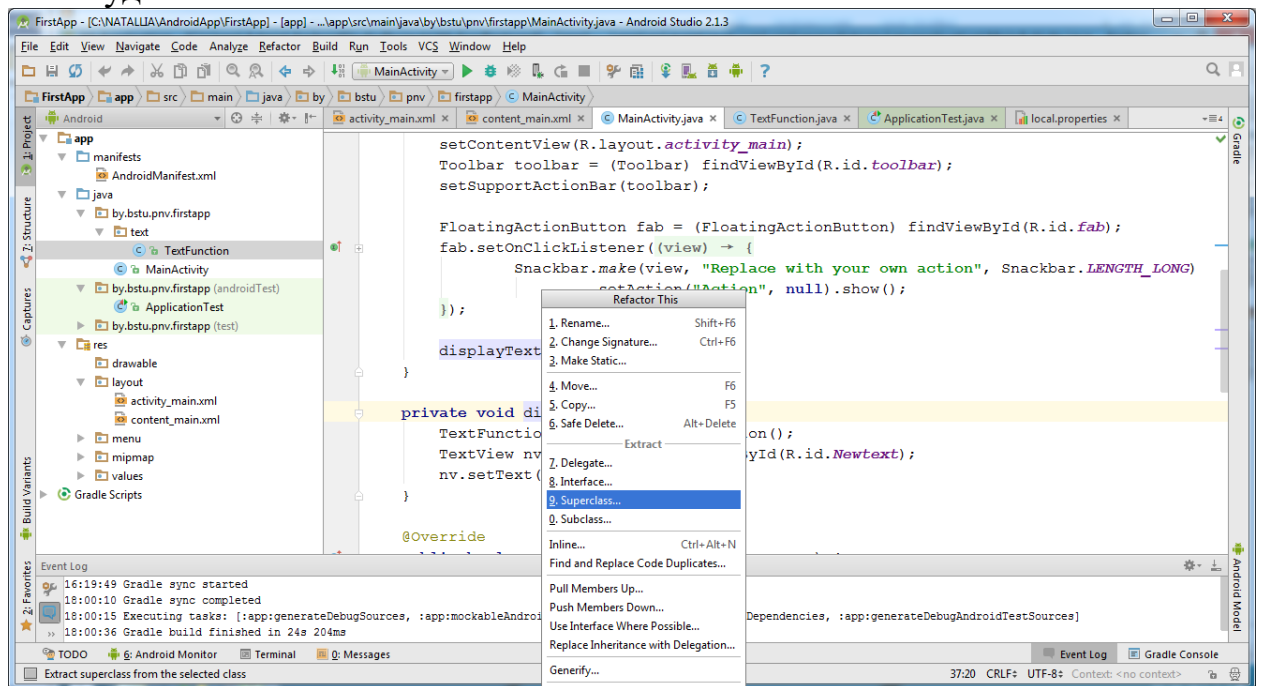


Попробуйте разные варианты.

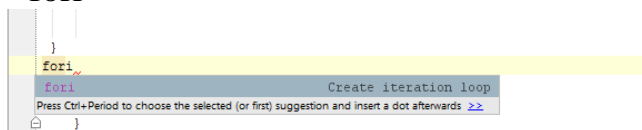
В class TextFunction проверьте возможности генерации кода (Alt+insert ).  
Сгенерируйте конструктор, сеттеры и геттеры, посмотрите что можно переопределить.



Изучите возможности рефакторинга Alt-Shift-Ctrl-T. Они достаточно удобные.



г) Использование средств отладки  
Создайте цикл. Наберите в методе onCreate  
for



Поменяйте имя переменной в выделенной области

```

    });

    for (int i = 0; i < ; i++) {

    }
}

```

Нажмите enter и вставьте диапазон

```

for (int count = 0; count < ; count++) {

```

```

for (int count = 0; count < 10; count++) {
    System.out.println("Counter " + count);
}

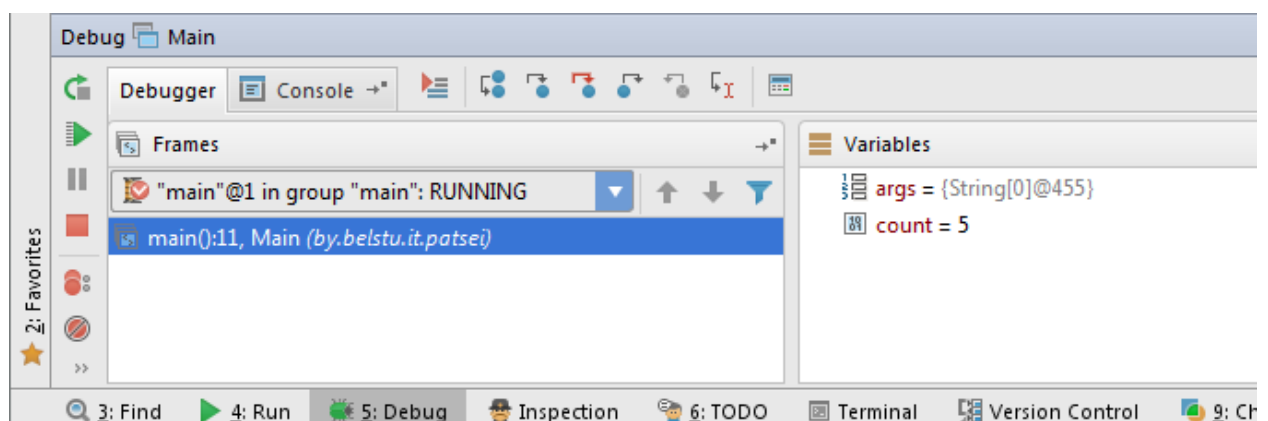
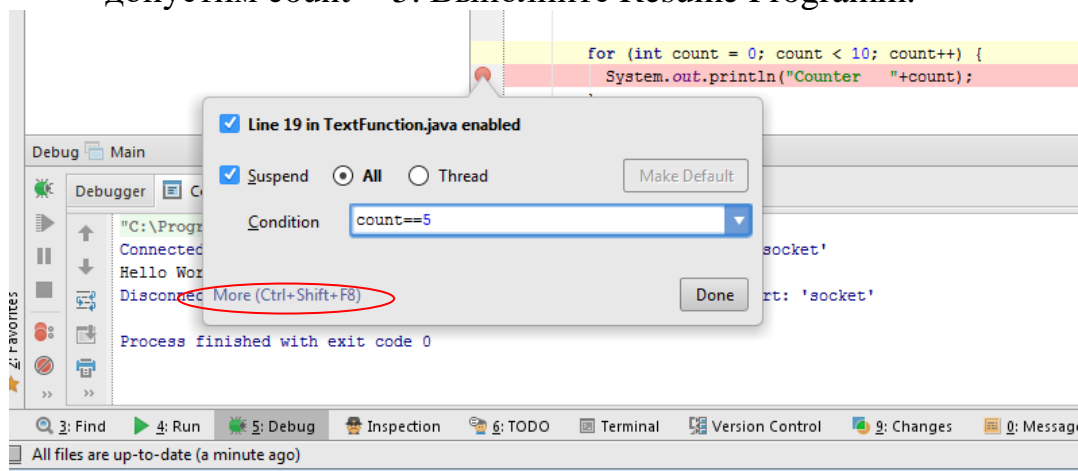
```

Установите точку останова и запустите приложение в отладчике Run- -Debug (Shift-f9).

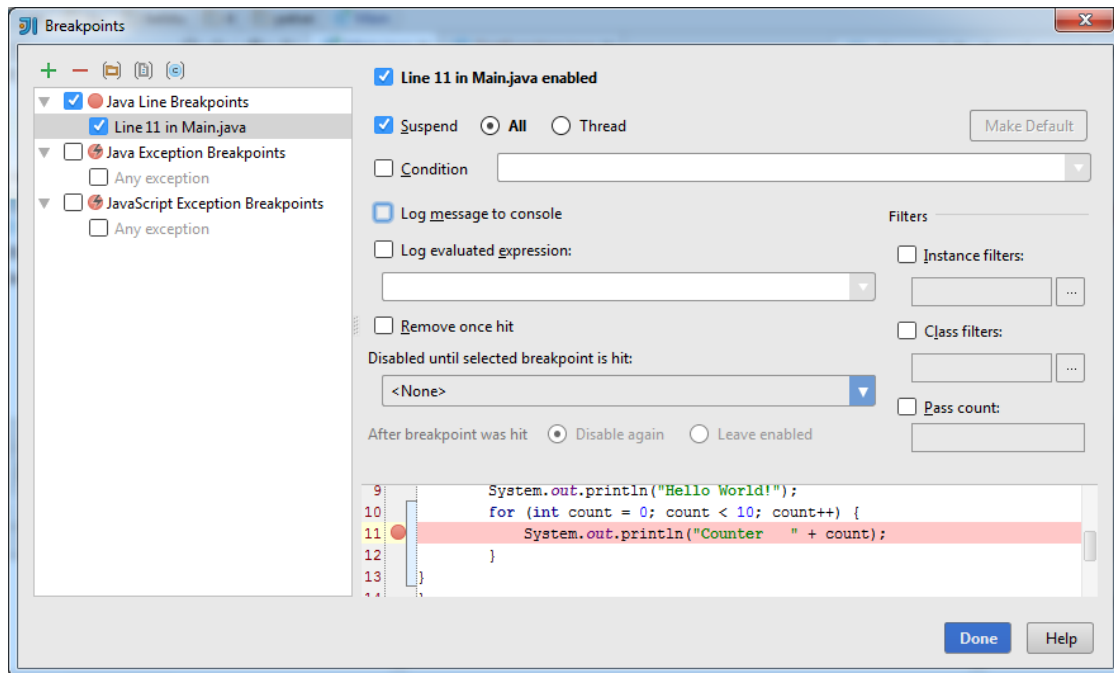
В окне Debugger посмотрите как изменяется переменная. Зайдите в Меню Run посмотрите доступные команды. Выполните Resume Programm.

Добавьте count в окно Watches.

Во время отладки щелкните по точке основа и установите условие установки допустим count==5. Выполните Resume Programm.



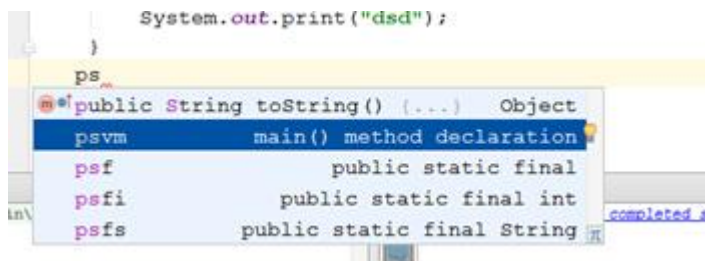
Нажмите на More в окне условий и разберитесь с возможностями наплойки точек останова



Держите под рукой список быстрого вызова для ускорения работы  
[https://resources.jetbrains.com/assets/products/intellij-idea/IntelliJIDEA\\_ReferenceCard.pdf](https://resources.jetbrains.com/assets/products/intellij-idea/IntelliJIDEA_ReferenceCard.pdf)

### 3) Напишите текст программы на языке Java с выводом в консоль

- a) Использовать пакеты `by.belstu.it.фамилия.basejava`  
Создайте в нем класс `JavaTest` с методом `main` (наберите просто `psvm`). В дальнейшем запускайте `main`.



- b) Определить переменные типа char, int, short, byte, long, boolean.  
Выполнить над ними следующие операции:

- ✓ `String + int`
- ✓ `String + char`
- ✓ `String + double`
- ✓ `byte = byte + int`

- ✓ `int = double+long`
- ✓ `long = int + 2147483647;` // при выводе значение должно быть положительное
- ✓ `static int sint;` // выведите значение без инициализации
- ✓ `boolean = boolean && boolean`
- ✓ `boolean= boolean ^ boolean`
- ✓ проверьте можно ли выполнить `boolean + boolean`
- ✓ подберите типы для чисел 9223372036854775807 и 0x7fff\_ffff\_fff
- ✓ проинициализируйте и выведите `char - 'a' ; \u0061'; 97;` после чего сложите все `char`.
- ✓ Проверьте результат операции `3.45 % 2.4`
- ✓ Проверьте результат операции `1.0/0.0;`
- ✓ Проверьте результат операции `0.0/0.0;`
- ✓ Проверьте результат операции `log(-345);`
- ✓ Проверьте результат `Float.intBitsToFloat(0x7F800000);`  
Проверьте результат `Float.intBitsToFloat(0xFF800000);`

В каких системах счисления можно записывать целые константы?

c) Задайте две **константы** в классе

```
final int
public final int
public static final int
```

Объясните разницу между ними.

d) Выполните статический импорт

```
import static java.lang.Math.*;
```

- ✓ выведите значения **Math.PI**; **Math.E**; округлите их (`Math.round()`); найдите минимальное среди них `Math.min(p,e)`;
- ✓ сгенерируйте случайное число из диапазона [0,1)

e) Создать объекты разных **классов оболочек** (`Boolean`, `Character`, `Integer`, `Byte`, `Short`, `Long`, `Double`)

- ✓ выполните на ними арифметические, логические и битовые операторы (`,`, `>>>`, `>>`, `~`, `&`, `*`, `-`, `+`) – выборочно
- ✓ введите `MIN_VALUE` и `MAX_VALUE` для `Long` и `Double`
- ✓ выполнить упакровку и распаковку для типов `Integer` и `Byte`
- ✓ вызовите для `Integer` методы : `parseInt` ; `toHexString` ; `compare` ; `toString` ; `bitCount` ; `isNaN` ...

f) Выполните преобразование числа типа **String** (`String s34 = "2345";`) к `int` , используя: конструктор, `valueOf`, `parse`....

- ✓ переводите строку в массив байтов и обратно из массива байтов в строку

- ✓ преобразуйте строку в логический тип 2-мя способами.
- ✓ определите две строки (String) с одинаковыми инициализаторами. Выполните ==, equals, compareTo. В чем разница, поясните результат. Одной из строк присвойте null. Повторите все три варианта сравнения.
- ✓ для произвольной строки выполните функции split, contains, hashCode, indexOf, length, replace.
- ✓

g) Проверьте какая из форм объявлений **многомерного массива** допустима:

```
char[][] c1;
char[] c2[];
char c3[][];
```

Можно ли определить массив нулевой длины?

Что случится, если индекс массива превысит его длину?

- ✓ для c1

```
c1 = new char[3][];
```

сделайте так, чтобы каждая последующая строка содержала на один элемент больше чем предыдущая. Выведите c1.length; c1[0].length и т.д.

- ✓ проинициализируйте c2 и c3 и выполните :

```
boolean comRez = c2==c3;
c2 = c3;
```

поясните результат

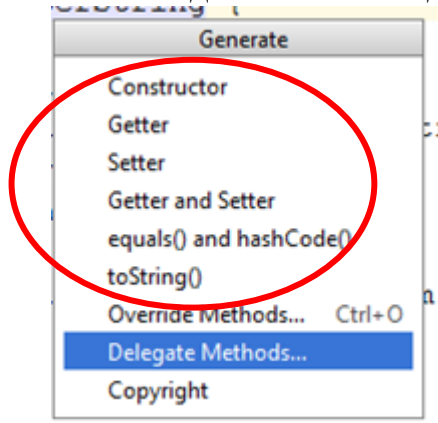
- ✓ выведите один из массивов с помощью цикла (foreach)

```
for(тип итер_пер : коллекция)
```

- ✓ выйдите преднамеренно за границы массива. Что будет получено?

h) Создайте **класс** WrapperString с приватным полем типа String.

- ✓ Используя Code->Generate (Alt+Insert) сгенерируйте выделенные методы



Определите метод

```
public void replace (char oldchar, char newchar)
```

- ✓ Создайте объект, выполните метод
- ✓ Определите анонимный класс с переопределенной реализацией `replace (char oldchar, char newchar)` и дополнительным методом `delete (char newchar)`

i) **Документируйте код.** Добавить комментарии в программы в виде `/** комментарий */`,

- ✓ Добавьте в комментарии дескрипторы

Для класса – `@author`, `@version`

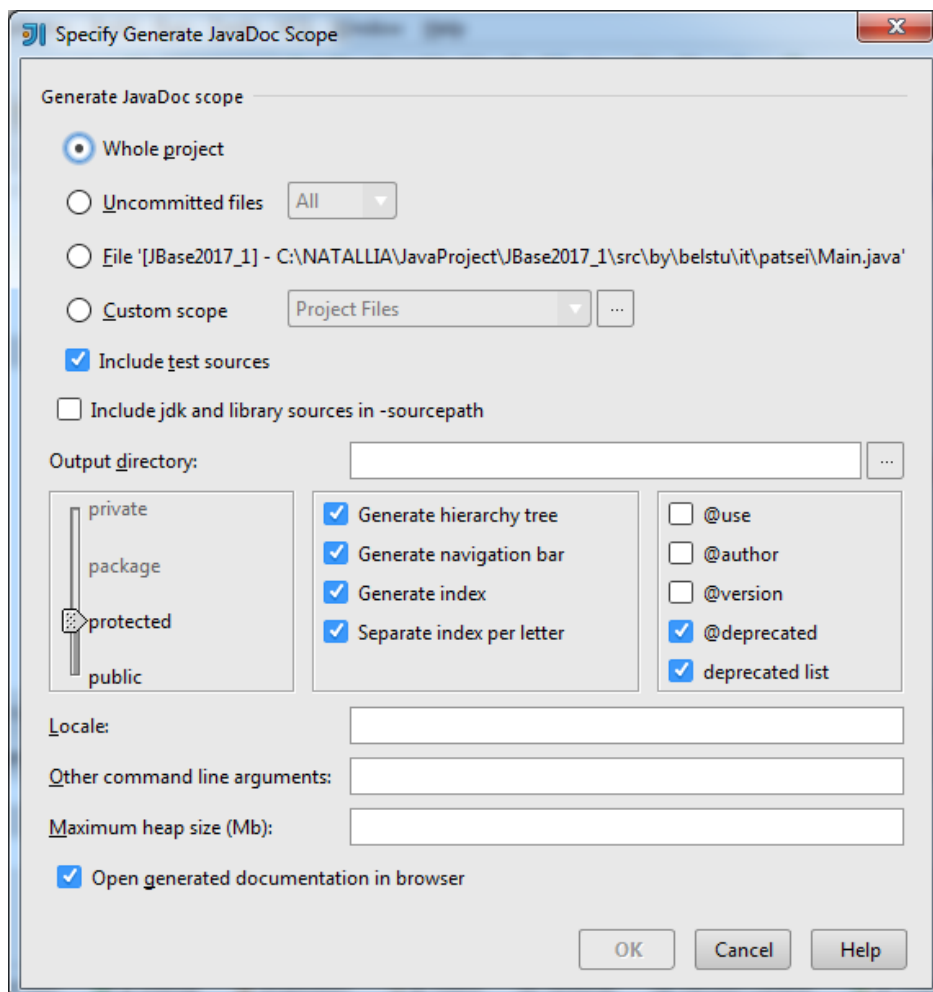
Для метода `main` - `@return @throws @param`

Для произвольной переменной - `@value @see`

- ✓ сгенерируйте html-файл документации.

Можно через утилиту `javadoc.exe` – извлекает информацию из классов

Или `Tools --> Javadoc` Настройте `Generate Javadoc` диалог и выберите `Custom scope`



Что такое аннотация?