# Как я написал свой blockchain

# Введение

Для начала загуглил и нашёл два руководства, но оба написаны для Python (который я почти не знаю), да ещё и с дополнительными библиотеками. А так как сейчас активно изучаю программирование под Android и Java – решил создать проект на Java, сразу разместив его на Bitbucket (который в этот день хотел меня о чём-то предупредить разместив радужный логотип)

Итак, меньше слов и больше дела. Поехали. В модуле **Main.java** создаём объект класса Blockchain.

**public static void** main(String[] args) {  
 Blockchain blockchain = **new** Blockchain();  
 System.***out***.println(**"Hello, blockchain!"**);  
}

А в классе **Blockchain.java** будем создавать блокчейн. Пока только заглушки. В самом объекте 2 поля: chain – наша цепочка блоков, которую будем строить и messages – цепочка сообщений, которые будут храниться.

**private** LinkedList<Block> **chain**;  
**private** LinkedList<Message> **messages**;  
  
**public** Blockchain() {  
 **this**.**chain** = **new** LinkedList<Blockchain>();  
 **this**.**messages** = **new** LinkedList<Message>();  
}  
  
**public** Blockchain createBlock() {

*//* ***TODO: Создаём новый блок и добавляем его к цепочке* return new** LinkedList<String>();

}  
  
**public int** createMessage() {  
 *//* ***TODO: Создаём новое сообщение и добавляем его в список сообщений* return new** LinkedList<String>();  
}  
  
**public static** String getHash(Blockchain blockchain) {  
 *//* ***TODO: Формируем хеш блока и возвращаем его* return ""**;  
}  
  
**public** Blockchain getLastBlock() {  
 *//* ***TODO: Возвращаем последний блок в цепочке* return new** Blockchain();  
}

Хочу заодно попробовать метод TTD, поэтому добавил модули с unit-тестами заглушками: **MainTest.groovy** и **BlockChainTest.groovy**. Все тесты прошли успешно – можно продолжать.

Разбираюсь дальше. Добавил 2 новых класса **Block.java** и **Message.java**. То, как работает код и какие unit-тесты написал - расскажу позже, когда буду дорабатывать unit-тесты.

# Добавление документации

## Правила написания документации

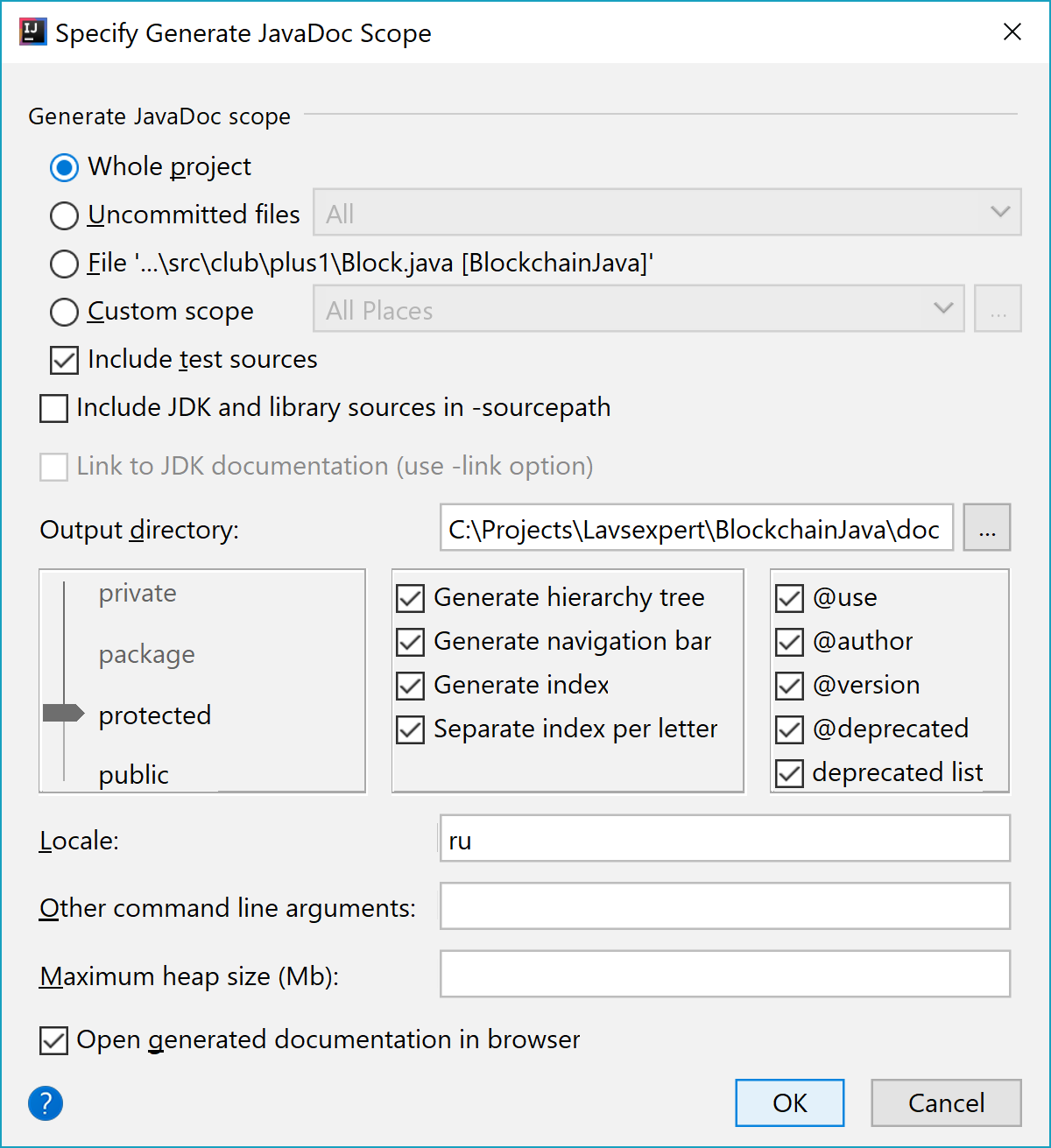
Далее изучал то, как правильно пишется JavaDoc. Эффективная JavaDoc должна содержать описания того:

* Как классы взаимосвязаны друг с другом
* Как методы влияют на состояние объекта
* Как методы сообщают вызывающим устройствам о ситуациях ошибки, и о каких ошибках они могут оповещать
* Как класс справляется с использованием в многопоточном приложении
* Домен аргументов метода и множество их возвращаемых значений

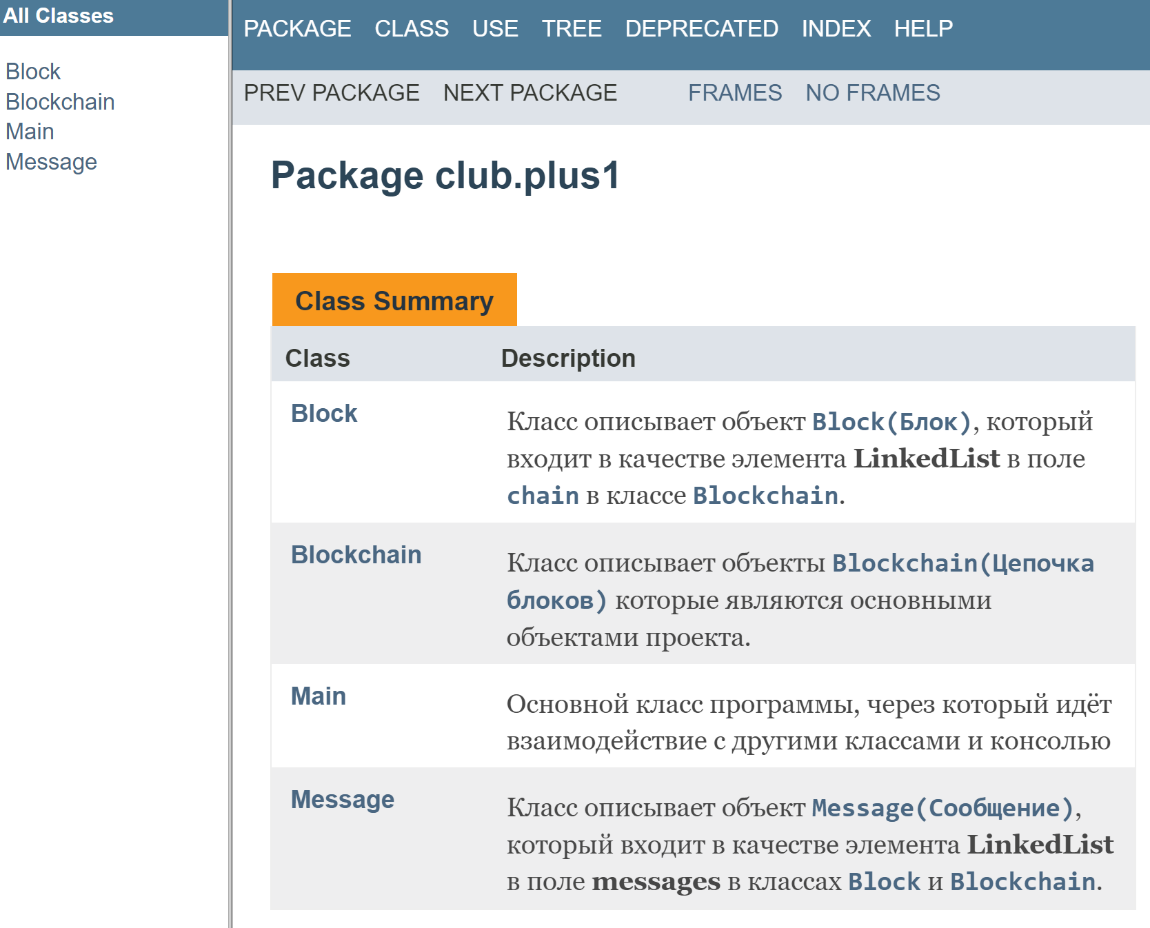
Плохая документация - или что еще хуже, ее отсутствие - может привести хороший код в состояние непригодности или одноразового пользования. Потратив немного дополнительного времени на документацию, вы сэкономите несчетное количество часов ваших пользователей (а может и своих).

## Выгрузка документации

В итоге добавил JavaDoc во все классы. Cделал выгрузку JavaDoc (в IDEA: Tools -> Generate JavaDoc) Теперь есть документация в виде html.

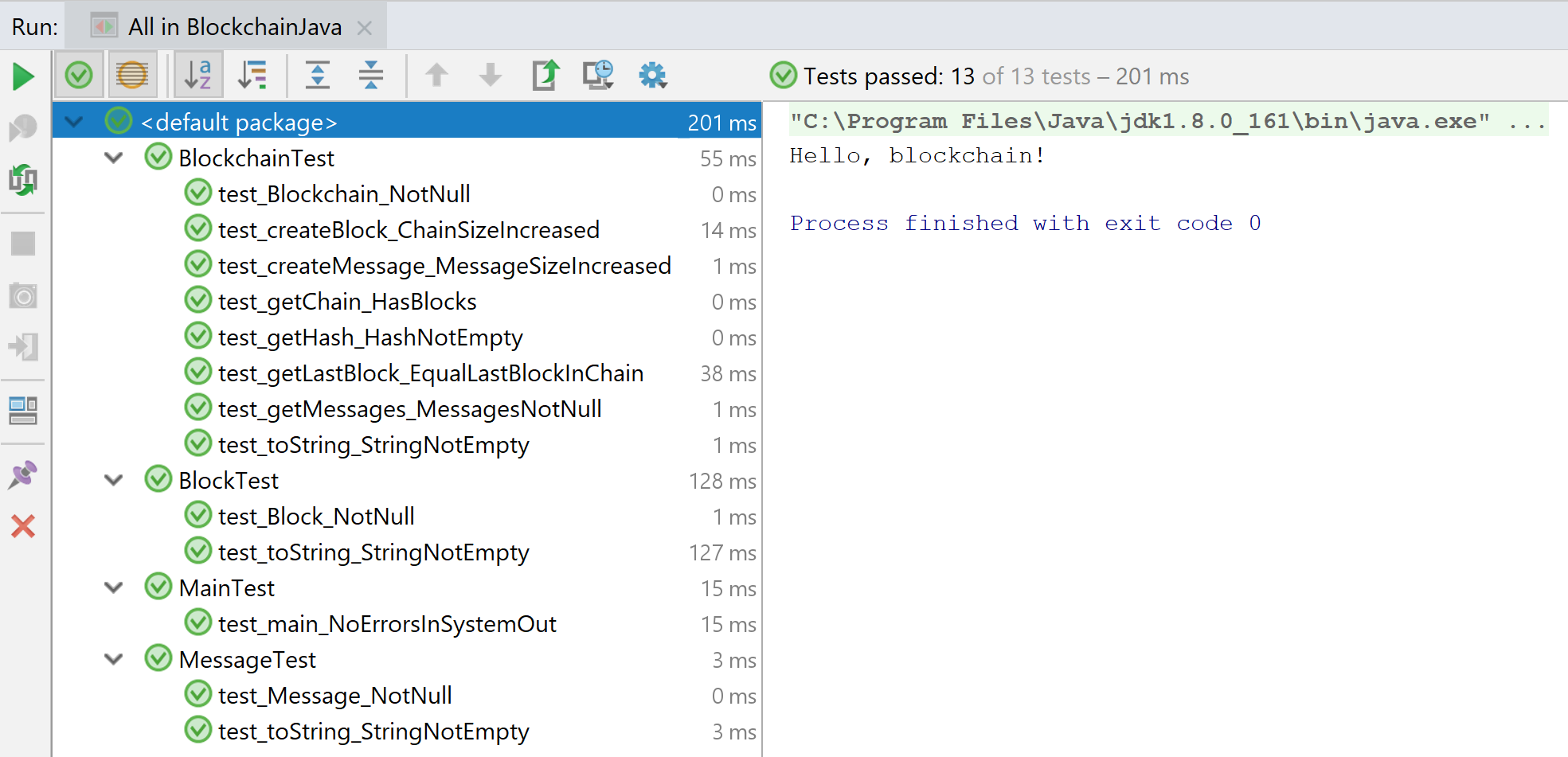


## Документация в html

В итоге документация выглядела так:

# Покрытие модулей unit-тестами

Перечитав несколько статей по тому, как писать тесты – полностью переписал все тесты, а также обновил сами модули. Тесты сделал по одному на каждый метод, и имя метода теста формировал по шаблону: test\_<ИмяМетода>\_<ПроверяемыйРезультат>. И во всех тестах ставил throws Exception – чтобы отлавливать исключения от Assert. Это поможет отличать ошибки в тестах от ошибок в Assert (ошибки в тестах будут красными, а в Assert – жёлтыми). Потом специально во всех тестах сделал ошибку в Assert – чтобы тесты стали жёлтыми. А потом исправил все эти ошибки, сделав все тесты зелёными. Вот что получилось после прогона всех тестов:



# Приложения

## Ссылки

1. Основная статья по основам создания blockchain: <https://proglib.io/p/learn-blockchains-by-building-one/>
2. Дополнительная статья по созданию blockchain: <https://habr.com/company/otus/blog/342974/> (перевод статьи <https://bigishdata.com/2017/10/17/write-your-own-blockchain-part-1-creating-storing-syncing-displaying-mining-and-proving-work/> )
3. Получение SHA256: <https://stackoverflow.com/questions/5531455/how-to-hash-some-string-with-sha256-in-java>
4. Как писать JavaDoc: <https://www.ibm.com/developerworks/ru/library/j-jtp0821/index.html>
5. Пример кода с JavaDoc: <http://www.fandroid.info/dokumentirovanie-koda/>
6. Подробнее о возможностях JavaDoc: <http://www.codenet.ru/webmast/java/JavaDoc/>
7. Виды тестов: <https://medium.com/@arturbasak/тесты-которые-должен-писать-разработчик-a04cab35f45b>
8. Подробно про unit-тестирование: <https://habr.com/post/169381/>
9. Подключение unit-тестов в IDEA: <http://qaru.site/questions/156160/setting-up-junit-with-intellij-idea>
10. Написание простейших unit-тестов в IDEA: <http://javastudy.ru/junit/junit-hello-world/>