Как я написал свой blockchain

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 основном модуле создаём объект класса Block:

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

А в самом классе Block будем создавать блокчейн. Пока только заглушки, по которым видно, что собираемся дальше сделать. В самом объекте 2 поля: chain – наша цепочка блоков, которую будем строить и transaction – цепочка транзакций, которые будут храниться.

**import** java.util.LinkedList;  
  
**public class** Block {  
  
 **private** LinkedList<String> **chain**;  
 **private** LinkedList<String> **transaction**;  
  
 **public** Block(){  
 **this**.**chain** = **new** LinkedList<String>();  
 **this**.**transaction** = **new** LinkedList<String>();  
 }  
  
 **public** LinkedList<String> newBlock(){  
 *//* ***TODO: Создаём новый блок и добавляем его к цепочке* return new** LinkedList<String>();  
 }  
  
 **public** LinkedList<String> newTransaction(){  
 *//* ***TODO: Создаём новую транзакцию и добавляем её в список транзакций* return new** LinkedList<String>();  
 }  
  
 **public static** String hash(Block block){  
 *//* ***TODO: Формируем хеш блока и возвращаем его* return ""**;  
 }  
  
 **public** Block last(){  
 *//* ***TODO: Возвращаем последний блок в цепочке* return new** Block();  
 }  
}

А так же добавил соответствующие unit-тесты заглушки. Хочу заодно попробовать метод TTD.

**import** org.junit.After  
**import** org.junit.Assert  
**import** org.junit.Before  
**import** org.junit.Test  
  
**class** MainTest **extends** GroovyTestCase {  
 **private** Main **main**;  
  
 @Before  
 **void** initTest() {  
 **main** = **new** Main();  
 }  
  
 @After  
 **void** afterTest() {  
 **main** = **null**;  
 }  
  
 @Test  
 **void** testMain() {  
 Assert.*assertTrue*(**true**);  
 }  
  
 @Test(timeout = 500l)  
 **void** timeStampTest() {  
 **while** (**true**);  
 }  
}

И соответствующие тесты для Block

**import** org.junit.After  
**import** org.junit.Assert  
**import** org.junit.Before  
**import** org.junit.Test  
  
**class** BlockTest **extends** GroovyTestCase {  
 **private** Block **block**;  
  
 @Before  
 **void** initTest() {  
 **block** = **new** Block();  
 }  
  
 @After  
 **void** afterTest() {  
 **block** = **null**;  
 }  
  
 @Test  
 **void** testBlock() {  
 Assert.*assertTrue*(**true**);  
 }  
  
 @Test(timeout = 500l)  
 **void** timeStampTest() {  
 **while** (**true**);  
 }  
  
 @Test  
 **void** testNewBlock() {  
 Assert.*assertTrue*(**true**);  
 }  
  
 @Test  
 **void** testNewTransaction() {  
 Assert.*assertTrue*(**true**);  
 }  
  
 @Test  
 **void** testHash() {  
 Assert.*assertTrue*(**true**);  
 }  
  
 @Test  
 **void** testLast() {  
 Assert.*assertTrue*(**true**);  
 }  
}

Все тесты прошли успешно – можно продолжать.

Ссылки

1. <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/> )
2. <https://proglib.io/p/learn-blockchains-by-building-one/>