Week 06 - Unit Testing

Topics covered in this week

- Unit testing in Java
- Issues with floating point operations

Reading material

Definitions:

- https://martinfowler.com/bliki/UnitTest.html
- http://softwaretestingfundamentals.com/unit-testing/ (http://softwaretestingfundamentals.com/software-testing-levels/)
- https://content.pivotal.io/blog/what-is-a-unit-test-the-answer-might-surprise-you
- https://smartbear.com/learn/automated-testing/what-is-automated-testing/

Implementations:

- https://dzone.com/articles/how-to-start-with-unit-testing-in-java-with-junit5
- https://alexecollins.com/given-when-then-junit/
- https://www.baeldung.com/mockito-annotations
- https://www.vogella.com/tutorials/Mockito/article.html

Other:

- https://cs.lmu.edu/~ray/notes/unittesting/

Homework

Difficulty	Problem	Notes
EASY	In the attached sources find Account.java and AccountTest.java. Write test cases in AccountTest which exercise all the methods of the Account class. If your tests reveal any defects in the logic of the Account class, repair them.	unitTesting-homework.zip
EASY	Open BankSimulation class, examine the code without running it and predict the output. If your prediction was incorrect, research the cause and write up a detailed technical explanation for the observed results.	
EASY	Write unit tests for the EngineFactory class from the com.iquestgroup.plant.**initial**.factory package. The goal is to test all possible cases. (Write your tests in InitialEngineFactoryTest class) Do not change the existing code, and do not write any other classes except the tests. Use JUnit for the unit tests.	
MEDIUM	Write unit tests for the EngineFactory class from the com.iquestgroup.plant.**reorg**.factory package. (Write your tests in ReOrgEngineFactoryTest class) Do not change the existing code, and do not write any other classes except the tests. Use JUnit for the unit tests. To be able to test EngineFactory, you will have to use mocking.	

Kahoot