- Purpose:
    - Combinatorial testing (testing multiple combos of input parameters/configurations)
    - JUnit is more general purpose unit testing
    - JCUnit is useful for devs/testers in complex systems (inputs in embedded systems, API's, or GUI's)
- Focus:
    - To test input configuration/combinations
    - JUnit tests more individual methods/classes
- Test Cases:
    - Automated generation of combinations
    - JUnit has manual/parameterized inputs

- Issues we faced:
    - java.lang.Exception: No public static parameters method on class your.package.YourTest - required manual Parameter extension (https://github.com/dakusui/jcunit/issues/185)
    - The plugin refused to work on the latest release version 0.10.x-SNAPSHOT, reverted to 0.8.12 instead (this is not mentioned in the repo).

- Line 30, scenario(): Regex creates all possible scenarios of deposit, withdraw, transfer
- Lines 34-47, depositAmount(), withdrawAmount(), transferAmount(): creates lists for methods to test
- Lines 50-60, depositUsed(): creates booleans to check if scenario uses deposit, withdraw, transfer
    - **However, he hard coded in checking if the value is -1, meaning you can deposit, withdraw, and transfer any negative value that isn't -1. This means that if we test for -100, then it would pass the test**
    - **Line 49, : conditions restrict the generation of combinatorial pairs in order to weed out invalid test cases (i.e. if the scenario contains a deposit, deposit amount must be over 0 (in our case not -1))**
- Lines 102-116, calculateBalance(): calculates amount in bank account after tested methods, but will not accept any more methods the instant balance is negative. (i.e. if you have 10$ cash in your account, then withdraw $20, then you will not be able to deposit, withdraw, or transfer more)
- Line 135, printScenario(): iterates through scenarios printing how much money you have after each method is called