2017 Java Coding Competition

The Problem

SF University is a unique learning environment where people can challenge themselves. You are tasked with coming up with the solution for the newest coding problem – Blocks. The problem is separated in multiple pieces and allows you to build confidence as you pass the simple problem of a building block and move on to the complex idea of a poly block. There are some basic requirements (JUnits) that must be followed, but SF University appreciates fresh ideas and will gladly accept additional enhancements.

Description of the Basic Requirements (Successfully pass 104 JUnit tests)

Building Block - A building block may connect to only one other building block above, and one below.
BuildingBlockTest

- Run JUnits 1st time – expected results (Runs: 104/104, Errors: 25, Failures: 74)
- Test calling same method multiple times with same result – Block 1 stack under block 2
- Test preconditions are valid – Blocks are not stacked to begin the activity (Junit – completed!)
- Test stacking only two blocks
- Etc.

Poly Block – An Object which can have an unlimited amount of connections to other PolyBlocks, but can be sequentially iterated.
PolyBlockTest

- Test Iterator with 1, 2, or 3 elements
- Test connect and disconnect with null
- Test connect and disconnect with another PolyBlock
- Etc.

StorageBuildingBlockTest/StoragePolyConnectorTest

- Etc.

JUnit tests are the highest authority related to requirements and can be found by following the path *src/test/java*. Additional details/comments are found in the JUnits.

********* **Do not change anything in the JUnit tests!** **********

First Actions:

- Import the problem statement into your IDE.
- We have provided Maven dependency for JUnit 4. If you are not set up with the recommended IDE (Spring Tool Suite), you may need to add JUnit 4.
- If you identify any additional libraries you would like to use, please add them to the pom.xml file or copy the .jar files into the resources folder
- Run your JUnit tests, code, and repeat. "How to run JUnits"

Questions:

- Private questions can be submitted via email - codingcompetition@statefarm.com
- Public questions can be submitted by opening an issue against the 2017 Coding Competition repository on GitHub.
  - Used to provide clarity about the problem or report issues
  - Not a forum to find a solution to the problem

2017 Java Coding Competition

When you are done:

- Update the feedback.txt file and include the following information:
    - Your team – name of each individual participating.
    - How many JUnits you were able to execute successfully.
    - Document and describe the additional "nice to have" features included, to help the judges properly grade your submission and explain how to properly execute new enhancements.

- Push your changes to one single branch for you and your teammate. Open a single pull request against the main State Farm Coding Competition repository before 11:59PM CDT on October 7, 2017.
    - If you make any commits after midnight without prior approval from codingcompetition@statefarm.com, your submission will be disqualified.
    - If you so choose, you may open a pull request at any time during the competition and continue to update it as long as you do not make any commits after midnight.

Rules

- Contestants cannot seek help from individuals outside their team.
- Teams are expected to have the necessary tools and JARs preloaded on their machines **prior** to the competition.
- If you believe this document and the JUnit tests conflict, the JUnit tests are the highest authority.

How you will be Graded

Components of submission will be weighted as follows.

- 100% core requirements met, including:
    - Number of JUnits that pass using correct functionality in the program
    - Maintaining Object Oriented Programming principles
    - Code documentation
    - Code must compile and execute
- Do not complete any Bonus unless you have all the JUnit tests completed
    - Bonus credit awarded for any extra features added (up to 10%)

In the event of a tie, we will further judge your solution based on: code cleanliness, maintainability, and adherence to object-orientated principles.

<u>Bonus "Nice to have" features</u>:

- Bonus Credit – Do not complete any bonus features unless you have completed all the required functionality and all JUnits pass.
    - Create additional functionality and demonstrate through a Graphical User Interface (GUI), command line interface, JUnits, or web UI.
    - If you can think of any other useful features to add, we appreciate ingenuity and will gladly accept any useful enhancements. Be sure to document any bonus features in the feedback text file.