To: Da Boss
From: Rohit Rastogi and Upasna Madhok
Subject: Swapping Codebases
Date: 5/15/2018

Moving forward, we will be working off of Upasna and Amulya's Python codebase. We have identified a few concrete pros of the Python implementation that make it a stronger foundational codebase than Rohit and Chris' Java implementation. We have also identified some opportunities to improve the Python codebase and practice using software engineering best practices that we would not have if we chose to work off of the Java codebase.

First, we will discuss why the Python implementation provides a stronger base than the Java implementation. The representation of the board and the tiles used in the game is a critical part of the implementation of the rules of Tsuro. We have determined that the representation of the board in the Python implementation is simpler and more straightforward. In this implementation, players do not have to move across tile boundaries and a player's absolute position can be determined at any time. This significantly removes the overhead in playing turns in the game and reduces the scope for errors. It is also easier to retrieve all players that are standing adjacent to a tile which further simplifies implementing the rules of Tsuro. Furthermore, the Tile class in the Python implementation contains identifiers for the tiles which removes the need for equality checks when checking if a tile has already been played, is a duplicate of another tile, etc. Additionally, the inclusion of the identifier tags makes rotating the tiles easier and removes the overhead of storing information about individual rotations. Lastly, the Python codebase is smaller and more maintainable than the Java codebase which will make testing and debugging more efficient.

Next, we will discuss some of the learning opportunities offered by the decision to use the Python codebase. First, as Python enthusiasts, we'd like to gain experience using the new 'typing' module included in Python 3.6's Standard Library. We recognize the fact that Java's type system allows us to catch type-related bugs early (at compile time, instead of at runtime). But, we also recognize that Python is more concise and arguably more clear due to its dynamic typing. The 'typing' module provides a mix of both benefits as it adds a layer of static analysis to Python, allowing us to catch bugs before runtime while preserving Python's conciseness. Second, we'd also like experience working with the PyContract Python package. This library will allow us to write contracts in a clear and organized way. In the Java implementation, behavioral contracts are often implemented by a sequence of complicated if-statements intertwined with a function's implementation. This can be difficult to interpret and we believe that the PyContract package will make this easier. Lastly, the Python implementation was recently refactored and the play-a-turn function was broken up into a series of helper functions. Currently, these helper functions are untested. By choosing the Python implementation, will be able to gain more experience writing thorough and comprehensive tests to ensure that game play is correct, predictable, and reliable.