Codebase Swap Analysis

Codebase 1 - Java | Codebase 2 - Python

<u>Code structure</u>

Both codebases have mostly the same classes such as server player, machine player, player interface, server, board, deck, and tile.

Codebase 1 uses a token class to represent the token position on board and index on tile. It also has a color field so that a token can be associated with a player. Codebase 2 uses a player location class that has two attributes - position on board and index on tile, but uses a tile location wrapper class that contains the row and column indices when a new position is generated for purposes such as tile placement.

Codebase 1 has an array of tiles as an attribute in the server player class and methods to manage the hand in the same class. Codebase 2 uses a hand class to represent the tiles in a player's hand. The hand class keeps track of a list of tiles and the number of tiles. The server player class has an attribute called hand that is of the hand type.

<u>Level of abstraction</u>

Both codebases manage to hide operation and management of class object within their class definition. In addition, they handle the registration of a player in a similar manner by taking in a client player, creating a corresponding server player, and associating these two types of players.

In PlayATurn, codebase 1 undertakes three steps to first check against a list of contracts, then manage player movement and elimination, and finally determine game over conditions. Each step uses different helper functions such as eliminating a player and draw and pass dragon as abstractions to handle the logic of nitty-gritty details. Codebase 2 also sequentially manages all operations within the function but did not have the same abstraction level for some operations such as passing the dragon tile.

In LegalPlay, Codebase 2 checks whether placing a tile on the board kills the player by calling a separate legal play function associated with the board class. Codebase 1 has this check in the server class instead. Since this check is based on the board state, having this check in the Board class allows Codebase 2 to separate board logic from server logic.

<u>Test coverage</u>

Both codebases contain a test suite for each class and conducted a tournament to compare performance among machine players with different strategies. In addition, both codebases also support running all test suites from the console/command line.

<u>Choice of language</u>

Both team members have prior experience in developing programs in Java and Python and thus they are comfortable with writing in either programming language.

In conclusion, both codebases have implemented essential components of the Tsuro game and created extensive test suites to validate different functionality implementations. The team decided to proceed with codebase 1 because some components and code style are more organized, and key operations in the server implementation use more abstractions.