Team ABC
Alec Vaughn, Babita Kurmi, Chris Rogers
CS471 Object Oriented Design
Detail Report

Responsibilities:

- Chris was responsible for implementing the GUI observer based on the diagrams created in our meetings. It was also required to extend observer. Would connect with the logic when finished. Also creating the GUI mock up.
- Alec was responsible for creating the database and connecting it to a storage adapter. Also, implementing the text observer, database storage Adapter, and other design patterns. Also, creating the sequence diagrams.
- Babita was responsible for constructing the base code, and implemented a first version of the Singleton, Factory, and Observer patterns. Also, responsible for implementing a csv storage Adapter and creating the class diagrams.

Implementation: Design Patterns

0. MonopolyGame, Player, and Square were the most modified of the original code to provide adequate updates to the Observers, store data properly in the Adapters, and implement the Factory.
    a. getSomeSquare(index) was added to Square to be called by Player
    b. instance variable squareNum was added to record the last square index the player landed on for the load() implementation
    c. p:Player.setLocation() was also necessarily important for this purpose
    d. instance variable roll was added to simplify passing the last dice roll to the Observers
1. Observer: MonopolyGame is the Observable, and calls update on the Observers on a player's turn one or more times. Player is the common argument. Others are Integer and String for round and player name, respectively.
    a. The GUI observer moves each player piece across a board. It shows the squares of which the pieces land on.
    b. The text Observer prints out the round, current location, dice roll, and new location and repeats.
2. Factory: Square was the ideal candidate for the Factory method since there are two types of Squares in iteration 1 and more are added later. Square was made abstract and GoSquare and RegularSquare were created to extend it. SquareFactory was created to be called from Board and build each Square type.
3. Singleton: MonopolyGame was a trivial candidate for the Singleton pattern. Its constructor was made private and it aggregates a static instance of itself which is accessible through getInstance()
4. Adaptor: the two storage methods (db, csv) were both programmed to a separate interface, which they implemented. Then an additional StorageAdapter interface was created and implemented by two Adapter classes (one of each type).