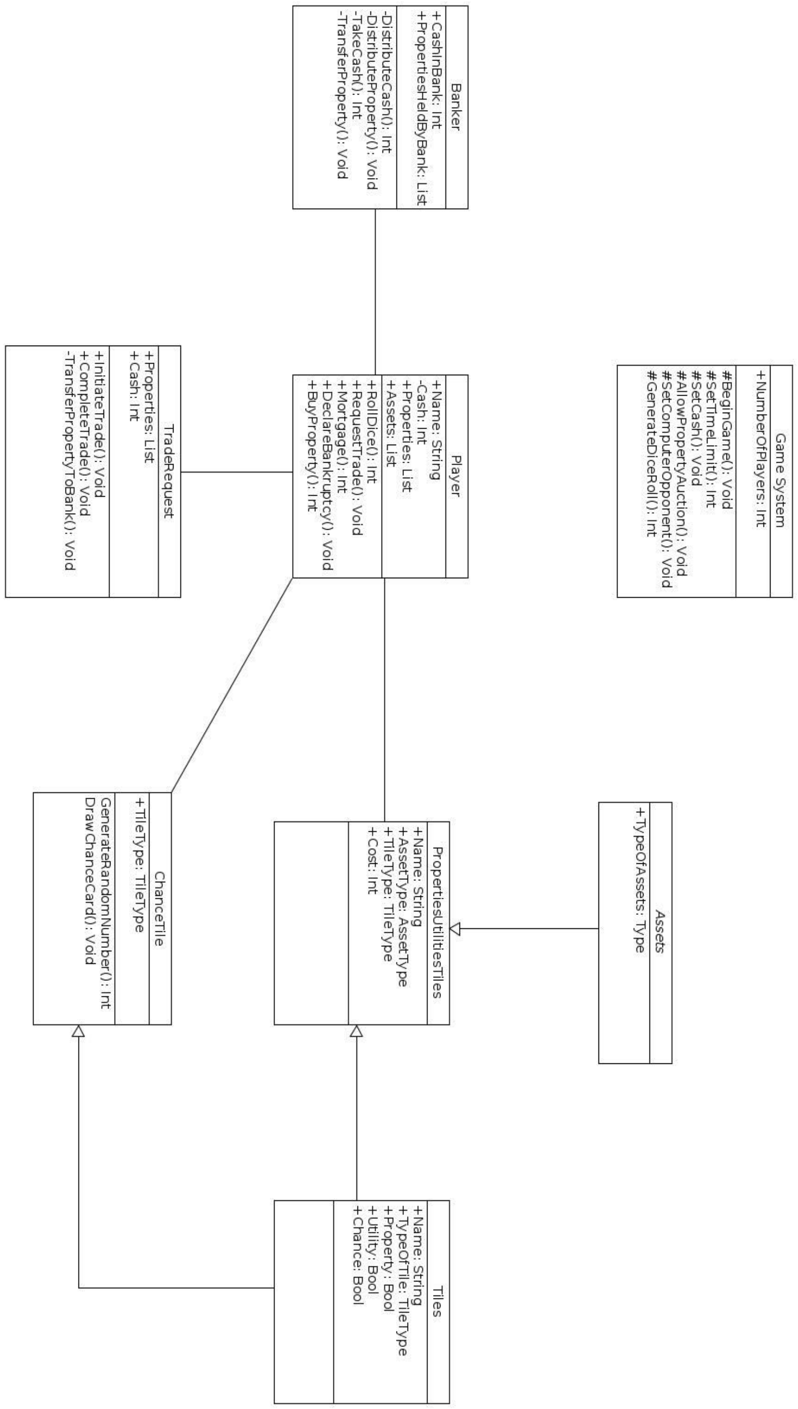
William Diment

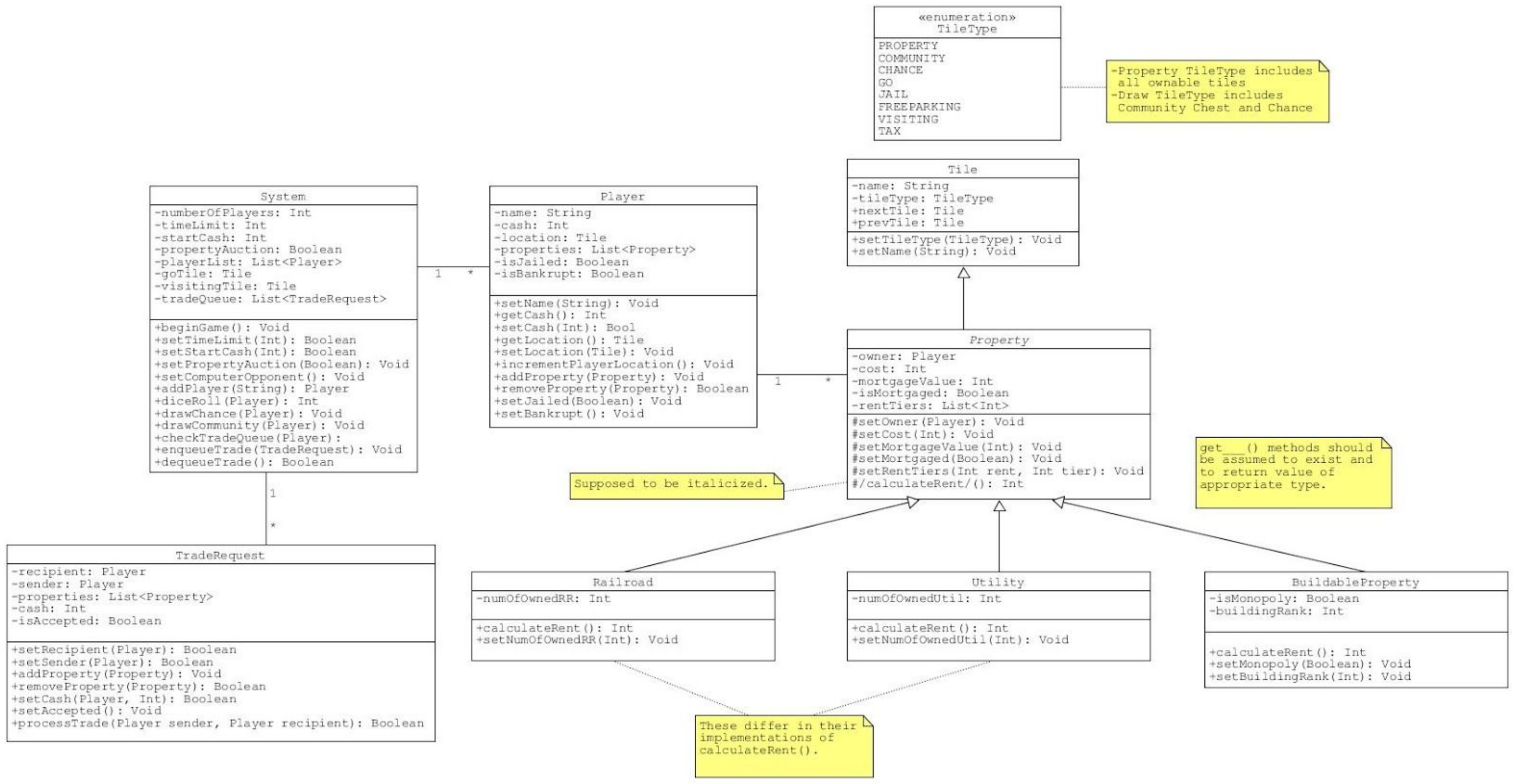
Thomas Trieu

CSCI 4448 Project

Part 3 - Refactoring

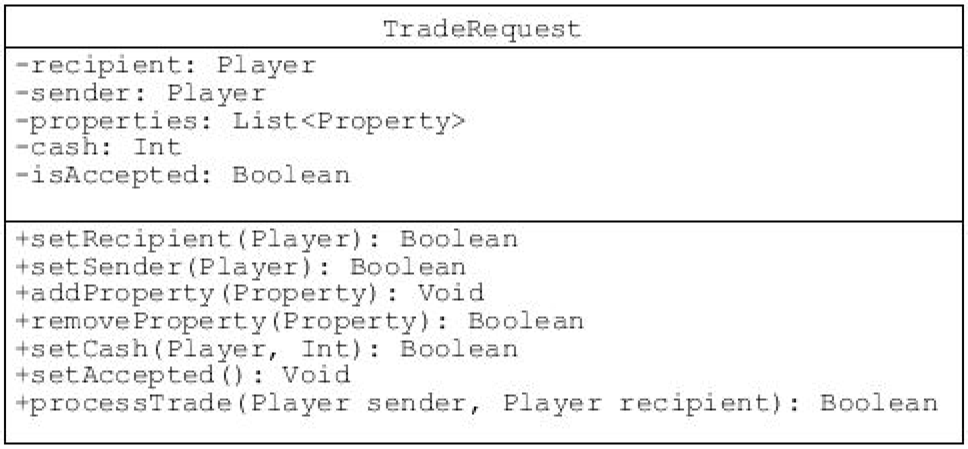


Part 2 class diagram


«enumeration» TileType
PROPERTY
COMMUNITY
CHANCE
GO
JAIL
FREEPARKING
VISITING
TAX
-Property TileType includes all ownable tiles
-Draw TileType includes Community Chest and Chance
Tile
-name: String
-tileType: TileType
+nextTile: Tile
+prevTile: Tile
+setTileType(TileType): Void
+setName(String): Void
System
-numberOfPlayers: Int
-timeLimit: Int
-startCash: Int
-propertyAuction: Boolean
-playerList: List<Player>
-goTile: Tile
-visitingTile: Tile
-tradeQueue: List<TradeRequest>
+beginGame(): Void
+setTimeLimit(Int): Boolean
+setStartCash(Int): Boolean
+setPropertyAuction(Boolean): Void
+setComputerOpponent(): Void
+addPlayer(String): Player
+diceRoll(Player): Int
+drawChance(Player): Void
+drawCommunity(Player): Void
+checkTradeQueue(Player):
+enqueueTrade(TradeRequest): Void
+dequeueTrade(): Boolean
1
*
Player
-name: String
-cash: Int
-location: Tile
-properties: List<Property>
-isJailed: Boolean
-isBankrupt: Boolean
+setName(String): Void
+getCash(): Int
+setCash(Int): Bool
+getLocation(): Tile
+setLocation(Tile): Void
+incrementPlayerLocation(): Void
+addProperty(Property): Void
+removeProperty(Property): Boolean
+setJailed(Boolean): Void
+setBankrupt(): Void
1
*
Property
-owner: Player
-cost: Int
-mortgageValue: Int
-isMortgaged: Boolean
-rentTiers: List<Int>
#setOwner(Player): Void
#setCost(Int): Void
#setMortgageValue(Int): Void
#setMortgaged(Boolean): Void
#setRentTiers(Int rent, Int tier): Void
#/calculateRent/(): Int
Supposed to be italicized.
get___() methods should be assumed to exist and to return value of appropriate type.
1
*
TradeRequest
-recipient: Player
-sender: Player
-properties: List<Property>
-cash: Int
-isAccepted: Boolean
+setRecipient(Player): Boolean
+setSender(Player): Boolean
+addProperty(Property): Void
+removeProperty(Property): Boolean
+setCash(Player, Int): Boolean
+setAccepted(): Void
+processTrade(Player sender, Player recipient): Boolean
Railroad
-numOfOwnedRR: Int
+calculateRent(): Int
+setNumOfOwnedRR(Int): Void
Utility
-numOfOwnedUtil: Int
+calculateRent(): Int
+setNumOfOwnedUtil(Int): Void
BuildableProperty
-isMonopoly: Boolean
-buildingRank: Int
+calculateRent(): Int
+setMonopoly(Boolean): Void
+setBuildingRank(Int): Void
These differ in their implementations of calculateRent().

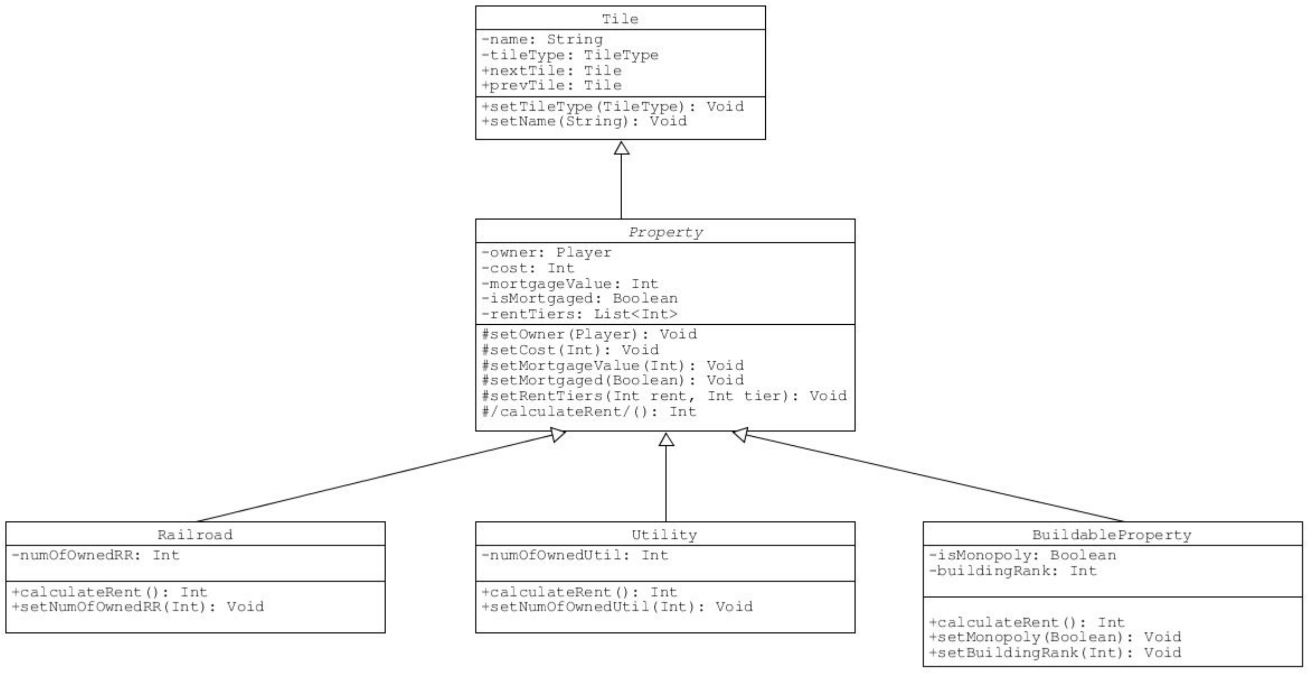
Part 3 class diagram

One design pattern that has been included is the Facade Pattern. This is done in through the TradeRequest class.


TradeRequest
-recipient: Player
-sender: Player
-properties: List<Property>
-cash: Int
-isAccepted: Boolean
+setRecipient(Player): Boolean
+setSender(Player): Boolean
+addProperty(Property): Void
+removeProperty(Property): Boolean
+setCash(Player, Int): Boolean
+setAccepted(): Void
+processTrade(Player sender, Player recipient): Boolean

The class simplifies the process of transferring assets between Player objects by consolidating their interfaces. Its interfaces doesn't have all of the functionality of what it encapsulates. The TradeRequest class can also be said to reflect the Command Pattern by encapsulating an operation that can be undone if needed (based on whether the trade request is accepted or declined).

Another design pattern used is the Decorator Pattern.



Starting with the Tile class, we add functionality based on what is required of the subclass. The Tile class is built upon by the Property abstract class. This addition allows for Tile's subclasses to behave as ownable properties instead of static tiles like Go and Free Parking.

In refactoring, we've removed the Banker class. This is because it seemed unnecessary and could be consolidated into the System class. While this might seem to be a step towards the Blob Antipattern, it was done to avoid the Poltergeist Antipattern. The Banker, as it was, essentially existed as an 'operation class'.