Final Project Report

1. Name of project and names of all team members

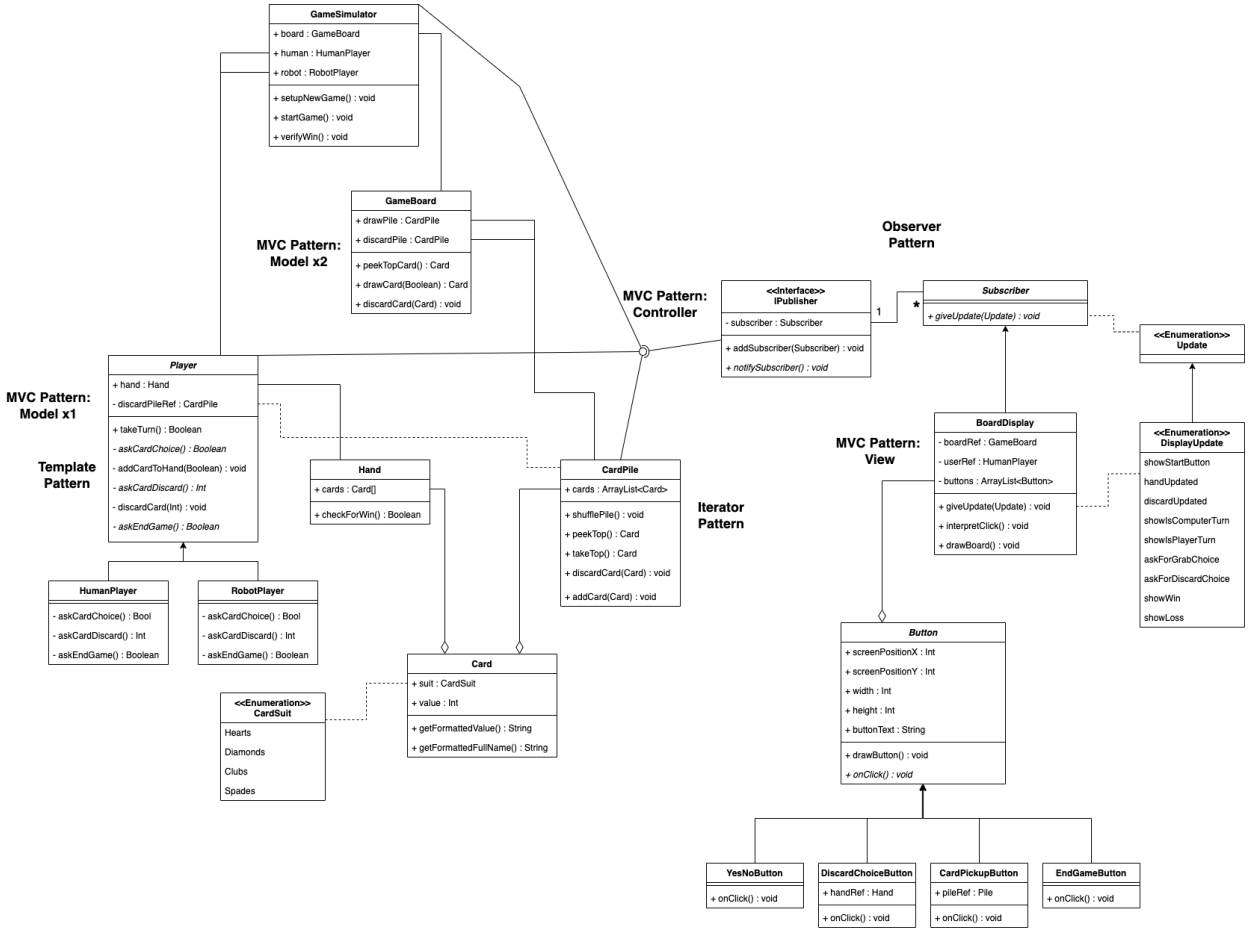
Rummy
Lara Chunko, Maria Stull, Jake Swartwout

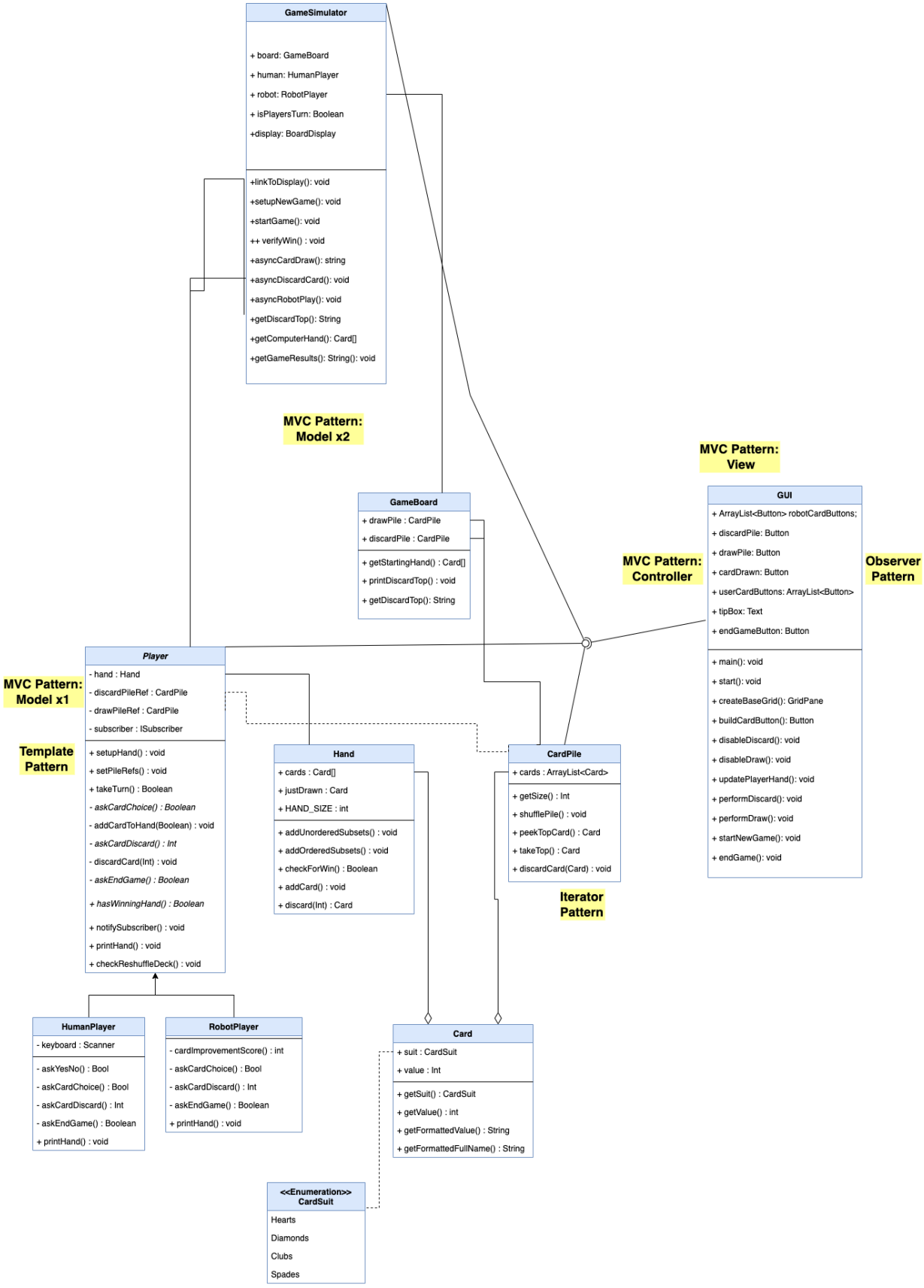
2. Final State of System Statement

We ended up with a dual system--you can play either via command-line or via a GUI. Originally the text-based version was a temporary measure while we sorted out the GUI, but we kept it so our game would run whether or not the user has JavaFX set up. We made some changes to a lot of the GUI support portions of the code, since JavaFX worked a little differently than we expected--so we replaced most of what we had intended to design with our GUI class. We did implement all of the features we intended to, so the game is completely playable and correct. The only thing we didn't implement was images for the cards. This was a stretch goal to begin with, but we ended up struggling so much with implementing the GUI that we didn't have time to add graphics.

3. Final Class Diagram and Comparison Statement

The project 5 class diagram is black and white, and the Project 7 class diagram has blue labels. Our biggest changes were replacing BoardDisplay/DisplayUpdate/Button with GUI--since JavaFX has its own rules, and buttons are defined already, we didn't have to redefine our own. We also planned on having some extra support for an observer/subscriber model, but a lot of that was already built-in to JavaFX, so we were able to implement that without creating separate classes. Other than that (and minor changes to hook everything together), our design largely stayed the same between project 5 and 7.


GameSimulator
+ board : GameBoard
+ human : HumanPlayer
+ robot : RobotPlayer
+ setupNewGame() : void
+ startGame() : void
+ verifyWin() : void
GameBoard
+ drawPile : CardPile
+ discardPile : CardPile
+ peekTopCard() : Card
+ drawCard(Boolean) : Card
+ discardCard(Card) : void
MVC Pattern: Model x2
Observer Pattern
MVC Pattern: Controller
<<Interface>> IPublisher
- subscriber : Subscriber
+ addSubscriber(Subscriber) : void
+ notifySubscriber() : void
1
*
Subscriber
+ giveUpdate(Update) : void
<<Enumeration>> Update
Player
+ hand : Hand
- discardPileRef : CardPile
+ takeTurn() : Boolean
- askCardChoice() : Boolean
- addCardToHand(Boolean) : void
- askCardDiscard() : Int
- discardCard(Int) : void
- askEndGame() : Boolean
MVC Pattern: Model x1
Template Pattern
Hand
+ cards : Card[]
+ checkForWin() : Boolean
CardPile
+ cards : ArrayList<Card>
+ shuffilePile() : void
+ peekTop() : Card
+ takeTop() : Card
+ discardCard(Card) : void
+ addCard(Card) : void
Iterator Pattern
MVC Pattern: View
BoardDisplay
- boardRef : GameBoard
- userRef : HumanPlayer
- buttons : ArrayList<Button>
+ giveUpdate(Update) : void
+ interpretClick() : void
+ drawBoard() : void
<<Enumeration>> DisplayUpdate
showStartButton
handUpdated
discardUpdated
showIsComputerTurn
showIsPlayerTurn
askForGrabChoice
askForDiscardChoice
showWin
showLoss
HumanPlayer
- askCardChoice() : Bool
- askCardDiscard() : Int
- askEndGame() : Boolean
RobotPlayer
- askCardChoice() : Bool
- askCardDiscard() : Int
- askEndGame() : Boolean
Card
+ suit : CardSuit
+ value : Int
+ getFormattedValue() : String
+ getFormattedFullName() : String
<<Enumeration>> CardSuit
Hearts
Diamonds
Clubs
Spades
Button
+ screenPositionX : Int
+ screenPositionY : Int
+ width : Int
+ height : Int
+ buttonText : String
+ drawButton() : void
+ onClick() : void
YesNoButton
+ onClick() : void
DiscardChoiceButton
+ handRef : Hand
+ onClick() : void
CardPickupButton
+ pileRef : Pile
+ onClick() : void
EndGameButton
+ onClick() : void


GameSimulator
+ board: GameBoard
+ human: HumanPlayer
+ robot: RobotPlayer
+ isPlayersTurn: Boolean
+display: BoardDisplay
+linkToDisplay(): void
+setupNewGame(): void
+startGame(): void
++ verifyWin() : void
+asyncCardDraw(): string
+asyncDiscardCard(): void
+asyncRobotPlay(): void
+getDiscardTop(): String
+getComputerHand(): Card[]
+getGameResults(): String(): void
MVC Pattern: Model x2
MVC Pattern: View
GameBoard
+ drawPile : CardPile
+ discardPile : CardPile
+ getStartingHand() : Card[]
+ printDiscardTop() : void
+ getDiscardTop(): String
GUI
+ ArrayList<Button> robotCardButtons;
+ discardPile: Button
+ drawPile: Button
+ cardDrawn: Button
+ userCardButtons: ArrayList<Button>
+ tipBox: Text
+ endGameButton: Button
+ main(): void
+ start(): void
+ createBaseGrid(): GridPane
+ buildCardButton(): Button
+ disableDiscard(): void
+ disableDraw(): void
+ updatePlayerHand(): void
+ performDiscard(): void
+ performDraw(): void
+ startNewGame(): void
+ endGame(): void
MVC Pattern: Controller
Observer Pattern
MVC Pattern: Model x1
Player
- hand : Hand
- discardPileRef : CardPile
- drawPileRef : CardPile
- subscriber : ISubscriber
+ setupHand() : void
+ setPileRefs() : void
+ takeTurn() : Boolean
- askCardChoice() : Boolean
- addCardToHand(Boolean) : void
- askCardDiscard() : Int
- discardCard(Int) : void
- askEndGame() : Boolean
+ hasWinningHand() : Boolean
+ notifySubscriber() : void
+ printHand() : void
+ checkReshuffleDeck() : void
Template Pattern
Hand
+ cards : Card[]
+ justDrawn : Card
+ HAND_SIZE : int
+ addUnorderedSubsets() : void
+ addOrderedSubsets() : void
+ checkForWin() : Boolean
+ addCard() : void
+ discard(Int) : Card
CardPile
+ cards : ArrayList<Card>
+ getSize() : Int
+ shufflePile() : void
+ peekTopCard() : Card
+ takeTop() : Card
+ discardCard(Card) : void
Iterator Pattern
HumanPlayer
- keyboard : Scanner
- askYesNo() : Bool
- askCardChoice() : Bool
- askCardDiscard() : Int
- askEndGame() : Boolean
+ printHand() : void
RobotPlayer
- cardImprovementScore() : int
- askCardChoice() : Bool
- askCardDiscard() : Int
- askEndGame() : Boolean
+ printHand() : void
Card
+ suit : CardSuit
+ value : Int
+ getSuit() : CardSuit
+ getValue() : int
+ getFormattedValue() : String
+ getFormattedFullName() : String
<<Enumeration>>
CardSuit
Hearts
Diamonds
Clubs
Spades

4. Third-Party code vs. original code Statement

We based our JavaFX GUI on the example at https://docs.oracle.com/javafx/2/get_started/form.htm — virtually all elements have been changed, but this was our starting point.

5. Statement on the OOAD process for your overall Semester Project • List three key design process elements or issues (positive or negative) that your team experienced in your analysis and design of the OO semester project

- Starting off with a highly object-oriented design made it very easy to switch between a command-line / text-based game and a GUI. Since all the game elements and logic were implemented without a particular display option in mind, it was very easy to modify our code to work with either a command-line interface or a GUI.
- We struggled to keep an OO approach when coding the GUI--none of us were familiar with JavaFX to begin with, and it has enough of its own quirks (i.e. a lot of the logic having to be in its start() method) that our final design for that class wasn't quite as neat as we would have liked.
- Having the UML diagram designed from the beginning made it much easier for everyone to understand how all the pieces fit together; when one of us was looking at code someone else had written, we were able to easily figure out how it worked within the overall system, because already we had a design worked out and largely stuck to it.