CPSC 2720 SPRING 2021

Children's Card Game

Team Incinii

Julie Wojtiw-Quo, David Neufeld, Fairooz Afia Saiyara, Jenna McDonnell

February 26, 2021 11:59 pm.

TABLE OF CONTENTS

INTRODUCTION

Children's Card Games is a software that implements three card games (Go Fish, Crazy Eights, and Uno). The rules for each game are as follows:

Go Fish

# Players: 2 - 5

Deck

- Standard 52-card deck (One of each card from Ace to King in four different suits (Club, Diamond, Heart, Spade)

Objective

- Win the most books (i.e. Four of a kind)

Dealing

- 2/3 players: 7 cards
- 4/5 players: 5 cards
- All remaining cards are put in a pile to form the stock

Play Rules

- Player to the left of the dealer requests a card of the same rank as one of the cards in their hand from any other player. When their turn ends, play continue in a clockwise fashion.
    - If the player the card rank was requested from has the requested card(s), they must hand it/them over to the player requesting (this is called a "catch"). The player may then ask another player for another card rank for as long as they get a card(s) each time they request it. If the player does not get a card, the requesting player is told to "Go Fish!" (pick up a card from the stock). If the player makes a catch, their turn continues. The turn ends when no catch is made.
    - If the player the card rank was requested from does not have the requested card(s), the requesting player is told to "Go Fish!" If the player makes a catch from the stock, they may continue their turn, otherwise it ends.
    - If a player has no cards, they may pick a card from the stock and play with that card, otherwise, if there are no cards remaining in the stock, they are done.
- When a player has four cards from a rank, they must place down their book.
- The game ends when all 13 books have been obtained.

Crazy Eights

# Players: 2 - 5

Deck

- Standard 52-card deck (One of each card from Ace to King in four different suits (Club, Diamond, Heart, Spade)

Objective

- Be the first player to get rid of all the cards in their hand

Dealing

- Each player gets 5 cards
- Remaining cards are left in the stock
- Top card of the stock is placed face up in a separate pile (this is the "starter"). If the card is an 8 it is buried in the middle of the pile and a new starter is turned.

Play Rules

- Player to the left of the dealer places down one card of the same suit or rank on the starter. Play continues in a clockwise fashion of players placing down one card of the same suit or rank as the top card.
- If there is no playable card in the player's hand, they must pull cards from the stock until they have a playable card that can be placed down. If the stock is exhausted and the player can't place down a card, they must pass.
- Players may pull cards from the stock even if they have a playable card.
- Eights are wild; players may place it down on any card and proceed to specify a suit to be played on the next play. A card of that suit must be placed down on the eight, or another eight can be played.
- After a player has won, they gather the cards from the other players and add up the points of their hands as follows:
  Each eight = 50 points
  Each K, Q, J or 10 = 10 points
  Each ace = 1 point
  Each other card is the face/pip value

UNO

# Players 2 – 5

Deck

- 108 card deck containing 4 each of "Wild" and "Wild Draw 4" cards and 25 each of 4 different colours (red, yellow, green, blue). Each color consists of one 0 card, 2 cards numbered 1-9, and 2 each of "Skip", "Draw 2" and "Reverse".
- Skip, Draw 2 and Reverse cards are known as "Action cards".

- Skip card: If a player plays this card, next player will not take their turn.
- Draw 2: The next player will draw 2 cards from the pile if he or she does not have draw 2 in his/her personal deck of cards.
- Reverse: The direction of the game changes.
- Wild Draw 4: The next player will draw 4 cards from the pile if he or she does not have a wild draw 4 in his/her personal deck of cards. The color is set by the person who played that card to one of the four (blue, green, red or yellow).
- Wild Card: The color can be changed by the person who played that card to either of the four: blue, green, red or yellow.

Objective

- Be the first player to get rid of all the cards in your hand.

Dealing

- Each player gets 7 cards
- Remaining cards are left in the stock
- At first, one card is selected at random from the pile and put on the table as a discard pile

Play Rules

- Based on the first card in the discard pile, the player to the left of the dealer will attempt to play a card matching either the color or the number/symbol or both.
- If a card cannot be played, or the player chooses not to play a card, they must draw a card from the stock. If that card can be played, it is played; otherwise, the turn is over.
- When a player only has one card left in hand, they must say "UNO" before the end of their turn.

The penalty for not saying UNO when a player has got only one card would be drawing two cards from the pile.

- This game is played by following a clockwise direction and repeated every time based on the last card played.
- After a player has won, they gather the cards from the other players and add up the points of their hands as follows:
  Each numbered card = face/pip value
  Each Draw 2/Skip/Reverse card = 20 points
  Each Wild/Wild Draw 4 card = 50 points

The rest of the document proceeds as follows. First, how the development of the project will be managed is described. Next, the development process that will be followed is presented. Finally, the initial design of our project is given.

PROJECT MANAGEMENT

TEAM ORGANIZATION

Team Member	Design - Draft	Design – Final	Implementation - Basic	Implementation - Final
Julie	Phase Lead	Design Lead	QA Lead	Reporting Lead
David	Design Lead	QA Lead	Reporting Lead	Phase Lead
Jenna	QA Lead	Reporting Lead	Phase Lead	Design Lead
Fairooz	Reporting Lead	Phase Lead	Design Lead	QA Lead

TEAM ROLES/RESPONSIBILITIES:

• PHASE LEAD

o Understands what needs to be accomplished in the phase.
o Asks questions/clarification from instructor(s) on behalf of the group.
o Identifies (in collaboration with the team) tasks to be done and distributes them.
o Follows up with team members about assigned tasks.
o Identifies problems and leads group discussions about how to solve them.
o Makes sure the team meets deadlines.

• DESIGN LEAD

o Proposes and collects design ideas from the group.
o Assures (in collaboration with the team) that the software design is following SOLID+DRY principles.
o Creates or maintains the UML diagrams so that they reflect the current design of the software.

• QUALITY ASSURANCE LEAD

o Ensures that the team is making a quality software product.
o Reviews pull requests to the master branch to ensure that they meet the team’s quality standards.
o Manages merge requests
o Creates a testing plan / schedule.
o Identifies test cases.

- **REPORTING LEAD**

o Organizes the work to be done for the report(s).
o Collects team's contributions to the reports (e.g., design diagrams from Design Lead)
o Records the team's ideas / action items during meetings.
o Writes (in collaboration with the team) the phase and/or team report.

RISK MANAGEMENT

- **REQUIREMENTS/DESIGN/ESTIMATION:**

1. The team planned a project that is too large (i.e. "eyes bigger than stomach").
   - Discuss with other members and divide our work
   - If someone gets stuck with their part of work, team members will try to find solution together
   - Make the Computational Intelligence simple
2. The team underestimated how long parts of the project would take.
   - Set an achievable deadline for each part of work
   - Member(s) who finishes their work earlier, he/she can help other members.
   - If a team member is struggling under a lot of work, divide up the tasks.
3. Major changes to design are needed during implementation.
   - Decide which classes can be kept
   - Call an emergency team meeting to reach a conclusion

- **PEOPLE:**

1. Addition or loss of team member (i.e. someone dropped the course, a new person joins the team)
   - (For loss of a team member) Re-divide/Re-arrange the project among existing team members
   - (For addition of a team member) Bring them up to speed so that all of us are on the same page
2. Unproductive team member(s)
   - Write in the report that the member was being unproductive
   - Talk to the lazy member at first. If that doesn't help, inform the instructor
   - Take away maintainer access
3. Team member(s) lacking expected technical background (e.g. don't know C++)
   - Have them start with roles that doesn't require C++, for example: making diagrams with DIA
   - Assuming that the team member knows other programming languages like Java or C#, we can ask him/her to talk to instructors for more assistance since they would have to complete their individual assignments in C++ anyway
4. Major life events (e.g. long-term illness, death of family member, birth of a child)
   - Take their work for the next week or two and divide it among existing team members

- Evaluate how the event had impacted the team member and whether he/she can be back anytime soon
- If a team member is gone for a month or longer, then re-assign roles for the whole project
- Mental health comes first and check with that member if he/she is doing okay

- **LEARNING & TOOLS**

1. Inexperience with new tools
   - Practice and help each other out. If all of us are clueless, seek help from the instructor for further assistance
2. Learning curve for tools steeper than expected
   - Schedule extra meetings to discuss and work together
   - GOOGLE! (for example: Khan Academy, C++ Tutorials, etc.)
3. Tools don't work together in an integrated way
   - Ask the instructor for help
   - Check if the software was installed properly by following each instruction again
   - Re-install and see if it works
   - Check Makefiles
   - If the tools don't work only for a single member, divide that work among other team members and assign that member with some other work of equal amount

DEVEVLOPMENT PROCESS

CODE REVIEW PROCESS

- All members will be expected to be working on their individually assigned branches and classes, and if any issues arise or they finish early, they will be expected to contact the design leader.

- Team members will message the design leader with a @mention and will document questions in the GitLab repo. The design leader will review the proposed changes and prepare a recommendation for the next meeting.

- Any changes or merge requests will be managed by the Quality Assurance Lead, and any substantial changes or complicated, confuzzling merge requests beyond the understanding of one person will be discussed at a team meeting.

- Any changes that are contrary to the design plan and/or will have undesirable consequences will be rejected and added to the list of items to be discussed at the next team meeting.

COMMUNICATION TOOLS

We are using Discord and a separate GitLab repo (using our 2720 uLeth accounts) to communicate and store our files.

During the final testing phase, group members and any other recruited testers will send bug reports via the GitLab issue tracker in the assigned repo.

Change Management

The Quality Assurance Lead will triage all bug reports.

- If the bug report identifies an error in a method, then the QA lead will determine which team member wrote the code and assign them to fix their code.

- If the bug report identifies an error in resulting from the interaction of code written by two or more group members and if it is clear the error is due to one member not following the design specifications the QA lead will assign resolution of the bug to that individual. If the problem is a feature not a bug, then a proposal for design change will be created and/or a team meeting will be called to discuss the changes.

- If the bug report identifies an error in resulting from the interaction of code written by two or more group members and it is not clear why the bug has appeared it will be discussed at the next group meeting and after discussing it with the other team members the QA lead will assign resolution of the bug at the meeting.

- Otherwise, the QA lead will reply to the bug report appropriately.

After a bug is resolved (fixed or confirmed but it is decided by the team it will not be fixed) the individual that resolved it will respond to the bug. The response will have two parts: An appropriate tag: It's a Feature, Works for Me, Confirmed, or Fixed; and a brief description/explanation.

e.g.: If it is not a bug, indicate the design specification related to the issue. If it won't be fixed, indicate why. If it has been fixed, indicate which files were updated.

SOFTWARE DESIGN

Design

Each game has

1. A deck of cards.
2. Computational intelligence to play against, or possibly other players.
3. A scoring system - these will be different for the three games.
4. A comparison to historical data.

All elements which are not common will appear in subclasses named for the separate games.

Design Rationale

The goal of this project, which informed all design decisions, is to develop examples that illustrate the principles taught in CPSC2720 as well as utilize several design patterns taught in the course. We will be using various techniques just as the SOLID and DRY principles, Agile development techniques, and Test Driven Development. We are also implementing object-oriented programming and practicing inheritance and polymorphism.

We are starting by implementing the components mentioned above independently of each other. Once those are built and tested, we will use them to create each of the games. We will start with Go Fish first and then move on to the other, somewhat more complex games.

We may utilize various design patterns that will be noted after learning about them (such as the Factory Method design pattern, the composite design pattern, the state design pattern, or the template design pattern). The specific outcomes of the design decision can be found in the UML diagrams in the appendices, with class diagrams found in Appendix A and sequence diagrams found in Appendix B.

Class Descriptions

Game: The abstract interface which will hold the general attributes for each of the games. It will also define the common variables used within each of the games, such as player turn, score and number of rounds.

getInstructions(): Displays the instructions for the specified game.
addPlayer(): Adds a player to the game.
removePlayer(): Removes a player from the game.
startGame(): Starts the chosen game.

GoFish: The class containing the rules and methods only implemented in Go Fish (such as saying go fish and other aspects of gameplay).

sayGoFish(): The method allowing a player to say go fish if they do not have the specified card.
askForCard(): The method allowing a player to choose the card they will ask others for.
displayBook(): The method called once a player reaches 4 of a kind in their hand.

CrazyEights: The class containing the rules and methods only implemented in Crazy Eights (such as picking up cards and other aspects of gameplay).

placeCard(): Plays a card
pickUpCard(): Picks up a card

Uno: The class containing the rules and methods only implemented in Uno (such as saying uno and other aspects of gameplay).

sayUno(): The method allowing a player to call uno whether or not they have one card left in their hand.
placeCard(): Plays a card

Player: The class that contains the information relating to the player, such as name, number/seating order, current hand and current points.

changeName(): Changes the specified player's name.

Computational Intelligence: The class setting up the computational intelligence.

Hand: Sets the hand for a player.

addCard(): adds a card to the hand
removeCard(): removes a card from the hand
dealHand(): deals a hand of cards as specified in the chosen game

Deck: Represents a deck of cards.

shuffleDeck() will "shuffle" the deck of cards by sorting them randomly
getTopCard() will display the top card of a deck (this will likely be implemented using a queue)
buildDeck() will "create" a new deck of cards by resetting the values to a full, complete pile in order

GoFishDeck: Represents a standard deck of cards used in Go Fish

CrazyEightsDeck: Represents a standard deck of cards used in Crazy Eights, with special data for the eights…

UnoDeck: Represents a standard deck of Uno cards (108 cards of four colors plus wild cards)

reverse(): implements the reverse card
plusTwo(): implements the plus two card

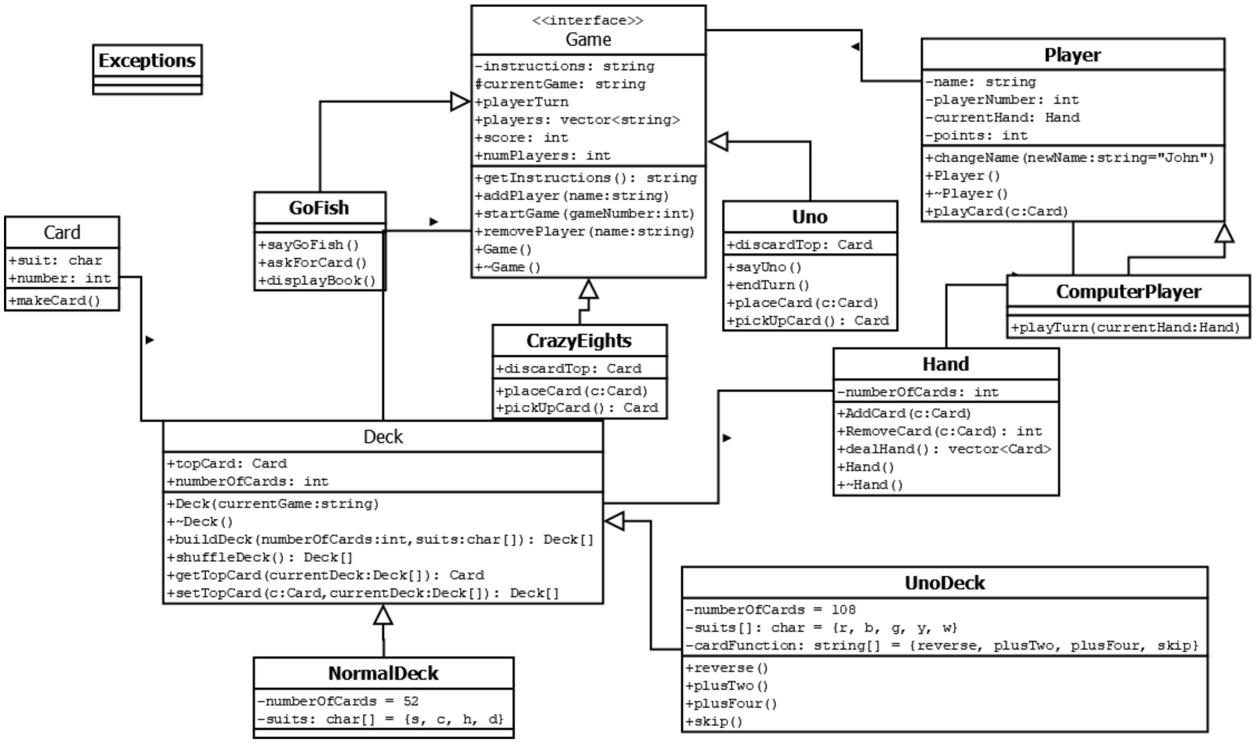
plusFour(): implements the plus four wild card
skip(): implements the skip card

Card: Represents a singular card.

drawCard(): draws a card

APPENDICES

APPENDIX A: CLASS DIAGRAMS



Our overall classes except for Exceptions which holds Exceptions.h which will be used by every class for testing purposes.

APPENDIX B: SEQUENCE DIAGRAMS

Diagram for Go Fish start:

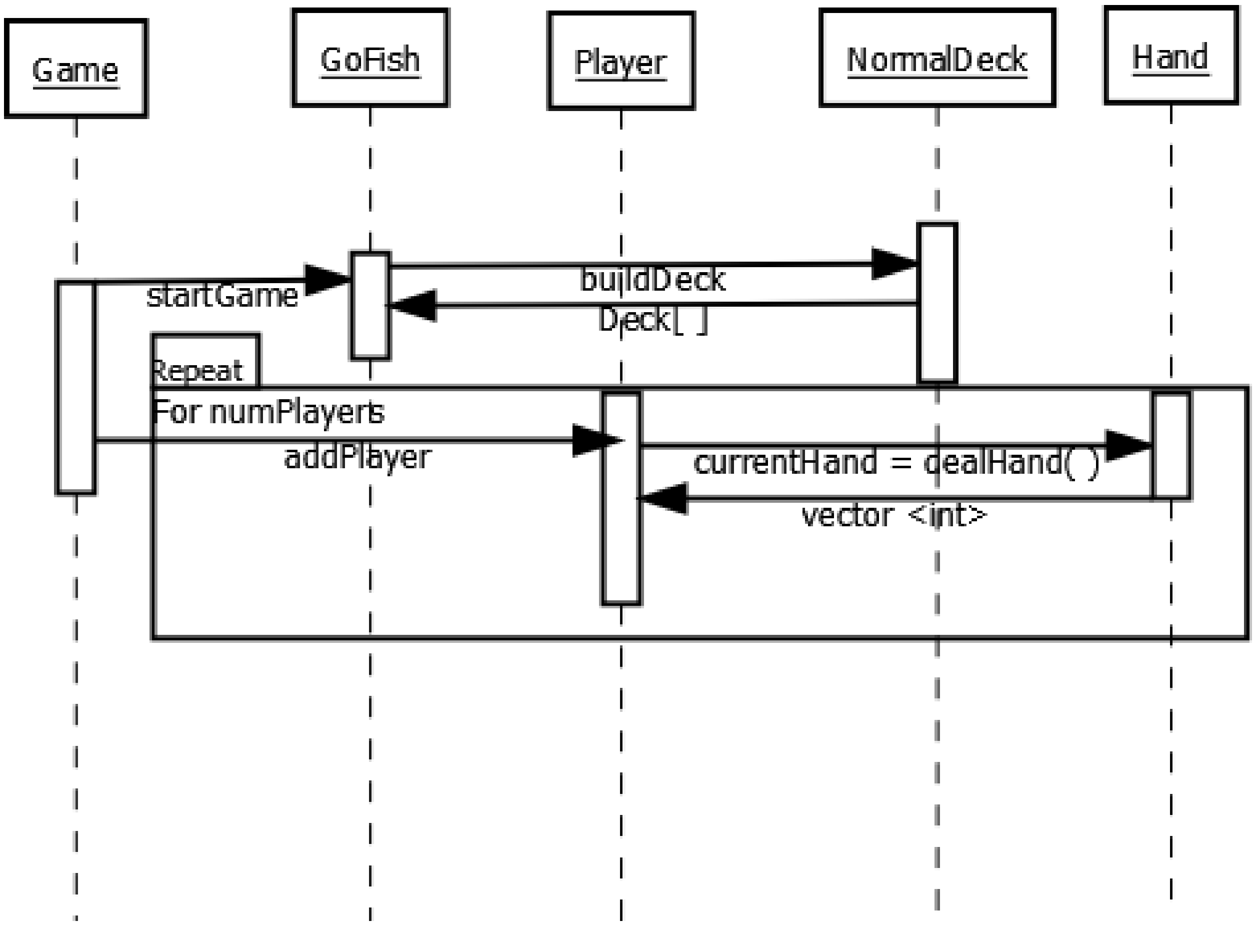


Diagram for Go Fish turn:

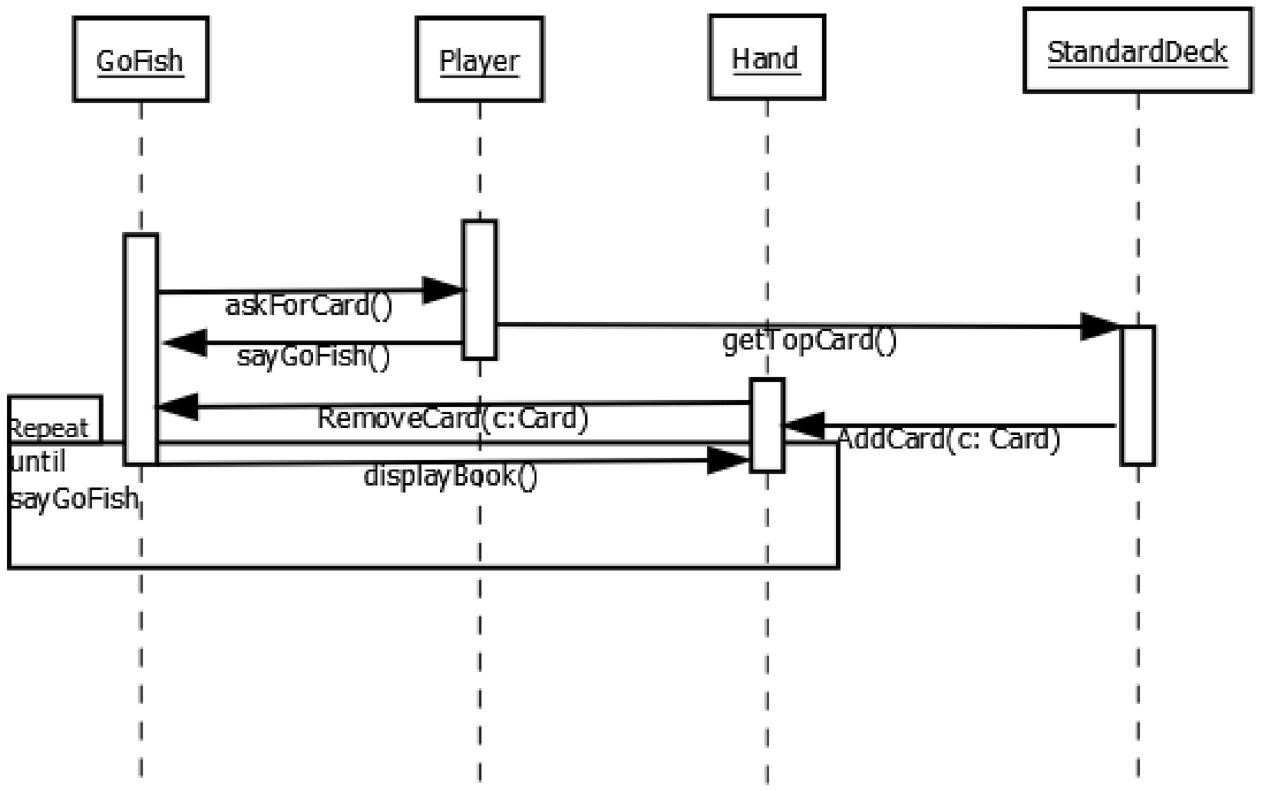


Diagram for Go Fish end:

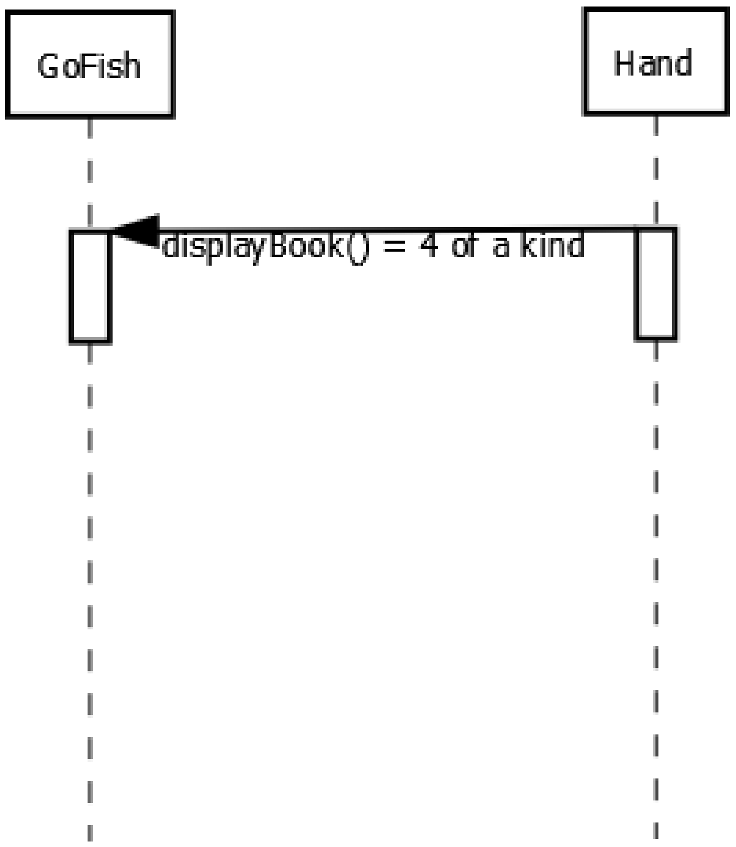


Diagram for starting Crazy Eights:

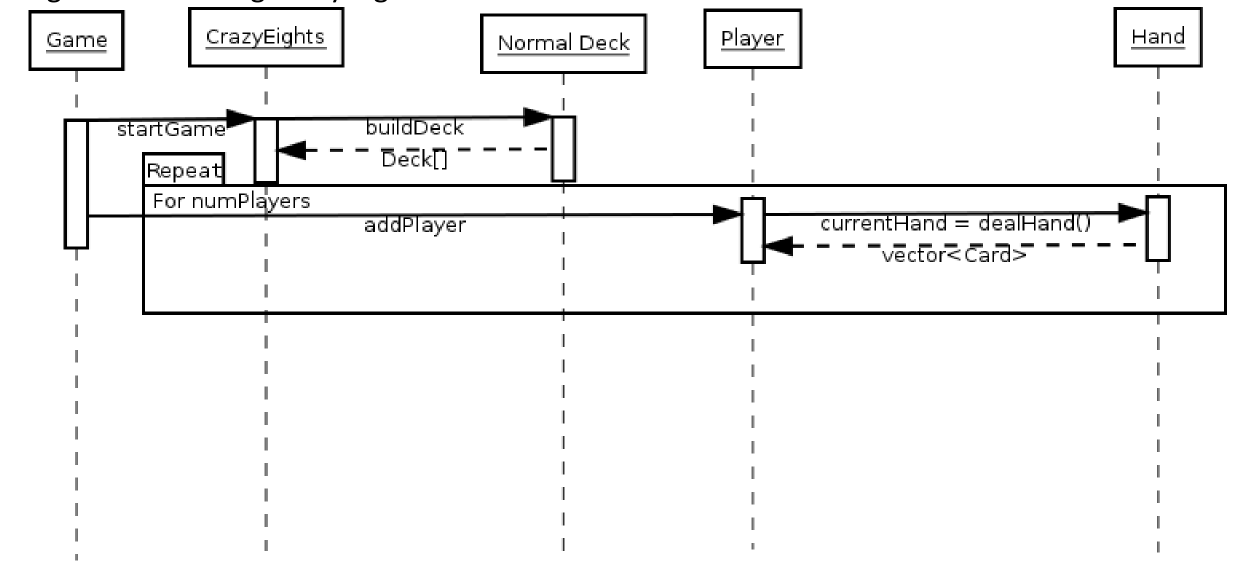


Diagram for Crazy Eights turn:

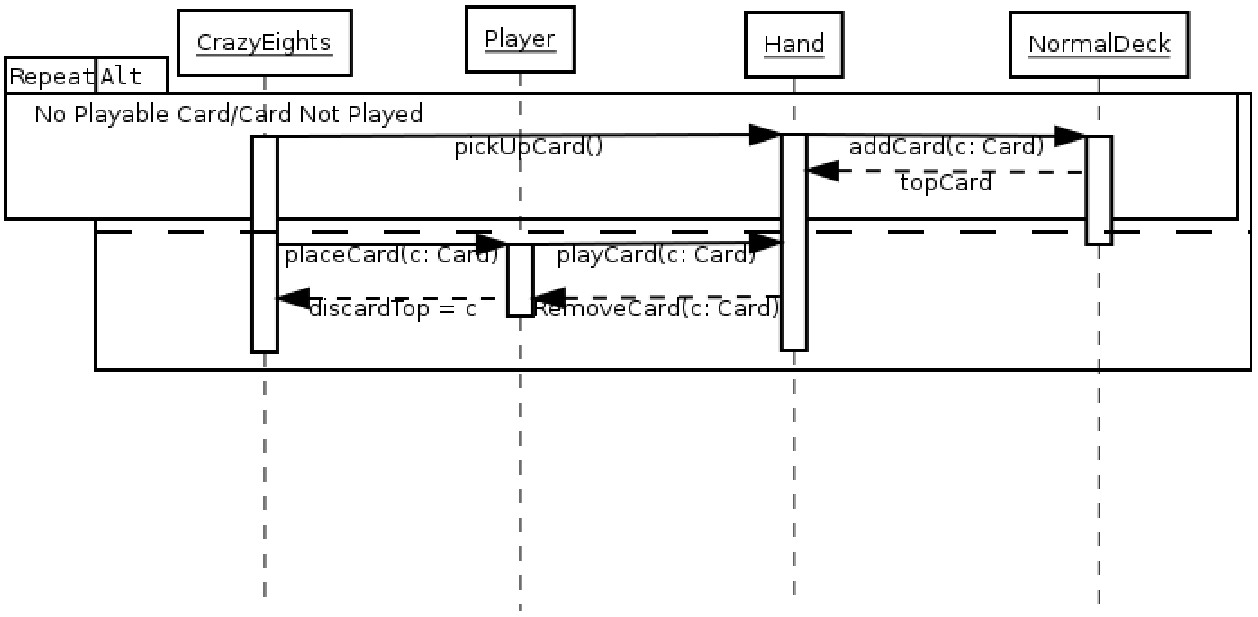


Diagram for Crazy Eights end:

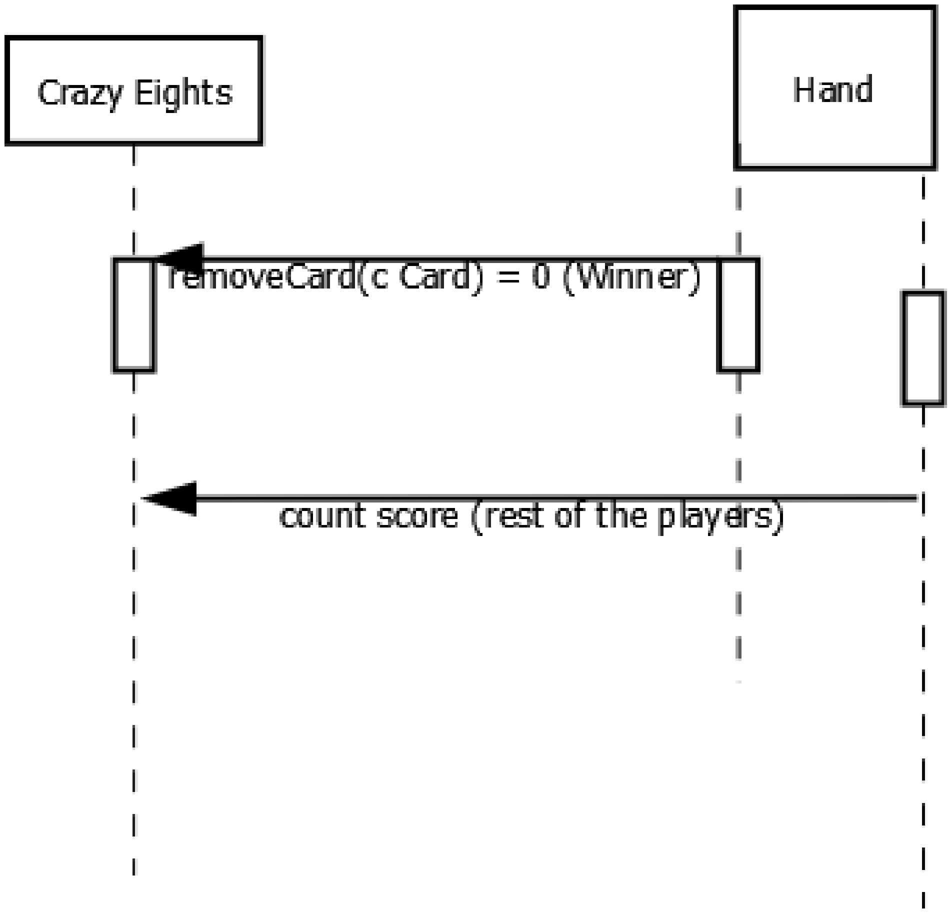

Crazy Eights
Hand
removeCard(c Card) = 0 (Winner)
count score (rest of the players)

Diagram for starting the Uno game:

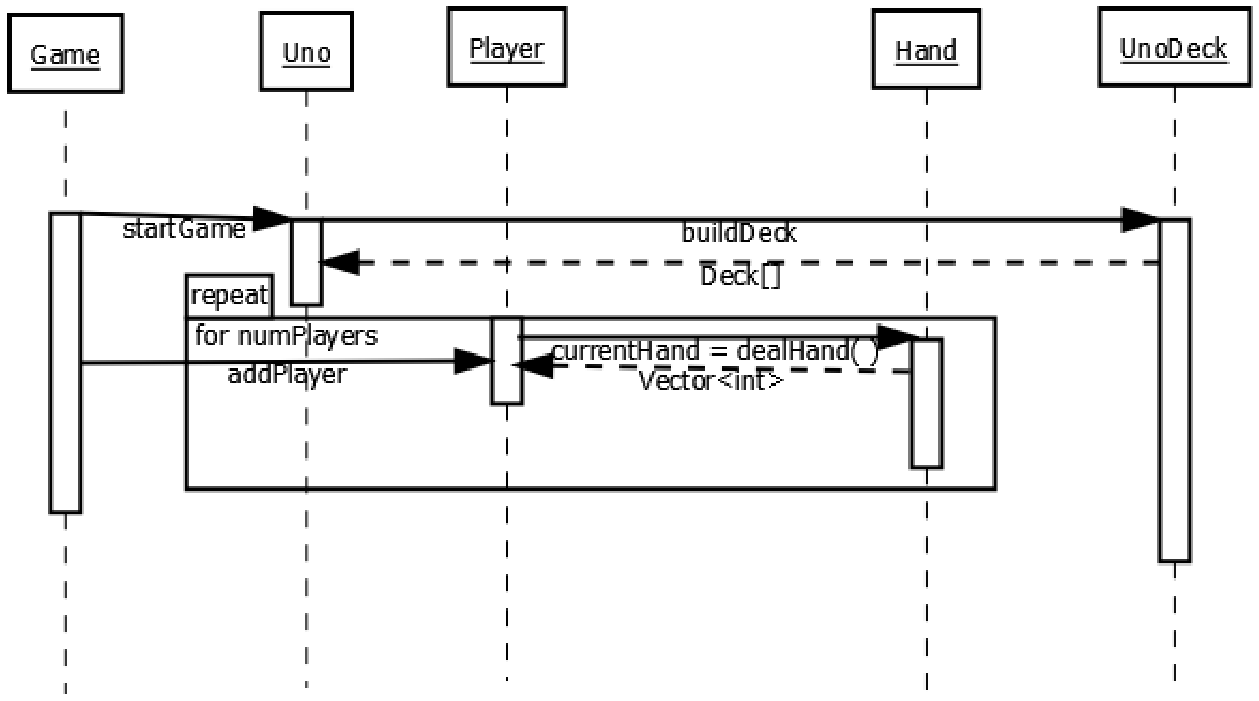


Diagram for an Uno turn:

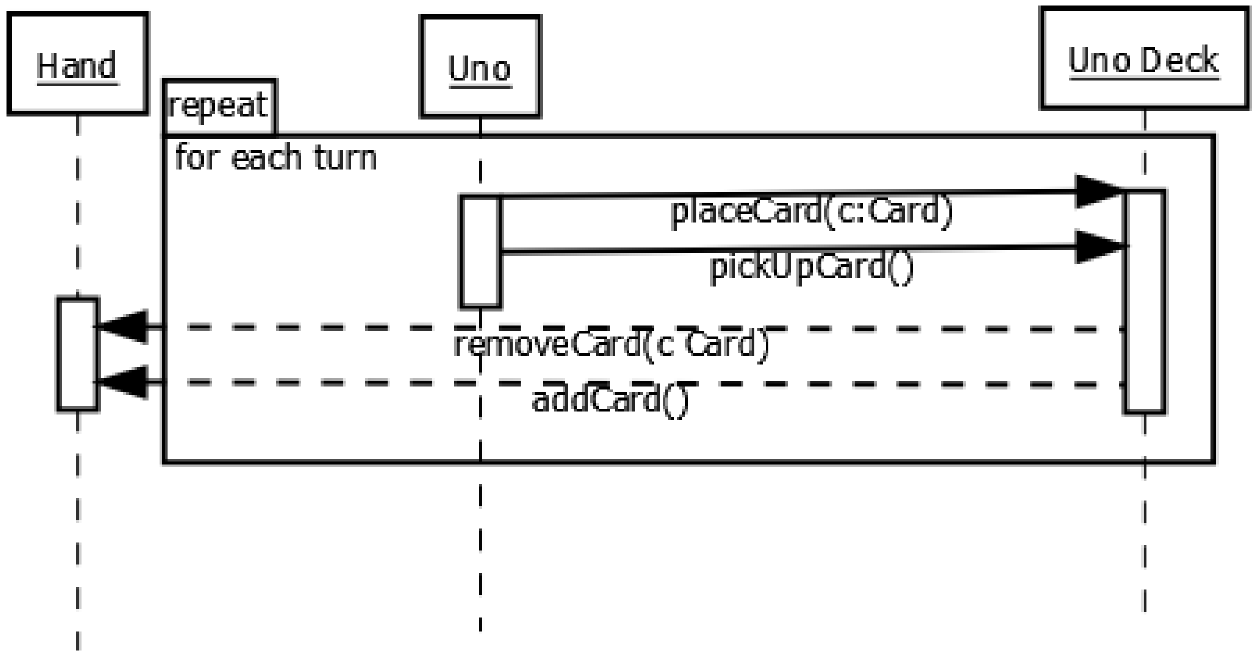
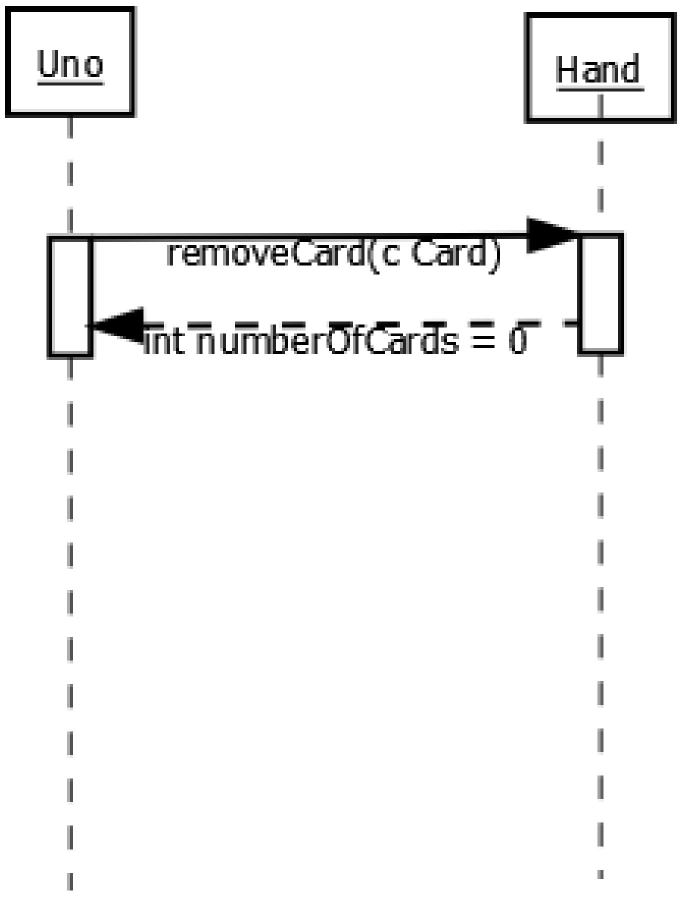


Diagram for Uno end:


Uno
Hand
removeCard(c Card)
int numberOfCards = 0