CS 342: PROJECT 1
5 Card Draw

Bryan Spahr
Khurratul-Ain Naseer

TABLE OF CONTENTS

1. INTRODUCTION

1.1. Purpose

This design document describes the architecture and system design of this project, 5 Card Draw.

1.2. Scope

This project is a simple command-line game of chance, written in Java and consisting of 7 classes.

1.3. Reference Material

Mainly the project handout; also various Java API web sources

2. SYSTEM OVERVIEW

The goal of this game is to get the highest poker hand. The process of this program can best be described by these steps:

a) The user runs the program, which will prompt for the desired number of computer opponents.
b) The user enters their desired number of computer opponents.
c) Deck of 52 cards is created and dealt one card at a time until all players have 5 cards.
d) Each computer player is given the opportunity to discard up to 3 cards from their hand. The computer opponents will discard theirs automatically.
e) The user is given the opportunity to evaluate their poker hand and then discard 3 cards. If their hand contains an ace, they may discard 4 cards.
f) All players' hands are replenished, one at a time, until each player again has 5 cards.
g) All hands are evaluated and the player with the highest poker hand wins.

3. SYSTEM ARCHITECTURE

3.1. Architectural Design/Overview of Classes

Card: Object representation of a card. Each Card contains a two-character ID (e.g. 5C = five of clubs) to represent its rank and suit and a numerical representation of its rank (2-14), along with many getters, setters, and various methods. This class is sortable (descending) by its numerical rank by implementing Comparable.

Hand: Contains a list of 5 cards and all methods for determining its poker value and (for computer players) which cards it will discard. As cards are added to the hand, it will wait until it has 5 cards, sort the list by card rank, and automatically determine its poker value.

CardPile: Wrapper class that contains an collection of Cards and various methods, getters, and setters. Included is a function that will initialize a new 52-card deck of cards for you.

Player: Superclass representation of a game participant. Each player has a Hand and various getters, setters, and methods. This class is sortable (descending) by its hand's numerical poker rank by implementing Comparable.

OpponentPlayer: Represents the computer player. Simply extends superclass Player.

UserPlayer: Represents the human player. Simply extends superclass Player.

3.2. Design Rationale

The Card, CardPile, UserPlayer, OpponentPlayer, and Game classes were required. The Player class was created as a superclass so that a list could be made containing both UserPlayer and OpponentPlayer objects that could be sorted by poker rank. The Hand class was created as a concise way to contain players' hands and determine its own poker value.

4. DATA DESIGN

4.1. Data Description

All collections of objects in this codebase are stored in ArrayLists.

4.2. Data Dictionary

Card:
```
char[] id;
int rank;
int suit;
boolean keep;
public Card(String in);

public String getRankString();
public String getSuitString();
public String getNameAsString();
public char[] getId();
public String getIdString();
public int getSuit();
public void setSuit(int suit);
public int getRank();
public void setRank(int rank);
public void setRanks();
public boolean willKeep();
public void setKeep(boolean keep);
public int compareTo(Card c);
```

CardPile:
```
private List<Card> pile;
```

```
public CardPile(boolean newDeck);

public void initializeDeck();
public void shuffleDeck(int numTimes);
public String pileAsString();
public void add(Card c);
public void remove(Card c);
public Card get(int x);
public int size();
```

Game:

```
public static void main(String[] args);
```

Hand:

```
private int pokerRank;
private int handSum;
private List<Card> hand;
public Hand();

public void determinePoker();
public void determineDiscards();
public int getRankOccurences(int n);
public int getSuitOccurences(int n);
public boolean areSequential(List<Card> cards);
public void setHandSum();
public String handAsString(boolean numbering);
public void sortHand();
public String getPokerString();
public void add(Card c);
public void remove(Card c);
public Card get(int x);
public void setKeep(int x, boolean val);
public void setAllKeep(boolean val);
public void setAllKeep(int from, int to, boolean val);
public int getPokerRank();
public void setPokerRank(int pokerRank);
public List<Card> getHand();
public int getHandSum();
```

OpponentPlayer:

```
public OpponentPlayer(String s);
```

Player:

```
private Hand hand;
private int handRank;
private String name;
public Player(String s);

public void refresh();
public void addToHand(Card c);
public void removeFromHand(Card c);
public int getPokerRank();
public void setPokerRank();
public int getHandSum();
public String getName();
public Hand getHand();
public int compareTo(Player p);
```

UserPlayer:

```
public UserPlayer(String s);
```

5. COMPONENT DESIGN

Tie-breaking was resolved by maintaining the sum of all the card ranks in a hand. If two cards are tied with the same poker value, the card with the higher rank sum wins.
The main algorithms used are in the Hand class, given it does most of the work. Here are its main methods:
(Note: the hand is already sorted by rank in descending order at this point)

`public void determinePoker();`
This function's purpose is to determine what the poker value of its hand is. To do this, it has a series of if – else if statements checking for each distinct poker hand from highest (straight flush) to lowest (high card). This way if multiple conditions match only the highest poker rank will actually be stored. This function makes heavy use of **`getRankOccurences().`**

`public int getRankOccurences(int n);`
Here is where the main algorithm happens. n represents the expected total number of matching cards within a hand (e.g. n = 3 means you're looking for 3 of a kind).
First the function gets the rank of the first card in the hand. It then takes that rank and counts the number of matches to all the cards in the hand.
Then it creates a subset of the original hand from the 2nd card to the end, gets the rank of the first card, and compares with this hand. This process is repeated until the subset hand size = n. The function then returns the sum of matches from each check. For example:

Hand: K T 5 5 2
n = 3 (checking for 3 of a kind)

1st check:	K T 5 5 2	rank = 13 (K)	matches = 1
2nd check:	T 5 5 2	rank = 10 (T)	matches = 1
3rd check:	5 5 2	rank = 5	matches = 2

The function would return 1 + 1 + 2 = 4. Had there actually been 3 of a kind, the function would have returned 6.

So based on the input n and what it returns, I can determine the hand.

```
public int getSuitOccurences(int n);
```
Same as **`getRankOccurences`**. This function didn't really need the same algorithm as **`getRankOccurences`**, but it was easier to just use the same code and check for matching suits instead of ranks.

```
public boolean areSequential(List<Card> cards);
```
Takes in a list of cards. Since sequential is defined as n, n-1, n-2, n-3, n-4 (e.g. Q, J, T, 9, 8) the function defines n as the rank of the first card and creates an int array to contain n, n-1, n-2, n-3, n-4. It then loops through **`cards`** and compares the integers in the array with the rank of each card. If there are no mismatches, the list of cards is sequential; otherwise, the list of cards is not sequential.
If there is an ace in the list, a new hand is created based on the input list (but with the ace reassigned to rank = 1) and then sorted. The same process is repeated. If either sets of checks are true, the function returns true.

6. HUMAN INTERFACE DESIGN

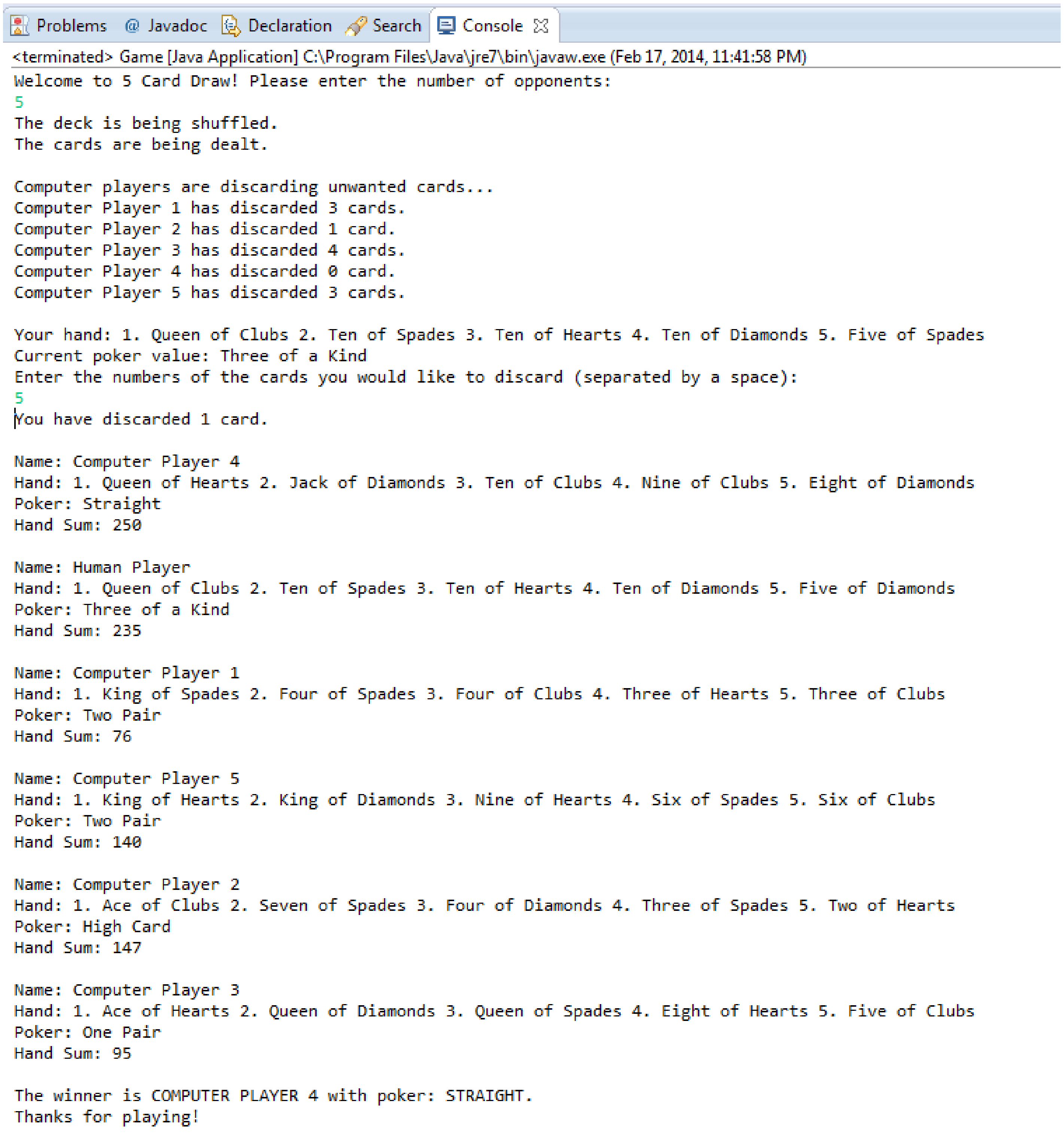
6.1. Overview of User Interface

Gameplay is as described in Section 2. Given this is a command-line program, the only interaction with the user is when prompting for input and when outputting to the console. After prompting for the number of opponents, the program will say:
```
The deck is being shuffled.
The cards are being dealt.
Computer players are discarding unwanted cards...
```

As each computer player performs their discards, the console will state how many cards they discarded, e.g.
```
Computer Player 2 has discarded 3 cards.
```

Then the user is shown their hand and its poker value and given the opportunity to choose their discards.
Finally, all the players' hands are printed, along with their poker value, and the winner is stated.

6.2. **Screenshots**



```
<terminated> Game [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (Feb 17, 2014, 11:41:58 PM)
Welcome to 5 Card Draw! Please enter the number of opponents:
5
The deck is being shuffled.
The cards are being dealt.

Computer players are discarding unwanted cards...
Computer Player 1 has discarded 3 cards.
Computer Player 2 has discarded 1 card.
Computer Player 3 has discarded 4 cards.
Computer Player 4 has discarded 0 card.
Computer Player 5 has discarded 3 cards.

Your hand: 1. Queen of Clubs 2. Ten of Spades 3. Ten of Hearts 4. Ten of Diamonds 5. Five of Spades
Current poker value: Three of a Kind
Enter the numbers of the cards you would like to discard (separated by a space):
5
You have discarded 1 card.

Name: Computer Player 4
Hand: 1. Queen of Hearts 2. Jack of Diamonds 3. Ten of Clubs 4. Nine of Clubs 5. Eight of Diamonds
Poker: Straight
Hand Sum: 250

Name: Human Player
Hand: 1. Queen of Clubs 2. Ten of Spades 3. Ten of Hearts 4. Ten of Diamonds 5. Five of Diamonds
Poker: Three of a Kind
Hand Sum: 235

Name: Computer Player 1
Hand: 1. King of Spades 2. Four of Spades 3. Four of Clubs 4. Three of Hearts 5. Three of Clubs
Poker: Two Pair
Hand Sum: 76

Name: Computer Player 5
Hand: 1. King of Hearts 2. King of Diamonds 3. Nine of Hearts 4. Six of Spades 5. Six of Clubs
Poker: Two Pair
Hand Sum: 140

Name: Computer Player 2
Hand: 1. Ace of Clubs 2. Seven of Spades 3. Four of Diamonds 4. Three of Spades 5. Two of Hearts
Poker: High Card
Hand Sum: 147

Name: Computer Player 3
Hand: 1. Ace of Hearts 2. Queen of Diamonds 3. Queen of Spades 4. Eight of Hearts 5. Five of Clubs
Poker: One Pair
Hand Sum: 95

The winner is COMPUTER PLAYER 4 with poker: STRAIGHT.
Thanks for playing!
```