CSCI212 Project #4

Simulate the card game, Go Fish. (See rules below)
There should be 2 players, *(you and the computer).*
To accomplish this, the program should have the following classes:

- Card – a suit & a rank
- Hand – LinkedList of Cards
- Deck – ArrayList of Cards
- Player – a name, a Hand, points
- Game – array of Players, a Deck

Program and all methods should be well documented.

Go Fish!

Rules:
A standard 52 card deck is used.
7 cards are dealt to each player.
The remaining cards are left in the deck.

Players take turns asking each other for cards.

A turn consists of asking the other player for a specific rank. The player asking must already hold at least one card of the requested rank.

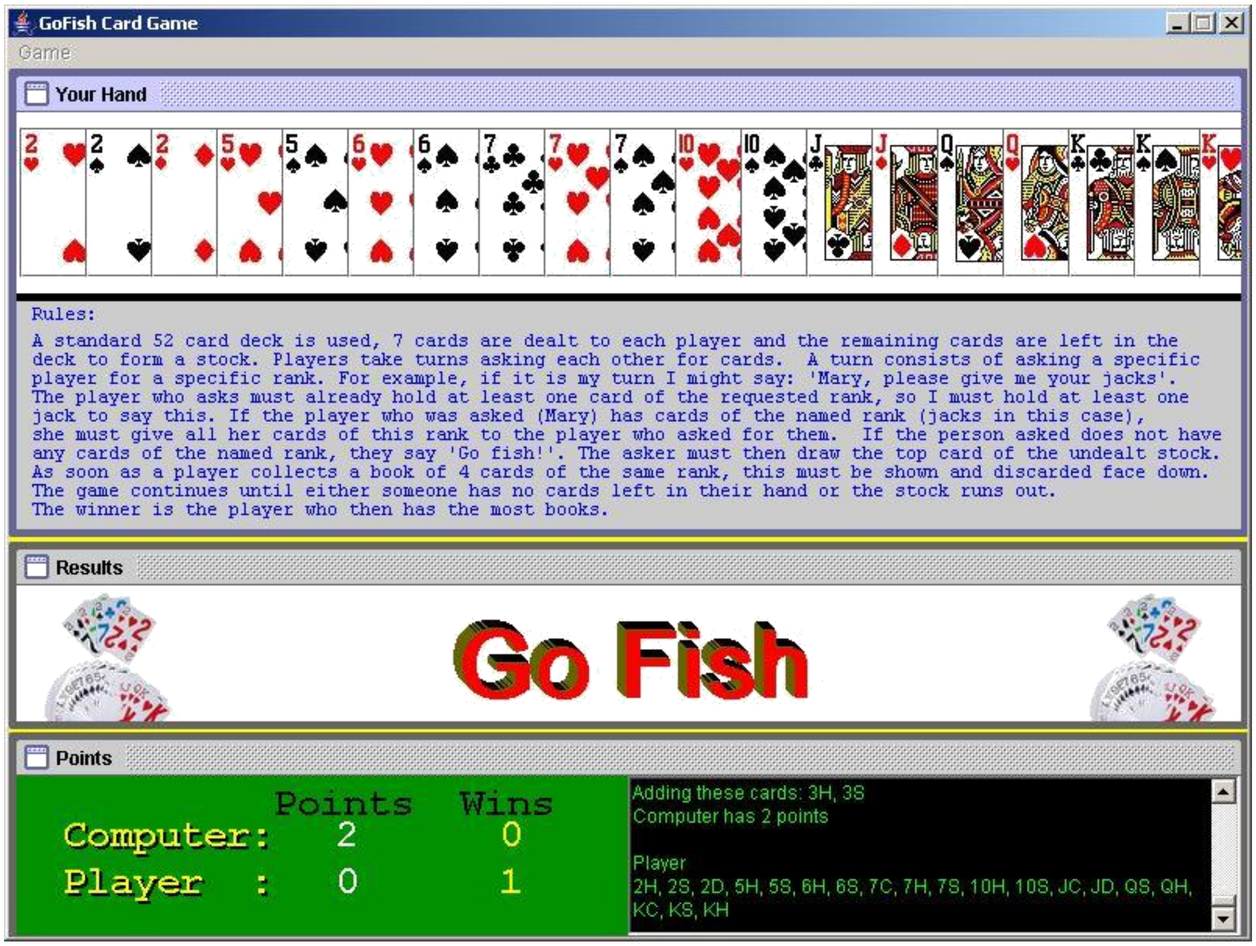
If the other player has cards of the specified rank, that player must give all of the cards of that rank to the player requesting them.

If the other player does not have any cards of the named rank, they should say 'Go fish!'.
The requester must then draw the top card from the deck.

As soon as a player collects a book of 4 cards of the same rank, the cards should be displayed and discarded.

The game continues until either someone has no cards left in their hand or the deck is empty.

The winner is the player with the most books.



Rules:

A standard 52 card deck is used, 7 cards are dealt to each player and the remaining cards are left in the
deck to form a stock. Players take turns asking each other for cards. A turn consists of asking a specific
player for a specific rank. For example, if it is my turn I might say: 'Mary, please give me your jacks'.
The player who asks must already hold at least one card of the requested rank, so I must hold at least one
jack to say this. If the player who was asked (Mary) has cards of the named rank (jacks in this case),
she must give all her cards of this rank to the player who asked for them. If the person asked does not have
any cards of the named rank, they say 'Go fish!'. The asker must then draw the top card of the undealt stock.
As soon as a player collects a book of 4 cards of the same rank, this must be shown and discarded face down.
The game continues until either someone has no cards left in their hand or the stock runs out.
The winner is the player who then has the most books.

	Points	Wins
Computer:	2	0
Player :	0	1

```
Adding these cards: 3H, 3S
Computer has 2 points

Player
2H, 2S, 2D, 5H, 5S, 6H, 6S, 7C, 7H, 7S, 10H, 10S, JC, JD, QS, QH,
KC, KS, KH
```

Figure 1 - GUI Implementation

Text (Command Line) Implementation

```
***************************************************************
Computer [Books: 0]
3D 3C 3H 4S 4H 6H 9H 9C 10S 10D 10C JD QS QC QH AC AS
John, do you have any: 9
John says "No, Go Fish!!"
Book: [QC, QD, QH, QS]
Computer [Books: 1]
3D 3C 3H 4S 4H 6H 9H 9C 10S 10D 10C JD AC AS
***************************************************************
John [Books: 1]
2S 2C 2H 4C 5H 5S 5C 6D 6S 6C 8H 8D 10H JC JH KC KH
Computer, do you have any: 4 Computer says "Yes!!"

John [Books: 1]
2S 2C 2H 4C 4S 4H 5H 5S 5C 6D 6S 6C 8H 8D 10H JC JH KC KH
***************************************************************
Computer [Books: 1]
3D 3C 3H 6H 9H 9C 10S 10D 10C JD AC AS
John, do you have any: A
John says "No, Go Fish!!"
Computer [Books: 1]
3D 3C 3H 6H 8C 9H 9C 10S 10D 10C JD AC AS
************************************************************
John [Books: 1]
2S 2C 2H 4C 4S 4H 5H 5S 5C 6D 6S 6C 8H 8D 10H JC JH KC KH
Computer, do you have any: 5
Computer says "No, Go Fish!!"
John [Books: 1]
2S 2C 2H 3S 4C 4S 4H 5H 5S 5C 6D 6S 6C 8H 8D 10H JC JH KC KH
***************************************************************
Computer [Books: 1]
3D 3C 3H 6H 8C 9H 9C 10S 10D 10C JD AC AS
John, do you have any: 10
John says "Yes!!"
Book: [10C, 10D, 10H, 10S]
Computer [Books: 2]
3D 3C 3H 6H 8C 9H 9C JD AC AS
***************************************************************
John [Books: 1]
2S 2C 2H 3S 4C 4S 4H 5H 5S 5C 6D 6S 6C 8H 8D JC JH KC KH
Computer, do you have any: 7
You must choose a rank that you have in your hand.  Try again.
Computer, do you have any: k
Computer says "No, Go Fish!!"
John [Books: 1]
2S 2C 2H 3S 4C 4S 4H 5H 5S 5C 6D 6S 6C 8H 8D 8S JC JH KC KH
***************************************************************
Computer [Books: 2]
3D 3C 3H 6H 8C 9H 9C JD AC AS
John, do you have any: 3
John says "Yes!!"
Book: [3C, 3D, 3H, 3S]
Computer [Books: 3]
6H 8C 9H 9C JD AC AS
***************************************************************
John [Books: 1]
2S 2C 2H 4C 4S 4H 5H 5S 5C 6D 6S 6C 8H 8D 8S JC JH KC KH
Computer, do you have any:
```

```
import java.io.*;
import java.util.*;

public class Driver {
     public static void main(String[] args) throws IOException
     {
          GoFish game = new GoFish();
          game.playGame();
     }
}
```

GoFish	
-**players** : Player[] -**deck** : Deck	Array of Players Deck of Cards
+**GoFish**() +**getNames**():void +**dealCards**():void +**playGame**():void +**displayHands**():void +**getRank**(Player): int +**gameResults**():void	Default constructor Input Players' names Deal 7 Cards to each Player Display Player info for all players Input rank from the keyboard Display game results

`public class Card extends java.lang.Object`

Field Summary

`private int`	**rank**	- an integer between 0 - 12 representing the card's rank
`private int`	**suit**	- an integer between 0 - 3 representing the card's suit

Constructor Summary

Card`()` Card default constructor -- gets called when an object of the Card class is instantiated – values for *rank* and *suit* are randomly assigned
Card`(int n)` Card constructor -- gets called when an object of the Card class is instantiated -- the rank and suit of the card are determined based on the number received (0 - 51)
Card`(int r, int s)` Card constructor -- gets called when an object of the Card class is instantiated -- r represents the rank, and s represents the suit of the card

Method Summary

`int`	**compareByRank**`(Card otherCard)` -- this method compares 2 Card objects by rank and returns a negative integer, zero, or a positive integer as this Card is less than, equal to, or greater than the other Card.
`int`	**compareBySuit**`(Card otherCard)` -- this method compares 2 Card objects by suit and returns a negative integer, 0, or a positive integer indicating if this Card is less than, equal to, or greater than the other Card.
`boolean`	**equals**`(Card otherCard)` -- indicates whether some other Card is "equal to" this one.
`int`	**getRank**`()` -- returns what's stored in the instance variable rank
`java.lang.String`	**getRankAsString**`()` -- returns a String representation of the instance variable rank
`int`	**getSuit**`()` -- returns what's stored in the instance variable suit
`java.lang.String`	**getSuitAsString**`()` -- returns a String representation of the instance variable suit
`void`	**setRank**`(int r)` -- modifies the value of the instance variable rank
`void`	**setSuit**`(int s)` -- modifies the value of the instance variable suit
`java.lang.String`	**toString**`()` -- returns a String representation of the Card

```
public class GoFishCard extends Card
      implements java.lang.Comparable<GoFishCard>
```

Constructor Summary

`GoFishCard()`	Default Constructor
`GoFishCard(int n)`	Parameterized Constructor
`GoFishCard(int r, int s)`	Parameterized Constructor – *r* is the rank and *s* is the suit

Method Summary

`int`	`compareTo(GoFishCard otherCard)` Compares this GoFishCard to another specified GoFishCard, returns -1 if
`static int`	`convertToRank(java.lang.String str)` A static method that converts a string to a card's equivalent rank
`boolean`	`equals(GoFishCard otherCard)` Compares this GoFishCard to another specified GoFishCard
`boolean`	`equals(java.lang.Object otherCard)` Compares this GoFishCard to the specified object

```java
public class GoFishCard extends Card implements Comparable<GoFishCard>{

      public GoFishCard(){  super();   }
      public GoFishCard(int n){ super(n); }
      public GoFishCard(int r, int s){ super(r,s);  }

      public int compareTo(GoFishCard otherCard) {
            return compareByRank((Card) otherCard);
      }
      public boolean equals(GoFishCard otherCard) {
            return (getRank() == otherCard.getRank());
      }

      public boolean equals(Object otherCard) {
            return (getRank() == ((GoFishCard)otherCard).getRank()
                        && getSuit() == ((GoFishCard)otherCard).getSuit());
      }
      public static int convertToRank(String str){
            String[] ranks = { "2", "3", "4", "5", "6", "7", "8", "9", "10", "J",
                        "Q", "K", "A" };
            for(int i=0; i<ranks.length; i++)
                  if(ranks[i].equalsIgnoreCase(str))
                        return i;
            return -1;
      }
}
```

```
public class Deck extends java.lang.Object
```

Field Summary

private java.util.ArrayList<Card>	cards	ArrayList of Cards

Constructor Summary

Deck()

Method Summary

Card	deal()	Returns a card from the Deck or *null* if the Deck is empty
void	initialize()	Generates 52 Cards and stores them in the ArrayList
boolean	isEmpty()	Returns *true* if the Deck is empty, *false* otherwise
void	shuffle()	Shuffles the Deck of Cards
java.lang.String	toString()	Returns a string representation of the Deck

```java
import java.util.ArrayList;
import java.util.Collections;

public class Deck {
	public static final int CARDS_IN_DECK = 52;
	private ArrayList<Card> cards = new ArrayList<Card>();
	public Deck() {
		cards.ensureCapacity(CARDS_IN_DECK);
		initialize();
	}

	public void initialize(){
		for(int i = 0;i < CARDS_IN_DECK; i++)   {
			cards.add(new GoFishCard(i));
		}
	}

	public String toString(){
		return "No. of cards: " + cards.size() +"\n" + cards.toString();
	}

	public void shuffle() {    Collections.shuffle(cards);      }

	public Card deal() {
		if(!cards.isEmpty())
			return cards.remove(0);
		return null;
	}

	public boolean isEmpty(){ return cards.isEmpty();    }
}
```

`public class **Hand** extends java.lang.Object`

Field Summary

`private java.util.LinkedList<GoFishCard>`	**hand**	LinkList of GoFish Cards

Constructor Summary

Hand()	**Default constructor**

Method Summary

`int`	**countRank**(int rank) Counts the number of cards of a particular rank in the hand
`int`	**evaluate**() Returns 1 if a book (all 4 cards of a particular suit) is in the hand and removes the book from the hand
`java.util.LinkedList<GoFishCard>`	**findRank**(int rank) Finds and returns all cards of the specified rank
`GoFishCard`	**getCardAt**(int index) Returns the card at the specified position in this list
`java.util.LinkedList<GoFishCard>`	**getCards**(int rank) Returns a list of cards of a specified rank
`int`	**getCount**() Returns the number of cards in the hand
`java.util.LinkedList<GoFishCard>`	**getHand**() Returns the hand as LinkedList of GoFish cards
`boolean`	**hasRank**(int rank) Returns *true* if this rank is the hand
`void`	**insertByRank**(GoFishCard card) Adds a Card to the hand, the hand is sorted by rank
`void`	**insertHand**(java.util.Collection<? extends GoFishCard> otherHand) Adds a LinkList of Cards to the hand, the hand is sorted by rank
`boolean`	**isEmpty**() Determines if the hand is empty
`java.lang.String`	**toString**() Returns a string representation of the hand

public class **Player** extends java.lang.Object

Field Summary

private Hand	**hand**
private java.lang.String	**name**
private int	**points**

Constructor Summary

Player(java.lang.String n) Parameterized constructor

Method Summary

void	**addCard**(GoFishCard card) Adds a card to the hand
void	**addCards**(java.util.LinkedList<GoFishCard> otherHand) Adds a LinkedList of Cards to the hand
java.util.LinkedList<GoFishCard>	Returns the cards of a specified rank as a Linkedlist
GoFishCard	**getCard**(int rank)
GoFishCard	**getCardAt**(int index) Returns the card at a specified index in the hand
java.util.LinkedList<GoFishCard>	**getCards**(int rank) Returns all of the cards of the specified rank as a LinkedList
java.lang.String	**getName**() Returns the player's name
int	**getPoints**() Returns the number of books the player has
int	**getTotalCards**() Returns the number of cards the player has
boolean	**hasRank**(int rank) Returns *true* if the player has a specified rank
void	**setHand**(Hand hand)
void	**setName**(java.lang.String name) Sets the name
java.lang.String	**showHand**() Returns the string representation of the hand
java.lang.String	**toString**() Returns the string representation of the player