SYST17796 Deliverable 2
Java Go Fish

Aleksandar Lim
Sheridan College 2021

https://github.com/AleksLim13/Java-Go-Fish.git

"The overall grouping of classes falls into 5 main categories."

Go Fish Package Structure:

OVERVIEW PACKAGE CLASS USE TREE DEPRECATED INDEX HELP

JavaGoFish 1.0-SNAPSHOT API

Packages

Package	
ca.sheridancollege.javagofish.Cards	Cards package groups classes related to Cards
ca.sheridancollege.javagofish.Players	Players package has classes related to the Players
ca.sheridancollege.javagofish.Start	Start package has classes related to how the game begins
ca.sheridancollege.javagofish.Turns	Turns package has classes related to turn functionality.
ca.sheridancollege.javagofish.Utility	Utility package has classes for printing and user input.

"The Cards package has four java classes in total. The Card class is abstract. Go Fish Card extends card and Deck and Hand are concrete classes."

Cards Package Classes:

Package ca.sheridancollege.javagofish.Cards

All these classes revolve around functionality for what needs to happen to a Players hand during a game of Go Fish.

Class Summary

Class	Description
Card	This class models Card objects used in a Card game.
Deck	This is the Deck class.
GoFishCard	This class is an extension of the Card class.
Hand	This class has all the functionality for tasks relating to a players hand.

"The Card class is an independent class that many clients will depend on. How the central deck is created and ranges for possible Card asks by players are bound by two important constructs from this class."

Package ca.sheridancollege.javagofish.Cards

Class Card

java.lang.Object
    ca.sheridancollege.javagofish.Cards.Card

Direct Known Subclasses:

GoFishCard

```
public abstract class Card
extends java.lang.Object
```

This class models Card objects used in a Card game. Each card has a value and a suit. Both fields are string values. the range of values for each are stored in a static final array that's of public access allowance.

Author:

AllyCat13. This is the most general category of what a card can be and has. It's abstract and will be extended by classes like Go Fish card or Black Jack Card.

Field Summary

Fields

Modifier and Type	Field	Description
protected java.lang.String	suit	Field variable for a Cards suit.
static java.lang.String[]	suitsRange	A field variable for range of suits a Card can have.
protected java.lang.String	value	A Card needs to store is value characteristic.
static java.lang.String[]	valuesRange	A field variable for range of values a Card can have.

Constructor Summary

Constructors

Constructor	Description
Card()	Default constructor for creating a Card.
Card(java.lang.String value)	Constructs A Card object with value only.
Card(java.lang.String suit, java.lang.String value)	Constructs a Card object when value and suit are known ahead of time.

Method Summary

All Methods | **Instance Methods** | **Abstract Methods** | **Concrete Methods**

Modifier and Type	Method	Description
boolean	equals (java.lang.Object o)	Override the equals method inherited from Object so Cards can be compared and determined if their states are equal per a Cards attributes suit and value.
java.lang.String	getSuit()	Getter for the suit of the Card instance.
java.lang.String	getValue()	Value getter for the Card instance.
int	hashCode()	
abstract java.lang.String	toString()	Override objects toString for easy printing of Card's attributes.

Field Details

suitsRange

```
public static final java.lang.String[] suitsRange
```

A field variable for range of suits a Card can have. It's a String array because the suit values don't change. There's only ever four of them. Strings are easy to work with as much Java API classes can sort, split, or add them. Regular arrays are easy to perform operations on directly.

valuesRange

```
public static final java.lang.String[] valuesRange
```

A field variable for range of values a Card can have. There's thirteen constant values for Card values in a standard deck. It's final because these values don't change. It's static because only one copy is ever needed for reference. This range comes in hand for initializing the deck. All values are encapsulated in one String array variable.

value

```
protected java.lang.String value
```

A Card needs to store is value characteristic. A String indicating it's value is stored here.

Constructor Details

Card

```
public Card()
```

Default constructor for creating a Card. No parameters passed in. Sets Cards suit and value to default values. Ace of Hearts.

Card

```
public Card(java.lang.String value)
```

Constructs A Card object with value only. At times during go fish only a value is asked for. If player has two two's, they ask for any more twos. They don't care about the suit they just want four of a kind.

Parameters:

`value` - of String type.

Card

```
public Card(java.lang.String suit,
            java.lang.String value)
```

Constructs a Card object when value and suit are known ahead of time.

Parameters:

`suit` - of String type.

`value` - of String type.

Method Details

getValue

```
public java.lang.String getValue()
```

Value getter for the Card instance.

Returns:

String value of the Card object.

getSuit

```
public java.lang.String getSuit()
```

Getter for the suit of the Card instance.

Returns:

String suit value of Card object.

toString

```
public abstract java.lang.String toString()
```

Override objects toString for easy printing of Card's attributes.

Overrides:

`toString` in class `java.lang.Object`

Returns:

equals

```
public boolean equals(java.lang.Object o)
```

Override the equals method inherited from Object so Cards can be compared and determined if their states are equal per a Cards attributes suit and value.

Overrides:

`equals` in class `java.lang.Object`

Parameters:

`o` - is the object to be compared.

Returns:

true or false value depending if suits and values match.

hashCode

```
public int hashCode()
```

"The Deck class is a independent class that the Hand class is dependent on. The Hand class is the only class that has a direct reference to a Deck object. This class could easily be made abstract so it can model different kinds of decks for different kinds of Card games in the future."

Package ca.sheridancollege.javagofish.Cards

Class Deck

java.lang.Object
ca.sheridancollege.javagofish.Cards.Deck

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

This is the Deck class. It has behavior for initializing and shuffling a deck of cards. The deck is to be used throughout a card game. Most decks have 52 cards. There usually is 13 values and 4 types of suit.

Author:
AllyCat13 @ Sheridan High 2021

Constructor Summary

Constructors

Constructor	Description
`Deck(java.util.List<Card> deck)`	

Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method	Description
`java.util.List<Card>`	`getDeck()`	A getter method to access the values in a created deck.
`void`	`initDeck()`	This method populates the deck field var full of Card instances.
`java.util.List<Card>`	`removeCard(Card card)`	This method is for removing a Card from a players hand after they've handed it to a opponent.
`void`	`shuffle()`	This method repeats it's body of statements 52 times.

Constructor Details

Deck

```
public Deck(java.util.List<Card> deck)
```

Parameters:
`deck` - of Card list type. Pass empty array list for constructor dependency injection. This constructs a instance of a Deck.

Method Details

getDeck

```
public java.util.List<Card> getDeck()
```

A getter method to access the values in a created deck.

Returns:
the Deck field var of Card list type.

initDeck

```
public void initDeck()
```

This method populates the deck field var full of Card instances. Outer for loop repeats per number of suits in Cards suits range array. Inner for loop repeats per number of values in Cards values range array. So, it will make a whole values range of hearts. Then it will make a whole values range of clubs. Then it will make a whole values range of diamonds. Lastly, it will make a whole values range of spades.

shuffle

```
public void shuffle()
```

This method repeats it's body of statements 52 times. Set the counter limit to the size of the field var deck. This creates a random number from 0 to 52. in total it creates 52 random numbers from 0 to 52. It steps through each slot in the deck list and swaps every slot with a value from a random slot in the same deck list.

removeCard

```
public java.util.List<Card> removeCard(Card card)
```

This method is for removing a Card from a players hand after they've handed it to a opponent.

Parameters:

`card` - is of Card type and the target to be deleted.

Returns:

a Card list representing the updated hand as the result.

"The class Go Fish Card is an extension of a Card. This class is concrete and independent in terms of dependencies. It will be depended on by other clients."

Package ca.sheridancollege.javagofish.Cards

Class GoFishCard

java.lang.Object
  ca.sheridancollege.javagofish.Cards.Card
    ca.sheridancollege.javagofish.Cards.GoFishCard

```
public class GoFishCard
extends Card
```

This class is an extension of the Card class. It is a specific type of Card yet it still takes on values and suits of a standard deck. It is a Go Fish card to be used in the deck during a game. This class doesn't extend much data or functionality from Card other than being a unique implementation.

Author:
AllyCat13 @ Sheridan High 2021

Field Summary

Fields inherited from class ca.sheridancollege.javagofish.Cards.Card

`suit, suitsRange, value, valuesRange`

Constructor Summary

Constructors

Constructor	Description
`GoFishCard()`	This constructor call's Cards default no argument constructor.
`GoFishCard(java.lang.String value)`	Calls Cards value only constructor because players typically ask only for values.
`GoFishCard(java.lang.String suit, java.lang.String value)`	Creates a Card by calling Cards suit and value argument constructor.

Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method	Description
`java.lang.String`	`toString()`	Overrides Objects toString for custom printing of these Card instances.

Methods inherited from class ca.sheridancollege.javagofish.Cards.Card

`equals, getSuit, getValue, hashCode`

Methods inherited from class java.lang.Object

`clone, finalize, getClass, notify, notifyAll, wait, wait, wait`

Constructor Details

GoFishCard

```
public GoFishCard()
```

This constructor call's Cards default no argument constructor.

GoFishCard

```
public GoFishCard(java.lang.String suit,
                  java.lang.String value)
```

Creates a Card by calling Cards suit and value argument constructor.

Parameters:
`suit` - of String type. The suit of this card instance.

`value` - of String type. The value of this card instance.

GoFishCard

```
public GoFishCard(java.lang.String value)
```

Calls Cards value only constructor because players typically ask only for values.

Parameters:

`value` - of String type. The value of this Card instance.

Method Details

toString

```
public java.lang.String toString()
```

Overrides Objects toString for custom printing of these Card instances.

Specified by:

`toString` in class `Card`

Returns:

the String value of the Cards suit and value values.

"Class performs much actions on data of other classes but doesn't directly depend on them as they're mostly parameter arguments. It is dependent on a Deck and a Scoreboard to perform it's full functionality."

Package ca.sheridancollege.javagofish.Cards

Class Hand

java.lang.Object
    ca.sheridancollege.javagofish.Cards.Hand

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

This class has all the functionality for tasks relating to a players hand. Only one of these Hand objects need to be created during a games life cycle. It can be declared as a final field variable in the aggregating class. This class aggregates Deck and Scoreboard as it needs one of each along with their functionality.

Author:

AllyCat13:

Constructor Summary

Constructors

Constructor	Description
`Hand(Deck deck, ScoreBoard sb)`	This constructs a Hand instance with a Deck and Scoreboard object as argument thereby initializing the field variables.

Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method	Description
void	`createHand(int size, Player player)`	Calls this classes start deal method and adds the return value per a desired amount of Cards to a Players hand.
void	`getCardFromDeck(Player player)`	This method is for when a Players asks for a Card and the asked player doesn't have it That player who asked has to "Gi Fish" and draw a Card from the top of the deck.
Deck	`getDeck()`	This is a public getter interface for accessing the Deck field var of Hand instances.
void	`sort(Player player, char option)`	Two Player lists need to be sorted throughout the game for Players to make quick decisions based on their hand.
void	`updateHandAdd(Player inPlay, Card inPlaysDesireC, Player notInPlay)`	public interface for adding a Card to the Player who asked for it if it's also possessed by asked player.
void	`updateHandDelete(Player notInPlay, Card inPlaysDesireC)`	This is the public interface for removing the Card from the asked players hand.

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Hand

```
public Hand(Deck deck,
            ScoreBoard sb)
```

This constructs a Hand instance with a Deck and Scoreboard object as argument thereby initializing the field variables.

Parameters:

`deck` - is of Deck type.

`sb` - is of ScoreBoard type. This constructors calls this classes deck setup method to automatically initialize and shuffle the deck.

Method Details

getDeck

```
public Deck getDeck()
```

This is a public getter interface for accessing the Deck field var of Hand instances.

Returns:

the field variable of Deck type.

getCardFromDeck

```
public void getCardFromDeck(Player player)
```

This method is for when a Players asks for a Card and the asked player doesn't have it That player who asked has to "Gi Fish" and draw a Card from the top of the deck. This method calls another method from this class that chooses a value from a random slot in the deck.

Parameters:

`player` - is of top level Player type.

updateHandDelete

```
public void updateHandDelete(Player notInPlay,
                             Card inPlaysDesireC)
```

This is the public interface for removing the Card from the asked players hand. It deletes from the Players duplicates list to keep hand and duplicate list in sync.

Parameters:

`notInPlay` - top level Player type.

`inPlaysDesireC` - top level Card type.

updateHandAdd

```
public void updateHandAdd(Player inPlay,
                          Card inPlaysDesireC,
                          Player notInPlay)
```

public interface for adding a Card to the Player who asked for it if it's also possessed by asked player. This calls this classes private method add card to hand.

Parameters:

`inPlay` - top level Player type.

`inPlaysDesireC` - top level card type.

`notInPlay` - top level Player type.

createHand

```
public void createHand(int size,
                       Player player)
```

Calls this classes start deal method and adds the return value per a desired amount of Cards to a Players hand.

Parameters:

`size` - as integer for how many cards to include in a Hand.

`player` - top level Player type. The Hand to add cards to.

sort

```
public void sort(Player player,
                 char option)
```

Two Player lists need to be sorted throughout the game for Players to make quick decisions based on their hand. Sort either Players hand or duplicates list.

Parameters:

`player` - top level Player type.

`option` - char type indicating which Player list to work on.

"Many client classes if not all of them will depend on classes from the Player package."

Player Package Classes:

Package ca.sheridancollege.javagofish.Players

This package contains 3 java classes in total. Player is the parent and the two sibling classes are CompPlayer and Human player.

Class Summary

Class	Description
CompPlayer	This a specific type of Player for modeling the computer.
HumanPlayer	This models human players.
Player	Parent class that models a generic Player.

"The CompPlayer class is an extension of Player. This class is a independent concrete class that much clients will depend on for overall functionality of the game."

Class CompPlayer

java.lang.Object
  ca.sheridancollege.javagofish.Players.Player
    ca.sheridancollege.javagofish.Players.CompPlayer

```
public class CompPlayer
extends Player
```

This a specific type of Player for modeling the computer. This class inherits data and functionality from Player.

Author:

AllyCat13 @ Sheridan High 2021.

Field Summary

Fields inherited from class ca.sheridancollege.javagofish.Players.Player

`books, desirableList, hand, name, numOfPlayers, playerId`

Constructor Summary

Constructors

Constructor	Description
`CompPlayer(java.util.List<Card> books, java.util.List<Card> dL)`	Prevent null pointer exception with Card lists copies to initialize field variables.

Method Summary

Methods inherited from class ca.sheridancollege.javagofish.Players.Player

`getBooks, getDesirableList, getHand, getName, getNumOfPlayers, getPlayerId, getPlayerID, printStats, setBooks, setDesirableList, setHand, setName, setNumOfPlayers, setPlayerId`

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Details

CompPlayer

```
public CompPlayer(java.util.List<Card> books,
                  java.util.List<Card> dL)
```

Prevent null pointer exception with Card lists copies to initialize field variables.

Parameters:

`books` - Card list data type.

`dL` - card list data type.

"HumanPlayer class is an extension of Player. It is a independent class and other clients will depend on it."

Package ca.sheridancollege.javagofish.Players

Class HumanPlayer

java.lang.Object
  ca.sheridancollege.javagofish.Players.Player
    ca.sheridancollege.javagofish.Players.HumanPlayer

```
public class HumanPlayer
extends Player
```

This models human players. This class inherits data and functionality from Player.

Author:

alllyCat13 @ Sheridan High 2021

Field Summary

Fields inherited from class ca.sheridancollege.javagofish.Players.Player

`books, desirableList, hand, name, numOfPlayers, playerId`

Constructor Summary

Constructors

Constructor	Description
`HumanPlayer(java.lang.String name, java.util.List<Card> books, java.util.List<Card> dL)`	Constructs a Player and initializes name, books, and duplicate list.

Method Summary

Methods inherited from class ca.sheridancollege.javagofish.Players.Player

`getBooks, getDesirableList, getHand, getName, getNumOfPlayers, getPlayerId, getPlayerID, printStats, setBooks, setDesirableList, setHand, setName, setNumOfPlayers, setPlayerId`

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Details

HumanPlayer

```
public HumanPlayer(java.lang.String name,
                   java.util.List<Card> books,
                   java.util.List<Card> dL)
```

Constructs a Player and initializes name, books, and duplicate list.

Parameters:

`name` - of String type.

`books` - card list type.

`dL` - Card list type.

"Class Player is an abstract class. 2 different child classes extend this class. Other classes are directly dependent on the children of this class."

Package ca.sheridancollege.javagofish.Players

Class Player

java.lang.Object
  ca.sheridancollege.javagofish.Players.Player

Direct Known Subclasses:

`CompPlayer`, `HumanPlayer`

```
public abstract class Player
extends java.lang.Object
```

Parent class that models a generic Player. All the data a Player needs during a Card game is declared and defined here.

Author:

AllyCat13 @ Sheridan High 2021.

Field Summary

Fields

Modifier and Type	Field	Description
protected java.util.List<Card>	books	Each player needs a Card list variable to store their number of four of a kinds.
protected java.util.List<Card>	desirableList	Each Player needs a Card list to store their duplicates to know what to ask for.
protected java.util.List<Card>	hand	Every Player needs a Card list to hold their Hand.
protected java.lang.String	name	Every player needs a String variable to store their name.
protected static int	numOfPlayers	This variable keeps track of number of Players created to set Ids automatically.
protected int	playerId	Each Player needs a number variable to store their Id to differentiate them from other Players.

Constructor Summary

Constructors

Constructor	Description
Player(java.lang.String name, java.util.List<Card> books, java.util.List<Card> dL)	Constructs a Player and initializes the Players name, book, and duplicate list.

Method Summary

All Methods | Static Methods | Instance Methods | Concrete Methods

Modifier and Type	Method	Description
java.util.List<Card>	getBooks()	
java.util.List<Card>	getDesirableList()	
java.util.List<Card>	getHand()	
java.lang.String	getName()	
static int	getNumOfPlayers()	
int	getPlayerId()	
int	getPlayerID()	
java.lang.String	printStats()	Returns String representation of all a Players data.
void	setBooks(java.util.List<Card> books)	
void	setDesirableList(java.util.List<Card> desirableList)	
void	setHand(java.util.List<Card> newHand)	
void	setName(java.lang.String newName)	
void	setName(java.lang.String newName)	
static void	setNumOfPlayers(int numOfPlayers)	
void	setPlayerId(int playerId)	

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Details

name

protected java.lang.String name

Every player needs a String variable to store their name.

name

protected java.lang.String name

Every player needs a String variable to store their name.

hand

protected java.util.List<Card> hand

Every Player needs a Card list to hold their Hand.

numOfPlayers

protected static int numOfPlayers

This variable keeps track of number of Players created to set Ids automatically.

playerId

```
protected int playerId
```

Each Player needs a number variable to store their Id to differentiate them from other Players.

books

```
protected java.util.List<Card> books
```

Each player needs a Card list variable to store their number of four of a kinds.

desirableList

```
protected java.util.List<Card> desirableList
```

Each Player needs a Card list to store their duplicates to know what to ask for.

Constructor Details

Player

```
public Player(java.lang.String name,
              java.util.List<Card> books,
              java.util.List<Card> dL)
```

Constructs a Player and initializes the Players name, book, and duplicate list.

Parameters:

`name` - String type.

`books` - card list type.

`dL` - card list type.

Method Details

setName

```
public void setName(java.lang.String newName)
```

setHand

```
public void setHand(java.util.List<Card> newHand)
```

getName

```
public java.lang.String getName()
```

getHand

```
public java.util.List<Card> getHand()
```

getPlayerID

```
public int getPlayerID()
```

getNumOfPlayers

```
public static int getNumOfPlayers()
```

setNumOfPlayers

```
public static void setNumOfPlayers(int numOfPlayers)
```

getPlayerId

```
public int getPlayerId()
```

setPlayerId

```
public void setPlayerId(int playerId)
```

getDesirableList

```
public java.util.List<Card> getDesirableList()
```

setDesirableList

```
public void setDesirableList(java.util.List<Card> desirableList)
```

getBooks

```
public java.util.List<Card> getBooks()
```

setBooks

```
public void setBooks(java.util.List<Card> books)
```

printStats

```
public java.lang.String printStats()
```

Returns String representation of all a Players data. Easy printing and display of all a Players attributes.

Returns:
concatenated String of all a Players data.

"These classes are important for advancement of the game. Both classes in here are concrete implementations that can be directly instantiated."

Turns Package Classes:

Package ca.sheridancollege.javagofish.Turns

This package contains 2 java classes in total. classes related to what happens during a turn of Go Fish are grouped here for clarity sake.

Class Summary

Class	Description
ScoreBoard	
TurnManager	This class has all the data and functionality needed for tasks that occur during a Players turn.

"Class Score Board is a independent class that Hand and Turn Manager directly depend on or their complete functionality."

Class ScoreBoard

java.lang.Object
    ca.sheridancollege.javagofish.Turns.ScoreBoard

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

Constructor Summary

Constructors

Constructor	Description
`ScoreBoard()`	Default constructor for creating a ScoreBoard to be used in a aggregating class.
`ScoreBoard(Player winner)`	Construct a instance of ScoreBoard when the winner is known ahead of time.

Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method	Description
void	`calcBooks(Card tNum, Player player)`	This method determines if a Player has a four of a kind and then records it.
Player	`determineWinner(Player human, Player computer)`	
void	`getDupes(Player player)`	This method creates a list of duplicates cards a Player has.
Player	`getWinner()`	
void	`setWinner(Player winner)`	

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Details

ScoreBoard

```
public ScoreBoard()
```

Default constructor for creating a ScoreBoard to be used in a aggregating class.

ScoreBoard

```
public ScoreBoard(Player winner)
```

Construct a instance of ScoreBoard when the winner is known ahead of time.

Parameters:

`winner` - top level Player type.

Method Details

getWinner

```
public Player getWinner()
```

setWinner

```
public void setWinner(Player winner)
```

calcBooks

```
public void calcBooks(Card tNum,
                      Player player)
```

This method determines if a Player has a four of a kind and then records it.

Parameters:

`tNum` - top level Card type.

`player` - top level Player type.

getDupes

```
public void getDupes(Player player)
```

This method creates a list of duplicates cards a Player has.

Parameters:

`player` - top level Player type.

determineWinner

```
public Player determineWinner(Player human,
                              Player computer)
```

Parameters:

`human` - top level Player type.

`computer` - top level Player type.

Returns:

Player determined to be the one who made the most progress during the game.

"Class Turn Manager is a major aggregating class. It is directly dependent on Hand, ScoreBoard, HumanPlayer, CompPlayer etc. All the steps of the game have their designated methods in this class."

Class TurnManager

java.lang.Object
ca.sheridancollege.javagofish.Turns.TurnManager

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

This class has all the data and functionality needed for tasks that occur during a Players turn. During a turn, a Players asking for cards is determined to be a success or failure. Whether or not the player should keep asking needs to be determined. the way a computer and human each uniquely ask for cards needs to be defined. A way for the Players status as the one asking versus the one being asked needs to be defined.

Author:

AllyCat13 @ Sheridan High 2021.

Constructor Summary

Constructors

Constructor	Description
`TurnManager(Player human, Player computer, Hand hand, ScoreBoard scoreBoard)`	Constructs a TurnManager instance and initializes Players and helpers.

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method	Description
`Hand`	`getClassHand()`	
`Player`	`getComputer()`	
`Player`	`getHuman()`	
`Player`	`getInPlay()`	
`Player`	`getNotInPlay()`	
`ScoreBoard`	`getScoreBoard()`	
`void`	`setInPlay(Player inPlay)`	
`void`	`setNotInPlay(Player notInPlay)`	
`boolean`	`shouldKeepGoing()`	This method iterates for as long as a players ask was successful.

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Details

TurnManager

```
public TurnManager(Player human,
                   Player computer,
                   Hand hand,
                   ScoreBoard scoreBoard)
```

Constructs a TurnManager instance and initializes Players and helpers.

Parameters:

`human` - top level Player type.

`computer` - top level Player type.

`hand` - Hand class type.

`scoreBoard` - ScoreBoard class type.

Method Details

getInPlay

```
public Player getInPlay()
```

setInPlay

```
public void setInPlay(Player inPlay)
```

getNotInPlay

```
public Player getNotInPlay()
```

getHuman

```
public Player getHuman()
```

getComputer

```
public Player getComputer()
```

setNotInPlay

```
public void setNotInPlay(Player notInPlay)
```

getClassHand

```
public Hand getClassHand()
```

getScoreBoard

```
public ScoreBoard getScoreBoard()
```

shouldKeepGoing

```
public boolean shouldKeepGoing()
```

This method iterates for as long as a players ask was successful. it asks the not asking player for a card, checks their hand if it's there, updates both players hands accordingly, and switches who's during the asking for next round.

Returns:

signal to if the players turn was successful or not.

"The main containers for overall steps of the game are structured in this package. The use cases are contained in RoundOne.java."

Package Start Classes:

Package ca.sheridancollege.javagofish.Start

This package has 5 java classes in total. Classes related to the game as a whole are grouped here.

Class Summary

Class	Description
Game	top level class for modeling generic card games.
GoFish	Specific example of a Game.
Main	The Main class with the main method for where the program starts.
RoundOne	This class contains the steps for a full round of the Go Fish game.
Start	This class models how a Game starts and plays.

"Class game is a parent for child classes that are types of Car games. We differentiate between one game and another through extending and providing a implementation for this class."

Package ca.sheridancollege.javagofish.Start

Class Game

java.lang.Object
  ca.sheridancollege.javagofish.Start.Game

Direct Known Subclasses:

GoFish

```
public abstract class Game
extends java.lang.Object
```

top level class for modeling generic card games. An instance like Black Jack or Big Two can extend this class.

Author:

AllyCat13 : Sheridan High 2021.

Constructor Summary

Constructors

Constructor	Description
Game()	Basic constructor for creating a game example.

Method Summary

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Method Summary

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Details

Game

```
public Game()
```

Basic constructor for creating a game example.

"Class Go Fish is a extension of Game. We can differentiate a game of Go Fish versus Black Jack now."

Package ca.sheridancollege.javagofish.Start

Class GoFish

java.lang.Object
  ca.sheridancollege.javagofish.Start.Game
    ca.sheridancollege.javagofish.Start.GoFish

```
public class GoFish
extends Game
```

Specific example of a Game. This is a Go Fish version of a Card game.

Author:

AllyCat13 : Sheridan High 2021

Constructor Summary

Constructors

Constructor	Description
`GoFish(TurnManager turnController)`	Constructs a instance of Go Fish and initializes TurnManager.
`GoFish(java.lang.String hName, TurnManager tN)`	Constructs a Go Fish instance and initializes name and turn manager.

Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method	Description
`TurnManager`	`getTurnController()`	

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Details

GoFish

```
public GoFish(java.lang.String hName,
              TurnManager tN)
```

Constructs a Go Fish instance and initializes name and turn manager.

Parameters:

`hName` - String type.

`tN` - TurnManager type.

GoFish

```
public GoFish(TurnManager turnController)
```

Constructs a instance of Go Fish and initializes TurnManager.

Parameters:

`turnController` - TurnManager type.

Method Details

getTurnController

```
public TurnManager getTurnController()
```

“Class Main is the java compiler official program starter. I have to have a class like this.”

Package ca.sheridancollege.javagofish.Start

Class Main

java.lang.Object
    ca.sheridancollege.javagofish.Start.Main

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

The Main class with the main method for where the program starts.

Author:
aleks

Constructor Summary

Constructors

Constructor	Description
`Main()`	

Method Summary

All Methods | **Static Methods** | **Concrete Methods**

Modifier and Type	Method	Description
`static void`	`main(java.lang.String[] args)`	The main method.

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Details

Main

```
public Main()
```

Method Details

main

```
public static void main(java.lang.String[] args)
```

The main method. Creates the RoundOne object. Calls the class RounOne play method.

Parameters:
`args` -

“All use cases are in here. This class is a child of Start. It extends Starts play method. Each type of game starts and plays in their own unique way. This class uses a TurnManager that aggregates all of the games functionality together in one class.”

Package ca.sheridancollege.javagofish.Start

Class RoundOne

java.lang.Object
ca.sheridancollege.javagofish.Start.Start
ca.sheridancollege.javagofish.Start.RoundOne

```
public class RoundOne
extends Start
```

This class contains the steps for a full round of the Go Fish game. This class extends the Start class and overrides the play method declared in Start.

Author:
AlllyCat13 @ Sheridan High 2021.

Constructor Summary

Constructors

Constructor	Description
`RoundOne(Game game)`	Constructs a instance of RoundOne and initializes the Game field.

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method	Description
`void`	`play()`	Overrides Start play method per the unique steps of a GoFish round.

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Details

RoundOne

```
public RoundOne(Game game)
```

Constructs a instance of RoundOne and initializes the Game field.

Parameters:
`game` -

Method Details

play

```
public void play()
```

Overrides Start play method per the unique steps of a GoFish round. This method creates hands, sets who's in play, and calls keep going method repeatedly.

Specified by:
`play` in class `Start`

"This is a abstract class. Child classes that represent different types of rounds a game can have will extend this class."

Package ca.sheridancollege.javagofish.Start

Class Start

java.lang.Object
    ca.sheridancollege.javagofish.Start.Start

Direct Known Subclasses:

RoundOne

```
public abstract class Start
extends java.lang.Object
```

This class models how a Game starts and plays. Each instance of this class will define the way that the particular game plays.

Author:

AllyCat13 : Sheridan High 2021.

Constructor Summary

Constructors

Constructor	Description
`Start()`	

Method Summary

All Methods | **Instance Methods** | **Abstract Methods**

Modifier and Type	Method	Description
`abstract void`	`play()`	This method is where the programs overall steps will be defined.

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Details

Start

```
public Start()
```

Method Details

play

```
public abstract void play()
```

This method is where the programs overall steps will be defined.

"Package containing classes for printing and getting user input from the console."

Utility Package Classes:

Package ca.sheridancollege.javagofish.Utility

there's 2 classes in total in this package. They are both independent concrete classes.

Class Summary

Class	Description
Printer	Class for printing Player lists to the console so the human player knows what's happening and why.
UInput	

"Class Printer has one method to print a List of Cards."

Package ca.sheridancollege.javagofish.Utility

Class Printer

java.lang.Object
ca.sheridancollege.javagofish.Utility.Printer

```
public final class Printer
extends java.lang.Object
```

Class for printing Player lists to the console so the human player knows what's happening and why.

Author:
AllyCat13 @ Sheridan High 2021

Constructor Summary

Constructor Summary

Constructors

Constructor	Description
`Printer()`	We don't need to create a instance of Printer because we have simple static display methods.

Method Summary

All Methods | **Static Methods** | **Concrete Methods**

Modifier and Type	Method	Description
`static void`	`printHand(java.util.List<Card> tHand)`	Static method for looping through a list and printing each item.

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Details

Printer

```
public Printer()
```

We don't need to create a instance of Printer because we have simple static display methods.

Method Details

printHand

```
public static void printHand(java.util.List<Card> tHand)
```

Static method for looping through a list and printing each item.

Parameters:
`tHand` - Card list type.

Package ca.sheridancollege.javagofish.Utility

Class UInput

java.lang.Object
    ca.sheridancollege.javagofish.Utility.UInput

```
public final class UInput
extends java.lang.Object
```

Constructor Summary

Constructors

Constructor	Description
UInput()	

Method Summary

All Methods | Static Methods | Instance Methods | Concrete Methods

Modifier and Type	Method	Description
java.util.Scanner	getInput()	
static double	promptDoubleUser()	When a decimal value is needed from the user.
static int	promptIntUser()	When a whole number is needed from the user.
static java.lang.String	promptStringUser()	When we need a word or sentence from the user.
static void	setInput(java.util.Scanner input)	

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Details

UInput

```
public UInput()
```

Method Details

setInput

```
public static void setInput(java.util.Scanner input)
```

getInput

```
public java.util.Scanner getInput()
```

promptDoubleUser

```
public static double promptDoubleUser()
```

When a decimal value is needed from the user.

Returns:
Double type result.

promptIntUser

```
public static int promptIntUser()
```

When a whole number is needed from the user.

Returns:
integer as the result.

promptStringUser

```
public static java.lang.String promptStringUser()
```

When we need a word or sentence from the user.

Returns:
String type as result.

"RoundOne.java Use Cases. There's 7 in total and they are each main section in RoundOne's play method definition."

1. Creating Hands.
2. Creating Duplicate Lists.
3. Setting who goes first.
4. Displaying Hands and Duplicate Lists.
5. Sorting hands and Duplicate Lists.
6. Beginning A Turn.
7. Calculating the previous in play players books.

Creating Hands:

- Use Turn Controller reference that the Game reference of RoundOne has access to to access class hand reference variable contained in Turn Controller.
- Call class hands create hand method and pass in number for how many cards and the player who's hand is being created.
- Create Hand method calls start deal in Hand class and adds the return value to the passed in Player hand field variable.
- Start Deal in hand class use a Deck reference to take a card from it and remove it from the deck.

Creating Duplicate Lists:

Use Games reference to Turn Controller to access ScoreBoard reference.
Execute ScoreBoard's get dupes method and pass in the Player to create the list for.
Get Dupes will compare every card in the players hand to every other and if a duplicate is found will add it to a different list each player has called desirable list.

Setting Who Goes First:

Use Games reference to TurnController and use it's setter for A in play or not in play player.
Set the in play player to human by passing the human player to the setter.
Set the not in play player to computer by passing in the computer player to the setter method.

Displaying Hands and Duplicate Lists:

Use Games reference to TurnController to access the Hands sort method.
Pass in the player and a char option of ‘d’ or ‘h’ to indicate which Player list to sort.
The Hands sort method compares each card to the adjacent one in the list and swaps slots according to a increasing in value order.
Use Games reference to TurnController to get the desired player to pass into the static method print hand of Printer class.

Beginning A Turn:

Use Games reference to TurnController to call TurnController should keep going method.
Should keep going starts a iteration loop of statement that repeat per the player’s ask being successful.
Check whether in play player is a human or computer.
Start asking for a Card using the appropriate Card ask method.
Do this by calling either Human asking for a card or Computer asking for a Card.
Take card returned from above step and pass into TurnController’s Go Fish method.
Go Fish then takes Card passed in and not in play player and checks the hand to see if the card is in there.
If the card is in not in play players hand, take target card from not in play player, give it to player in play, remove from not in play players hand, remove from not in play’s desirable list if there and iterate all steps again.
If card is not in players hand as Go Fish returned true, call hands draw card from deck method and add it to in play players hand.
Call TurnController’s switch player method to change status of which player is currently asking for cards.
Return false immediately to break the loop because in play players turn is complete.

Calculate Books for Previous Player In Play:

Use games reference to TurnController to get that classes reference to ScoreBoard to call the calculate books method contained in ScoreBoard.
Calculate books takes in the Player to be inspected which is now not in play status.
Calculate books will compare every Card in the players hand to duplicate list to see if there are any four of a kinds.
If there are then calculate books will add the value to books list and remove all duplicates from the players hand and duplicate list.

All steps will keep iterating as long as there still are Cards in the central deck.

```java
package ca.sheridancollege.javagofish.Start;


import ca.sheridancollege.javagofish.Utility.Printer;

/**
 * This class contains the steps for a full round of the Go Fish game.
 * This class extends the Start class and overrides the play method declared in Start.
 * @author AlllyCat13 @ Sheridan High 2021.
 */
public class RoundOne extends Start {

    /**
     * A reference to a top level Game example.
     * Only one copy is needed and it's state determined once throughout the game.
     */

    private final Game game;

    /**
     * Constructs a instance of RoundOne and initializes the Game field.
     * @param game
     */
    public RoundOne(Game game)
    {
        this.game = game;
    }//End C:*
     */
    @Override
    public void play()
    {

        if (game instanceof GoFish)
        {
            //Divide: Part One:_____________________________________________

            System.out.println("Creating Hands:");

            ((GoFish)game)
                        .getTurnController()
                        .getClassHand()
                        .createHand(7,  ((GoFish)game)
                                                .getTurnController()
                                                .getHuman());

             ((GoFish)game)
                         .getTurnController()
                         .getClassHand()
                         .createHand(7,  ((GoFish)game)
                                                .getTurnController()
                                                .getComputer());
```

```
//Divide: Part Two:_____________________________________________________
System.out.println("");
System.out.println("Creating Duplicate Lists:");

((GoFish)game)
            .getTurnController()
            .getScoreBoard()
            .getDupes(((GoFish)game)
                                    .getTurnController()
                                    .getHuman());



((GoFish)game)
             .getTurnController()
             .getScoreBoard()
             .getDupes(((GoFish)game)
                                     .getTurnController()
                                     .getComputer());
```

```
//Divide: Part Three:___________________________________________________

System.out.println("");
System.out.println("Setting who goes first");
 ((GoFish)game)
            .getTurnController()
            .setInPlay(((GoFish)game)
                                    .getTurnController()
                                    .getHuman()
            );//End S:*

((GoFish)game)
            .getTurnController()
            .setNotInPlay(((GoFish)game)
                                     .getTurnController()
                                     .getComputer()
            );//End S:*


//Divide: Part Four:____________________________________________________
System.out.println("");
System.out.println("Round One");
```

```
//Define: main while loop control structure. Per if deck still had cards.
while(!((GoFish)game)
                    .getTurnController()
                    .getClassHand()
                    .getDeck()
                    .getDeck()
                    .isEmpty()
       )//End W:*
{

System.out.println("");
System.out.println( "[" + ((GoFish)game)
                                    .getTurnController()
                                    .getInPlay()
                                    .getName() + "]" + " -In Play Hand:");



((GoFish)game)
            .getTurnController()
            .getClassHand()
            .sort(((GoFish)game)
                              .getTurnController()
                               .getInPlay(), 'h');
```

```
Printer.printHand(((GoFish)game)
                                .getTurnController()
                                .getInPlay()
                                .getHand()
                                );

System.out.println("");
System.out.println("[" + ((GoFish)game)
                                    .getTurnController()
                                    .getInPlay()
                                    .getName() + "]" + " -In Play Desirable List:");

((GoFish)game)
            .getTurnController()
            .getClassHand()
            .sort(((GoFish)game)
                                .getTurnController()
                                 .getInPlay(), 'd');
Printer.printHand(((GoFish)game)
                            .getTurnController()
                            .getInPlay()
                            .getDesirableList()
                            );
System.out.println("");
System.out.println("[" + ((GoFish)game)
                                .getTurnController()
                                .getNotInPlay()
                                .getName() + "]" + " -Not In Play Hand:");


((GoFish)game)
            .getTurnController()
            .getClassHand()
            .sort(((GoFish)game)
                            .getTurnController()
                             .getNotInPlay(), 'h');
Printer.printHand(((GoFish)game)
                            .getTurnController()
                            .getNotInPlay()
                            .getHand()
                            );

System.out.println("");
System.out.println("[" + ((GoFish)game)
                                    .getTurnController()
                                    .getNotInPlay()
                                    .getName() + "]" + " -Not In Play Desirable List:");
((GoFish)game)
            .getTurnController()
            .getClassHand()
            .sort(((GoFish)game)
                                .getTurnController()
                                 .getNotInPlay(), 'd');

Printer.printHand(((GoFish)game)
                                .getTurnController()
                                .getNotInPlay()
                                .getDesirableList()
                                );
    ((GoFish)game)
                .getTurnController()
                .shouldKeepGoing();

System.out.println("");
   System.out.println("Calculating books for " + ((GoFish)game)
                .getTurnController()
                .getNotInPlay().getName());
```

```
        for(int i = 0; i < ((GoFish)game)
                                    .getTurnController()
                                    .getNotInPlay()
                                            .getDesirableList()
                                                .size(); i++)
        {

            ((GoFish)game)
                        .getTurnController()
                        .getScoreBoard()
                        .calcBooks(((GoFish)game)
                                                .getTurnController()
                                                .getNotInPlay()
                                                .getDesirableList()
                                                .get(i),
                                                 ((GoFish)game)
                                                            .getTurnController()
                                                            .getNotInPlay());
        }//End F:*
         }//End W:*



    }//End I:*

}//End M:*
```