The University of Hong Kong

Department of Computer Science

COMP2396 Object-oriented Programming and Java

Assignment 3

Deadline: 11:55pm, 27th Oct, 2016.

Overview

This assignment tests your understanding of inheritance and polymorphism, and their implementations in Java. You are going to implement a card game called Big Two. A number of classes will be provided to aid your implementation. These include a Card class which models a card, a CardList class which models a list of cards, a Deck class which models a deck of cards, a CardGamePlayer class which models a card game player, and a BigTwoConsole class which models an user interface for the Big Two card game, respectively. You may refer to their Javadoc for details of these classes. You should **NOT** modify any of these classes in completing your assignment.

As a minimum requirement, you are required to implement the following classes: BigTwo, BigTwoCard, BigTwoDeck, Hand, Single, Pair, Triple, Straight, Flush, FullHouse, Quad and StraightFlush. The BigTwo class models the Big Two card game logics. The BigTwoCard class and the BigTwoDeck class model a card and a deck of cards used in a Big Two card game, respectively. The Hand class models a hand of cards in general card games. The Single, Pair, Triple, Straight, Flush, FullHouse, Quad and StraighFlush classes model hands of legal combinations of cards in a Big Two card game. You are free to introduce new instance variables and methods to these classes. Besides, you are also free to design and introduce new classes in the inheritance trees as appropriate. You should write Javadoc for all non-private classes and their non-private class members.

Specifications

General game rules

You may refer to https://www.pagat.com/climbing/bigtwo.html for a detailed description of the Big Two card game. To simplify your implementation, we will adopt the following rules:

- A standard 52 card pack is used.
- The order of ranks from high to low is 2, A, K, Q, J, 10, 9, 8, 7, 6, 5, 4, 3.
- The order of suits from high to low is Spades, Hearts, Clubs, Diamonds.
- There are always four players in a game.
- Each player holds 13 (randomly assigned) cards at the beginning of the game.
- The player holding the Three of Diamonds will begin the game by playing a hand of legal combination of cards that includes the Three of Diamonds.
- Players take turns to play by either playing a hand of legal combination of cards that beats the last hand of cards played on the table, or by passing his turn to the next player.
- A hand of legal combination of cards can only be beaten by another better hand of legal combination of cards with the same number of cards.

- A player cannot pass his turn to the next player if he is the one who played the last hand of cards on the table. In this case, he can play a hand of any legal combination of cards regardless of the last hand he played on the table.
- The game ends when any of the players has no more cards in his hand.

Legal combinations of Cards

- **Single**. This hand consists of only one single card. The only card in a single is referred to as the top card of this single. A single with a higher rank beats a single with a lower rank. For singles with the same rank, the one with a higher suit beats the one with a lower suit.
- **Pair**. This hand consists of two cards with the same rank. The card with a higher suit in a pair is referred to as the top card of this pair. A pair with a higher rank beats a pair with a lower rank. For pairs with the same rank, the one containing the highest suit beats the other.
- **Triple**. This hand consists of three cards with the same rank. The card with the highest suit in a triple is referred to as the top card of this triple. A triple with a higher rank beats a triple with a lower rank.
- **Straight**. This hand consists of five cards with consecutive ranks. For the sake of simplicity, 2 and A can only form a straight with K but not with 3. The card with the highest rank in a straight is referred to as the top card of this straight. A straight having a top card with a higher rank beats a straight having a top card with a lower rank. For straights having top cards with the same rank, the one having a top card with a higher suit beats the one having a top card with a lower suit.
- **Flush**. This hand consists of five cards with the same suit. The card with the highest rank in a flush is referred to as the top card of this flush. A flush always beats any straights. A flush with a higher suit beats a flush with a lower suit. For flushes with the same suit, the one having a top card with a higher rank beats the one having a top card with a lower rank.
- **Full House**. This hand consists of five cards, with two having the same rank and three having another same rank. The card in the triplet with the highest suit in a full house is referred to as the top card of this full house. A full house always beats any straights and flushes. A full house having a top card with a higher rank beats a full house having a top card with a lower rank.
- **Quad**. This hand consists of five cards, with four having the same rank. The card in the quadruplet with the highest suit in a quad is referred to as the top card of this quad. A quad always beats any straights, flushes and full houses. A quad having a top card with a higher rank beats a quad having a top card with a lower rank.
- **Straight Flush**. This hand consists of five cards with consecutive ranks and the same suit. For the sake of simplicity, 2 and A can only form a straight flush with K but not with 3. The card with the highest rank in a straight flush is referred to as the top card of this straight flush. A straight flush always beats any straights, flushes, full houses and quads. A straight flush having a top card with a higher rank beats a straight flush having a top card with a lower rank. For straight flushes having top cards with the same rank, the one having a top card with a higher suit beats one having a top card with a lower suit.

The BigTwo class

The BigTwo class is used to model a Big Two card game. It has private instance variables for storing a deck of cards, a list of players, a list of hands played on the table, an index of the current player, and a console for providing the user interface. Below is a detailed description for the BigTwo class.

Specification of the BigTwo class:

public constructor:

`BigTwo()` – a constructor for creating a Big Two card game. You should create 4 players and add them to the player list. You should also create a 'console' (i.e., a BigTwoConsole object) for providing the user interface.

private instance variables:

`Deck deck` – a deck of cards.

`ArrayList<CardGamePLayer> playerList` – a list of players.

`ArrayList<Hand> handsOnTable` – a list of hands played on the table.

`int currentIdx` – an integer specifying the index of the current player.

`BigTwoConsole bigTwoConsole` – a BigTwoConsole object for providing the user interface.

public methods:

`Deck getDeck()` – a method for retrieving the deck of cards being used.

`ArrayList<CardGamePlayer> getPlayerList()` – a method for retrieving the list of players.

`ArrayList<Hand> getHandsOnTable()` – a method for retrieving the list of hands played on the table.

`int getCurrentIdx()` – a method for retrieving the index of the current player.

`void start(BigTwoDeck deck)` – a method for starting the game with a (shuffled) deck of cards supplied as the argument. It implements the Big Two game logics.

public static methods:

`void main(String[] args)` – a method for starting a Big Two card game. It should create a Big Two card game, create and shuffle a deck of cards, and start the game with the deck of cards.

`Hand composeHand(CardGamePlayer player, CardList cards)` – a method for returning a valid hand from the specified list of cards of the player. Returns `null` is no valid hand can be composed from the specified list of cards.

The BigTwoCard class

The BigTwoCard class is a subclass of the Card class, and is used to model a card used in a Big Two card game. It should override the `compareTo()` method it inherited from the Card class to reflect the ordering of cards used in a Big Two card game. Below is a detailed description for the BigTwoCard class.

Specification of the BigTwoCard class:

public constructor:

`BigTwoCard(int suit, int rank)` – a constructor for building a card with the specified suit and rank. `suit` is an integer between 0 and 3, and `rank` is an integer between 0 and 12.

overriding method:

`int compareTo(Card card)` – a method for comparing this card with the specified card for order. Returns a negative integer, zero, or a positive integer as this card is less than, equal to, or greater than the specified card.

The BigTwoDeck class

The BigTwoDeck class is a subclass of the Deck class, and is used to model a deck of cards used in a Big Two card game. It should override the `initialize()` method it inherited from the Deck class to create a deck of Big Two cards. Below is a detailed description for the BigTwoDeck class.

Specification of the BigTwoDeck class:

overriding method:

`void initialize()` – a method for initializing a deck of Big Two cards. It should remove all cards from the deck, create 52 Big Two cards and add them to the deck.

The Hand class

The Hand class is a subclass of the CardList class, and is used to model a hand of cards. It has a private instance variable for storing the player who plays this hand. It also has methods for getting the player of this hand, checking if it is a valid hand, getting the type of this hand, getting the top card of this hand, and checking if it beats a specified hand. Below is a detailed description for the Hand class.

Specification of the Hand class:

public constructor:

`Hand(CardGamePlayer player, CardList cards)` – a constructor for building a hand with the specified player and list of cards.

private instance variable:

`CardGamePlayer player` – the player who plays this hand.

public methods:

`CardGamePlayer getPlayer()` – a method for retrieving the player of this hand.

`Card getTopCard()` – a method for retrieving the top card of this hand.

`boolean beats(Hand hand)` – a method for checking if this hand beats a specified hand.

abstract methods:

`boolean isValid()` – a method for checking if this is a valid hand.

`String getType()` – a method for returning a string specifying the type of this hand.

The Single, Pair, Triple, Straight, Flush, FullHouse, Quad, StraightFlush classes

These classes are a subclass of the Hand class, and are used to model a hand of single, pair, triple, straight, flush, full house, quad and straight flush in a Big Two card game, respectively. They should override methods of the Hand class as appropriate. In particular, the `getType()` method should return the name of the class as a String object in these classes modelling legal hands in a Big Two card game. For examples, calling the `getType()` method on a Triple object should return `"Triple"`, while calling the `getType()` method on a FullHouse object should return `"FullHouse"`.

Sample output

Showing cards on the table

At each player's turn, your program should print out the cards held by each player as well as the last hand played on the table (see figure 1). You can achieve this by calling the `repaint()` method of the BigTwoConsole object.

```
<Player 0>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ] 9 [  ] 10 [  ] 11 [  ] 12 [  ]
<Player 1>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ] 9 [  ] 10 [  ] 11 [  ] 12 [  ]
<Player 2>
==> 0 [♦3] 1 [♠3] 2 [♦5] 3 [♦6] 4 [♣7] 5 [♥7] 6 [♠8] 7 [♣0] 8 [♥0] 9 [♦Q] 10 [♠Q] 11 [♥A] 12 [♦2]
<Player 3>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ] 9 [  ] 10 [  ] 11 [  ] 12 [  ]
<Table>
  [Empty]
```

Figure 1. Showing cards on the table.

Getting user input

At each player's turn, your program should read from the keyboard a space-separated list of indices that represent the list of cards played by the player. You can achieve this by calling the `getSelected()` method of the BigTwoConsole object.

Showing the hand played by a player

After the current player has selected a list of cards to play, your program should produce the followings to the console (see figure 2):

- Print `"Not a legal move!!!"` to the console if the cards selected do not compose a valid hand or the move is not legal, and prompt the player to select again.
- Print to the console the type of the hand followed by the cards in the hand if the cards selected do compose a valid hand and is a legal move.
- Print `"{pass}"` to the console if the player enters an empty list and {pass} is a legal move.

```
<Player 0>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ] 9 [  ] 10 [  ] 11 [  ] 12 [  ]
<Player 1>
==> 0 [♦3] 1 [♣4] 2 [♦5] 3 [♣5] 4 [♠5] 5 [♦6] 6 [♠8] 7 [♥9] 8 [♠J] 9 [♣K] 10 [♣A] 11 [♥A] 12 [♣2]
<Player 2>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ] 9 [  ] 10 [  ] 11 [  ] 12 [  ]
<Player 3>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ] 9 [  ] 10 [  ] 11 [  ] 12 [  ]
<Table>
  [Empty]
Player 1's turn: 0 1 2
Not a legal move!!!
Player 1's turn: 2 3 4 10 11
Not a legal move!!!
Player 1's turn: 0
{Single} [♦3]
```

Figure 2. Sample output after the current player has selected a list of cards to play.

The game ends when any of the players has no more cards in his hand. Your program should then print out the number of cards held by each player (see figure 3).

```
Game ends
Player 0 has 2 cards in hand.
Player 1 has 8 cards in hand.
Player 2 wins the game.
Player 3 has 5 cards in hand.
```

Figure 3. Sample output when the game ends.

You may refer to the Appendix for an example game play.

Marking Scheme

Marks are distributed as follows:

- Design and implementation of the BigTwo class (25%)
- Design and implementation of the BigTwoCard and BigTwoDeck classes (5% each)
- Design and implementation of the Hand class and its subclasses (5% each)
- Javadoc and comments (20%)

Submission

Please pack the source code (`*.java`) of your application into a single zip file, and submit it to the course Moodle page.

A few points to note:

- Always remember to write Javadoc for all non-private classes and their non-private class members.
- Always remember to submit the source code files (`*.java`) but **NOT** the bytecode files (`*.class`).
- Always double check after your submission to ensure that you have submitted the most up-to-date source code files.
- Your assignment will not be marked if you have only submitted the bytecode files (`*.class`). You will get zero mark for the assignment.
- Please submit your assignment on time. Late submission will not be accepted.

~ End ~

Appendix

Example game play

```
<Player 0>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ] 9 [  ] 10 [  ] 11 [  ] 12 [  ]
<Player 1>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ] 9 [  ] 10 [  ] 11 [  ] 12 [  ]
<Player 2>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ] 9 [  ] 10 [  ] 11 [  ] 12 [  ]
<Player 3>
==> 0 [♦3] 1 [♣4] 2 [♦6] 3 [♦7] 4 [♣7] 5 [♣8] 6 [♠8] 7 [♣9] 8 [♠9] 9 [♥Q] 10 [♥A] 11 [♣2] 12 [♥2]
<Table>
  [Empty]
Player 3's turn: 0
{Single} [♦3]

<Player 0>
==> 0 [♥4] 1 [♦5] 2 [♣5] 3 [♥5] 4 [♠5] 5 [♠6] 6 [♦8] 7 [♠0] 8 [♣J] 9 [♦Q] 10 [♠Q] 11 [♦A] 12 [♦2]
<Player 1>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ] 9 [  ] 10 [  ] 11 [  ] 12 [  ]
<Player 2>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ] 9 [  ] 10 [  ] 11 [  ] 12 [  ]
<Player 3>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ] 9 [  ] 10 [  ] 11 [  ]
<Table>
    <Player 3> {Single} [♦3]
Player 0's turn: 0
{Single} [♥4]

<Player 0>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ] 9 [  ] 10 [  ] 11 [  ]
<Player 1>
==> 0 [♣3] 1 [♦4] 2 [♠4] 3 [♠7] 4 [♥8] 5 [♦9] 6 [♥9] 7 [♣0] 8 [♥0] 9 [♦J] 10 [♣Q] 11 [♣K] 12 [♠A]
<Player 2>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ] 9 [  ] 10 [  ] 11 [  ] 12 [  ]
<Player 3>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ] 9 [  ] 10 [  ] 11 [  ]
<Table>
    <Player 0> {Single} [♥4]
Player 1's turn: 3
{Single} [♠7]

<Player 0>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ] 9 [  ] 10 [  ] 11 [  ]
<Player 1>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ] 9 [  ] 10 [  ] 11 [  ]
<Player 2>
==> 0 [♥3] 1 [♠3] 2 [♣6] 3 [♥6] 4 [♥7] 5 [♦0] 6 [♥J] 7 [♠J] 8 [♦K] 9 [♥K] 10 [♠K] 11 [♣A] 12 [♠2]
<Player 3>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ] 9 [  ] 10 [  ] 11 [  ]
<Table>
    <Player 1> {Single} [♠7]
Player 2's turn: 0 1
Not a legal move!!!
Player 2's turn: 5
{Single} [♦0]

<Player 0>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ] 9 [  ] 10 [  ] 11 [  ]
<Player 1>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ] 9 [  ] 10 [  ] 11 [  ]
<Player 2>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ] 9 [  ] 10 [  ] 11 [  ]
<Player 3>
==> 0 [♣4] 1 [♦6] 2 [♦7] 3 [♣7] 4 [♣8] 5 [♠8] 6 [♣9] 7 [♠9] 8 [♥Q] 9 [♥A] 10 [♣2] 11 [♥2]
<Table>
```

```
    <Player 2> {Single} [♦0]
Player 3's turn: 8
{Single} [♥Q]

<Player 0>
==> 0 [♦5] 1 [♣5] 2 [♥5] 3 [♠5] 4 [♠6] 5 [♦8] 6 [♠0] 7 [♣J] 8 [♦Q] 9 [♠Q] 10 [♦A] 11 [♦2]
<Player 1>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ] 9 [  ] 10 [  ] 11 [  ]
<Player 2>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ] 9 [  ] 10 [  ] 11 [  ]
<Player 3>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ] 9 [  ] 10 [  ]
<Table>
    <Player 3> {Single} [♥Q]
Player 0's turn: 6
Not a legal move!!!
Player 0's turn: 10
{Single} [♦A]

<Player 0>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ] 9 [  ] 10 [  ]
<Player 1>
==> 0 [♣3] 1 [♦4] 2 [♠4] 3 [♥8] 4 [♦9] 5 [♥9] 6 [♣0] 7 [♥0] 8 [♦J] 9 [♣Q] 10 [♣K] 11 [♠A]
<Player 2>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ] 9 [  ] 10 [  ] 11 [  ]
<Player 3>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ] 9 [  ] 10 [  ]
<Table>
    <Player 0> {Single} [♦A]
Player 1's turn: 11
{Single} [♠A]

<Player 0>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ] 9 [  ] 10 [  ]
<Player 1>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ] 9 [  ] 10 [  ]
<Player 2>
==> 0 [♥3] 1 [♠3] 2 [♣6] 3 [♥6] 4 [♥7] 5 [♥J] 6 [♠J] 7 [♦K] 8 [♥K] 9 [♠K] 10 [♣A] 11 [♠2]
<Player 3>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ] 9 [  ] 10 [  ]
<Table>
    <Player 1> {Single} [♠A]
Player 2's turn: 11
{Single} [♠2]

<Player 0>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ] 9 [  ] 10 [  ]
<Player 1>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ] 9 [  ] 10 [  ]
<Player 2>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ] 9 [  ] 10 [  ]
<Player 3>
==> 0 [♣4] 1 [♦6] 2 [♦7] 3 [♣7] 4 [♣8] 5 [♠8] 6 [♣9] 7 [♠9] 8 [♥A] 9 [♣2] 10 [♥2]
<Table>
    <Player 2> {Single} [♠2]
Player 3's turn:
{Pass}

<Player 0>
==> 0 [♦5] 1 [♣5] 2 [♥5] 3 [♠5] 4 [♠6] 5 [♦8] 6 [♠0] 7 [♣J] 8 [♦Q] 9 [♠Q] 10 [♦2]
<Player 1>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ] 9 [  ] 10 [  ]
<Player 2>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ] 9 [  ] 10 [  ]
<Player 3>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ] 9 [  ] 10 [  ]
<Table>
    <Player 2> {Single} [♠2]
Player 0's turn:
```

```
{Pass}

<Player 0>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ] 9 [  ] 10 [  ]
<Player 1>
==> 0 [♣3] 1 [♦4] 2 [♠4] 3 [♥8] 4 [♦9] 5 [♥9] 6 [♣0] 7 [♥0] 8 [♦J] 9 [♣Q] 10 [♣K]
<Player 2>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ] 9 [  ] 10 [  ]
<Player 3>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ] 9 [  ] 10 [  ]
<Table>
    <Player 2> {Single} [♠2]
Player 1's turn:
{Pass}

<Player 0>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ] 9 [  ] 10 [  ]
<Player 1>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ] 9 [  ] 10 [  ]
<Player 2>
==> 0 [♥3] 1 [♠3] 2 [♣6] 3 [♥6] 4 [♥7] 5 [♥J] 6 [♠J] 7 [♦K] 8 [♥K] 9 [♠K] 10 [♣A]
<Player 3>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ] 9 [  ] 10 [  ]
<Table>
    <Player 2> {Single} [♠2]
Player 2's turn: 0 1
{Pair} [♥3] [♠3]

<Player 0>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ] 9 [  ] 10 [  ]
<Player 1>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ] 9 [  ] 10 [  ]
<Player 2>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ]
<Player 3>
==> 0 [♣4] 1 [♦6] 2 [♦7] 3 [♣7] 4 [♣8] 5 [♠8] 6 [♣9] 7 [♠9] 8 [♥A] 9 [♣2] 10 [♥2]
<Table>
    <Player 2> {Pair} [♥3] [♠3]
Player 3's turn: 1 2
Not a legal move!!!
Player 3's turn: 2 3
{Pair} [♦7] [♣7]

<Player 0>
==> 0 [♦5] 1 [♣5] 2 [♥5] 3 [♠5] 4 [♠6] 5 [♦8] 6 [♠0] 7 [♣J] 8 [♦Q] 9 [♠Q] 10 [♦2]
<Player 1>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ] 9 [  ] 10 [  ]
<Player 2>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ]
<Player 3>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ]
<Table>
    <Player 3> {Pair} [♦7] [♣7]
Player 0's turn: 8 9
{Pair} [♦Q] [♠Q]

<Player 0>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ]
<Player 1>
==> 0 [♣3] 1 [♦4] 2 [♠4] 3 [♥8] 4 [♦9] 5 [♥9] 6 [♣0] 7 [♥0] 8 [♦J] 9 [♣Q] 10 [♣K]
<Player 2>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ]
<Player 3>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ]
<Table>
    <Player 0> {Pair} [♦Q] [♠Q]
Player 1's turn:
{Pass}
```

```
<Player 0>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ]
<Player 1>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ] 9 [  ] 10 [  ]
<Player 2>
==> 0 [♣6] 1 [♥6] 2 [♥7] 3 [♥J] 4 [♠J] 5 [♦K] 6 [♥K] 7 [♠K] 8 [♣A]
<Player 3>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ]
<Table>
    <Player 0> {Pair} [♦Q] [♠Q]
Player 2's turn:
{Pass}

<Player 0>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ]
<Player 1>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ] 9 [  ] 10 [  ]
<Player 2>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ]
<Player 3>
==> 0 [♣4] 1 [♦6] 2 [♣8] 3 [♠8] 4 [♣9] 5 [♠9] 6 [♥A] 7 [♣2] 8 [♥2]
<Table>
    <Player 0> {Pair} [♦Q] [♠Q]
Player 3's turn:
{Pass}

<Player 0>
==> 0 [♦5] 1 [♣5] 2 [♥5] 3 [♠5] 4 [♠6] 5 [♦8] 6 [♠0] 7 [♣J] 8 [♦2]
<Player 1>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ] 9 [  ] 10 [  ]
<Player 2>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ]
<Player 3>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ]
<Table>
    <Player 0> {Pair} [♦Q] [♠Q]
Player 0's turn: 0 1 2 3 4
{Quad} [♦5] [♣5] [♥5] [♠5] [♠6]

<Player 0>
    0 [  ] 1 [  ] 2 [  ] 3 [  ]
<Player 1>
==> 0 [♣3] 1 [♦4] 2 [♠4] 3 [♥8] 4 [♦9] 5 [♥9] 6 [♣0] 7 [♥0] 8 [♦J] 9 [♣Q] 10 [♣K]
<Player 2>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ]
<Player 3>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ]
<Table>
    <Player 0> {Quad} [♦5] [♣5] [♥5] [♠5] [♠6]
Player 1's turn:
{Pass}

<Player 0>
    0 [  ] 1 [  ] 2 [  ] 3 [  ]
<Player 1>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ] 9 [  ] 10 [  ]
<Player 2>
==> 0 [♣6] 1 [♥6] 2 [♥7] 3 [♥J] 4 [♠J] 5 [♦K] 6 [♥K] 7 [♠K] 8 [♣A]
<Player 3>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ]
<Table>
    <Player 0> {Quad} [♦5] [♣5] [♥5] [♠5] [♠6]
Player 2's turn: 5 6 7 0 1
Not a legal move!!!
Player 2's turn:
{Pass}

<Player 0>
    0 [  ] 1 [  ] 2 [  ] 3 [  ]
```

```
<Player 1>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ] 9 [  ] 10 [  ]
<Player 2>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ]
<Player 3>
==> 0 [♣4] 1 [♦6] 2 [♣8] 3 [♠8] 4 [♣9] 5 [♠9] 6 [♥A] 7 [♣2] 8 [♥2]
<Table>
    <Player 0> {Quad} [♦5] [♣5] [♥5] [♠5] [♠6]
Player 3's turn:
{Pass}

<Player 0>
==> 0 [♦8] 1 [♠0] 2 [♣J] 3 [♦2]
<Player 1>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ] 9 [  ] 10 [  ]
<Player 2>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ]
<Player 3>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ]
<Table>
    <Player 0> {Quad} [♦5] [♣5] [♥5] [♠5] [♠6]
Player 0's turn: 0
{Single} [♦8]

<Player 0>
    0 [  ] 1 [  ] 2 [  ]
<Player 1>
==> 0 [♣3] 1 [♦4] 2 [♠4] 3 [♥8] 4 [♦9] 5 [♥9] 6 [♣0] 7 [♥0] 8 [♦J] 9 [♣Q] 10 [♣K]
<Player 2>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ]
<Player 3>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ]
<Table>
    <Player 0> {Single} [♦8]
Player 1's turn: 3
{Single} [♥8]

<Player 0>
    0 [  ] 1 [  ] 2 [  ]
<Player 1>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ] 9 [  ]
<Player 2>
==> 0 [♣6] 1 [♥6] 2 [♥7] 3 [♥J] 4 [♠J] 5 [♦K] 6 [♥K] 7 [♠K] 8 [♣A]
<Player 3>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ]
<Table>
    <Player 1> {Single} [♥8]
Player 2's turn: 8
{Single} [♣A]

<Player 0>
    0 [  ] 1 [  ] 2 [  ]
<Player 1>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ] 9 [  ]
<Player 2>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ]
<Player 3>
==> 0 [♣4] 1 [♦6] 2 [♣8] 3 [♠8] 4 [♣9] 5 [♠9] 6 [♥A] 7 [♣2] 8 [♥2]
<Table>
    <Player 2> {Single} [♣A]
Player 3's turn: 6
{Single} [♥A]

<Player 0>
==> 0 [♠0] 1 [♣J] 2 [♦2]
<Player 1>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ] 9 [  ]
<Player 2>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ]
```

```
<Player 3>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ]
<Table>
    <Player 3> {Single} [♥A]
Player 0's turn: 2
{Single} [♦2]

<Player 0>
    0 [  ] 1 [  ]
<Player 1>
==> 0 [♣3] 1 [♦4] 2 [♠4] 3 [♦9] 4 [♥9] 5 [♣0] 6 [♥0] 7 [♦J] 8 [♣Q] 9 [♣K]
<Player 2>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ]
<Player 3>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ]
<Table>
    <Player 0> {Single} [♦2]
Player 1's turn:
{Pass}

<Player 0>
    0 [  ] 1 [  ]
<Player 1>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ] 9 [  ]
<Player 2>
==> 0 [♣6] 1 [♥6] 2 [♥7] 3 [♥J] 4 [♠J] 5 [♦K] 6 [♥K] 7 [♠K]
<Player 3>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ]
<Table>
    <Player 0> {Single} [♦2]
Player 2's turn:
{Pass}

<Player 0>
    0 [  ] 1 [  ]
<Player 1>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ] 9 [  ]
<Player 2>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ]
<Player 3>
==> 0 [♣4] 1 [♦6] 2 [♣8] 3 [♠8] 4 [♣9] 5 [♠9] 6 [♣2] 7 [♥2]
<Table>
    <Player 0> {Single} [♦2]
Player 3's turn: 6
{Single} [♣2]

<Player 0>
==> 0 [♠0] 1 [♣J]
<Player 1>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ] 9 [  ]
<Player 2>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ]
<Player 3>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ]
<Table>
    <Player 3> {Single} [♣2]
Player 0's turn:
{Pass}

<Player 0>
    0 [  ] 1 [  ]
<Player 1>
==> 0 [♣3] 1 [♦4] 2 [♠4] 3 [♦9] 4 [♥9] 5 [♣0] 6 [♥0] 7 [♦J] 8 [♣Q] 9 [♣K]
<Player 2>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ]
<Player 3>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ]
<Table>
    <Player 3> {Single} [♣2]
```

```
Player 1's turn:
{Pass}

<Player 0>
    0 [  ] 1 [  ]
<Player 1>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ] 9 [  ]
<Player 2>
==> 0 [♣6] 1 [♥6] 2 [♥7] 3 [♥J] 4 [♠J] 5 [♦K] 6 [♥K] 7 [♠K]
<Player 3>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ]
<Table>
    <Player 3> {Single} [♣2]
Player 2's turn:
{Pass}

<Player 0>
    0 [  ] 1 [  ]
<Player 1>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ] 9 [  ]
<Player 2>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ]
<Player 3>
==> 0 [♣4] 1 [♦6] 2 [♣8] 3 [♠8] 4 [♣9] 5 [♠9] 6 [♥2]
<Table>
    <Player 3> {Single} [♣2]
Player 3's turn: 2 3
{Pair} [♣8] [♠8]

<Player 0>
==> 0 [♠0] 1 [♣J]
<Player 1>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ] 8 [  ] 9 [  ]
<Player 2>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ]
<Player 3>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ]
<Table>
    <Player 3> {Pair} [♣8] [♠8]
Player 0's turn:
{Pass}

<Player 0>
    0 [  ] 1 [  ]
<Player 1>
==> 0 [♣3] 1 [♦4] 2 [♠4] 3 [♦9] 4 [♥9] 5 [♣0] 6 [♥0] 7 [♦J] 8 [♣Q] 9 [♣K]
<Player 2>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ]
<Player 3>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ]
<Table>
    <Player 3> {Pair} [♣8] [♠8]
Player 1's turn: 3 4
{Pair} [♦9] [♥9]

<Player 0>
    0 [  ] 1 [  ]
<Player 1>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ]
<Player 2>
==> 0 [♣6] 1 [♥6] 2 [♥7] 3 [♥J] 4 [♠J] 5 [♦K] 6 [♥K] 7 [♠K]
<Player 3>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ]
<Table>
    <Player 1> {Pair} [♦9] [♥9]
Player 2's turn: 3 4
{Pair} [♥J] [♠J]

<Player 0>
```

```
    0 [  ] 1 [  ]
<Player 1>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ]
<Player 2>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ]
<Player 3>
==> 0 [♣4] 1 [♦6] 2 [♣9] 3 [♠9] 4 [♥2]
<Table>
    <Player 2> {Pair} [♥J] [♠J]
Player 3's turn:
{Pass}

<Player 0>
==> 0 [♠0] 1 [♣J]
<Player 1>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ]
<Player 2>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ]
<Player 3>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ]
<Table>
    <Player 2> {Pair} [♥J] [♠J]
Player 0's turn:
{Pass}

<Player 0>
    0 [  ] 1 [  ]
<Player 1>
==> 0 [♣3] 1 [♦4] 2 [♠4] 3 [♣0] 4 [♥0] 5 [♦J] 6 [♣Q] 7 [♣K]
<Player 2>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ]
<Player 3>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ]
<Table>
    <Player 2> {Pair} [♥J] [♠J]
Player 1's turn:
{Pass}

<Player 0>
    0 [  ] 1 [  ]
<Player 1>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ]
<Player 2>
==> 0 [♣6] 1 [♥6] 2 [♥7] 3 [♦K] 4 [♥K] 5 [♠K]
<Player 3>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ]
<Table>
    <Player 2> {Pair} [♥J] [♠J]
Player 2's turn:
Not a legal move!!!
Player 2's turn: 0 1 3 5 4
{FullHouse} [♣6] [♥6] [♦K] [♥K] [♠K]

<Player 0>
    0 [  ] 1 [  ]
<Player 1>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ]
<Player 2>
    0 [  ]
<Player 3>
==> 0 [♣4] 1 [♦6] 2 [♣9] 3 [♠9] 4 [♥2]
<Table>
    <Player 2> {FullHouse} [♣6] [♥6] [♦K] [♥K] [♠K]
Player 3's turn:
{Pass}

<Player 0>
==> 0 [♠0] 1 [♣J]
<Player 1>
```

```
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ]
<Player 2>
    0 [  ]
<Player 3>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ]
<Table>
    <Player 2> {FullHouse} [♣6] [♥6] [♦K] [♥K] [♠K]
Player 0's turn:
{Pass}

<Player 0>
    0 [  ] 1 [  ]
<Player 1>
==> 0 [♣3] 1 [♦4] 2 [♠4] 3 [♣0] 4 [♥0] 5 [♦J] 6 [♣Q] 7 [♣K]
<Player 2>
    0 [  ]
<Player 3>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ]
<Table>
    <Player 2> {FullHouse} [♣6] [♥6] [♦K] [♥K] [♠K]
Player 1's turn:
{Pass}

<Player 0>
    0 [  ] 1 [  ]
<Player 1>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ] 5 [  ] 6 [  ] 7 [  ]
<Player 2>
==> 0 [♥7]
<Player 3>
    0 [  ] 1 [  ] 2 [  ] 3 [  ] 4 [  ]
<Table>
    <Player 2> {FullHouse} [♣6] [♥6] [♦K] [♥K] [♠K]
Player 2's turn: 0
{Single} [♥7]

<Player 0>
    0 [♠0] 1 [♣J]
<Player 1>
    0 [♣3] 1 [♦4] 2 [♠4] 3 [♣0] 4 [♥0] 5 [♦J] 6 [♣Q] 7 [♣K]
<Player 2>
    [Empty]
<Player 3>
    0 [♣4] 1 [♦6] 2 [♣9] 3 [♠9] 4 [♥2]
<Table>
    <Player 2> {Single} [♥7]

Game ends
Player 0 has 2 cards in hand.
Player 1 has 8 cards in hand.
Player 2 wins the game.
Player 3 has 5 cards in hand.
```