```
public class Client {

    /**
     * Xavier Quinn
     *
     * I affirm that I have carried out the attached academic endeavors with full academic
honesty, in
     * accordance with the Union College Honor Code and the course syllabus.
     */
        public static void main(String[] args) {
                // TODO Auto-generated method stub
                CrazyEights game = new CrazyEights();
                game.playGame();

        }

}
```

```
import java.util.Scanner;
public class CrazyEights {
        private final int DEF_DECK_SIZE=104;
        private final int DEF_NUM_SUITES=4;
        private final int DEF_CARD_PER_SUITE=13;
        private final int DEF_INIT_HAND=7;


        private Hand[] player;
        private Scanner scan;
        private int numCards;
        private Deck deck;
        private int turn;
        private String[] playerName;
```

```
private boolean endGame;
private Card faceCard;
private String pseudoSuite;
private Card tmpCard;
private boolean cont;


public CrazyEights() {
	scan=new Scanner(System.in);
	deck=new Deck(DEF_DECK_SIZE, DEF_NUM_SUITES, DEF_CARD_PER_SUITE);
	numCards=DEF_DECK_SIZE;

	deck.shuffle();
}


public void playGame() {
	System.out.println("input number of players");
	int numPlayers = scan.nextInt();
	endGame=false;
	turn=0;
	pseudoSuite="Nothing, becuase it hasn't been assigned a value yet";
	cont=false;


	if(numPlayers<=1) {
		System.out.println("This is not enough players");
	}

	else if(numPlayers>(numCards/DEF_INIT_HAND)) {
		System.out.println("You do not have a large enough deck");
	}
	else {
		//The actual gameplay
		player = new Hand[numPlayers];
		playerName=new String[numPlayers];

		for(int i=0;i<numPlayers;i++) {
			System.out.println("Please input the name of the next player");
			playerName[i]=scan.next();

		}

		//Makes a hand for each player
		for(int i=0;i<numPlayers;i++) {
			player[i] = new Hand(numCards);
			for(int j=0; j<7;j++) {
				player[i].drawCard(deck.deal());
			}

		}

		//Places the starter card
		faceCard=deck.deal();

		while(endGame==false) {

			//Says who's turn it is
```

```
                    System.out.println("It is now " + playerName[turn] +"s turn\n");

                    //Shows the top card
                    System.out.println("The top card is:\n" + faceCard.toString());
                    if (faceCard.getValue()==8) {
                        System.out.println("But is acting as " + pseudoSuite);
                    }

                    //Shows their hand
                    System.out.println("Their hand:\n");
                    player[turn].showHand();
                    System.out.println("Input " + (int)(player[turn].handSize()+1) + " to draw
\n");



                    while (cont==false) {

                        //Gets their move
                        System.out.println(playerName[turn] + "s move?");

                        tmpCard=player[turn].removeCard(scan.nextInt());

                        if(tmpCard==null) {
                            player[turn].drawCard(deck.deal());
                            System.out.println("New card is " + player[turn].seekCard
(player[turn].handSize()-1).toString());
                            cont=true;
                        }
                        else {

                            if(tmpCard.getValue()==8) {
                                System.out.println("CRAZY EIGHT\nWhich Suite?\n0)
Clubs\n1) Diamonds\n2) Hearts\n3) Spades");
                                switch (scan.nextInt()) {

                                case 0: pseudoSuite = "Clubs";
                                break;
                                case 1: pseudoSuite = "Diamonds";
                                break;
                                case 2: pseudoSuite = "Hearts";
                                break;
                                case 3: pseudoSuite = "Spades";
                                break;
                                }

                                faceCard=tmpCard;
                                cont=true;
                            }
                            else if(tmpCard.getValue()==faceCard.getValue() ||
tmpCard.getSuite()==faceCard.getSuite() || tmpCard.getValue()==8) {
                                faceCard=tmpCard;
                                cont=true;
                            }
                            else {
                                System.out.println("This is not a valid move, please try
again\n");
                            }
```

```
                        }
                    }
                    cont=false;


                    //If the game is over
                    if(player[turn].handSize()==0) {
                        System.out.println("GAME OVER");
                        endGame=true;
                    }
                    else if(player[turn].handSize()==1) {
                        System.out.println(playerName[turn] + " Only has one card left!");
                    }

                    //loops
                    if(turn==numPlayers-1) {
                        turn=0;
                    }
                    else {
                        turn++;
                    }
                }
            }
        }
}
```

```
import java.util.Random;



public class Deck {
    final int DEFAULT_DECK_SIZE=52;
    final int DEFAULT_NUM_SUITES=4;
    final int DEFAULT_CARDS_PER_SUITE=13;

    Card[] deck;
```

```
Random Random;
private int dealer;
private int deckItterator;
private int deckSize;
private int numSuites;
private int cardsPerSuite;

/**
 * The constructor takes no inputs and fills itself with cards.
 * @param This constructor takes no params
 */
Deck() {
	deck = new Card[DEFAULT_DECK_SIZE];
	deckSize=DEFAULT_DECK_SIZE;
	numSuites=DEFAULT_NUM_SUITES;
	cardsPerSuite=DEFAULT_CARDS_PER_SUITE;

	dealer = 0;
	deckItterator=0;
	while(deckItterator<DEFAULT_DECK_SIZE) {
		for(int i=0;i<DEFAULT_NUM_SUITES;i++) {
			for(int j=0;j<DEFAULT_CARDS_PER_SUITE;j++) {
				deck[deckItterator] = new Card(i,j);
				deckItterator++;
			}
		}
	}
}
/**
 * Constructor for the deck
 * @param newDeckSize The size of the deck
 * @param newNumSuites The number of suites
 * @param newCardsPerSuite The number of cards per suite
 */

Deck(int newDeckSize, int newNumSuites, int newCardsPerSuite) {
	deckSize=newDeckSize;
	deck = new Card[deckSize];
	numSuites=newNumSuites;
	cardsPerSuite=newCardsPerSuite;


	dealer = 0;
	deckItterator=0;
	while(deckItterator<deckSize) {
		for(int i=0;i<(numSuites);i++) {
			for(int j=0;j<cardsPerSuite;j++) {
				deck[deckItterator] = new Card(i,j);
				deckItterator++;
			}
		}
	}
}



/**
 * This function evaluates the contents of the deck
```

```
	 * @return This returns the contents of the deck in a printable string
	 */
	public String tostring() {
		String alldeck = "The current deck:\n";
		for(int i=0;i<deckSize;i++){
			alldeck = alldeck + deck[i].toString() + "\n";
		}
		return alldeck;
	}

	/**
	 *
	 * @return This returns a new card
	 */
	public Card deal() {
		Card returnval=deck[dealer];
		dealer++;
		return returnval;

	}
	/**
	 * Gets the dealer value
	 * @return returns the Int dealer value
	 */
	public int getDealerVal() {

		return dealer;
	}


	/**
	 * This is not used, but I feel like it would be useful if I have to use this again
	 * @return The number of suites
	 */
	public int numSuites() {
		return numSuites;

	}

	/**
	 * Shuffles the deck
	 */
	public void shuffle() {
		dealer=0;
		Random = new Random();
		for(int i=0;i<deckSize;i++){
			int randomval = Random.nextInt(deckSize);
			Card firstarr = deck[i];
			Card secondarr = deck[randomval];
			deck[i] = secondarr;
			deck[randomval] = firstarr;
		}
	}


}
```

```
import java.util.Random;


public class Deck {
	final int DEFAULT_DECK_SIZE=52;
	final int DEFAULT_NUM_SUITES=4;
	final int DEFAULT_CARDS_PER_SUITE=13;

	Card[] deck;
	Random Random;
	private int dealer;
	private int deckItterator;
	private int deckSize;
	private int numSuites;
	private int cardsPerSuite;

	/**
	 * The constructor takes no inputs and fills itself with cards.
	 * @param This constructor takes no params
	 */
	Deck() {
		deck = new Card[DEFAULT_DECK_SIZE];
		deckSize=DEFAULT_DECK_SIZE;
		numSuites=DEFAULT_NUM_SUITES;
		cardsPerSuite=DEFAULT_CARDS_PER_SUITE;

		dealer = 0;
		deckItterator=0;
		while(deckItterator<DEFAULT_DECK_SIZE) {
			for(int i=0;i<DEFAULT_NUM_SUITES;i++) {
				for(int j=0;j<DEFAULT_CARDS_PER_SUITE;j++) {
					deck[deckItterator] = new Card(i,j);
					deckItterator++;
```

```
                }
            }
        }
}
/**
 * Constructor for the deck
 * @param newDeckSize The size of the deck
 * @param newNumSuites The number of suites
 * @param newCardsPerSuite The number of cards per suite
 */

Deck(int newDeckSize, int newNumSuites, int newCardsPerSuite) {
        deckSize=newDeckSize;
        deck = new Card[deckSize];
        numSuites=newNumSuites;
        cardsPerSuite=newCardsPerSuite;


        dealer = 0;
        deckItterator=0;
        while(deckItterator<deckSize) {
                for(int i=0;i<(numSuites);i++) {
                        for(int j=0;j<cardsPerSuite;j++) {
                                deck[deckItterator] = new Card(i,j);
                                deckItterator++;
                        }
                }
        }
}



/**
 * This function evaluates the contents of the deck
 * @return This returns the contents of the deck in a printable string
 */
public String tostring() {
        String alldeck = "The current deck:\n";
        for(int i=0;i<deckSize;i++){
                alldeck = alldeck + deck[i].toString() + "\n";
        }
        return alldeck;
}

/**
 *
 * @return This returns a new card
 */
public Card deal() {
        Card returnval=deck[dealer];
        dealer++;
        return returnval;

}
/**
 * Gets the dealer value
 * @return returns the Int dealer value
 */
```

```
	public int getDealerVal() {

		return dealer;
	}


	/**
	 * This is not used, but I feel like it would be useful if I have to use this again
	 * @return The number of suites
	 */
	public int numSuites() {
		return numSuites;

	}

	/**
	 * Shuffles the deck
	 */
	public void shuffle() {
		dealer=0;
		Random = new Random();
		for(int i=0;i<deckSize;i++){
			int randomval = Random.nextInt(deckSize);
			Card firstarr = deck[i];
			Card secondarr = deck[randomval];
			deck[i] = secondarr;
			deck[randomval] = firstarr;
		}
	}


}
```