```java
package club.westcs.BlackjackCeciliaB;

import java.util.ArrayList;
import java.util.Scanner;

public class StrippedGameLogic {
    //Attributes
            /*
             * String
             *      playerName
             * Objects
             *      Deck
             *      Chips
             *      Scanner
             *      ArrayLists (2)
             *          playerHand<StrippedCard>
             *          dealerHand<StrippedCard>
             */
    /**
     * A string of playerName
     * A deck
     * A currency
     * A Scanner
     * Two ArrayLists for each hand(playerHand and dealerHand)
     */
    private String playerName;
    private StrippedDeck deck;
    private StrippedCurrency currency;
    private Scanner scan;
    private ArrayList<StrippedCard> playerHand;
    private ArrayList<StrippedCard> dealerHand;


        //Constructor
            //Regular constructor stuff
```

```java
    /**
     * New Scanner
     * get the player's name
     * make a new deck
     * make a new currency
     * make a new ArrayList for each hand
     * Run the game
     */
    public StrippedGameLogic() {
        scan = new Scanner(System.in);
        setName();
        deck = new StrippedDeck();
        currency = new StrippedCurrency();
        playerHand = new ArrayList<>();
        dealerHand = new ArrayList<>();
        runGame();
    }

        //Methods

            /*
             * #1 setName()
             * #2 runGame()
             *      like the pet live method
             *      while the player has currency
             *          call the game logic methods (deal, printHands, bet, hitStay, dealerTurn, determineWinner, resetHands/deck
             * #3 dealStrippedCard(ArrayList<StrippedCard> hand)
             *      if the hand is empty add two StrippedCards
             *      otherwise add one StrippedCard
             * #4 printHand(AL<StrippedCard> hand, boolean firstStrippedCard)
             *      for every StrippedCard in the hand
             *          if the firstStrippedCard is true
             *              print the StrippedCard's toString()
             *          else
             *              print mystery StrippedCard
             *              set firstStrippedCard to true
             * #5 totalHand(AL<StrippedCard> hand)
             *      need an int for the total
```

```
75              *      need a boolean for the ace
76              *      for every StrippedCard in the hand
77              *          add the numValue to the total
78              *          if the StrippedCard is an ace
79              *              set the bool to true
80              *      if the bool is true and the total is less than 12
81              *          add 10 to the total
82              *      return the total
83              * #6 hitStay()
84              *      while the playerTotal is < 21
85              *          ask if the player wants to hit or stay
86              *          if they hit give the playerHand a StrippedCard
87              *          else break
88              * #7 dealerTurn()
89              *      while the dealertotal is less than 17
90              *          add a StrippedCard to the dealerhand
91              * #8 determineWinner()
92              *      -Decides who wins and calls currency methods
93              *      --check if the player and dealer have tied or both gone bust (over 21) (T)
94              *      --check if the player has gone bust (L)
95              *      --Check if the dealer has gone bust or if the playerhandTotal > dealerHandTotal (W)
96              *      --else (L)
97              * #9 resetHands()
98              *      - clear the playerhand and dealerhand
99              *      - reset the deck
100             * #10 create the currency class stuff
101             * #11 resetGame()
102             *      - like the resetPet method
103             *      -if the player is out of currency
104             *          -ask if they want to play again
105             *          -if yes reset the currency
106             *          -else end the game
107             */
108
109         /**
110          * Ask what is your name
111          * save and name the scanner scan
112          * @return the player's name
```

```
113      */
114
115      public String setName() {
116          System.out.println("What is your name?");
117          playerName = scan.nextLine();
118          return playerName;
119      }
120
121      /**
122       * give the player their money
123       * have them bet their money
124       * @while the player has money
125       * deal them a card
126       * print what they have in their hand and drew
127       * hit or stay
128       * dealer's turn
129       * print what the player drew
130       * print the hand
131       * print what the dealer has
132       * print what the dealer drew
133       * determine the winner
134       * reset the hand and deck
135       * reset the game and ask if they want to play again
136       * shuffle the deck
137       * reset the deck
138       */
139
140      public void runGame() {
141          this.currency.getPlayerCurrency();
142          this.currency.bet();
143          while(this.currency.playerCurrency > 0) {
144              dealStrippedCard(playerHand);
145              printHand(playerHand,true);
146              hitStay();
147              dealerTurn();
148              System.out.println(this.playerName + " has:");
149              printHand(playerHand,true);
150              System.out.println("The dealer has: ");
```

```
151             printHand(dealerHand,true);
152             determineWinner();
153             resetHand();
154             resetGame();
155             shuffling();
156         }
157             this.deck.resetDeck();
158 
159         }
160 
161     /**
162      * Mr young's code for shuffling
163      * all i know is that is does it correctly and has a try catch
164      */
165 
166     private void shuffling() {
167         String drama = "Shuffling";
168         for(int i = 0; i < 4; i++) {
169             drama += ".";
170             System.out.println(drama);
171             try {
172                 Thread.sleep(1000);
173             } catch (InterruptedException e) {
174                 e.printStackTrace();
175             }
176         }
177 
178     }
179 
180     /**
181      * @if the hand is empty
182      * add two cards
183      * @param hand
184      * @else
185      * give one card
186      */
187 
188     public void dealStrippedCard(ArrayList<StrippedCard> hand) {
```

```
189         if(hand.isEmpty()) {
190             hand.add(this.deck.deal());
191             hand.add(this.deck.deal());
192         }
193         else {
194             hand.add(this.deck.deal());
195 
196         }
197     }
198 
199     /**
200      * prints the hand
201      * @param hand
202      * @param firstStrippedCard
203      * if the dealer has drawn their first card it is labeled as the first card
204      */
205 
206     public void printHand(ArrayList<StrippedCard> hand, boolean firstStrippedCard) {
207 
208         for(StrippedCard c: hand) {
209             if(firstStrippedCard) {
210                 System.out.println(c.toString());
211             }
212             else{
213                 System.out.println("Hidden StrippedCard");
214                 firstStrippedCard = true;
215             }
216         }
217 
218     }
219 
220     /**
221      * the ace is by defult false
222      * if the player has an ace and their total is less that 10 then the ace 10
223      * adds the card value to the total
224      * @param hand
225      * @return the total
226      */
```

```
227
228     public int totalHand(ArrayList<StrippedCard> hand) {
229         int total = 0;
230         boolean ace = false;
231         for(StrippedCard c: hand) {
232
233                 total += c.getNumValue();
234             if(c.getNumValue() == 1)
235                 ace = true;
236             }
237         if (ace && total< 12){
238
239             total += 10;
240         }
241         return total;
242     }
243
244     /**
245      * @while the player has less then 21
246      * ask if they want to hit or stay
247      * save the answer
248      * @if the answer says hit
249      * deals 1 card
250      * prints what they drew
251      * @else it breaks the method as they have a bust
252      */
253
254     public void hitStay() {
255         while(totalHand(playerHand) < 21) {
256             System.out.println("Do you want to hit or to stay?");
257             String answer =scan.nextLine().toLowerCase();
258             if(answer.equals("hit")) {
259                 dealStrippedCard(playerHand);
260                 System.out.println("You drew: ");
261                 System.out.println("      " + playerHand.get(playerHand.size()-1).toString());
262             }
263             else {
264                 break;
```

```
265             }
266         }
267     }
268 
269     /**
270      * @while the dealer has less then 17 as its total
271      * it adds 1 card
272      */
273 
274     public void dealerTurn() {
275         while(totalHand(dealerHand) < 17) {
276             dealerHand.add(this.deck.deal());
277         }
278     }
279 
280     /**
281      * @if the dealer and the player both have 21
282      * they tie
283      * resets the game so there can be tie breaker
284      * @else if the player has gone over 21
285      * player automatically loses
286      * @else if the dealer has gone over 21 or the player has more points then the dealer
287      * then the player wins
288      * @else the player loses
289      */
290 
291     public void determineWinner() {
292         if(totalHand(playerHand) > 21 && totalHand(dealerHand) > 21) {
293             System.out.println("You and the dealer have a bust and have tied");
294             System.out.println("Starting another game as a tie breaker.");
295             this.deck.resetDeck();
296             resetHand();
297             resetGame();
298         }
299         else if(totalHand(playerHand) > 21) {
300             System.out.println("You have a bust and therefore has lost.");
301             this.currency.lose();
302         }
```

```
303         else if(totalHand(dealerHand) > 21 || totalHand(playerHand) > totalHand(dealerHand)) {
304             System.out.println("You have won!");
305             this.currency.win();
306         }
307         else {
308             System.out.println("You have lost.");
309             this.currency.lose();
310         }
311     }
312 
313     /**
314      * clears the player hand
315      * clears the dealer hand
316      * resets the deck
317      */
318 
319     public void resetHand() {
320         playerHand.clear();
321         dealerHand.clear();
322         this.deck.resetDeck();
323     }
324 
325     /**
326      * asks if you want to play again
327      * saves the answer
328      * @if the answer is yes
329      * resets the hand
330      * resets the currency
331      * run the game again
332      * @else
333      * thanks them for playing
334      * terminates
335      */
336 
337     public void resetGame() {
338         System.out.println("Do you want to play again?");
339         String answer = scan.nextLine().toLowerCase();
340         if(answer.equals("yes")) {
341             resetHand();
342             this.currency.resetCurrency();
343             runGame();
344         }
345         else {
346             System.out.println("Thanks for playing!");
347             System.exit(0);
348         }
349     }
350 
351 }
352 
```