```
/*
    - Layout on screen OK
    - Incomplete design, final task not handled -0.8
    - Deep copy  -0.1
    - Overall design OK
    - Marks allocated subject to OK in demo

    5.1 / 6
*/

// Full Name - Hay Munn Hnin Wai
// Tutorial - T04
// Hi Sir, This is my own work & Kindly check the below code with some comments

import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;

enum SuitEnum
{
    Spade('S'),
    Heart('H'),
    Diamond('D'),
    Club('C');

private final char suit;

      // Constructor
    SuitEnum (char suit)
    {
        this.suit = suit;
    }
      //Accessor method
    public char getsuit ()
    {
        return this.suit;
    }
}
enum RankEnum
{
    Two('2'),
    Three('3'),
    Four('4'),
    Five('5'),
    Six('6'),
    Seven('7'),
    Eight('8'),
    Nine('9'),
    Ten ('T'),
    Jack('J'),
    Queen('Q'),
    King('K'),
    Ace('A');

    private final char rank;

      //Constructor
    RankEnum (char rank)
    {
        this.rank = rank;
    }

    //Accessor method
    public char getrank ()
    {
        return this.rank;
```

```
67          }
68      }
69      class PlayingCard
70      {
71          private SuitEnum suit;
72          private RankEnum rank;
73
74          public PlayingCard()
75          {
76
77          }
78          public PlayingCard ( SuitEnum suit, RankEnum rank)
79          {
80              setCard(suit, rank);
81
82          }
83          public PlayingCard (PlayingCard pc)
84          {
85              this(pc.suit, pc.rank);
86
87          }
88          public SuitEnum getSuit()
89          {
90              return this.suit;
91
92          }
93          public RankEnum getRank()
94          {
95              return this.rank;
96
97          }
98          public void setCard ( SuitEnum suit, RankEnum rank)
99          {
100             this.suit = suit;
101             this.rank = rank;
102
103         }
104         public String toString()
105         {
106             return String.format( "%c%c ", suit.getsuit(), rank.getrank());
107
108         }
109
110     }
111     class MyInt
112     {
113         private int n;
114
115         MyInt()
116         {
117
118         }
119
120         public void setInt(int n)
121         {
122             this.n = n;
123         }
124
125         public int getInt()
126         {
127             return n;
128         }
129     }
130     class HayMunnHninWai_A1
131      {
132         private final int MAXC = 13 ;
```

```
133       private final int MAXD = 52;
134
135       public static String [] cardArray;
136
137       // Create an Object to access PlayingCard Method
138       private static PlayingCard c = new PlayingCard();
139
140        // ArrayList of Playing Card
141       private void deckOfCards (ArrayList <PlayingCard> cardList)
142       {
143           for (SuitEnum suit : SuitEnum.values())
144           {
145               for(RankEnum rank : RankEnum.values())
146               cardList.add (new PlayingCard(suit,rank));
147           }
148       }
149        // Print Deck from ArrayList
150       private void printDeck (ArrayList <PlayingCard> values)
151       {
152           System.out.printf("Printing from ArrayList\n\n");
153           int a = 0;
154
155           for (int j = 0; j < (MAXD/MAXC) ; j++)
156           {
157               for(int k = 0; k < MAXC ; k++)
158               {
159                   System.out.printf("%s ",values.get(a));
160                   a++;
161               }
162               System.out.println();
163           }
164           System.out.println("--------------------------" +
165                              "--------------------------");
166
167       }
168        // Copy the ArrayList to cardArray
169       private void listToArray (ArrayList <PlayingCard> cardList, PlayingCard[]
    cardArray)
170       {
171           int x = 0;
172
173           for(PlayingCard c : cardList)
174           {
175               cardArray[x] = c;       /* deep copy */
176               x++;
177           }
178       }
179       // Print all the 52 playing cards from cardArray
180       private void printDeck(PlayingCard[] cardArray)
181       {
182           System.out.printf("Printing from Array\n\n");
183
184           int b = 0;
185
186               // Print the rows
187           for (int i = 0; i<(MAXD/MAXC); i++)
188           {
189               // Print the columns
190               for(int j = 0; j < MAXC; j++)
191               {
192                   System.out.printf("%s ", cardArray[b]);
193                   b++;
194               }
195               System.out.println();
196           }
197           System.out.println("--------------------------" +
```

```
198                             "--------------------------");
199
200         }
201         //transfer PlayingCard array to String array
202         private void transfer (PlayingCard[] cardArray, String[] strArray)
203         {
204             int i = 0;
205             for (PlayingCard c : cardArray)
206             {
207                 strArray[i] = c.toString();
208                 i++;
209             }
210         }
211
212         //display the string array
213         private void displayStringArray (String[] strArray)
214         {
215             System.out.printf("Printing from string array%n%n");
216             int c = 0;
217             for (int i = 0; i < (MAXD/MAXC); i++)
218             {
219                 for (int j = 0; j < MAXC; j++)
220                 {
221                     System.out.printf("%s ",strArray [c]);
222                     c++;
223                 }
224                 System.out.println();
225             }
226             System.out.println("--------------------------" +
227                             "--------------------------");
228         }
229
230         //Shuffle The Deck 50 times
231         private void shuffle (PlayingCard[] cardArray)
232         {
233             int x = 65;
234             int i,j;
235             for (int a = 0; a < x; a++)
236             {
237                 i = (int)(Math.random()*MAXD);
238                 j = (int)(Math.random()*MAXD);
239                 PlayingCard temp1 = cardArray[i];
240                 PlayingCard temp2 = cardArray[j];
241                 cardArray[i] = temp2;
242                 cardArray[j] = temp1;
243             }
244         }
245
246         // Swap the positions of the playing cards
247         private void swap(String[] str, int i, int j)
248         {
249             String temp = str[i];
250             str[i] = str[j];
251             str[j] = temp;
252         }
253
254         public static void main ( String [] args )
255         {
256             HayMunnHninWai_A1 a1 = new HayMunnHninWai_A1 ();
257             // Task-3a
258             ArrayList<PlayingCard> cardList = new ArrayList<PlayingCard> ();
259             a1.deckOfCards (cardList );
260             a1.printDeck(cardList);
261
262             // Task-3b
263             PlayingCard[]cardArray= new PlayingCard[ cardList.size()];
```

```
264             a1.listToArray(cardList, cardArray);
265             a1.printDeck(cardArray);
266 
267             // Task- 3c
268             String[] cardString = new String [cardList.size()];
269             a1.transfer(cardArray, cardString);
270             a1.displayStringArray(cardString);
271             a1.shuffle(cardArray);
272 
273             System.out.println("Shuffle the cards - Array version");
274             a1.printDeck(cardArray);
275 
276 
277         }
278 
279     }
280 
281 
282 
283 
284 
285 
286 
287 
```