

10/26/15 01:43:46 /home/15504319/DSA120/DSAAssignment/connorLib/Yard.java

```

1  /*****
2  *   FILE: Yard.java
3  *   AUTHOR: Connor Beardsmore - 15504319
4  *   UNIT: DSA120 Assignment S2- 2015
5  *   PURPOSE: Yard StockRoom for use in the DC ( GENERAL ARRAY )
6  *   LAST MOD: 19/10/15
7  *   REQUIRES: NONE
8  *****/
9  package connorLib;
10
11  public class Yard implements IStockRoom
12  {
13      //CLASS FIELDS
14      private Carton[] array;
15      private int count;
16
17      //CLASS CONSTANTS
18      private static final int MAX_VALID_CAP = 10000;
19      private static final int MIN_VALID_CAP = 1;
20      private static final String YARD = "Y";
21  //-----
22      //ALTERNATE Constructor
23      //IMPORT: maxCap (int)
24      //ASSERTION: Array allocated 'maxCap' elements.Count to default 0
25
26      public Yard(int maxCap)
27      {
28          //maxCap must be a value between 2 and 10,000
29          if ( (maxCap < MIN_VALID_CAP) || ( maxCap > MAX_VALID_CAP ) )
30          {
31              throw new IllegalArgumentException("Array Capacity Not Valid");
32          }
33          array = new Carton[maxCap];
34          count = 0;
35      }
36  //-----
37      //ACCESSOR getCount
38      //EXPORT: count (int)
39
40      public int getCount()
41      {
42          return count;
43      }
44  //-----
45      //ACCESSOR getCapacity
46      //EXPORT: array length (int)
47
48      public int getCapacity()
49      {
50          return array.length;
51      }
52  //-----
53      //ACCESSOR isEmpty
54      //EXPORT: empty (boolean)
55
56      public boolean isEmpty()
57      {
58          return ( count == 0 );
59      }
60  //-----
61      //ACCESSOR isFull
62      //EXPORT: full (boolean)
63
64      public boolean isFull()
65      {
66          //Length stored in array itself
67          return ( count == array.length );
68      }
69  //-----
70      //MUTATOR addCarton
71      //IMPORT: inCart (Carton), index (int)
72      //PURPOSE: Add new Carton to array
73
74      public void addCarton(Carton inCart, int index)
75      {
76          //Can't add anymore values if array is full
77          if ( isFull() )
78          {
79              throw new IllegalStateException("Array Is Full. Cannot Add");
80          }
81
82          //Can't add value if index is full
83          if ( array[index] instanceof Carton )
84          {

```

```

85         throw new IllegalStateException("Index is occupied. cannot add");
86     }
87
88     array[index] = inCart;
89     inCart.setRIndex(index);
90     count++;
91 }
92 //-----
93 //MUTATOR addCarton
94 //IMPORT: inCart (Carton)
95 //PURPOSE: Add new Carton to array
96
97 public void addCarton(Carton inCart)
98 {
99     //Add to Array, increment counter
100     int i = 0;
101     boolean done = false;
102     while ( ( i < array.length ) && ( done == false ) )
103     {
104         if ( array[i] == null )
105         {
106             array[i] = inCart;
107             done = true;
108         }
109         i++;
110     }
111
112     inCart.setRIndex(i - 1);
113     count++;
114 }
115 //-----
116 //MUTATOR removeCarton
117 //IMPORT: index (int)
118 //EXPORT: outCart (Carton)
119 //PURPOSE: Remove Carton from the specific index in array
120
121 public Carton removeCarton(int index)
122 {
123     //Can't remove anymore values if array is empty
124     if ( isEmpty() )
125     {
126         throw new IllegalStateException("Yard Is Empty. Cannot Remove");
127     }
128
129     Carton outCart = array[index];
130     array[index] = null;
131     outCart.setDIndex(-1);
132     outCart.setRIndex(-1);
133     count--;
134     return outCart;
135 }
136 //-----
137 //MUTATOR removeCarton
138 //EXPORT: outCart (Carton)
139 //PURPOSE: Remove front Carton from the array
140
141 public Carton removeCarton()
142 {
143     //Can't remove anymore values if array is empty
144     if ( isEmpty() )
145     {
146         throw new IllegalStateException("Yard Is Empty. Cannot Remove");
147     }
148
149     int ii = 0;
150     boolean done = false;
151     Carton outCart = array[0];
152
153     //Iterate until first non null element in the Yard
154     while ( ( ii < array.length ) && ( done == false ) )
155     {
156         if ( outCart == null )
157         {
158             ii++;
159             outCart = array[ii];
160         }
161         else
162         {
163             done = true;
164         }
165     }
166
167     array[ii] = null;
168     outCart.setDIndex(-1);
169     outCart.setRIndex(-1);
170     count--;
171     return outCart;

```

```
172     }
173 //-----
174 //ACCESSOR toString
175 //EXPORT: stateString (String)
176 //PURPOSE: Prints out room Carton's in DC Geometry file format
177
178 public String toString()
179 {
180     String stateString = YARD;
181     for (int i = 0; i < array.length; i++)
182     {
183         //Accounts for empty slots via ":" print outside
184         stateString += ":";
185         if ( array[i] != null )
186         {
187             stateString += array[i].getNote();
188         }
189     }
190     return stateString;
191 }
192 //-----
193 //ACCESSOR contentString
194 //EXPORT: stateString (String)
195 //PURPOSE: Output All Carton Contents in Queue As a String
196
197 public String contentString()
198 {
199     String stateString = "";
200     for ( int ii = 0; ii < array.length; ii++ )
201     {
202         if ( array[ii] != null )
203         {
204             stateString += array[ii].toString() + "\n";
205         }
206     }
207     return stateString;
208 }
209 //-----
210 }
```