10/26/15 01:54:38 /home/15504319/DSA120/DSAAssignment/connorLib/DistroCentre.java

```
  1 /*******************************************************************************
  2 *    FILE: DistroCentre.java
  3 *    AUTHOR: Connor Beardsmore - 15504319
  4 *    UNIT: DSA120 Assignment S2- 2015
  5 *    PURPOSE: Container Class For a DC and it's stock rooms
  6 *    LAST MOD: 23/10/15
  7 *  REQUIRES: NONE
  8 *******************************************************************************/
  9 package connorLib;
 10
 11 public class DistroCentre
 12 {
 13     //TYPEREF CONSTANTS
 14     //MAKES IT EASY TO CHECK WHAT TYPE A STOCKROOM IS
 15     public static final int DEADEND = 0;
 16     public static final int ROLLING = 1;
 17     public static final int YARD = 2;
 18
 19     //CLASSFIELDS
 20     private IStockRoom[] stockRooms;
 21     private Carton[] indexArray;
 22     private int[] typeRef;
 23     private int count;
 24
 25 //------------------------------------------------------------------------------
 26     //ALTERNATE Constructor
 27     //IMPORT: numRooms (int)
 28
 29     public DistroCentre(int numRooms)
 30     {
 31         stockRooms = new IStockRoom[numRooms];
 32         //Since array is 0-indexed, we increment before creation.
 33         //indexArray[0] will always be empty
 34         indexArray = new Carton[Carton.MAX_CON_NOTE + 1];
 35         typeRef = new int[numRooms];
 36         count = 0;
 37     }
 38 //------------------------------------------------------------------------------
 39     //ACCESSOR getCount
 40     //EXPORT: count (int)
 41
 42     public int getCount()
 43     {
 44         return count;
 45     }
 46 //------------------------------------------------------------------------------
 47     //ACCESSOR copyIndexArray
 48     //EXPORT: Copy of index array (Carton)
 49     //PURPOSE: Returns a copy of the index array, if modification is required
 50
 51     public Carton[] copyIndexArray()
 52     {
 53         return indexArray.clone();
 54     }
 55 //------------------------------------------------------------------------------
 56     //ACCESSOR getCartonIndex
 57     //IMPORT: index (int)
 58     //PURPOSE: O(1) access to and Carton given its index, will return null
 59     //           if no Carton of that index exists
 60
 61     public Carton getCartonIndex(int index)
 62     {
 63         return indexArray[index];
 64     }
 65 //------------------------------------------------------------------------------
 66     //ACCESSOR setCartonIndex
 67     //IMPORT: inCart (Carton)
 68     //PURPOSE: O(1) access to place a Carton into the indexArray
 69
 70     public void setCartonIndex(Carton inCart)
 71     {
 72         indexArray[ inCart.getNote() ] = inCart;
 73     }
 74 //------------------------------------------------------------------------------
 75     public void nullCartonIndex(int index)
 76     {
 77         indexArray[index] = null;
 78     }
 79 //------------------------------------------------------------------------------
 80     //ACCESSOR getStockRoom
 81     //IMPORT: index (int)
 82     //EXPORT: stockRoom (IStockRoom)
 83
 84     public IStockRoom getStockRoom(int index)
```

```
85      {
86          return stockRooms[index];
87      }
88  //-------------------------------------------------------------------------
89      //ACCESSOR getType
90      //IMPORT: index (int)
91      //EXPORT: typeRef (int)
92      //PURPOSE: Get type of a specific stockroom in DC, via type reference array
93
94      public int getType(int index)
95      {
96          return typeRef[index];
97      }
98  //-------------------------------------------------------------------------
99      //ACCESSOR totalItems
100     //EXPORT: totalItems (int)
101     //PURPOSE: Calculate total items across entire DC
102
103     public int totalItems()
104     {
105         int totalItems = 0;
106         for (int ii = 0; ii < count; ii++)
107         {
108             totalItems += stockRooms[ii].getCount();
109         }
110         return totalItems;
111     }
112 //-------------------------------------------------------------------------
113     //ACCESSOR totalCapacity
114     //EXPORT: totalCapacity (int)
115     //PURPOSE: Calculate total capacity across entire DC
116
117     public int totalCapacity()
118     {
119         int totalCapacity = 0;
120         for (int ii = 0; ii < count; ii++)
121         {
122             totalCapacity += stockRooms[ii].getCapacity();
123         }
124         return totalCapacity;
125     }
126 //-------------------------------------------------------------------------
127     //ACCESSOR isFull
128     //EXPORT: full (boolean)
129     //PURPOSE: Returns true if DC has no free slots, false otherwise
130
131     public boolean isFull()
132     {
133         boolean full = true;
134         for (int ii = 0; ii < count; ii++)
135         {
136             //Break or return would be more effiecient here
137             full = ( full && stockRooms[ii].isFull() );
138         }
139         return full;
140     }
141 //-------------------------------------------------------------------------
142     //createDeadEnd
143     //IMPORT: maxCap (int)
144     //PURPOSE: Create a new DeadEnd stockroom in the DC based on maxCap
145
146     public void createDeadEnd(int maxCap)
147     {
148         stockRooms[count] = new DeadEnd(maxCap);
149         typeRef[count] = DEADEND;
150         count++;
151     }
152 //-------------------------------------------------------------------------
153     //createRolling
154     //IMPORT: maxCap (int)
155     //PURPOSE: Create a new Rolling stockroom in the DC based on maxCap
156
157     public void createRolling(int maxCap)
158     {
159         stockRooms[count] = new Rolling(maxCap);
160         typeRef[count] = ROLLING;
161         count++;
162     }
163 //-------------------------------------------------------------------------
164     //createYard
165     //IMPORT: maxCap (int)
166     //PURPOSE: Create a new Yard stockroom in the DC based on maxCap
167
168     public void createYard(int maxCap)
169     {
170         stockRooms[count] = new Yard(maxCap);
171         typeRef[count] = YARD;
```

```
172          count++;
173      }
174  //---------------------------------------------------------------------------
175      //createOutString
176      //EXPORT: outputString (String)
177      //PURPOSE: Create String representation of entire DC
178
179      public String createOutString()
180      {
181          String outputString = "";
182
183          for (int ii = 0; ii < count; ii ++)
184          {
185              outputString += stockRooms[ii].toString() + "\n";
186          }
187          //Formatting to split DC section and Carton section
188          outputString += "\n%\n\n";
189
190          for (int ii = 0; ii < count; ii ++)
191          {
192              outputString += stockRooms[ii].contentString();
193          }
194
195          return ( outputString + "\n" );
196      }
197  //---------------------------------------------------------------------------
198  }
```