10/26/15 01:54:53 /home/15504319/DSA120/DSAAssignment/connorLib/DeadEnd.java

```
  1  /*******************************************************************************
  2  *    FILE: DeadEnd.java
  3  *    AUTHOR: Connor Beardsmore - 15504319
  4  *    UNIT: DSA120 Assignment S2- 2015
  5  *    PURPOSE: Dead-End StockRoom for use in the DC ( stack )
  6  *    LAST MOD: 19/10/15
  7  *  REQUIRES: NONE
  8  *******************************************************************************/
  9  package connorLib;
 10
 11  public class DeadEnd implements IStockRoom
 12  {
 13      //CLASS FIELDS
 14      private Carton[] stack;
 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 DEADEND = "D";
 21  //-------------------------------------------------------------------------
 22      //ALTERNATE Constructor
 23      //IMPORT: maxCapacity (int)
 24      //ASSERTION: stack allocated 'maxCapacity' elements. Count to default 0
 25
 26      public DeadEnd(int maxCapacity)
 27      {
 28          //maxCapacity must be a value between 2 and 10,000
 29          if ( (maxCapacity < MIN_VALID_CAP) || (maxCapacity > MAX_VALID_CAP) )
 30          {
 31              throw new IllegalArgumentException("DeadEnd Capacity Not Valid");
 32          }
 33          stack = new Carton[maxCapacity];
 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 stack.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 stack itself
 67          return ( count == stack.length );
 68      }
 69  //-------------------------------------------------------------------------
 70      //MUTATOR addCarton
 71      //IMPORT: inCart (Carton)
 72      //PURPOSE: Add new value to the top of the stack
 73
 74      public void addCarton(Carton inCart)
 75      {
 76          //Can't add anymore values if stack is full. Must remove first
 77          if ( isFull() )
 78          {
 79              throw new IllegalStateException("DeadEnd Is Full. Cannot Add");
 80          }
 81          //Add to stack, increment counter
 82          stack[count] = inCart;
 83          stack[count].setRIndex(count);
 84          count++;
```

```
85      }
86  //--------------------------------------------------------------------------
87      //MUTATOR removeCarton
88      //EXPORT: outCart (Carton)
89      //PURPOSE: Remove top value from the stack
90
91      public Carton removeCarton()
92      {
93          Carton outCart = top();
94          stack[count - 1] = null;
95          //Indexes set back to default state. Doesn't exist in DC
96          outCart.setDIndex(-1);
97          outCart.setRIndex(-1);
98          count--;
99          return outCart;
100     }
101 //--------------------------------------------------------------------------
102     //ACCESSOR top
103     //IMPORT: value (Carton)
104     //PURPOSE: View top value on the stack. Not removed
105
106     public Carton top()
107     {
108         if ( isEmpty() )
109         {
110             throw new IllegalStateException("DeadEnd Is Empty. No Top");
111         }
112         //Top Element
113         return stack[count-1];
114     }
115 //--------------------------------------------------------------------------
116     //ACCESSOR toString
117     //EXPORT: stateString (String)
118     //PURPOSE: Prints out room Carton's in  DC Geometry file format
119
120     public String toString()
121     {
122         String stateString = DEADEND;
123         for (int i = 0; i < stack.length; i++)
124         {
125             //Accounts for empty slots via ":" print outside
126             stateString += ":";
127             if ( stack[i] != null )
128             {
129                 stateString += stack[i].getNote();
130             }
131         }
132         return stateString;
133     }
134 //--------------------------------------------------------------------------
135     //ACCESSOR contentString
136     //EXPORT: stateString (String)
137     //PURPOSE: Output All Carton Contents in Stack As a String
138
139     public String contentString()
140     {
141         String stateString = "";
142         for ( int ii = 0; ii < count; ii++ )
143         {
144             stateString += stack[ii].toString() + "\n";
145         }
146         return stateString;
147     }
148 //--------------------------------------------------------------------------
149 }
```