10/26/15 11:00:21 /media/TAKAMORI/DSAAssignment/connorLib/Rolling.java

```java
/***************************************************************************
*    FILE: Rolling.java
*    AUTHOR: Connor Beardsmore - 15504319
*    UNIT: DSA120 Assignment S2-
2015
*    PURPOSE: Rolling StockRoom for use in the DC ( queue )
*    LAST MOD: 20/10/15
*  REQUIRES: NONE
***************************************************************************/
package connorLib;

public class Rolling implements IStockRoom
{
     //CLASS FIELDS
     private Carton[] queue;
     private int count;

     //CLASS CONSTANTS
     private static final int MAX_VALID_CAPACITY = 10000;
    private static final int MIN_VALID_CAPACITY = 1;
    private static final String ROLLING = "R";
//---------------------------------------------------------------------------
    //ALTERNATE Constructor
    //IMPORT: maxCap (int)
    //ASSERTION: queue allocated 'maxCap' elements.Count to default 0

    public Rolling(int maxCap)
    {
        //maxCapacity must be a value between 2 and 10,000
        if ( (maxCap < MIN_VALID_CAPACITY) || ( maxCap > MAX_VALID_CAPACITY ) )
        {
            throw new IllegalArgumentException("Rolling Capacity Not Valid");
        }
        queue = new Carton[maxCap];
        count = 0;
    }
//---------------------------------------------------------------------------
    //ACCESSOR getCount
    //EXPORT: count (int)

    public int getCount()
    {
        return count;
    }
//---------------------------------------------------------------------------
    //ACCESSOR getCapacity
    //EXPORT: array length (int)

    public int getCapacity()
    {
        return queue.length;
    }
//---------------------------------------------------------------------------
    //ACCESSOR isEmpty
    //EXPORT: empty (boolean)

    public boolean isEmpty()
    {
        return ( count == 0 );
    }
//---------------------------------------------------------------------------
    //ACCESSOR isFull
    //EXPORT: full (boolean)

    public boolean isFull()
    {
        return ( count == queue.length );
    }
//--------------------------------------------------------------------------
    //MUTATOR addCarton
    //IMPORT: inCart (Carton)
    //PURPOSE: Add new value to back of the queue

    public void addCarton(Carton inCart)
    {
        //Can't add anymore values if queue is full. Must dequeue first
        if ( isFull() )
        {
            throw new IllegalStateException("Rolling Is Full. Cannot Add");
        }
        //Add to queue, increment counter
        queue[count] = inCart;
        inCart.setRIndex(count);
        count++;
    }
//--------------------------------------------------------------------------
    //MUTATOR removeCarton
    //EXPORT: outCart (Carton)
    //PURPOSE: Remove front value from the queue (SHUFFLING)

    public Carton removeCarton()
    {
        //Is empty is checked within peek. No need to repeat check
        Carton outCart = peek();

        //Shuffles all elements down by one
        for (int i = 0; i < count - 1; i++)
        {
            queue[i] = queue[i+1];
            queue[i].setRIndex(i);
        }
        //Set indexes back to default state. Doesn't exist in DC
        outCart.setDIndex(-1);
        outCart.setRIndex(-1);
        queue[count - 1] = null;
        count--;
        return outCart;
    }
//--------------------------------------------------------------------------
    //ACCESSOR peek
    //IMPORT: value (Carton)
    //PURPOSE: View front value of the queue. Not removed

    public Carton peek()
    {
        if ( isEmpty() )
        {
            throw new IllegalStateException("Rolling is Empty. No Top");
        }
        return queue[0];
    }
//--------------------------------------------------------------------------
    //ACCESSOR toString
    //EXPORT: stateString (String)
    //PURPOSE: Prints out room Carton's in  DC Geometry file format

    public String toString()
    {
        String stateString = ROLLING;
        for (int i = 0; i < queue.length; i++)
        {
            //Accounts for empty slots via ":" print outside
            stateString += ":";
            if ( queue[i] != null )
            {
                stateString += queue[i].getNote();
            }
        }
        return stateString;
    }
//---------------------------------------------------------------------------
    //ACCESSOR contentString
    //EXPORT: stateString (String)
    //PURPOSE: Output All Carton Contents in Queue As a String

    public String contentString()
    {
        String stateString = "";
        for ( int ii = 0; ii < count; ii++ )
        {
            stateString += queue[ii].toString() + "\n";
        }
        return stateString;
    }
//---------------------------------------------------------------------------
}
```