10/26/15 01:55:34 /home/15504319/DSA120/DSAAssignment/connorLib/Carton.java

```
/****************************************************************************
*    FILE: Carton.java
*    AUTHOR: Connor Beardsmore - 15504319
*    UNIT: DSA120 Assignment S2- 2015
*    PURPOSE: Container Class For A Single Carton
*    LAST MOD: 24/10/15
*  REQUIRES: java.util, java.Lang
****************************************************************************/
package connorLib;
import java.util.*;
import java.lang.*;

public class Carton implements ISortable
{
    //Currently the maximum consignment number
    //Changing here will change values in all program files.
    public static final int MAX_CON_NOTE = 1023;

    //CLASSFIELDS
    private final int conNote;            // 1 - 1023 int [C]
    private final String product;        // All Strings Valid [P]
    private final String wholesaler;    // All Strings Valid [WH]
    private final DateClass warranty;    // Validated internally [WA]

    //FOR REFERNCING LOCATION IN DC
    private int distroIndex;            // -1 for carton not in DC
    private int roomIndex;                // -1 for carton not in DC

//---------------------------------------------------------------------------
    //CONSTRUCTOR Alternate
    //IMPORT: inNote (int), inProd (String), inWhole (String), inDate (String)

    public Carton(int inNote, String inProd, String inWhole, DateClass inDate)
    {
        if ( (inNote < 1) || (inNote > MAX_CON_NOTE) )
        {
            throw new IllegalArgumentException("Consignment Note Invalid");
        }
        warranty = inDate;
        conNote = inNote;
        product = inProd;
        wholesaler = inWhole;
        //DEFAULT VALUES
        distroIndex = -1;
        roomIndex = -1;
    }
//---------------------------------------------------------------------------
    //CONSTRUCTOR Alternate
    //IMPORT: itemString (String)
    //PURPOSE: Creates Carton from string of format C:P:WH:WA

    public Carton(String itemString)
    {
        StringTokenizer strTok  = new StringTokenizer(itemString, ":");
        int count = strTok.countTokens();

        //Assumes that Product/Wholesaler names cannot include colons
        if ( count != 4 )
        {
            String errMessage = "Invalid Carton Format, Too Many Fields";
            throw new IllegalArgumentException( errMessage );
        }
        //Validate conNote is a number

        try
        {
            conNote = Integer.parseInt( strTok.nextToken() );
        }
        catch (NumberFormatException e)
        {
            throw new NumberFormatException("Con Note Must Be Numbers Only");
        }

        if ( (conNote < 1) || (conNote > MAX_CON_NOTE) )
        {
            throw new IllegalArgumentException("Consignment Note Invalid");
        }

        warranty = new DateClass( strTok.nextToken() );
        product = strTok.nextToken();
        wholesaler = strTok.nextToken();
        distroIndex = -1;
        roomIndex = -1;
    }
//---------------------------------------------------------------------------
```

```java
    //ACCESSOR: getNote
    //EXPORT: conNote (int)

    public int getNote()
    {
        return conNote;
    }
//---------------------------------------------------------------------------
    //ACCESSOR: getProduct
    //EXPORT: product (String)

    public String getProduct()
    {
        return product;
    }
//---------------------------------------------------------------------------
    //ACCESSOR: getWhole
    //EXPORT: wholesaler (String)

    public String getWhole()
    {
        return wholesaler;
    }
//---------------------------------------------------------------------------
    //ACCESSOR: getWar
    //EXPORT: warranty (DateClass)

    public DateClass getWar()
    {
        return warranty;
    }
//---------------------------------------------------------------------------
    //ACCESSOR: getDIndex
    //EXPORT: distroIndex (int)

    public int getDIndex()
    {
        return distroIndex;
    }
//---------------------------------------------------------------------------
    //ACCESSOR: setDIndex
    //IMPORT: newIndex (int)

    public void setDIndex(int newIndex)
    {
        distroIndex = newIndex;
    }
//---------------------------------------------------------------------------
    //ACCESSOR: getRIndex
    //EXPORT: roomIndex (int)

    public int getRIndex()
    {
        return roomIndex;
    }
//---------------------------------------------------------------------------
    //ACCESSOR: setRIndex
    //IMPORT: newIndex (int)

    public void setRIndex(int newIndex)
    {
        roomIndex = newIndex;
    }
//---------------------------------------------------------------------------
    //ACCESSOR: toString
    //EXPORT: statestring (String)
    //PURPOSE: Export Carton as a readable String, format of C:WA:P:WH

    public String toString()
    {
        return new String( conNote + ":" + warranty.toString() + ":"
                                + product + ":" + wholesaler);
    }
//---------------------------------------------------------------------------
    //IMPERATIVE: lessThan
    //IMPORT: inCart (ISortable)
    //EXPORT: less (boolean)
    //PURPOSE: Compares two Cartons based on their DC location, returning
     //            true if this.Carton is lessThan inCart

    public boolean lessThan(ISortable inCart)
    {
        if ( !( inCart instanceof Carton ) )
        {
            throw new IllegalArgumentException("inCart Is Wrong Date Type");
        }

        Carton inItem = (Carton)inCart;
        boolean less = false;

        if ( distroIndex < inItem.getDIndex() )
        {
            less = true;
        }
        else if ( distroIndex == inItem.getDIndex() )
        {
            if ( roomIndex < inItem.getRIndex() )
            {
                less = true;
            }
        }

        return less;
    }
//---------------------------------------------------------------------------
    //IMPERATIVE: compareTo
    //IMPORT: inOne (Carton)
    //EXPORT: -1/0/+1 (int)
    //PURPOSE: Compare 2 Carton to each other, via their warranty date

    public int compareTo(Carton inOne)
    {
        return ( warranty.compareTo( inOne.getWar() ) );
    }
//---------------------------------------------------------------------------
}
```