

Utility.java

```
/* File: Utility.java
```

This file contains several utility methods that are designed to be used by other programs.

getMonthName - returns the name of the month

getDayName - returns the name of the day

padString - pads a numeric string value with a designated character to form a fixed length string

fixString - pads a string with trailing spaces to form a fixed length string

addComponent - adds a component to a container

readString - reads a String from the keyboard

readInt - reads an int value from the keyboard

readDouble - reads a double value from the keyboard

stringSplit - splits a String object into an array of Strings, based on a specified delimiter

```
*/
```

```
import java.awt.*;                // used by addComponent method
```

```
import java.text.*;              // needed for DecimalFormat class
```

```
import java.util.*;
```

```
public class Utility
```

```
{
```

```
/******
```

```
getMonthName(int)
```

Returns the month name of a desired month.

Parameters:

int monthNumber month number (1-12)

Returns:

a String object

example:

```
String monthName=Utility.getMonthName(7);
```

```
*/
```

```
public static String getMonthName(int monthNumber)
```

```
{
```

```
String[] monthNames=new String[12];
```

```
monthNames[0]="January";
```

```
monthNames[1]="February";
```

```
monthNames[2]="March";
```

```
monthNames[3]="April";
```

```
monthNames[4]="May";
```

```
monthNames[5]="June";
```

```
monthNames[6]="July";
```

```
monthNames[7]="August";
```

```
monthNames[8]="September";
```

```
monthNames[9]="October";
```

```
monthNames[10]="November";
```

```
monthNames[11]="December";
```

```
return monthNames[monthNumber-1];
```

```
}
```

```
/******
```

```
getDayName(int)
```

Returns the day name of a desired day number.

Parameters:

int dayNumber day number (1-7)

Returns:

a String object

example:

```
String dayName=Utility.getDayName(7);
```

```
*/
```

```
public static String getDayName(int dayNumber)
```

```

{
    String[] dayNames=new String[7];
    dayNames[0]="Sunday";
    dayNames[1]="Monday";
    dayNames[2]="Tuesday";
    dayNames[3]="Wednesday";
    dayNames[4]="Thursday";
    dayNames[5]="Friday";
    dayNames[6]="Saturday";

    return dayNames[dayNumber-1];
}

```

```

/*****

```

```

padString(Number,int,int,String)

```

Method which pads a string of digits with trailing zeroes and a specified leading padding character to form a fixed length string with optional decimal places.

Parameters:

Number number	input Number object
int chars	length of desired fixed length string
int decimals	desired number of digits to the right of the decimal point
	in returned fixed length string
String lead	character to be used for leading padding

Returns:

a String object

example:

```

String output=Utility.padString(50.4,6,2,"*");

```

output will have a String value of "*50.40"

```

*/

```

```

public static String padString(Number number,int chars,int decimals,String lead)
{
    DecimalFormat format;
    String s="";

```

```

if(decimals>0)
{
    s=".";
    for(int i=0;i<decimals;i++)
    {
        s=s+"0";
    }
}

```

```

int digits=chars-s.length();
for(int i=0;i<digits;i++)
{
    if(i>0 && i%3==0)
    {
        s=","+s;
    }
    s="#" +s;
}

```

```

format=new DecimalFormat(s);
s=format.format(number);
digits=chars-s.length();
for(int i=0;i<digits;i++)
{
    s=lead+s;
}
return s;
}

```

```

public static String padString(long number,int chars,int decimals,String lead)
{
    return Utility.padString(new Long(number),chars,decimals,lead);
}

```

```

public static String padString(double number,int chars,int decimals,String lead)
{
    return Utility.padString(new Double(number),chars,decimals,lead);
}

```

```

public static String padString(String number,int chars,int decimals,String lead)

```

```
{
    return Utility.padString(new Double(number),chars,decimals,lead);
}

public static String padString(Number number,int chars,int decimals)
{
    return Utility.padString(number,chars,decimals," ");
}

public static String padString(long number,int chars,int decimals)
{
    return Utility.padString(new Long(number),chars,decimals," ");
}

public static String padString(double number,int chars,int decimals)
{
    return Utility.padString(new Double(number),chars,decimals," ");
}

public static String padString(String number,int chars,int decimals)
{
    return Utility.padString(new Double(number),chars,decimals," ");
}

public static String padString(Number number,int chars)
{
    return Utility.padString(number,chars,0," ");
}

public static String padString(long number,int chars)
{
    return Utility.padString(new Long(number),chars,0," ");
}

public static String padString(double number,int chars)
{
    return Utility.padString(new Double(number),chars,0," ");
}

public static String padString(String number,int chars)
{
    return Utility.padString(new Double(number),chars,0," ");
}
```

```
return Utility.padString(new Double(number),chars,0," ");
}
```

```
public static String padString(Number number,int chars,String lead)
{
    return Utility.padString(number,chars,0,lead);
}
```

```
public static String padString(long number,int chars,String lead)
{
    return Utility.padString(new Long(number),chars,0,lead);
}
```

```
public static String padString(double number,int chars,String lead)
{
    return Utility.padString(new Double(number),chars,0,lead);
}
```

```
public static String padString(String number,int chars,String lead)
{
    return Utility.padString(new Double(number),chars,0,lead);
}
```

```
/******
fixString(String,int)
```

Method which pads a string with trailing spaces to form a fixed length field.

Parameters:

String inString	input string of digits
int chars	length of desired fixed length string

Returns:

a String object

example:

```
input="test";
output=Utility.fixString(input,10);
```

output will have a String value of "test "

```
*/
```

```
public static String fixString(String inString,int chars)
{
    String s=inString;                // get temp string
    for(int i=0;i<chars-inString.length();i++)    // add trailing spaces
    {
        s=s+" ";
    }
    return s.substring(0,chars);        // return fixed length field
}
```

```
/******
```

```
commaString(String,int)
```

Method which inserts commas into a number for readability

Parameters:

String inString	input string of digits
int chars	length of desired fixed length string

Returns:

a String object

example:

```
input="test";
output=Utility.fixString(input,10);
```

output will have a String value of "test "

```
*/
```

```
public static String commaString(String inString)
{
    String s="";
    int decimal=inString.indexOf(".");
    if(decimal!=-1)
    {
        if(inString.length()<3)
        {
            return inString;
```

```

    }
    s=inString.substring(inString.length()-3,inString.length());
    inString=inString.substring(0,inString.length()-3);
    while(inString.length()>2)
    {
        if(inString.length()==3)
        {
            s=inString+","+s;
            inString="";
        }
        else
        {
            s=inString.substring(inString.length()-3,inString.length())+","+s;
            inString=inString.substring(0,inString.length()-3);
        }
    }
    if(inString.length()>0)
    {
        s=inString+","+s;
    }
}
return s;
}

```

```

public static String commaString(long number)
{
    return Utility.commaString(number+"");
}

```

```

public static String commaString(double number)
{
    return Utility.commaString(number+"");
}

```

```

/*****
addComponent(Container container,Component component,int align)

```

Method which adds a component to a container. This method will use a components preferred size, not necessarily filling up the container (as GridLayout would do).

Parameters:

Container container the container to which the component is to be added

Component component the component

int align a constant from the FlowLayout class

Returns:

nothing

example:

```
Panel myPanel=new Panel();
```

```
myPanel.setLayout(new GridLayout(4,2));
```

```
Button myButton=new Button("click me");
```

```
Utility.addComponent(myPanel,myButton);
```

```
*/
```

```
public static void addComponent(Container container,Component component,int align)
```

```
{
```

```
    Panel tempPanel=new Panel();
```

```
    tempPanel.setLayout(new FlowLayout(align));
```

```
    tempPanel.add(component);
```

```
    container.add(tempPanel);
```

```
}
```

```
public static void addComponent(Container container,Component component)
```

```
{
```

```
    addComponent(container,component,FlowLayout.LEFT);
```

```
}
```

```
/******
```

```
readString()
```

Method which reads a String value from the keyboard.

Parameters:

none

Returns:

a String object

example:

```
String inString=Utility.readString();
```

```
*/
```

```
public static String readString()
{
    int  input=0;           // define and initialize an int variable
    String s="";           // define and initialize a String object

    while(input!=10)        // loop until LF byte is read
    {
        try
        {
            input=System.in.read();    // read next byte from keyboard
        }
        catch(Exception e)
        {
            input=13;
        }
        if(input!=13 && input!=10)      // if byte is not CR and not LF, concatenate
        {
            s=s+(char)input;           // cast int input as char, then concatenate
        }
    }
    return s;                   // return String to calling program
}
```

```
/******
```

```
readInt()
```

Method which reads an int value from the keyboard.

Parameters:

none

Returns:

an int value

-999999f if error (invalid input)

example:

```
int i=Utility.readInt();
```

```
*/
```

```
public static int readInt()
```

```
{
```

```
int i;
```

```
String s=Utility.readString();           // read a String from the keyboard
```

```
try
```

```
{
```

```
    i=Integer.parseInt(s);               // try to parse String into an int
```

```
}
```

```
catch(Exception e)                       // catch exception thrown by Integer.parseInt
```

```
{
```

```
    i=-999999;                           // flag to indicate input error
```

```
}
```

```
return i;
```

```
}                                         // return value of i
```

```
/******
```

```
readDouble()
```

Method which reads a double value from the keyboard.

Parameters:

none

Returns:

a double value

-999999f if error (invalid input)

example:

```
double d=Utility.readDouble();
```

```
*/
```

```

public static double readDouble()
{
    double d;

    String s=Utility.readString();           // read a String from the keyboard

    try
    {
        Double doubleObject=Double.valueOf(s);       // try to parse String into a Double
        d=doubleObject.doubleValue();
    }
    catch(Exception e)                       // catch exception
    {
        d=-999999;                           // flag to indicate input error
    }

    return d;
}                                           // return value of f

```

```

/*****
stringSplit(String s,String d)

```

Method which splits a String into an array of Strings, based on a specified delimiter

Parameters:

s -- String to be split
d -- specified delimiter

Returns:

an array of Strings

example:

```
String[] splitStrings=Utility.stringSplit(inputRecord,",");
```

```
*/
```

```

public static String[] stringSplit(String s,String d)
{
    Vector<String> v=new Vector<String>();       // vector to hold parsed substrings

```

```

while(s.length()>0)                // loop while string is not null
{
    int i=s.indexOf(d);              // find next delimiter
    if(i>-1)
    {
        // delimiter was found
        v.addElement(s.substring(0,i));    // extract substring (left of delimiter)
        s=s.substring(i+1);              // then strip string for further processing
    }
    else
    {
        // no delimiter found
        v.addElement(s);                // add final string to vector
        s="";                          // set string to null
    }
}

String[] returnString=new String[v.size()];    // instantiate String[] for return
for(int i=0;i<returnString.length;i++)        // step through vector, assigning to String[]
{
    returnString[i]=(String) v.elementAt(i);
}

return returnString;                    // return array of String objects
}

}

```

Revision #1

Created 10 January 2019 00:18:58 by Aaron Kimbrell

Updated 12 January 2019 03:56:00 by Aaron Kimbrell