org.usfirst.frc.team272.robot

Class LCTelemetry

java.lang.Object
    org.usfirst.frc.team272.robot.LCTelemetry

```
public class LCTelemetry
extends java.lang.Object
```

General purpose tool to allow user to record and save robot variable data to xls spreadsheet.

```
 Changes ************************************************************************************
 Date          Ver.    BY      Description
 Dec. 14, 2015 1.00    FWL     Initial submission
 Dec. 16, 2015 1.01    FWL     Added commenting and changed names to relate it is a spread sheet.
 Dec. 17, 2015 1.01    FWL     Added Java docs and changed a few things.
                               1) changed method addHeadedr to addColumn.
                               2) changed headerName to columnName.
```

Version:

1.0

Author:

Frank Larkin, FRC272 - Lansdale Catholic Robotics - 12-14-2015

Field Summary

Fields

Modifier and Type	Field and Description
boolean	**b_DataSaved** Default: /tmp folder.(linked to /var/volitile/tmp) Lost after reboot, /home/lvuser is writable.
int	**ki_MaxColumns** Spreadsheet has been written.
int	**ki_MaxRows** Number of columns.

Constructor Summary

Constructors

Constructor and Description
LCTelemetry() Class Contructor initialize you variables here.

Method Summary

All Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
void	**addColumn(java.lang.String columnName)** Called to add a column header to the Telemetry object.
java.lang.String	**getFileName()** Call see what the file name will be.

void	**outputToDashBoard**(java.lang.Boolean b_MinDisplay) Called to show a few of the fields that can aid you in setup. **Note: This only works while in disabledPeriodic mode.**
void	**restartTimer**() Used to reset the Telemetry internal timer to zero.
void	**saveBoolean**(java.lang.String columnName, java.lang.Boolean value) Called to add a true/false boolean to a column in the Telemetry instance.
void	**saveDouble**(java.lang.String columnName, double value) Called to add an Double float point number to a column in the Telemetry instance.
void	**saveFalseBoolean**(java.lang.String columnName, java.lang.Boolean value) Called to add a false boolean to a column in the Telemetry instance.
void	**saveInteger**(java.lang.String columnName, int value) Called to add an Integer to a column in the Telemetry instance.
void	**saveSpreadSheet**() Called to tell Telemtry to save the rows and columns that are the spreadsheet data.
void	**saveString**(java.lang.String columnName, java.lang.String value) Called to add a string to a column in the Telemetry instance.
void	**saveTrueBoolean**(java.lang.String columnName, java.lang.Boolean value) Called to add a true boolean to a column in the Telemetry instance.
void	**updatePreferences**() Called to tell telemetry to read its Preferences.
void	**writeRow**() Called to tell telemetry to save the current saved data into a row.

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

b_DataSaved

public boolean b_DataSaved

Default: /tmp folder.(linked to /var/volitile/tmp) Lost after reboot, /home/lvuser is writable.

ki_MaxColumns

public final int ki_MaxColumns

Spreadsheet has been written. Once set to true you cannot do this again until reset.

See Also:

Constant Field Values

ki_MaxRows

public final int ki_MaxRows

Number of columns. Increase if you need to.

See Also:

Constant Field Values

Constructor Detail

LCTelemetry

```
public LCTelemetry()
```

Class Contructor initialize you variables here. Here the size of the arrays can be adjusted to allow for more columns and more rows. The constants ki_MaxRos and ki_MaxColumns control that.

Method Detail

restartTimer

```
public void restartTimer()
```

Used to reset the Telemetry internal timer to zero. You have to add this method to be called in every Init method of the main Iterative Robot class, disabledInit, teleopInit, autonomousInit.
This way you will be able to see the actual time spent in each section.

Example:

```
public void teleopInit(){

    telem.restartTimer();

}
```

addColumn

```
public void addColumn(java.lang.String columnName)
```

Called to add a column header to the Telemetry object. The field should have spaces between words to allow them to easily word wrap in Excel. The exact spelling is needed to add column data.
Example: I Left Driver Power **Note: The I tells us this is from the Inputs Class.**

The order these are entered in will be the columns of the final spread sheet. The order the data is add has no impact on this.

It is assumed that you will add a method to each class like addTelemetryHeaders(). In that class you will add the different fields you want to track. This is best called in the constructor of the Robot class after all the other classes have been initialized.

Example:

```
public void addTelemetryHeaders(LCTelemetry telem ){
    telem.addColumn("A Program Number");
    telem.addColumn("A Started");
    telem.addColumn("A Step Timer");
}
```

Parameters:

`columnName - string at the top of this column.`

saveInteger

```
public void saveInteger(java.lang.String columnName,
                        int value)
```

Called to add an Integer to a column in the Telemetry instance. The columnName is the exact spelling of the name entered in an addColumn call. The order this data is added will not change the order of the columns.

Example:

```
public void writeTelemetryValues(LCTelemetry telem ){
    telem.saveInteger("A Program Number",this.ip_ProgramNumber);
    telem.saveTrueBoolean("A Started",this.b_Started);
    telem.saveTrueBoolean("A Is Done",this.b_IsDone);
    telem.saveDouble("A Auton Timer",this.tim_AutonTimer.get());
```

```
        telem.saveString("A Program Desc",this.s_ProgramDescription);
        telem.saveString("A Step Desc",this.s_StepDescription);
        telem.saveInteger("A Step",this.i_Step);
        telem.saveDouble("A Step Timer",this.tim_StepTimer.get());
    }
```

Parameters:

columnName - string at the top of this column.

value - Integer number to be written.

saveDouble

```
public void saveDouble(java.lang.String columnName,
                       double value)
```

Called to add an Double float point number to a column in the Telemetry instance. The columnName is the exact spelling of the name entered in an addColumn call. The order this data is added will not change the order of the columns. The resulting data will have 2 decimal points of precision.

Parameters:

columnName - string at the top of this column.

value - Double number to be written.

saveString

```
public void saveString(java.lang.String columnName,
                       java.lang.String value)
```

Called to add a string to a column in the Telemetry instance. The columnName is the exact spelling of the name entered in an addColumn call. The order this data is added will not change the order of the columns. No additional formatting is done on the data.

Example: See saveInteger()

Parameters:

columnName - string at the top of this column.

value - String to be written.

saveBoolean

```
public void saveBoolean(java.lang.String columnName,
                        java.lang.Boolean value)
```

Called to add a true/false boolean to a column in the Telemetry instance. The columnName is the exact spelling of the name entered in an addColumn call. The order this data is added will not change the order of the columns. No additional formatting is done on the data but Excel will fore these to CAPs.

Example: See saveInteger()

Parameters:

columnName - string at the top of this column.

value - Boolean to be written.

saveTrueBoolean

```
public void saveTrueBoolean(java.lang.String columnName,
                            java.lang.Boolean value)
```

Called to add a true boolean to a column in the Telemetry instance. If the item is false then an empty field will be added. If true then the field will say true. Excel will set this to all CAPS.

The columnName is the exact spelling of the name entered in an addColumn call. The order this data is added will not change the order of the columns.

Example: See saveInteger()

Parameters:

columnName - string at the top of this column.

value - boolean to be written.

saveFalseBoolean

```
public void saveFalseBoolean(java.lang.String columnName,
                             java.lang.Boolean value)
```

Called to add a false boolean to a column in the Telemetry instance. If the item is true then an empty field will be added. If false then the field will say false. Excel will set this to all CAPS.

The columnName is the exact spelling of the name entered in an addColumn call. The order this data is added will not change the order of the columns.

Example: See saveInteger()

Parameters:

columnName - string at the top of this column.

value - boolean to be written.

writeRow

```
public void writeRow()
```

Called to tell telemetry to save the current saved data into a row. This will cause the data to be saved under the appropriate headers. Any columns missing data will show blank.

This is expected to be called at the bottom of the TeleopPeriodic and the autonomousPeriodic methods of the Robot class.

Example:

```
    public void autonomousPeriodic() {

        bla...bla...bla

        inputs.writeTelemetryValues(telem);
        robotbase.writeTelemetryValues(telem);
        wedge.writeTelemetryValues(telem);
        telem.writeRow();                               // write a record now
    }
```

updatePreferences

```
public void updatePreferences()
```

Called to tell telemetry to read its Preferences. The available fields are...
1) Tele_FileName - Default: telemetry - Name of the file to be written. This will have a .xls extension added.

2) Tele_FilePath - Default: /tmp - Location on the roboRio where the file is written. This is in /var/volitile/tmp. This will be erased when powered off. So get it before you power off. The folder /home/lvuser is writable and is not volitile. ***Use at your own risk.***

3) Tele_TimestampFile - Default: false - Use to tell the system to add a current time stamp to the file. This is applied just before the data is written. Example: telemetry.2015-12-14-13-25-23.xls

getFileName

```
public java.lang.String getFileName()
```

Call see what the file name will be. You can include on the SmartDashboard so you can see what the name will look like based upon the preferences. If you have the time stamp option enabled the time will be when the final file is written.

Returns:

String of the full file name with .xls extension.

outputToDashBoard

```
public void outputToDashBoard(java.lang.Boolean b_MinDisplay)
```

Called to show a few of the fields that can aid you in setup.
Note: This only works while in disabledPeriodic mode.

1) Tele_FileName - Final File name. Note: Is timestamp is used the timestamp will updated until you actually write the file.

2) Tele_DataSaved - a boolen that will show true when the spreadsheet is written. Once written you cannot write again until the code is restarted.

Example: /tmp/telemetry.2015-12-14-13-25-23.xls

Parameters:

```
b_MinDisplay - Ignored but kept for compatibility with other LC functions with this name.
```

saveSpreadSheet

```
public void saveSpreadSheet()
```

Called to tell Telemtry to save the rows and columns that are the spreadsheet data. Note: It is expected that this is called while in disabledPeriodic mode during testing or at the end of a competition round.

The best way is to have 2 buttons that must be pressed at the same time while in disabledPeriodic mode. Possibly one on the driver stick and the other on the operator stick. We do not want this to be triggered accidently during a round.

Example:

```
 public void disabledPeriodic() {

          // bla..bla..bla

     if( inputs.b_SaveTelemetry == true )
         telem.saveSpreadSheet();                                    // once done you cannot save more data there.
 }
```