**IRite Generator HERE IS A CHANGE**

***Marcus Brion, Scott Herscher, and Brian Ball***

Table of Contents

[General Overiew 2](#_Toc488757717)

[Running iRite Project 2](#_Toc488757718)

[Location of All Files 5](#_Toc488757719)

[All Files 5](#_Toc488757720)

[Runnable Jars 5](#_Toc488757721)

[NewSourceCode (Most up-to-date source code) 5](#_Toc488757723)

[Backup Code 5](#_Toc488757725)

[Pictures 5](#_Toc488757727)

[Templates 5](#_Toc488757728)

[Rollbacks 5](#_Toc488757730)

[How to Open Source Code in Eclipse 6](#_Toc488757731)

[Adding Additional Components 7](#_Toc488757732)

[How to Link/Add products 8](#_Toc488757733)

[Creating a new JAR 9](#_Toc488757734)

[General Overview of each Class 10](#_Toc488757735)

[*IRiteSource.java* 15](#_Toc488757736)

# General Overiew

This is a Java program developed at Rice Lake Weighing Systems® designed to be a code generator for the Specials Team. Users can generate all the necessary files for either a 920i or 1280 indicator or can copy and rename a standard product through the Git Enterprise repository.

## Running iRite Project

1. Start
   1. Select project name you would like the files to be named
      1. The SZ will not appear in the file names but is used in the .src file
   2. Select a project directory that the files will be saved to
      1. This should default to the user’s desktop if nothing is explicitly stated
   3. Select which indicator you would like code created for: 920i (default) or 1280
2. Components
   1. Select appropriate Hardware and Procedures/Methods in the Components Tab
      1. Dropdown boxes indicate which port will be used
      2. Text boxes indicate what name will be used
      3. Future components are place holders for future components to be added in
3. Products
   1. This page contains buttons that link to previously created programs
      1. These programs are pulled from Git through a batching program that Kyle Eggleston created
      2. DO NOT select any of these if you are creating a new program
      3. If selected, the generator will run one of Kyle’s batch files for that product
         1. Location of batch files: [Z:\Departments\58-Design Engineering\Specials Team\IRite-Generator\BatchLocation](file:///Z:\Departments\58-Design%20Engineering\Specials%20Team\IRite-Generator\BatchLocation)
4. Setup
   1. Creates variables, Setup Modes, Set Keys, User Keys, process Prompt Cancel, and nested/unnested PromptForEntrys in process data entry
      1. In the Enter Entry Modes enter the name of desired Setup mode
         1. The order entered here will determine the the order that the Set Keys will be created
         2. If you enter ‘Blank’, it will leave a SetKey blank for you
            1. To go to page 2 on the 1280 you need to generate blanks for lines 2-16 with the dedicated “Generate Blank Fields” button
      2. Select whether the variable will be a string, real, integer, or boolean
         1. Anything except one box is checked per Database entry, it will default to string
      3. Hit Generate Other Fields button on top (or Generate Blank Fields)
         1. This generates, as well as unlocks the Prompts and Variable name
            1. Example: Weigh Out – Prompt = Enter Weigh Out
            2. Example: Weigh Out – Variable = g\_(s,r, or I)WeighOut
            3. Example: Weigh Out – Setup mode = SetupWeighOutMode
      4. To nest any PromptForEntrys together, enter the row number/s separated by commas into the Nested Prompts column, and that one will be the first, and the rest will nest off of it
         1. Row numbers are the leftmost number on the left column
            1. The second set of numbers is the Setup mode number and the soft key it will be.
            2. Using ‘Blank’ or nesting will make these unreliable
         2. Make sure the row number/s being nested are greater than the current row \*\*\*Nesting shouldn’t work in V1.0\*\*\*
      5. Can change any of these fields but leave the Prompt Name untouched
         1. WARNING: Editing any of the prompt or variable fields and hitting Generate again will override prompt and variable fields based off Entry Modes Name
      6. Clear button will clear all fields
5. Add/Edit Database
   1. Will generate Setup modes, and initial PromptForEntrys for each database, as well as nesting any subsequent PromptForEntrys after the first for each database
   2. This page also adds the databases into the generated iRev or Revolution file

* \*\*\* The Revolution file has an issue where you need to open it in a text editor and copy and paste the database entries back into the file (anywhere) and resave for the databases to work properly \*\*\*
  + 1. In the database column, enter DatabaseName\_VariableName
       1. Example: Product\_TAG
       2. Nests everything with a similar Database Name
          1. Product\_TAG, Product\_ID, Product\_Weight are all nested
       3. Can handle multiple databases
    2. Select whether the variable will be a string, real or integer
       1. If anything except one box is checked per Database entry, it will default to string
    3. Hit Generate other Fields button on top to generate as well as unlocks the PromptForEntry prompt and variable name
       1. Example: Product\_TAG - prompt = Enter Product TAG
       2. Example: Product\_TAG - variable = ProductDB.TAG
    4. Can change any of these fields but leave the Database Name untouched
       1. WARNING: Editing any of the prompt or variable fields and hitting Generate again will override prompt and variable fields based off Database Name
    5. Clear button will clear all fields

1. Custom Prompt
   1. Similar to Databases, but nests multiple Custom prompts, as well as creating Variables, Setup Modes, and initial PromptForEntrys
      1. In the Prompt Name column enter the name of your custom Setupmode
         1. Nests everything unless user enters in a key word ‘done’ into Prompt Name
      2. Select whether the variable will be a string, real, or integer
         1. Anything except one box is checked per Database entry, it will default to string
      3. Hit Generate Other Fields button on top
      4. This generates, as well as unlocks the PromptForEntry prompt and variable name
         1. Example: Weigh In - Entered Prompt = Enter Weigh In
         2. Example: Weigh In - Variable Name = g\_(s,r, or i)WeighIn
         3. Example: Weigh In – Setup mode = SetupWeighInMode
      5. Users can change any of these fields but should leave the Prompt Name untouched
         1. WARNING: Editing any of the prompt or variable fields and hitting Generate again will override prompt and variable fields based off Prompt Name
      6. Clear button will clear all fields
2. Transaction Database
   1. This tab will generate a Transaction Database that will be added into the iRev or Revolution file. This Page functions closely to how the iRev or Revolution programs add a database
      1. In the Variable Name Column
         1. Enter names of additional database columns
      2. The Type Column
         1. Select the data type for that column
      3. The Size Column
         1. Depending on the data type, the user can alter the size of the data type
            1. This is only for *Fixed Length String* and *Variable Length String*
      4. The Records Column
         1. This field determines the number of records that the database will hold
      5. Hit Generate Other Fields button to generate the names into an array as well as lock in your entries
      6. Clear button will clear all fields
3. Clicking the Create Project button on any of these pages will immediately take the information entered in all tabs and create the project based on that information
   1. Fields left blank will be either blank in the .src or not included at all
   2. Hardware components selected will be included in the project as written in the template
   3. The Setup entries will be put into softkeys first and then followed by the databases.
      1. Database entries will be put in sequentially there after
4. Upon creation of the project, the user will have a folder created at the location specified in their project directory as well as a project.txt file created that is used in Kyle’s batch program
   1. A command prompt will pop up that is for Kyle’s batch program.

# Location of All Files

## All Files

[Z:\Departments\58-Design Engineering\Specials Team\IRite-Generator\IRiteProject](file:///Z:\Departments\58-Design%20Engineering\Specials%20Team\IRite-Generator\IRiteProject)

## Runnable Jars

[Z:\Departments\58-Design Engineering\Specials Team\IRite-Generator\IRiteProject\Functionalish Jars](file:///Z:\Departments\58-Design%20Engineering\Specials%20Team\IRite-Generator\IRiteProject\Functionalish%20Jars)

### Also contains a document about the updates made to the Jars for each version

## NewSourceCode (Most up-to-date source code)

### [Z:\Departments\58-Design Engineering\Specials Team\IRite-Generator\NewSourceCode](file:///Z:\Departments\58-Design%20Engineering\Specials%20Team\IRite-Generator\NewSourceCode)

* Used in Eclipse IDE primarily. Also functions in Netbeans IDE.

## Backup Code

[Z:\Departments\58-Design Engineering\Specials Team\IRite-Generator\NewSourceCode (old code, delete parenthesis to work with eclipse)](file:///Z:\Departments\58-Design%20Engineering\Specials%20Team\IRite-Generator\NewSourceCode%20(old%20code,%20delete%20parenthesis%20to%20work%20with%20eclipse))

## Backup code if things were to break at some point

## Pictures

[Z:\Departments\58-Design Engineering\Specials Team\IRite-Generator\IRiteProject\Pictures](file:///Z:\Departments\58-Design%20Engineering\Specials%20Team\IRite-Generator\IRiteProject\Pictures)

* Contains back up pictures (Most pictures are loaded internally in the program)

## Templates

### [Z:\Departments\58-Design Engineering\Specials Team\IRite-Generator\IRiteProject\NewTemplates](file:///Z:\Departments\58-Design%20Engineering\Specials%20Team\IRite-Generator\IRiteProject\NewTemplates)

* Templates parsed through to pull code for generator
* The alternate file “Templates” in the same folder contains the same templates as “NewTemplates”. This is a backup in case the templates get altered/destroyed

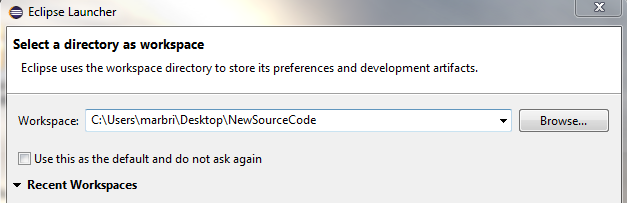
## Rollbacks

[Z:\Departments\58-Design Engineering\Specials Team\IRite-Generator\Rollbacks.zip](../Rollbacks.zip)

* A rollback file in case everything blows up for some reason

# How to Open Source Code in Eclipse

1. Locate newest version of source code in Z drive (NewSourceCode)
   1. [Z:\Departments\58-Design Engineering\Specials Team\IRite-Generator](file:///Z:\Departments\58-Design%20Engineering\Specials%20Team\IRite-Generator\IRiteProject\NewTemplates)
2. Copy folder to desired workspace (Most likely desktop)
3. Download most current version of Eclipse Java Neon (Most current version at time of writing Neon.3 Release (4.6.3) ) or IDE of your choice
4. Open eclipse and select NewSourceCode as the workspace



1. Once finished with editing code
   1. Export code into a new runnable .jar file
   2. Update the Functionalish Jars folder at: [Z:\Departments\58-Design Engineering\Specials Team\IRite-Generator\IRiteProject\Functionalish Jars](file:///Z:\Departments\58-Design%20Engineering\Specials%20Team\IRite-Generator\IRiteProject\Functionalish%20Jars)
   3. Update iRiteProject Updates word document (in same location as above) with the changes made
   4. Save the updated source code back into the Z drive (NewSourceCode)

# Adding Additional Components

1. Go to Feature.java class in Eclipse. ProjectCreator/src/com.rlws.iRiceProject.Template/Feature
   1. You will see a long list of features(enums) on the top
2. If you want to use a drop down list or textbox, go to #3, otherwise go to 4
3. Use drop down for port numbers or a textbox
   1. 4 fields/parameters 
      * Creation of JCheckBox
        + NameofFeature(new JCheckBox(“Name of Feature”),
      * Name of the feature for the token, current format uses all lowercase and one word(avoid symbols and space)
        + “nameoffeature”,
      * Name of variable how you spelled the enum(One word, capitalize the first letter of each word)
        + Variable.NameofFeature,
      * Category: this determines which column it shows up in on the main screen. Enter “Uno”, “Dos”, “Tres”, or “Quatro” for which row it needs to be in
      * Continue to step 5
4. Do not use drop down for port number
   1. 3 Fields/parameters



* + - Creation of JCheckBox
      * NameofFeature(new JCheckBox(“Name of Feature”),
    - Name of the feature, current format uses all lowercase and one word(avoid symbols)
      * “name of feature”,
    - Category: this determines which column it shows up in on the main screen. Enter either “Uno”, “Dos”, “Tres”, or “Quatro”
      * “Uno”, “Dos”, “Tres”, or “Quatro”)
    - ***Finished*** you do not need to continue with 5 or 6

1. Go to Variables class ProjectCreator/src/com.rlws.iRiceProject.Template/Variables



* 1. Write the name of enum(variable), one word, uppercase first letter of each word
     + NameOfVariable(
  2. Three parameters in parenthesis
     + Name: lowercase, all one word(same as step a: except all lowercase)
       - “nameofvariable”
     + The Type: ‘Type.’ Plus step a (Capitalize first letter of each word, all one word)
       - Type.NameOfVariable,
     + Configurable: Just put true
       - true),

1. Go to Types ProjectCreator/src/com.rlws.iRiceProject.Template/Types



* 1. This determines whether each selectable method has a Textfield(display list) or a combo box you can choose port numbers for(ex. IO Test, VFD display)
  2. Name of type: One word, capitalize first letter of each word
     + NameOfType(
     + Whether you want it to be a combo box, or text field
       - new JComboBox<String>(PORTNUMBERS), “dropdown”,
       - new JTextField("Initial text in box"), “textbox”,
  3. Category: “Uno”, “Dos”, “Tres”, or “Quatro” for the row number

# How to Link/Add products

1. Standard products are located in [Z:\Departments\58-Design Engineering\Specials Team\Standard Folders For Generator](file:///Z:\Departments\58-Design%20Engineering\Specials%20Team\Standard%20Folders%20For%20Generator)
   * + Bat files located at [Z:\Departments\58-Design Engineering\Specials Team\IRite-Generator\BatchLocation](file:///Z:\Departments\58-Design%20Engineering\Specials%20Team\IRite-Generator\BatchLocation)
   1. Just drag any new products from vault that you wish to add to the folder
      1. Make sure to remove all internal folders (source/standard/custom…) and just have the files directly inside the product folder
2. Open Eclipse
   1. Open ProductButton.java in com.rlws.iRiceProject.template
   2. Again add a new Product enum like with Hardware or Methods/Procedures following the ProductButton syntax



* 1. Name it what you want on the front, follow format of capitalize first letter of each word, one word, and not symbols
     1. Most of the address is already hardcoded in a different class, so this just needs the folder name to complete the address
  2. In Parenthesis:
     1. new JButton(“Button Text”)
     2. in Quotes, the exact name of the bat file you wish to run which will copy the Standard Folders for that product for the Generator
     3. then “Product”
     4. Remember to end every enum with a comma after closing parenthesis, and have last one end with a semi colon instead

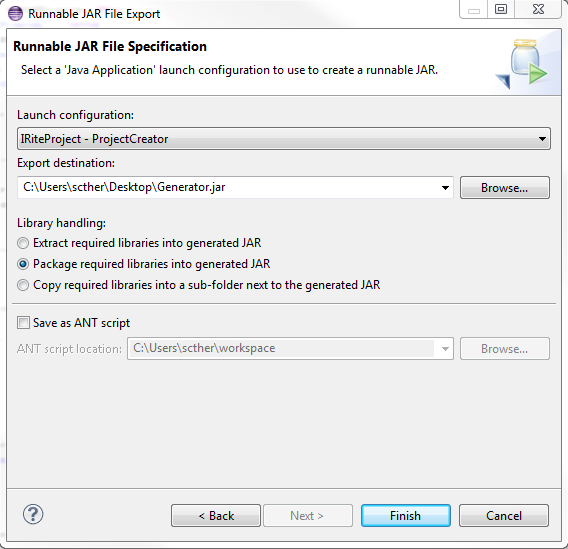
1. Go to the bat file location and create the appropriate bat file for the added product
   1. The bat files will go out to Git and copy the standard product into your generated project directory

\*\*\*To change hardcoded address of all the Product folders\*\*\*

1. Open Products.java in com.rlws.iRiteProject.project

* 1. At the top where variables are declared located private String partialAddress
  2. Change that address to wherever the new destination will be
     1. Use double slashes \\ because single slash is an ‘escape’ character in java and will not work

# Creating a new JAR

1. Open in Eclipse
2. Right click on ProjectCreator folder
   1. Click on Export
   2. Under java folder
      1. Click Runnable JAR file
   3. Click next
3. Set destination config
   1. Launch configuration at top: set to IRiteProject – ProjectCreator
   2. Export destination:
      1. Set it here, most likely set for desktop + \*JAR name*
   3. Library Handling: Check - Package required libraries into generated JAR (middle one)
   4. Leave the Save as ANT Script unchecked
   5. Box may pop up saying compiling errors occurred, just hit OK
4. Move to Functional JAR folder in Z Drive
   1. Update JAR updates word doc

# General Overview of each Class

#### com.rlws.iRiceProject.template

1. *CustomPromptMethods.java*
   1. Constructing class
      1. Used for generating Prompt entries for Custom Prompts tabs
2. *DataBase.java*
   1. Constructing class
      1. Used for storing data for DataBase and Custom Prompts tabs
3. *DatabaseMethods.java*
   1. Constructing class
      1. Used for generating Database entries for Database tab
      2. Creates database softkeys
4. *EntryMode.java*
   1. Constructing class
      1. Used for storing data in Setup Modes tab
5. *Feature.java*
   1. Enum class
      1. Contains names and fields of components (Hardware and Methods/Procedures)
      2. Where you begin if you want to edit/delete/add any components
6. *IRev.java*
   1. Template generator
      1. Copies and renames generic .920 file when a new project is created or .rev for 1280’s
7. *Iri.java*
   1. Template generator
      1. Copies and renames .iri file that only assists in compiling .src file initially
      2. Can delete after obtaining the work order .iri file
8. *IRiteSource.java*
   1. Template generator
      1. Copies/Generates all the code for the .src file
      2. Generates setup modes, process data entries, userkeys, set keys, global variables, process prompt cancel, refresh setup display, database nesting prompts, and custom nesting prompts
      3. See [IRiteSource.java](#_IRiteSource.java) section for more details
9. *Manual.java*
   1. Template generator
      1. Copies and renames generic word doc file
      2. Not being used in most current version
10. *MoveTemplate.java*
    1. Abstract Class
       1. Helps transfer of templates
       2. Don’t worry about this class
11. *ProductButton.java*
    1. Enum class
       1. Contains all the fields for the Products that are copied from Standard Folders for Generator
       2. Buttons are located on Products tabs
12. *Quote.java*
    1. Template generator assisting class
       1. Generates an address where files and directories can be created or copied to
13. *ReadAndWriteTemplate.java*
    1. Abstract Class
       1. Helps with the transferring of template data
       2. Don’t worry about this class
14. *SetupModesMethods.java*
    1. Constructing class
       1. Generates Setup Modes and adds softkeys
       2. Some database things are handled in here also
15. *Transaction.java*
    1. Constructing class
       1. Used for storing info about the Transaction database
16. *Transactions.java*
    1. Enumerated class
       1. This class holds the info about the data types for the Transaction tab
       2. Defines the Name, Number, Type, and Size of the data type
17. *Types.java*
    1. Enum Class:
       1. Contains fields for the Hardware and Methods/Procedures
       2. Determines whether said features are dropdowns or text fields
18. *Variables.java*
    1. Enum Class:
       1. Contains more fields for the Hardware and Methods/Procedures
       2. Determines the name of the variables, and what the type name is

#### com.rlws.iRiteProject.core

1. *IRiteProject.java*
   1. Driver class
      1. Beginning of the whole program
      2. Creates the new Display that sets up the tabbed panes
      3. Can change the *LookAndFeel* in the Main method here

#### com.rlws.iRiteProject.gui

1. *AuthorsPageButtonListener.java*
   1. Button Listening Class
      1. This class is dedicated as a button listener for the Rice Lake Weighing System logo built in button
      2. This brings up a new display with Author information and the current Version number
2. *ComponentDisplay.java*
   1. Display Class
      1. Creates and controls the Component tab in the generator and deals with all the actions that take place within it
      2. Creates all the fields for selection including: checkboxes, textboxes, dropdown menus, and text fields
3. *CreateBlankButton.java*
   1. Button Listening Class
      1. This class is for listening to when the create button gets selected
      2. This grabs all the information from the many of the classes and starts the project creation cycle
      3. This class also handles the progress bar that pops up when the button gets selected
      4. \*\*\* CreateBlankButton is misleading. It was named like this because it was a repurposed class from an older version
4. *CreateProductButton.java*
   1. Button Listening Class
      1. This class is for listening to a Create Product Button
         1. This is not implemented anywhere and could be removed in the future
5. *CustomPromptDisplay.java*
   1. Display Class
      1. Creates and controls the Custom Prompt tab in the generator and deals with all actions that take place within it
      2. Stores all the data located in its fields as an ArrayList<Database> called promptArray
6. *DatabasePromptDisplay.java*
   1. Display Class
      1. Creates and controls the Database tab in the generator and deals with all actions that take place within it
      2. Stores all the data located in its fields as an ArrayList<Database> called dataArray
7. *Display.java*
   1. Display creating class
      1. Creates the tabbed panes and displays for the generator GUI
      2. Has leftover code for scrolled panes that were previously used in past versions
8. *Displays.java*
   1. Abstract Class
      1. Don’t need to worry about
9. *HelpPageListener.java*
   1. Action Listening Class
      1. Class that handles when the help menu option is selected
      2. Creates a JDialog Panel with the tabbed panes
10. *LasherButtonListener.java*
    1. Button Listening Class
       1. Class to give praise to Logan the Lasher (YouTube him to meet Kyle’s younger self)
       2. This is hidden in the AuthorsPage Display.
          1. You need to click just to the left of the Rice Lake Weighing Systems Logo to activate the button (it’s a small invisible button)
11. *LasherIcon.java*
    1. Display Helping Class
       1. Used in the creation of the Lasher Appreciation Gallery
12. *MainDisplay.java*
    1. Display Class
       1. Creates the Start Display screen for the generator and deals with all the naming, project directory, and template selection
       2. \*\*\* Misleading naming for this class is because it is a leftover class that was repurposed
       3. \*\*\* To fully encapsulate the program into 1 file you will need to replace the template links to the resources package in ProjectCreator/src/resources
13. *MissingIcon.java*
    1. Display Class
       1. Used to help with the creation of the Lasher Appreciation Gallery
14. *ProductDisplay.java*
    1. Display Class
       1. Creates and controls the Products tab in the generator and deals with all the actions that take place within it
       2. Creates all the buttons based on the Product Enumerated class variables
15. *ProgressDialog.java*
    1. Unsure if used
16. *SetUpDisplay.java*
    1. Display Class
       1. Creates and controls the Setup tab in the generator and deals with all the actions that take place within it
       2. Stores all the info located in its fields as an ArrayList<EntryMode> called entryArray
17. *TransDatabaseDisplay.java*
    1. Display Class
       1. Creates and controls the Transaction tab in the generator and deals with all the actions that take place within it
          1. Stores all the info located in its fields as an ArrayList<Transaction> called transArray

#### com.rlws.iRiteProject.project

1. *NewProject.java*
   1. Project Creator
      1. Creates a project using an order number/name
      2. This is called to make a new Project, and copy over .src .iri .920 or .rev… files
      3. Also handles if a Product Button is selected and will run the appropriate batch file
2. *Products.java*
   1. Product copy
      1. Copies over a standard product from [Z:\Departments\58-Design Engineering\Specials Team\Standard Folders For Generator](file:///Z:\Departments\58-Design%20Engineering\Specials%20Team\Standard%20Folders%20For%20Generator)
3. *Project.java*
   1. Project Creator
      1. Creates a new Project using customer name and order number from Job list
   2. I don’t think this is ever used in the program anymore since the removal of the corresponding button
4. *ProjectMod.java*
   1. Not used at all; could delete
5. *RevEditor.java*
   1. File editor
      1. This class is used to alter the .920 or .rev files so that databases can be added automatically
      2. \*\*\* There is a known issue for the .rev files where you need to open them in a text editor and copy and paste the databases anywhere and re-save them \*\*\*
6. *RunCode.java*
   1. Template helper
      1. This file is used to help with the naming of the project files and other variables used in the .src and template parsing

#### com.rlws.iRiteProject.settings

1. *Settings.java*
   1. Preference Class
      1. Deals with the preferences from the Settings page that stores Template directory/Created project destination

#### com.rlws.iRiteProject.url

Whole class deals with going online and finding what is in user’s portal

\*Unsure what is used in this package

1. *IntQuotes.java*
   1. I believe this class tries to find the .src file
2. *IntWorkOrder.java*
3. *ProjectLog.java*
4. *Vault.java*
5. *WebSource.java*

#### com.rlws.bball.projectCreate.com

\*Unsure what any of these do. May be able to delete in future.

1. *Project.java*
2. *ProjectCreate.java*
3. *ProjectCreateGUI.java*
4. *ProjectCreateSettings.java*
5. *ProjectType.java*

## *IRiteSource.java*

1. IRiteSource has 4 ArrayLists objects passed into it
   1. features, array, dataArray, and promptArray
   2. The objects contain information such as name, variable, prompt, and nesting options
2. IRiteSource looks for special Tokens in the .src template to find where to add Setup modes, variables, etc…
   1. Tokens are comments in the Template, that the user can choose
      * Ex. --$(SomeVariable) or $$$
3. From there it checks each ArrayList to see if it is empty, and if not searches for each special Token as it passes through the Template
4. If a Token is found, it calls a method to generate the code
5. These methods are found in 1 of 3 ‘Method Classes’ based on which ArrayList it pertains to
   1. DatabaseMethods for Databases
   2. CustomPromptMethods for CustomPrompts
   3. SetupModesMethods for Setup Modes
   4. Uses dataArray to get access to DataBase objects
   5. Also creates dataNameArray, which is the name field of the DataBase object, but without spaces in the name
      * Arrays are the same size
   6. Iterates through the arrays generating code and inserting variables where needed
      * Reference comments in code
6. For CustomPromptMethods
   1. Uses promptArray to get access to DataBase objects that are used to create all the Setup Modes, and nested prompts
   2. Iterates through the promptArray and generates code while putting variables in the required places
      * Reference comments in code
7. For SetupModesMethods
   1. Uses array to get access to EntryMode objects that are used to create SetKeys, UserKeys, EntryModes, process prompt cancel, variables, and nested prompts
   2. Also creates numberArray , noSpaceNameArray, and nestedNumbersArray
      * Main ones, and numberArray, noSpaceNameArray, and array are all same size
   3. numberArray is an ArrayList<Integer> and the ints correspond to what number each entry mode is
      * numberArray is assigned a number at position ‘i’, which corresponds to the Setup mode number of the name of noSpaceNameArray at position ‘i’
   4. noSpaceNameArray is created because the name of each EntryMode has spaces, and noSpaceNameArray removes the spaces because it is applied to a setup mode name
   5. nestedNumbersArray is an ArrayList of all the row numbers that might be nested under a certain EntryMode

HERE IS ANOTHER CHANGE