2023 Jankster New Project Settings

Overview

The Janksters software build and source code organization make for a few requirements in order to build properly while keeping common files in common areas. Year after year, The Janksters have had two areas of commonality. Multi-year commonality comes in the form of the so-called *jankyLib* library. This library is utilized in multiple years and therefore must sit 'higher' in the directory structure. Within a given year, there are common classes and headers which are developed in a little experimental group that need to be used by the main project as the season begins to get to integration / convergence phase. These different common files exist in the following type of hierarchy:

`/code-java/`

`/code-java/jankyLib/` (multi-year common code area)

`/code-java/<year>/`

`/code-java/<year>/CommonClassesThisYear/` (single year common code area)

`/code-java/<year>/<projectname>`

The Problem

FIRST and WPI have created a gradle-based build environment that is used by a VS Code IDE. The desire is to make sure the team has easy visual access to all the code they may be using or modifying and have the build system support those files where they live. It is a requirement (from me) that source files do not simply get copied from one location to another to be "re-used." This will create splinters of versions of code and bugs that won't be well tracked or tolerated.

Prior to the 2023 season, there was a solution to these troubles which required a new project be modified once and once only. This solution worked well on Mac computers with symbolic links, but it didn't work on all Windows systems and so it was problematic.

2023 Solution

After several discussions with the WPI folks, we have an improved method to facilitate our requirements. This solution differs significantly from the prior solution. This solution uses gradle's *build.gradle* file to add the common areas as build/source areas. This solves the building issue and allowing those files to remain in a single place. The other requirement is to have those common files available to the students, visually, in the VS Code editor. This is facilitated by adding two folders to the project. However, these folders will only be preserved as

part of the project if the "Workspace" feature is used. This means a new project would be saved using File->Save Workspace As... which will create a file that ends in *.code-workspace*

Detail for build.gradle

Any WPI project that is created will have a build.gradle file in the project folder. To add support for our common file areas, one must edit the file and after the line reading:

```
def ROBOT_MAIN_CLASS = "frc.robot.Main"
```

the user (or script) must add the following lines:

```
sourceSets {
  main {
    java {
      srcDir "../../jankyLib"
      srcDir "../CommonClassesThisYear"
    }
  }
}
```

Detail for adding folders to the workspace

VS Code allows a project to be 'opened' simply by opening the folder in which it lives. It also allows for the concept of a workspace file for the project as well. This file has a long extension of .code-workspace

The user can facilitate the addition of the common folders into VS Code by using the File->Add Folder To Workspace... and finding the two folders which are "up on directory level" and "up two directory levels" for *CommonClassesThisYear/* and *jankyLib/* respectively. Once these folders are added, the workspace will need to be saved using File->Save Workspace As...
The resulting .code-workspace file is generally the following:

```
{
  "folders": [
        { "path": "." },
        { "path": "../CommonClassesThisYear" },
        { "path": "../../jankyLib" }
  ],
  "settings": {
        "java.configuration.updateBuildConfiguration": "automatic",
        "java.server.launchMode": "Standard"
  }
}
```

Setup Automation

There is an automation script which will work with Macs in /code-java/2023/setupNewProject.sh. After setting up a new WPILIB project, this script shoudl be run once in order to modify the project to support our common build files.
Usage for a project whos name is "JankstersTest" and who's project is found in /code-java/2023/JankstersTest/ :

```
cd code-java/2023
./setupNewProject JankstersTest
```

It will do the following and have the following limitations:

- It will modify the build.gradle file to add a block of `sourceSets{}` which will automatically build our CommonClassesThisYear/ and jankyLib/ files.
- So long as there is no JankstersTest.code-workspace file (this will only be created if the user does a File->Save Workspace As...), the script will create this file. The purpose of this file is to have the CommonClassesThisYear/ and jankyLib/ folders conveniently located in the workspace on the left side so that coders can easily edit those files which are not in the local project space.
- In terms of limitations, the script is pretty smart about handling the most important build.gradle file. However, it cannot easily *modify* an existing .code-workspace file and so it will simply warn the user than it cannot modify the file.

Once this script is ran, the project should be opened by double-clicking on the `.code-workspace` file or doing a File->Open Workspace from file... -- rather than doing a File->Open Folder... If only the folder is opened, the project *will build just fine*, however the workspace folders won't be shown. So, it's not a big "miss" or "problem" if only the folder is opened - it just means editing of common files won't be easy to the user.