**UBC SOLAR**

**Software Bootcamp**

**Part 1: Install software**

1. Eclipse: Go to [this website](http://www.eclipse.org/downloads/packages/eclipse-ide-java-developers/marsr) and use the download links on the right to install the Eclipse Java IDE, if you don’t already have it installed
2. GitHub: If you don’t already have a GitHub account, go to [this website](https://github.com/) and create one
3. SourceTree: If you don’t already have a Git client, go to [this website](https://www.sourcetreeapp.com/), download SourceTree and connect your Git account. Even if you do already have a Git client, I would recommend using SourceTree. During the SourceTree setup, when it asks you for a repository to connect to, use this address: <https://github.com/EmceeEscher/Solar-Simulator-Bootcamp>
4. WindowBuilder: If you DIDN’T download Eclipse using the link in step 1, go [here](https://eclipse.org/windowbuilder/) and install the WindowBuilder plugin for Eclipse. The Eclipse download link in step 1 includes WindowBuilder, so if you used that, you can skip this step.

**Part 2: Open the simulator**

1. Open Eclipse. Under File, click Import. Under the General tab, select “Existing Projects into Workspace”, and click Next. On the next page, next to “Select root directory:”, browse to the folder where you cloned the Solar-Simulator-Bootcamp directory and select it. In the window, select “StrategySystem” and then click Finish. In Eclipse, in the Package Explorer on the right, you should see an expandable folder labeled “StrategySystem.” Expand it, and look under “src” to find the code used in the simulation. Open files, and look around the code to see what it looks like.
2. To run the simulator, open the Main package, and then open Main.java and click the run button (green circle with white arrow) in the upper Quick Access toolbar.
3. In the simulator, try to do the following things:  
   -Load a route in the map window.  
   -Load some weather forecasts in the weather window.  
   -Add a location report through the debug menu.  
   -Add a car telemetry packet through the debug menu.  
   -Run a simulation.  
   -Add a custom weather forecast through the debug menu.

**Part 3: Change things**

1. Try to change the message that appears when you load a route for the first time in the map window. This will require you to dig through the code and try to figure out how some of this stuff works. Don’t move on to the next part until you’ve figured out how to change that message. Have fun!

**Part 4: Dealing with git…**

**(this is how you share your changes with the rest of the team)**

(These instructions will assume you’re using SourceTree)

1. First, create a new branch. Click the branch button near the top, and name your branch something unique (your name would be a good choice).
2. Next, commit your changes. Click the commit button at the top. On the commit screen, select the files you changed and click “Stage selected”. Then add an informative commit message and commit.
3. After that, push your changes. Click the push button, and select the box next to the name of your branch in the window that pops up. Click ok, and your changes will be pushed to the remote server that hosts all our code.
4. Now it’s time to switch branches. Scroll through the commit messages in SourceTree until you find a line that starts with “origin/merge-practice”. Right-click that line, and select checkout. You will now be making changes to the merge-practice branch.
5. Now it’s time to merge your changes! (dun dun dunnnnnn) Click the merge button at the top, and scroll through the commits until you find the commit that you made on your own branch. Double-click that line to choose that branch to merge.
6. You should get a merge error! These are fun (no, they’re not). Go back to Eclipse, and open the file that you made your changes in. You should see something like the following:  
     
   <<<<< HEAD  
     
   [not your code]  
     
   =============  
     
   [your code]  
     
   >>>>> [your branch name]  
     
   The section between “<<<<< HEAD” and “=======” is the code that is currently on the merge-practice branch. The other section is the code that is on your branch. They are different, so git complains. To fix it, delete the code you don’t want, and leave the code that you do (for now, replace the old message with your new message). Make sure that the program still compiles and runs, and that the message shown is the one you want. After that, commit and push your changes again.
7. That’s it! You now understand how to use git (hahahaha).