Documentation and Notes

For

General Telescope Control GUI

John Tran and Emily Cook

Last Updated: 8/10/16

Mentor: Waid Schlaegel

Table of Contents

Introduction

The main reason this software was developed was to have a general telescope control GUI that can easily be used and portable across multiple operating systems such as Windows and Linux. It is designed to be able to control the iOptron Minitower Pro mount. The program was written in Java using the jssc library, a third party library. It includes features such as time, positional control, and display of current position. The program was initially developed in Linux, Ubuntu 16.04.1 LTS so it is advised that you do the same. Then test and debug on newer and different operating systems such as Windows or Ubuntu 18. Currently it is incomplete, however it is hoped that other features will be added by future scholars as time permits. The development of this software was used primarily using the command line and a text editor. Not a single ide.

Usage

To use this program in its current state, you must:

1. Download and install the Java Development Kit (jdk). You can download it from the Oracle website. On Windows, you'll have to set the path in the environment variables to the jdk. Google this. On Linux, it's just a few simple install commands you can also Google. This will allow you to use javac and java commands for development. Make a simple hello world program to see if "javac HelloWorld.java" and "java HelloWorld" works before continuing. Of course if you already have the jdk, then you can skip this step.
2. Download and install the jssc library (allows for serial COM), Google it if the link is broken https://code.google.com/archive/p/java-simple-serial-connector/

On Linux, you have to add the jssc.jar to the ext file (or something similar) In my case:

/usr/lib/jvm/java-1.8.0-openjdk-amd64/jre/lib/ext

On Windows, you have to add the jssc.jar to the ext file (or something similar) In my case:

C:\Program Files\Java\jdk1.8.0_60\jre\lib\ext

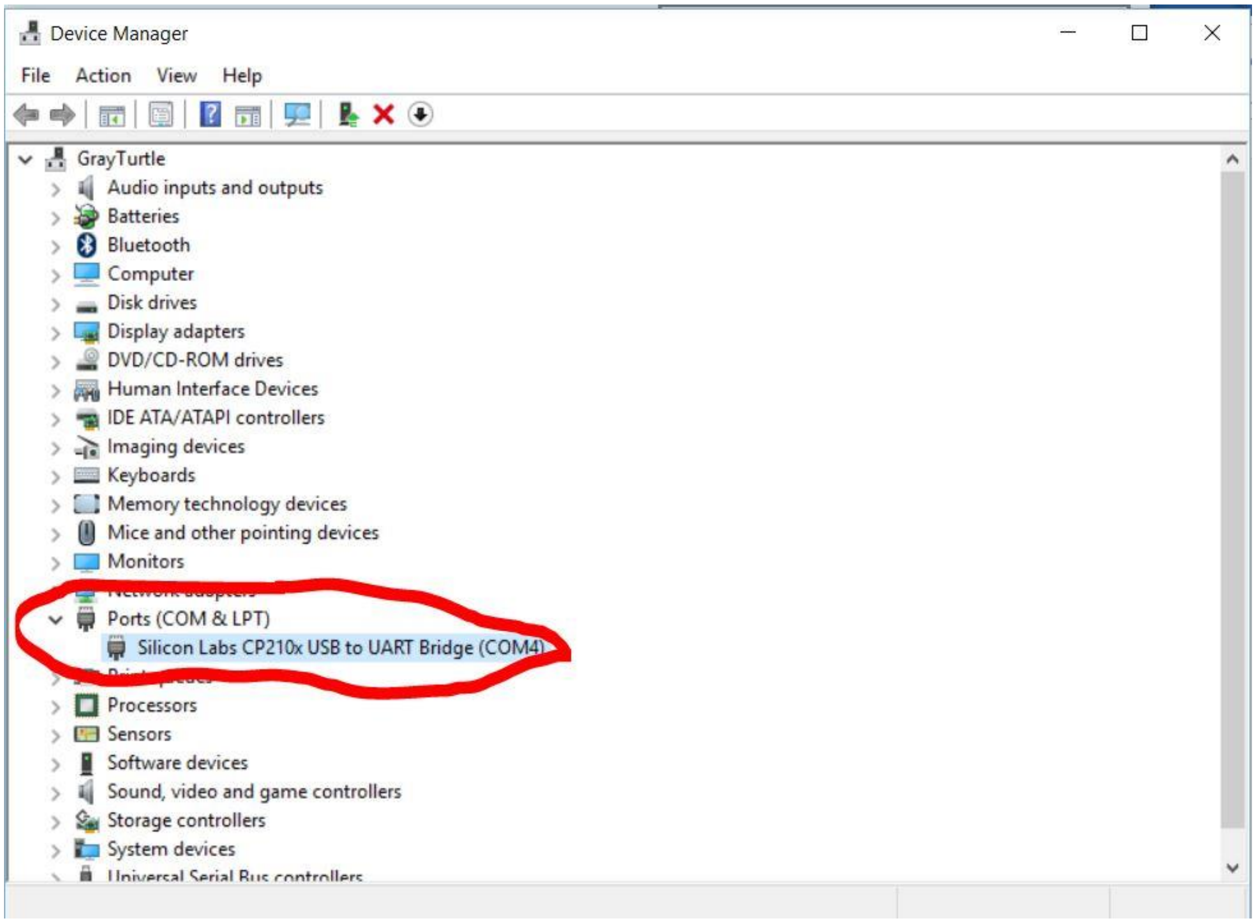
3. Plug in the telescope mount
4. Go to the TelescopeGUI folder
5. If using Windows, you'll have to run this command every time since jssc.jar is a 3rd party library.

   set classpath=path to jssc.jar*;*

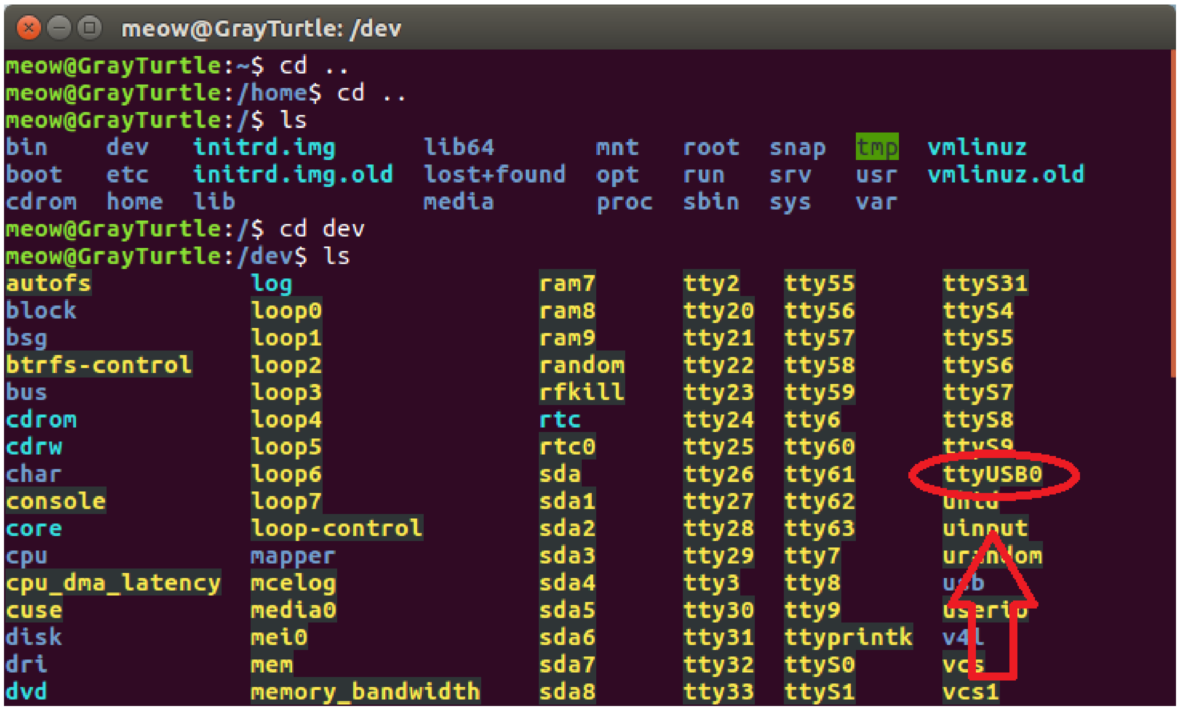
   example:

   set classpath=C:\Program Files\Java\jdk1.8.0_60\jre\lib\ext\jssc.jar;
6. Run *javac Tele.java* then *java Tele*. This should run the GUI.
7. Check to see what port the mount shows up on.

   If on Windows: Go to the device manager and go to Ports or whatever pops up that is similar. You may have to update your device driver for this to communicate properly.

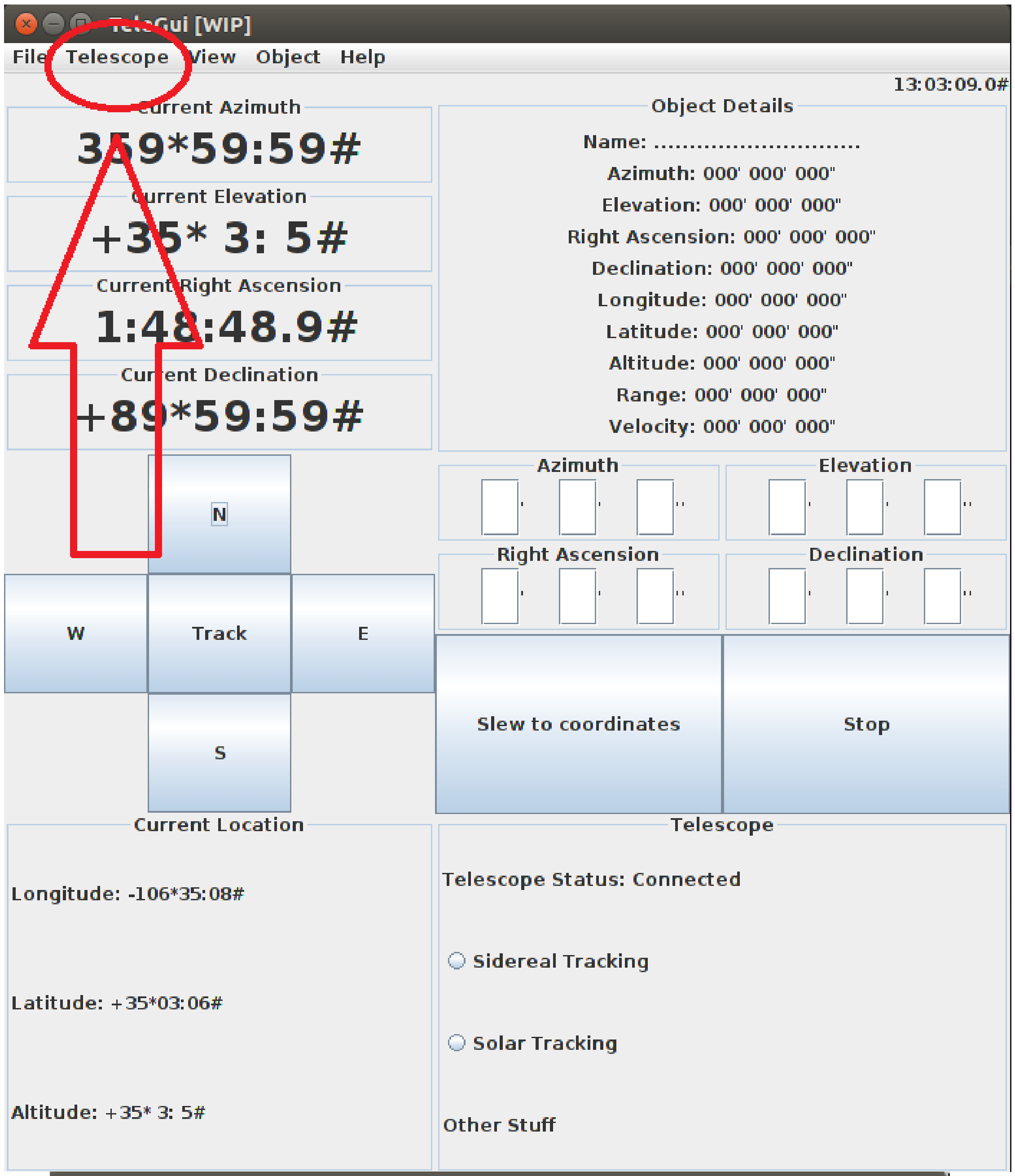


If on Linux: Go to your /dev/ in your root folder and look for the ttyUSB#



```
meow@GrayTurtle:~$ cd ..
meow@GrayTurtle:/home$ cd ..
meow@GrayTurtle:/$ ls
bin    dev   initrd.img      lib64       mnt   root  snap  tmp  vmlinuz
boot   etc   initrd.img.old  lost+found  opt   run   srv   usr  vmlinuz.old
cdrom  home  lib             media       proc  sbin  sys   var
meow@GrayTurtle:/$ cd dev
meow@GrayTurtle:/dev$ ls
autofs           log                 ram7      tty2   tty55      ttyS31
block            loop0               ram8      tty20  tty56      ttyS4
bsg              loop1               ram9      tty21  tty57      ttyS5
btrfs-control    loop2               random    tty22  tty58      ttyS6
bus              loop3               rfkill    tty23  tty59      ttyS7
cdrom            loop4               rtc       tty24  tty6       ttyS8
cdrw             loop5               rtc0      tty25  tty60      ttyS9
char             loop6               sda       tty26  tty61      ttyUSB0
console          loop7               sda1      tty27  tty62      uhid
core             loop-control        sda2      tty28  tty63      uinput
cpu              mapper              sda3      tty29  tty7       urandom
cpu_dma_latency  mcelog              sda4      tty3   tty8       usb
cuse             media0              sda5      tty30  tty9       userio
disk             mei0                sda6      tty31  ttyprintk  v4l
dri              mem                 sda7      tty32  ttyS0      vcs
dvd              memory_bandwidth    sda8      tty33  ttyS1      vcs1
```

8. Go to "Telescope" in the menu of the frame and choose the appropriate port.



TeleGui [WIP]

File Telescope View Object Help

13:03:09.0#

Current Azimuth
359*59:59#

Current Elevation
+35* 3: 5#

Current Right Ascension
1:48:48.9#

Current Declination
+89*59:59#

Object Details

Name: ....

Azimuth: 000' 000' 000"

Elevation: 000' 000' 000"

Right Ascension: 000' 000' 000"

Declination: 000' 000' 000"

Longitude: 000' 000' 000"

Latitude: 000' 000' 000"

Altitude: 000' 000' 000"

Range: 000' 000' 000"

Velocity: 000' 000' 000"

Current Location

Longitude: -106*35:08#

Latitude: +35*03:06#

Altitude: +35* 3: 5#

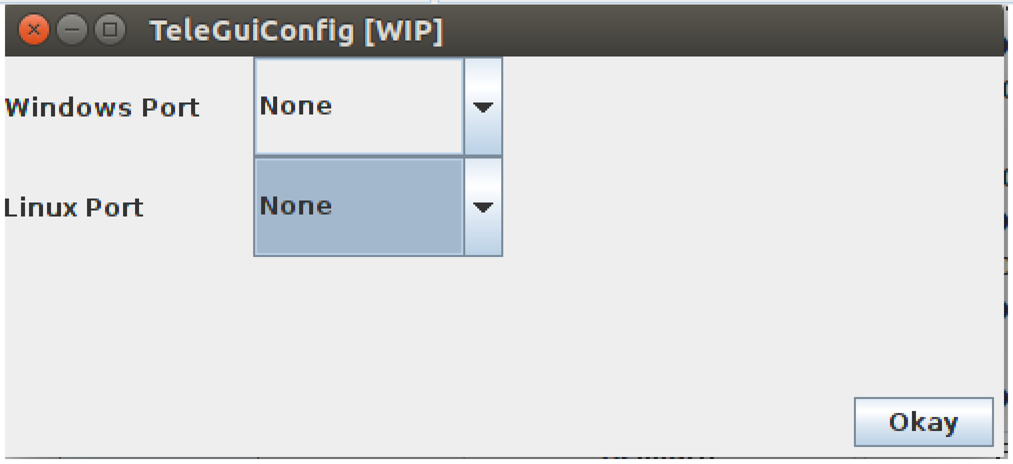
Telescope

Telescope Status: Connected

Sidereal Tracking

Solar Tracking

Other Stuff



TeleGuiConfig [WIP]

Windows Port: None

Linux Port: None

9. The GUI should be working now.

10. In order to send more commands in the code such as change the tracking rate to sidereal, refer to the page 42 in the PDF of the mount manual in this folder. TelePort.java is where the commands are being sent to the mount.

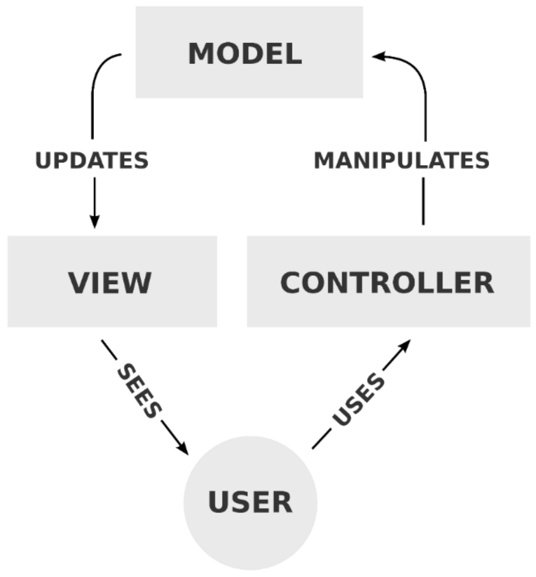
Organization

The program is organized in model view controller architecture.

“Model–view–controller (MVC) is a software architectural pattern for implementing user interfaces on computers. It divides a given software application into three interconnected parts, so as to separate internal representations of information from the ways that information is presented to or accepted from the user.

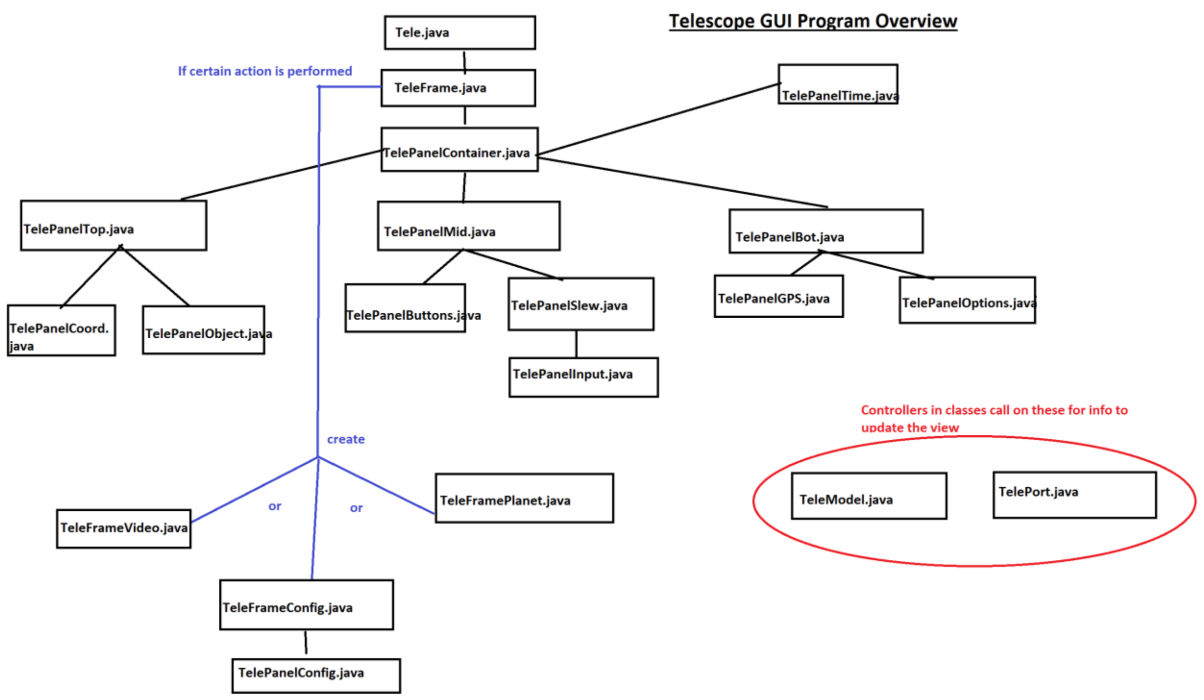
- A *model* stores data that is retrieved according to commands from the controller and displayed in the view.
- A *view* generates new output to the user based on changes in the model.
- A *controller* can send commands to the model to update the model's state (e.g. editing a document). It can also send commands to its associated view to change the view's presentation of the model (e.g. by scrolling through a document)” (Wikipedia).

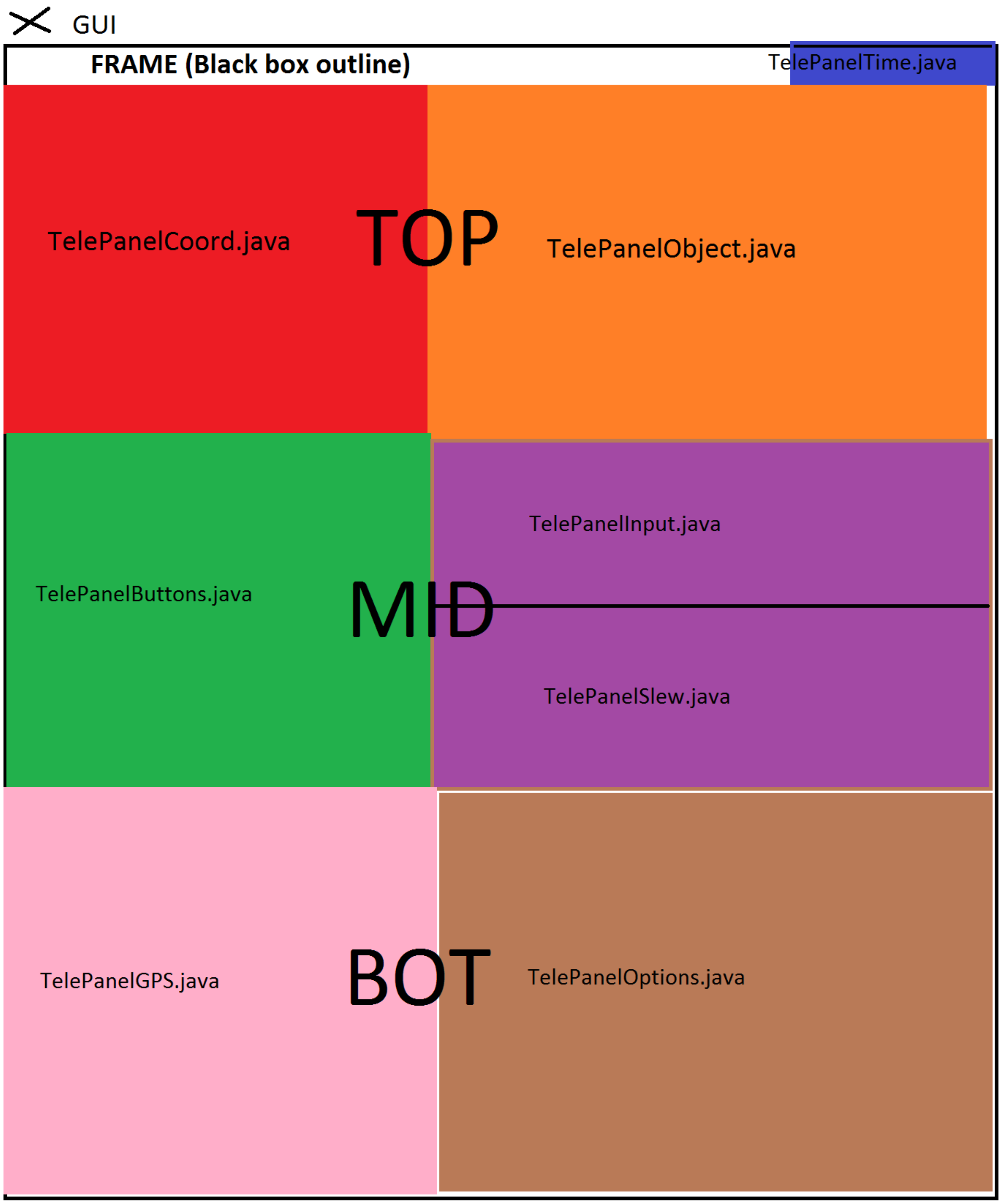
TeleModel.java and TelePort.java are purely for calculations and interpretation aka the model. The rest of the classes are views and controllers.



The program is also organized in such a way where there are panels within panels within panels. The Layouts are mainly GridBagLayout and GridLayout. Here's how the GUI creates itself:

1. The main method in Tele.java creates TeleFrame.java which creates and adds TelePanelContainer.java (this will contain all of the components).
2. TelePanelContainer.java creates and adds TelePanelTime.java, TelePanelTop.java, TelePanelMid.java, and TelePanelBot.java (the clock and TOP, MID, BOT sections of the GUI).
3. TelePanelTop.java creates and adds TelePanelCoord.java and TelePanelObject.java
4. TelePanelMid.java creates and adds TelePanelButtons.java and TelePanelSlew.java
5. TelePanelSlew.java also adds in TelePanelInput.java
6. TelePanelBot.java creates and adds TelePanelGPS.java and TelePanelOptions.java
7. Once the "Configure Telescope" option under "Telescope" in the frame menu is clicked, it creates TeleFrameConfig.java and that code adds TelePanelConfig.java.
8. When the "Video" option under "View" in the frame menu is clicked, it creates TeleFrameVideo.java.
9. When the "Planetarium" option under "View" in the frame menu is clicked, it creates TeleFramePlanet.java


Telescope GUI Program Overview
Tele.java
TeleFrame.java
If certain action is performed
TelePanelContainer.java
TelePanelTime.java
TelePanelTop.java
TelePanelMid.java
TelePanelBot.java
TelePanelCoord.java
TelePanelObject.java
TelePanelButtons.java
TelePanelSlew.java
TelePanelInput.java
TelePanelGPS.java
TelePanelOptions.java
create
TeleFrameVideo.java
or
or
TeleFramePlanet.java
TeleFrameConfig.java
TelePanelConfig.java
Controllers in classes call on these for info to update the view
TeleModel.java
TelePort.java


GUI
FRAME (Black box outline)
TelePanelTime.java
TelePanelCoord.java
TOP
TelePanelObject.java
TelePanelInput.java
TelePanelButtons.java
MID
TelePanelSlew.java
TelePanelGPS.java
BOT
TelePanelOptions.java

Known Bugs and Issues

- When the program opens on a smaller screen resolution, it is too large. A way to fix it is to have the GUI resize relative to the screen size.
- Choosing a second port after the first has been opened brings up a “port busy” exception from the jssc library.
- Response time is slow, it might be Java or the way it is programmed
- When the mount slews to a certain defined point, it starts spinning in circles. This is most likely an issue with the mount itself.

Software References

- Satellite Tracker
- TeleTrack

Things that aren’t finished and Waid’s/Rast’s Desires

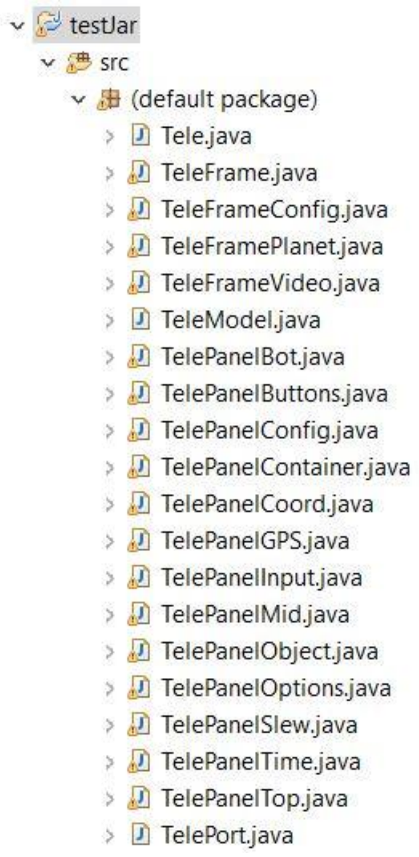
- In the menu of the frame: File, View, Object, and Help
- User Input of Azimuth, Elevation, Right Ascension, and Declination
- Telescope tracking: Sidereal and Solar tracking
- Night Mode. In the frame menu: View > Night Mode
- General formatting of presented data on GUI
- Waid wants video to work with a webcam. In the frame menu: View > Video
- Waid wants separate Planetarium software to be called on. View > Planetarium
- Rast wants input of several goto points to track a satellite (this is a theory on how to put a tracking option on the mount)

What to do when the code is completely finished

1. Update all documentation, including the comments in the code.
2. Make a general user manual for normal people (plebs)
3. Export the GUI to a .jar file (or try to figure out other executable files, but .jar should suffice)

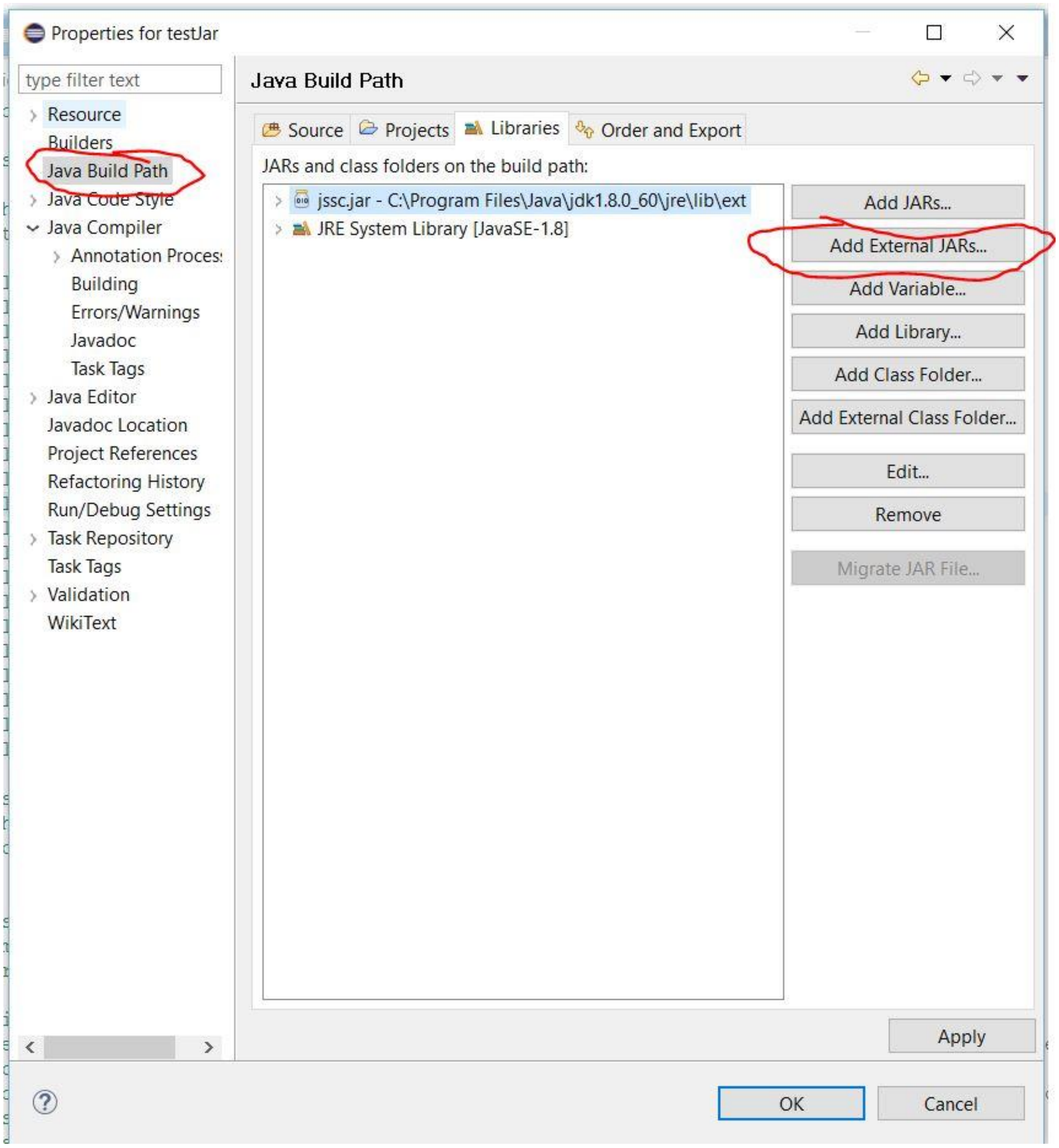
   The only way that I know how to do this is using an IDE (surprise). We'll be using Eclipse.

   1. Go to File > New > Java Project. Then copy and paste the GUI files into the src folder in the project (I named it testJar), and it should make a default package. Here what it looks like for me:

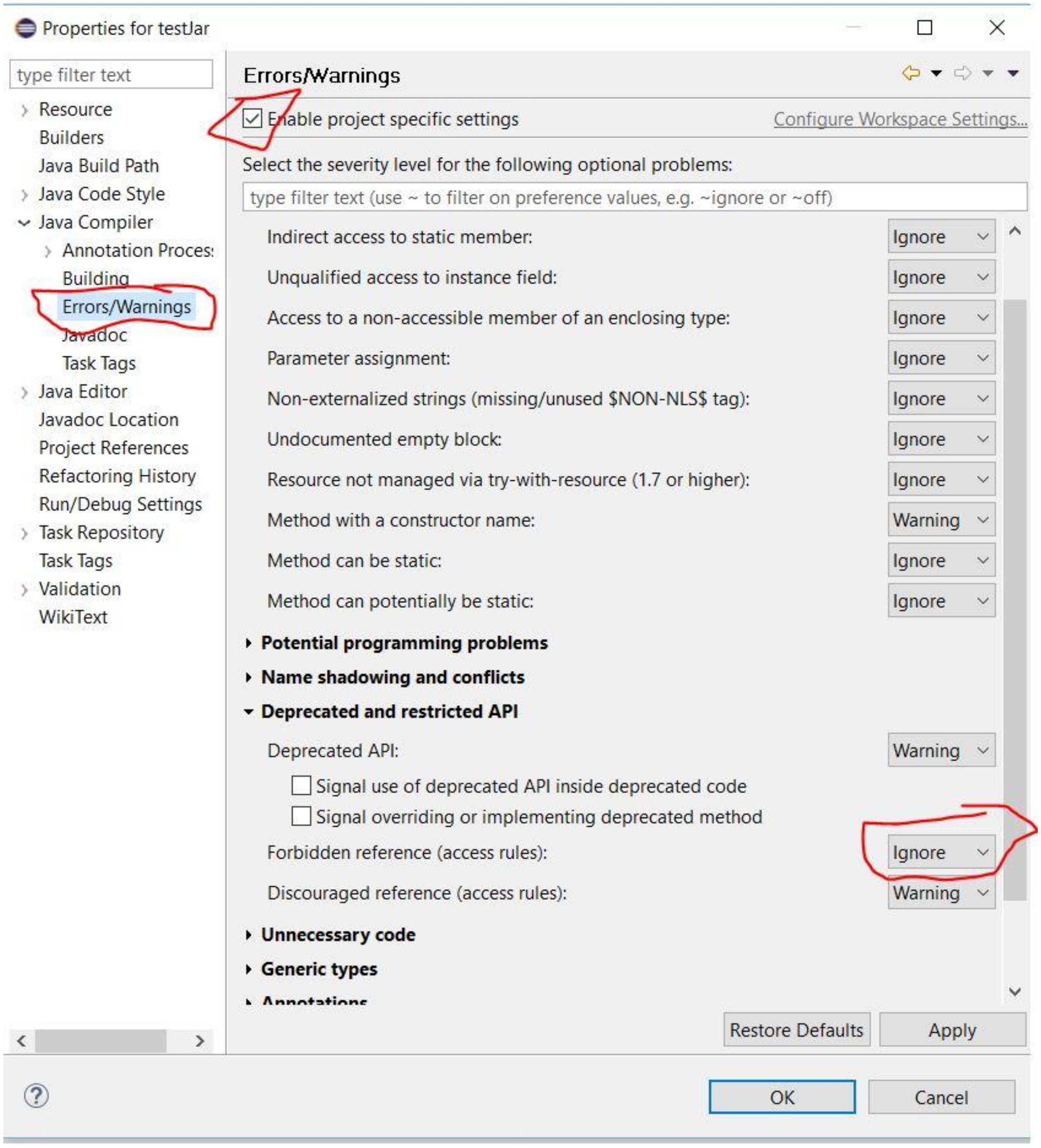


```
testJar
  src
    (default package)
      Tele.java
      TeleFrame.java
      TeleFrameConfig.java
      TeleFramePlanet.java
      TeleFrameVideo.java
      TeleModel.java
      TelePanelBot.java
      TelePanelButtons.java
      TelePanelConfig.java
      TelePanelContainer.java
      TelePanelCoord.java
      TelePanelGPS.java
      TelePanelInput.java
      TelePanelMid.java
      TelePanelObject.java
      TelePanelOptions.java
      TelePanelSlew.java
      TelePanelTime.java
      TelePanelTop.java
      TelePort.java
```

2. Right-click on your project and go to Properties > Java Build Path. Then click on the Libraries tab. Go to Add External JARs… and add in your jssc.jar file. This should add in the jssc library to the final .jar and the user will never need to go through the pains of installing jssc.



3. You might have noticed some errors that Eclipse picked up about the jssc library. We don't care about this. The reason Eclipse is acting this way because it doesn't like third party API. Obviously, in this case we need it. To disable this error right-click on your project and go to Properties > Java Compiler > Errors/Warnings and click Enable project specific settings. Then go to Forbidden reference (access rules) and choose ignore.



[✓] Enable project specific settings

Select the severity level for the following optional problems:

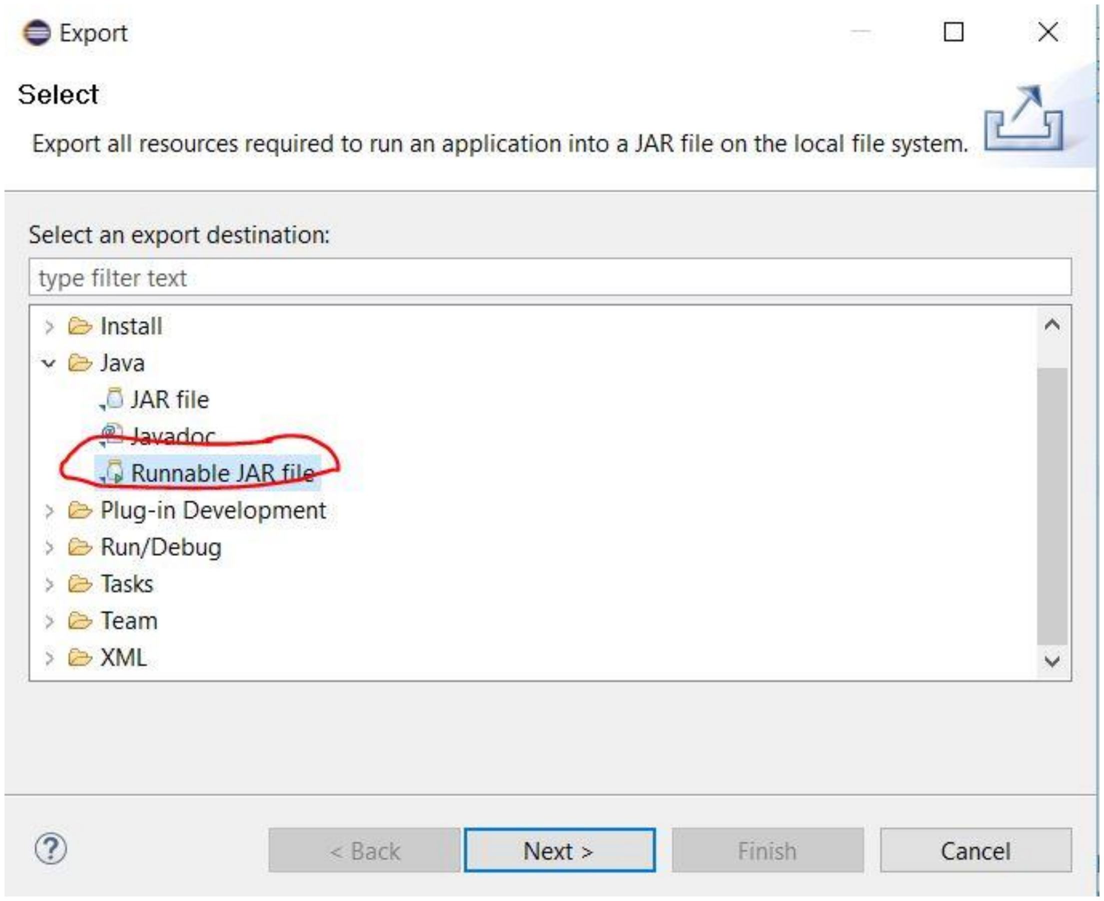
type filter text (use ~ to filter on preference values, e.g. ~ignore or ~off)

Indirect access to static member:	Ignore
Unqualified access to instance field:	Ignore
Access to a non-accessible member of an enclosing type:	Ignore
Parameter assignment:	Ignore
Non-externalized strings (missing/unused $NON-NLS$ tag):	Ignore
Undocumented empty block:	Ignore
Resource not managed via try-with-resource (1.7 or higher):	Ignore
Method with a constructor name:	Warning
Method can be static:	Ignore
Method can potentially be static:	Ignore

Deprecated and restricted API

Deprecated API:	Warning
[ ] Signal use of deprecated API inside deprecated code	
[ ] Signal overriding or implementing deprecated method	
Forbidden reference (access rules):	Ignore
Discouraged reference (access rules):	Warning

4. Finally, it is time to export the GUI to a single .jar file. Right-click on your project and go to Export…. Then go to Java > Runnable JAR file. Choose the main class (Tele) in your Launch Configuration, choose export destination and name the final GUI. Then choose Package required libraries into generated JAR. That should be it!

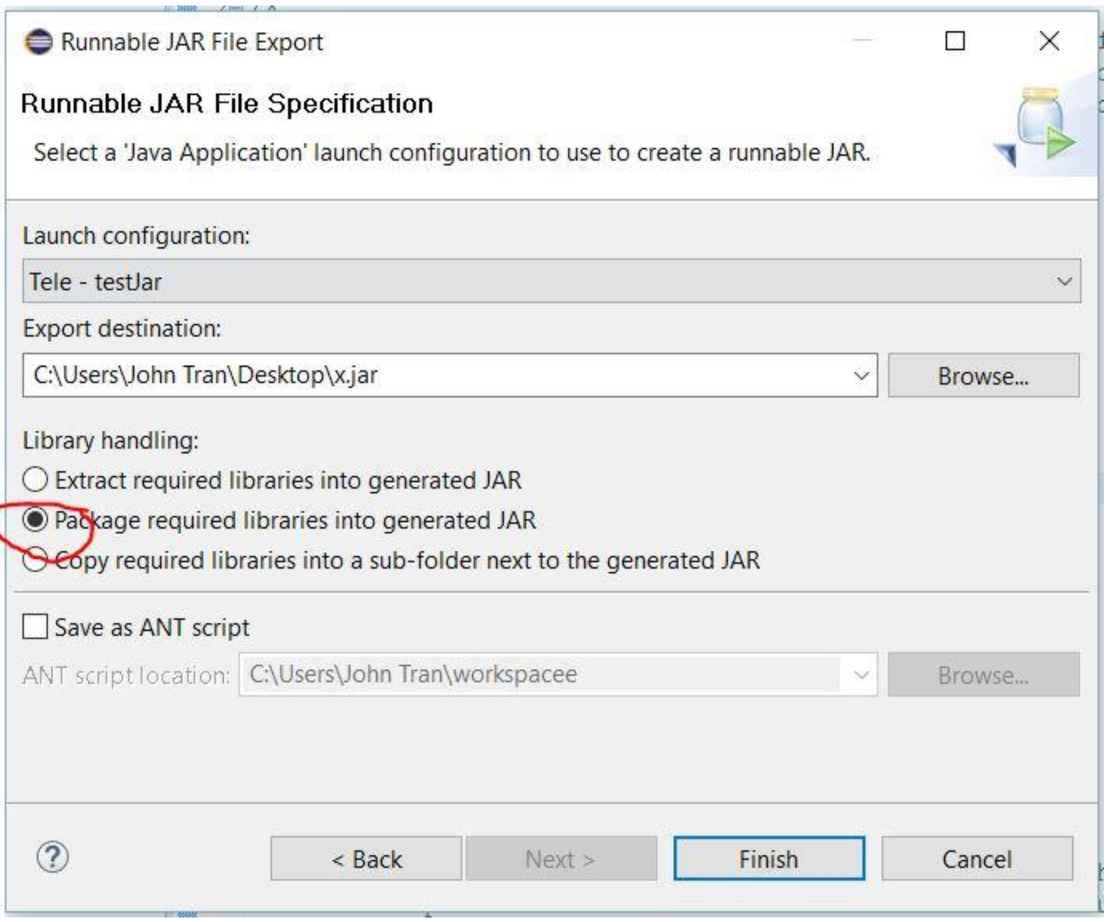

Export

Select

Export all resources required to run an application into a JAR file on the local file system.

Select an export destination:

type filter text



Runnable JAR File Export

Runnable JAR File Specification

Select a 'Java Application' launch configuration to use to create a runnable JAR.

Launch configuration: Tele - testJar

Export destination: C:\Users\John Tran\Desktop\x.jar

Library handling:

- ○ Extract required libraries into generated JAR
- [✓ ● Package required libraries into generated JAR]
- ○ Copy required libraries into a sub-folder next to the generated JAR

☐ Save as ANT script

ANT script location: C:\Users\John Tran\workspacee

5. **CELEBRATE.** The program should be able to run on both Windows and Linux at this point. Contact John Tran about your success.

Contact

John Tran – tran.john1337@gmail.com or johtr@okstate.edu , 918-851-8587