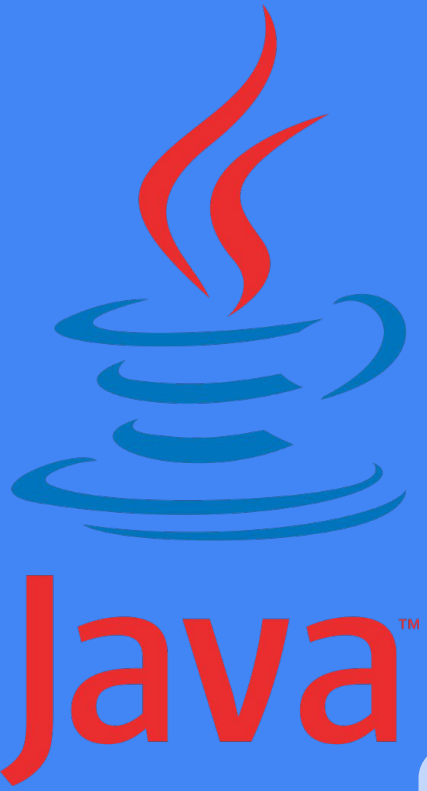
Introduction to FRC Programming

Part 7: GitHub, Version Control, Working Collaboratively



REVIEW FROM LAST TIME

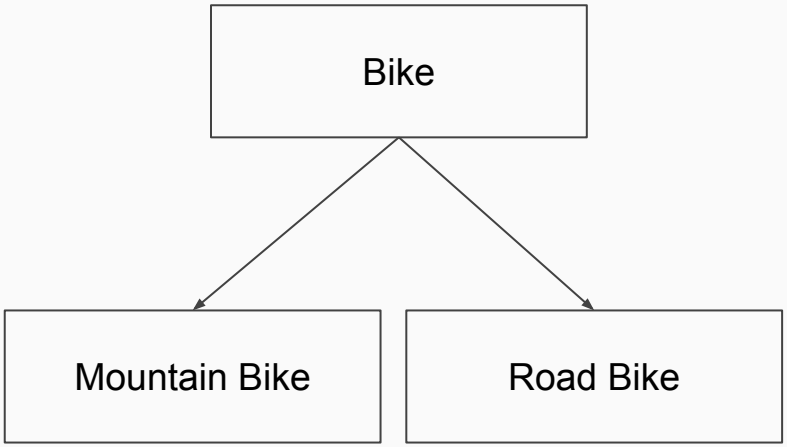
Has-A vs. Is-A Relationships

- All of the Object-Oriented Programming concepts we have discussed so far can be grouped into two main relationships: Has-A and Is-A.
- A class **has a** method or a member variable (class or instance) - in essence, a behavior or a state.
- A class **is a** superclass, interface, or abstract class that it inherits from or implements

- In our example, Dog **is an** animal, like Cat **is an** animal.
- Dog **has a** name, Dog **has a** behavior called bark.
- This concept is useful to remember when discussing inheritance and interfaces, especially when dealing with **polymorphism**

Inheritance Overview

- Inheritance allows classes that share code to “inherit” from one common superclass, as we have discussed.
- One superclass can be inherited by many different subclasses, but a subclass can only inherit one superclass.
- Nothing special has to be done to a class to allow it to be a superclass
- Private or protected modified fields or methods are not accessed by the subclass.



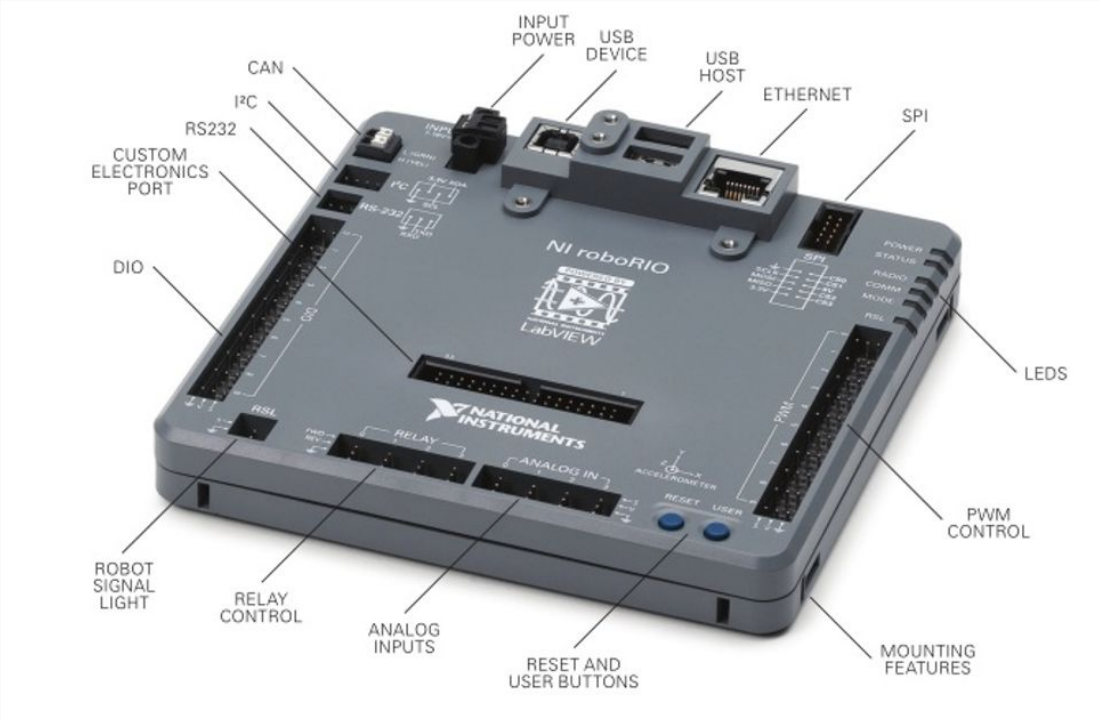
Abstract Classes

- An abstract class is a class that may or may not contain abstract methods - that is, methods without any body.
- An abstract class cannot be instantiated, but can be inherited from.
- It allows you to fully define some parts of the class that will be inherited, but leave the rest up to the subclasses to implement in a specific way.
- We turned our Animal class into an abstract class at the end of last time.

Interfaces

- Interfaces are like abstract classes, but take it a step further. They define a set of methods without implementations and constants.
- Interfaces are not inherited, but implemented.
- Interfaces can be thought of as a contract between the class that implements them and the compiler, promising that the class will provide an implementation for the specified methods.
- A class can implement any number of interfaces.

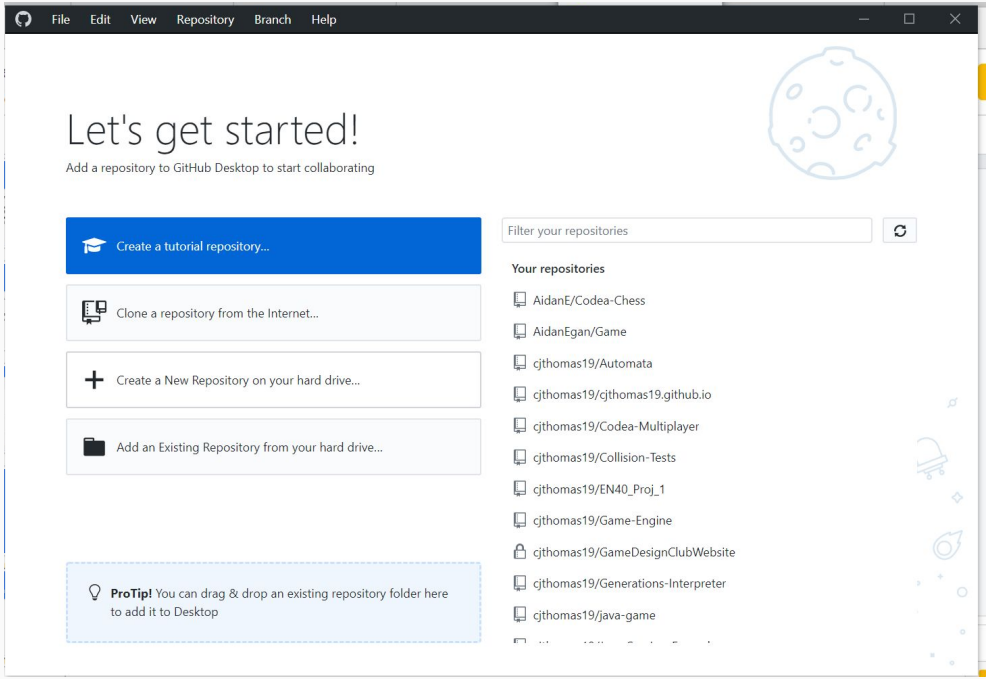
RoboRio - End 1 of CAN bus



END OF REVIEW

Let’s Start Experimenting With GitHub Desktop

- Hopefully to start off today we can all download GitHub Desktop and begin working on figuring out how to use version control and keep track of code collaboratively.
- When you download GitHub for the first time, you will be met with this screen:
- Let’s work through the tutorial together.

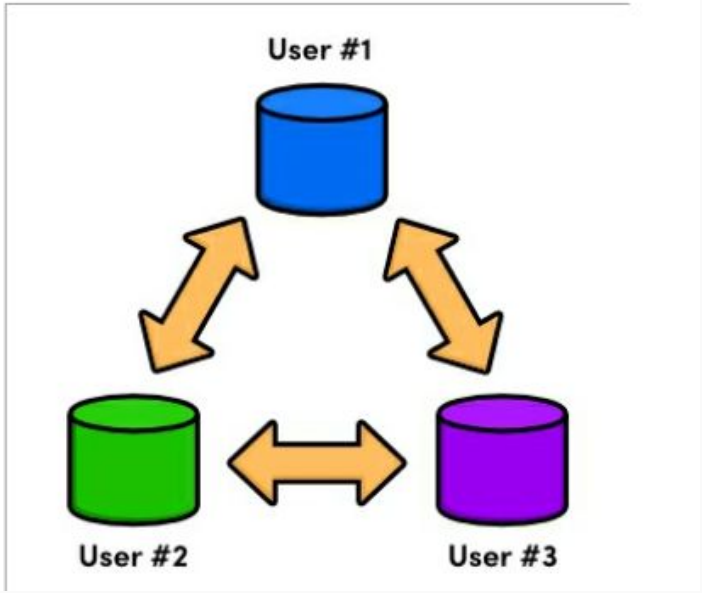


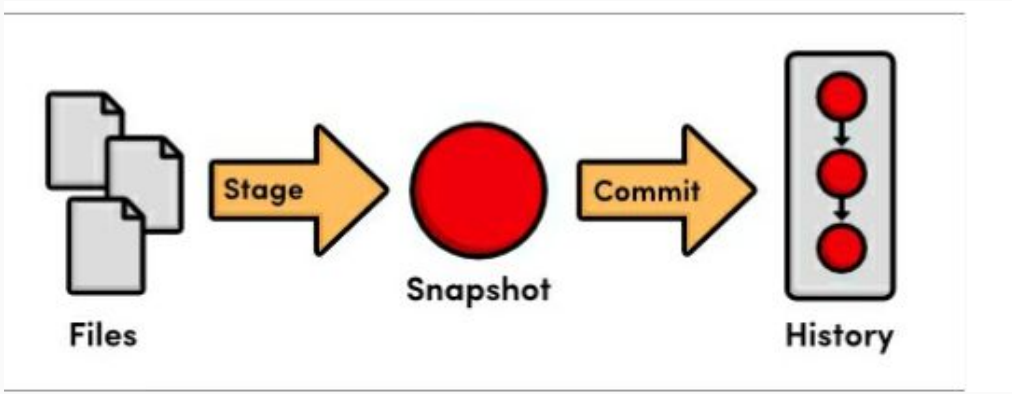
GitHub Collaboration Best Practices

- With Java, the .java source code files are saved with no encryption in plain text. This makes handling conflicts and seeing what has changed over the course of the version history much easier.
- However, handling conflicts is still time-consuming and can often result in headaches and difficulty. Therefore, there are a few key guidelines to follow when working collaboratively in a repository.

- Whenever possible, different teams should probably work in different branches. This prevents changes one team makes from conflicting with or affecting the functionality of another team's work. Any conflicts can be figured out all at once when the branches are merged.
- Try to limit the amount of people working on the same code at the same time.
- Before working on any critical code, fetch and pull any changes from the remote copy of the repository. When you're done, commit and push your changes.

How Git Works (photos from Ry's Git Tutorial)





Any General Programming Questions