Alejandro Hervella
Prof. Frank Tip

Homework 2

2)
From file singleton.ts:

Singleton:
This ProgramManager class is a singleton given its 'instance' variable and how it can only be accessed statically, and with its constructor as private. All other classes mostly reference classes that get actual variables and files being managed by the program here in a one stop shop.

Observer:
Although it's not spread out across a different class, each file has a list of subscribers tracked via the 'monitoredFileVars'. When a file is updated via the 'updateFileVariable' method, its associated variables are also updated (see the 'updateFileVariable' method for details below).

From file factory.ts:

This factory is responsible for making a specific IStatement instance using the provided type guard in the parsed data. We also see this for Expressions, and Conditions in a similar fashion with IExpressions and ICommands

3.1) When is it beneficial to use a Singleton?
- It is beneficial to use a singleton when we want to make sure that a variety of objects have easy access to frequently referenced variables or data, and do that there is only ever one reference to it in memory.

3.2) How does the Observer pattern decrease coupling between classes?
- The observer pattern decreases coupling because in doing so, the observer does not need to check the source it depends on for updates. Rather, the source sends an update to a list of subscribers, and the source only needs to worry about the list it currently has. This way there is no need for explicit references to the source and receiver (ie. parent and child), and there can be more than one source for each subscriber.

3.3) What kinds of changes to the code are easier to make when using a Visitor?
- Distinct and unrelated operations are easier to edit and perform with a visitor. They are useful when the programmer wants to frequently implement new operations and when changing the object class structure seldomly happens.

3.4) What is the difference between a Builder and a Factory?
- A builder creates a unique version of an object with different components that delivers a unique whole product, often dependent on the order of instructions for building the specific version of the object. A factory returns a different implementation of say an abstraction or interface. For example, a builder may always make a Car, with each model Car having different colors and configurations that are step orientated. A factory will build different types of Cars, such as a Truck or SUV, etc.