CSSE 374 UML Diagram Maker API and information

### Adding your own pattern detector:

- Create your own class to detect the pattern (note that this class must extend ClassVisitor)
  - The constructor must take in a ClassVisitor
- Make a call to UMLMaker.addPatternToPatternMap(String pattern, ClassVisitor visitor);
  - This can be done in the ClientRunner class before the MainRunner is called
- Add the pattern to your configuration file to let it be detected

### Access to data structures

- Arguments
  - The current list of arguments can be obtained via UMLMaker.getArguments()
  - The current class being iterated through can be obtained through DesignParser.getCurrentClass()
  - To check if a class is an argument, you can call UMLMaker.isArgument(String test)
- The list of all classes detected can be found using the class ArbitraryNodeNames
  - ArbitraryNodeNames is a singleton that maps the name of a class to the name of the node in graphviz.
  - List of all classes can be obtained from getting the keyset
- To add any arrows to the diagram, there are multiple methods that can be used depending on the type of arrow that needs to be drawn
- The list of interface extensions, class extensions, and the class method map can all be obtained from simple getter methods from UMLMaker (getInterfaceExtensions(), getClassExtensions(), getClassMethodMap())

### Phase Configuration

- The phases that the configuration goes through by default are the loading phase, and then the detection of every pattern that is contained within the pattern setting in the configuration file (in the order that they appear in), followed by the selection phase and the display phase.
- Note that all of the default phases must occur in their specific order (barring rearranging of the patterns in the configuration).
- To add a new phase such as a new pattern detection would be just adding the new pattern to detect to the end of the line, delimited by a comma.
- To add an entirely new phase, such as something that would push the output.txt file to a database, would require modifying the implementation of the private static void parseConfiguration() method in MainRunner.
  - All that is necessary is to add another statement to catch the new line that does not have a default configuration property name

### Pattern-specific Settings

- Pattern specific settings can be added in the configuration file
  - The format should be CLASS-VARIABLE-VALUE

  - Each variable should be presented on its own line
  - Implementation for handling the variable setting is done in the class, so if your class wants to implement specific settings for anything, then those need to be handled by the developer
    - For an example, look at SingletonClassVisitor