Maven Project Setup

We will be using the Eclipse AST framework in this discussion to demonstrate the visitor pattern. Using the Eclipse AST framework requires a lot of libraries, and installing them into a basic Java project manually is tedious. Therefore, to manage the dependencies, we will be using Apache maven https://maven.apache.org/what-is-maven.html to handle the build process.

To set up a maven project with the right dependencies and to work with the Eclipse AST parser, you can start from one of two ways.

1) Import the example visitor project uploaded to canvas. This method is likely simpler for this demonstration, but is less general. If you'd like to follow along on your own laptop during discussion, I'd recommend doing this before class.
    a) Download and unpack the visitorexample.zip file provided under the discussion 3 folder on Canvas. You should put the resulting folder into your Eclipse workspace.
    b) In Eclipse, select File -> Import. Then navigate to the "Maven" folder, expand it and select "Existing Maven Projects".
    c) Select the directory you extracted in your file system and hit finish.
2) Create a new Maven project and modify the pom.xml file to properly reference the Eclipse AST dependencies. This works more generally if you want to start your own project using the AST Parser. See the documentation here for a visual walkthrough: https://www.tech-recipes.com/rx/39279/create-a-new-maven-project-in-eclipse/
    a) Select File -> New -> Project. Under the Maven folder, select Maven Project.
    b) Use the default workspace and hit next.
    c) You will now be on the archetype page. Select the option "Internal" from the drop-down menu.
    d) There should now be many options visible in the window, select the one with Group Id: org.apache.maven.archetypes and Artifact Id: maven-archetype-quickstart, and hit next.
    e) Fill in the name fields group id and artifact id. For more on the meaning of these, see https://maven.apache.org/guides/mini/guide-naming-conventions.html
    If you want to match the names I choose in the example, use "ecs160.visitor" for the group id and "visitorexample" for the artifact id and hit finish.
    f) Now, in the package manager, there should be a file called "pom.xml", you'll need to add the following code between <dependencies></dependencies>

```
<dependency>
        <groupId>org.eclipse.jdt</groupId>
        <artifactId>org.eclipse.jdt.core</artifactId>
        <version>LATEST</version>
</dependency>
```

    g) Now, you should be able to reference the eclipse AST parser libraries.

AST View
The AST View Eclipse plugin provides a way to view the AST of Java files open in the editor. This can help you understand the structure of the tree. To install this plugin:

1. Go to Help -> Eclipse Marketplace
2. Search for AST View in the window and install
3. Accept the license agreements and restart Eclipse.

Now, to use this tool, go to Window -> Show View -> Other, and select AST View.

The AST View is a great way to get a sense of the structure of the AST of a file, what types make up the tree, and how statements are nested within each other. It will help you save time when trying to figure out how the tree is put together.

Eclipse AST Resources:

There is also fairly extensive documentation on the Eclipse implementation of a Java AST. The main classes to begin with are ASTNode and ASTVisitor. ASTNode corresponds to the **Component** in the composite pattern. ASTVisitor corresponds to the **Visitor** interface in the visitor pattern.

You can find documentation on these classes (and links to their subclasses) here:
ASTNode
https://help.eclipse.org/neon/index.jsp?topic=%2Forg.eclipse.jdt.doc.isv%2Freference%2Fapi%2Forg%2Feclipse%2Fjdt%2Fcore%2Fdom%2FASTNode.html
ASTVisitor
https://help.eclipse.org/neon/index.jsp?topic=%2Forg.eclipse.jdt.doc.isv%2Freference%2Fapi%2Forg%2Feclipse%2Fjdt%2Fcore%2Fdom%2FASTVisitor.html

I will go over some of the features of these classes and their subclasses in the discussion, but when working with the Eclipse AST, you need to become comfortable reading the JavaDoc style documentation used here.