

Class notes for CSC 520, Dale Parson, September 9, 2010

1. On Bill via putty:

```
cp ~parson/JavaLang/gamestomidi2010rev5.aug20mac.zip ~/JavaLang # If you haven't already
cp ~parson/JavaLang/generic_sequence_aclass.zip ~/JavaLang # Source code for tonight's lecture.
cp ~parson/JavaLang/models2010rev1.zip ~/JavaLang/models.zip # Basis of project 1.
```

2. These steps are on a local PC. If you brought a laptop, use that.

2a. Browse to the U:\ drive and create directory JavaLang.

2b. Use WinSCP to copy above three ZIP files into U:\JavaLang
(or FileZilla or ftp on others machines).

2c. Unzip all into current directory. You may have to move games2010rev5 and gamestomidi2010rev5 up into U:\JavaLang from the unzipped gamestomidi2010rev5.aug20mac subdirectory.

3. In c:/Program Files/eclipse copy a shortcut to the Eclipse executable and put it on your desktop.
On other machines you'll install Eclipse per instructions on the course web page.
Eclipse unzips as an executable that you run. There is no "install" to the registry.

4. Start Eclipse. The remainder of these steps occur in Eclipse.

Enter Java perspective.

File -> New Java Project -> from existing source.

We always use the file system, not Eclipse's project space, to manage projects.

~parson/JavaLang/models2010rev1.zip was captured under Eclipse, so we may need to import it as an existing model.

Use u:\JavaLang as the root for JavaLang, which is the only project we'll create all semester.

Also, whenever you re-copy JavaLang subdirectories onto your PC, make sure to refresh project JavaLang with an F5.

Under Project -> Properties note Java tool properties.

Under Window -> Preferences -> General -> Editors -> Text Editors, enable line numbers.

From Run -> Debug Configurations you can add a new executable configuration with a MAIN supplied from gamestomidi2010rev5.scrabble.ScrabbleToMultiMidiStartFrameUI. This should run under Eclipse after entering the Debug perspective. Also just double-clicking gamestomidi2010rev5.gamestomidi2010rev5.jar runs the stand-alone executable.

<http://bill.kutztown.edu/~parson/javadoc/> has up-to-date Javadoc for the Scrabble-to-MIDI packages.

5. Terminate the game. We are to hand capture some UML models inside Eclipse.

Relevant URLs from our course home page:

<http://www.vogella.de/articles/UML/article.html>

.....
<http://javareveng.sourceforge.net/>

I attempted the following and it failed on a Grim 307 lab machine because of a missing dependency.

It appears that the reverse engineering tools have not caught up without version of Eclipse.

We will do our reverse engineering by hand for project one -- these reverse engineering tools are a little flaky anyway.

In Eclipse Help -> Software Updates -> Add site
<http://javareveng.sourceforge.net/update/>
 Select and try to add Java Reverse Engineering for Eclipse.

The textbook lists three ways to use UML:

UML as sketch (in both forward and backward engineer – **We are taking this approach.**)

UML as blueprint – This approach is much more detailed concerning implementation.

UML as a programming language – Specialized tools generate code from constrained models.

5a. From the Project menu generate Javadoc HTML for package generic_sequence_aclass and inspect.

5b. From within Java perspective create sub-folder JavaLang/models.

5c. Highlight models/, File -> New -> Other -> Class Diagram, name it generic_sequence_aclass.

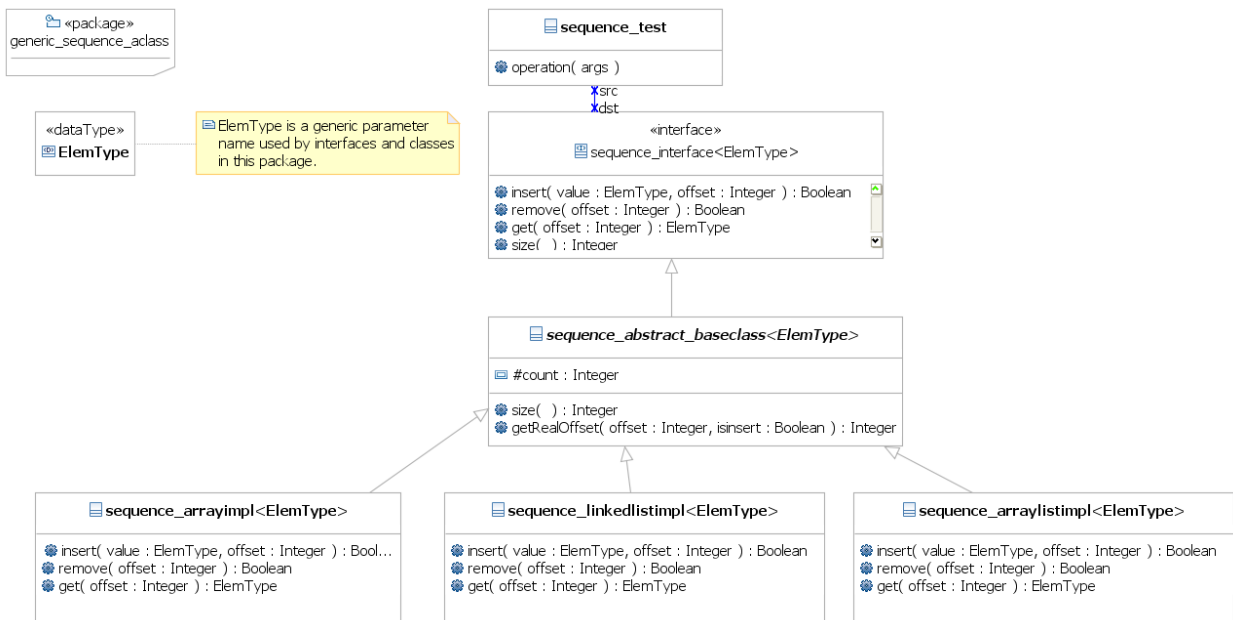
5d. Within package diagram generic_sequence_aclass create the following classifiers:

A *dataType* called *ElemType* that we will use to model generic types in the classes and interfaces.
 A *Comment* about ElemType that describes ElemType. Link it to ElemType in the diagram.

5e. Open Java file generic_sequence_aclass/sequence_interface.java in an editor.

5f. Within diagram generic_sequence_aclass create interface sequence_interface<ElemType> corresponding to the above Java class. Populate its interface, operations, classes and methods. Each operation and method has parameter types and return types. Populate those.

Repeat steps 5e and 5f for every class in Java package generic_sequence_aclass. It should look something like my handout.



IMPORTING IN ECLIPSE UML2: This tool is immature. Defining a class in one diagram and using it in other essentially requires defining all classes to be used together within a single **Model**. Classes defined in class diagrams in different Models or Packages typically will not copy-then-paste across diagrams. Classes defined within multiple diagrams within a single Model appear to work OK. The import machinery across models does not work consistently.

1. I have defined a single Model (not Package!) **games2010rev5.uml** under which we will place all diagrams this semester.
2. I have defined Class Diagrams `games2010rev5.umlclass`, `games2010rev5.scrabble.umlclass`, `gamestomidi2010rev5.umlclass` and `gamestomidi2010rev5.scrabble.umlclass` and populated the first two with their major interface and class definitions and relationships. The last two contain only model and package declarations. You will complete the last two to show all major relationships between the two latter MIDI packages and the two former GAME packages. Your diagrams need not focus on the internals of packages `gamestomidi2010rev5` and `gamestomidi2010rev5.scrabble`, but rather on their relationships back to `games2010rev5` and `games2010rev5.scrabble`.
3. Defining a new diagram requires identifying `games2010rev5.uml` in the Package Explorer panel, right clicking, and then initializing the appropriate diagram type. Always select `<Model>` `games2010rev5` as the **root** model. Doing so enables adding entities defined in other diagrams into your diagram. There is no need to do this for assignment 1, but we will test it out in class.
4. You are not required to use Eclipse UML for assignment 1. You may use any diagramming or illustration tool at your disposal. It should be possible to use this tool at this point – I have done a lot of trial and error work with it – and we will hit the essential points in class. If you get part way through using this tool and it becomes a nightmare, just print out a PNG graphic using File -> Save as Image File from each diagram, then hand draw the remaining elements. I do not think that this will become necessary.

EDIT ONLY 1 DIAGRAM AT A TIME. SAVE ALL BEFORE EDITING ANOTHER DIAGRAM.

