

CSC520, Advanced Object Oriented Programming, Fall, 2010, Dr. Dale Parson

Assignment 1: Model dependencies from packages `gamestomidi2010rev5` and `gamestomidi2010rev5.scrabble` to `games2010rev5` and `games2010rev5.scrabble`. Also model dependencies from `gamestomidi2010rev5.scrabble` to `gamestomidi2010rev5`.

Assignment is due before the beginning of class on September 23. We will use some class time to learn to use Eclipse on September 9 and to work through problems on September 16. Note that this is one of the few assignments for which I will collect both files and paper copies of diagrams. Normally I collect only files, but we need to make sure that the Eclipse UML tools are up to snuff. If Eclipse UML fails you, capture the two class diagrams of this assignment using some other drawing tool.

Summary: I have captured relationships among classes in Java packages `gamestomidi2010rev5` and `gamestomidi2010rev5.scrabble` as Eclipse UML class diagrams residing in `~parson/JavaLang/models2010rev1.zip`, which unzips to a `models/` directory under your `JavaLang`. Your goal is to model the relationships from classes in packages `gamestomidi2010rev5` and `gamestomidi2010rev5.scrabble` to classes that I have modeled in packages `games2010rev5` and `games2010rev5.scrabble`.

The purpose of this exercise is to understand the relationships from Scrabble-to-MIDI back to Scrabble. The reason for wanting this understanding is that you will be designing and integrating an artificially intelligent (AI) Scrabble opponent into the Scrabble game. Ideally, enhancing the game to add an AI opponent should not affect music generation, but the current design of Scrabble-to-MIDI may present hurdles. I do not know – I have not yet done this analysis myself.

There may be classes in `gamestomidi2010rev5` and `gamestomidi2010rev5.scrabble` that are not directly involved in relationships to `games2010rev5` and `games2010rev5.scrabble`. Examples would be private or package-protected implementation classes. You do not need to model such classes for this assignment. In fact you should not model them unless they are involved in the relationships of interest. Our goal is to determine MIDI to GAME associations and dependencies. For any classes that you do decide to include in your class diagrams, show any relationships among them, and any relationships from `gamestomidi2010rev5.scrabble` to `gamestomidi2010rev5`.

There are six related files in the `models/` directory:

`games2010rev5.uml` holds the root of the overall model. This file should not require any changes.

`games2010rev5.umlclass` is the class diagram for package `games2010rev5`. This file should not require any changes.

`games2010rev5.scrabble.umlclass` is the class diagram for package `games2010rev5.scrabble`. This file should not require any changes.

gamestomidi2010rev5.umlclass is the class diagram for package gamestomidi2010rev5. You will work on this file.

gamestomidi2010rev5.scrabble.umlclass is the class diagram for package gamestomidi2010rev5.scrabble. You will work on this file.

assignment1.txt is a currently empty file in which you can write any notes about ideas or problems you predict when thinking about extending the GAME packages due to hurdles created by the MIDI packages. The only requirement for this file is that you change “STUDENT NAME” to your name. Add any comments you like – these are notes to yourself.

You must model appropriate interfaces and classes to the level of detail used in `games2010rev5.umlclass` and `games2010rev5.scrabble.umlclass`. Do not model operations, methods, constants or fields unless they are relevant to your analysis goal. You might model a method that has a parameter type or a return type located in one of the GAME packages. I did **not** model methods for GAME classes. Instead I just showed method associations as associations.

Your links between classes need to show direction. For most links I used UML *Associations* onto which Eclipse UML attaches “src” and “dst” tags. These Associations must run from “src” to “dst” – the direction must be correct. Standard UML supports arrows on Association links.

I used an Eclipse UML *Dependency* link in two places, to show the private or package-protected relationships from `ScrabbleSwingFrameUI` and `ScrabbleSwingAppletUI` to helper class `ScrabbleSwingRootPaneHelper`. Because of limitations in Eclipse UML visual modeling, we will continue to use Association links with “src” and “dst” direction markers for public and protected associations, and Dependency relationships to denote private or package-protected relationships. If there is a need for bidirectional relationships, use two links. In general bidirectional Dependencies and Associations indicate possible design problems to be avoided. They tend to fuse all classes involved in a relationship cycle into one big conceptual class that is harder to maintain than with the case where relationships form an acyclic directed graph (DAG).

Finally, use Eclipse *Generalization* links with arrows for inheritance. See my example class diagrams.

Resources: <http://faculty.kutztown.edu/parson/fall2010/CSC520Fall2010.html>

The course web page contains this handout and the September 9 “Eclipse UML Modeling handout” that shows my class diagrams. The latter handout has instructions for obtaining the initial models/ directory with my class diagrams.

<http://bill.kutztown.edu/~parson/javadoc/> has up-to-date Javadoc documentation for the packages of this assignment.

<http://download.oracle.com/javase/6/docs/api/index.html> has the Java library documentation. You will need this in programming assignments later on.

The Java source code for these four packages is another resource in this analysis.

Once you have completed your modeling work, please regenerate the two PNG graphics files and the associated HTML files for your class diagrams in the models/ directory using right click File -> save as Image File for each of the two class diagrams in Eclipse. Print a copy of each for me. Make sure to include your name in the comment boxes in these diagrams and in file assignment1.txt.

In order to turn the assignment in some time before class on September 23, copy all of your models/ directory back onto bill.kutztown.edu under your ~/JavaLang directory. Then you can:

```
cd ~/JavaLang/models  
gmake turnitin
```

Observe the prompts and hit Enter (Carriage Return) to complete the turn in. For this assignment I will collect both the files via this mechanism and printouts of the two class diagram PNG files. Please bring the printouts to class on September 23.