

CSC 520, Advanced Object Oriented Programming, Fall 2010, Dr. Dale E. Parson

Assignment 2, designing the public interface of Class ScrabbleAI, due October 1, 2010

Below is my UML class diagram for package games2010rev5.scrabble, the environment for our current analysis and design project, which is an intelligent Scrabble opponent.

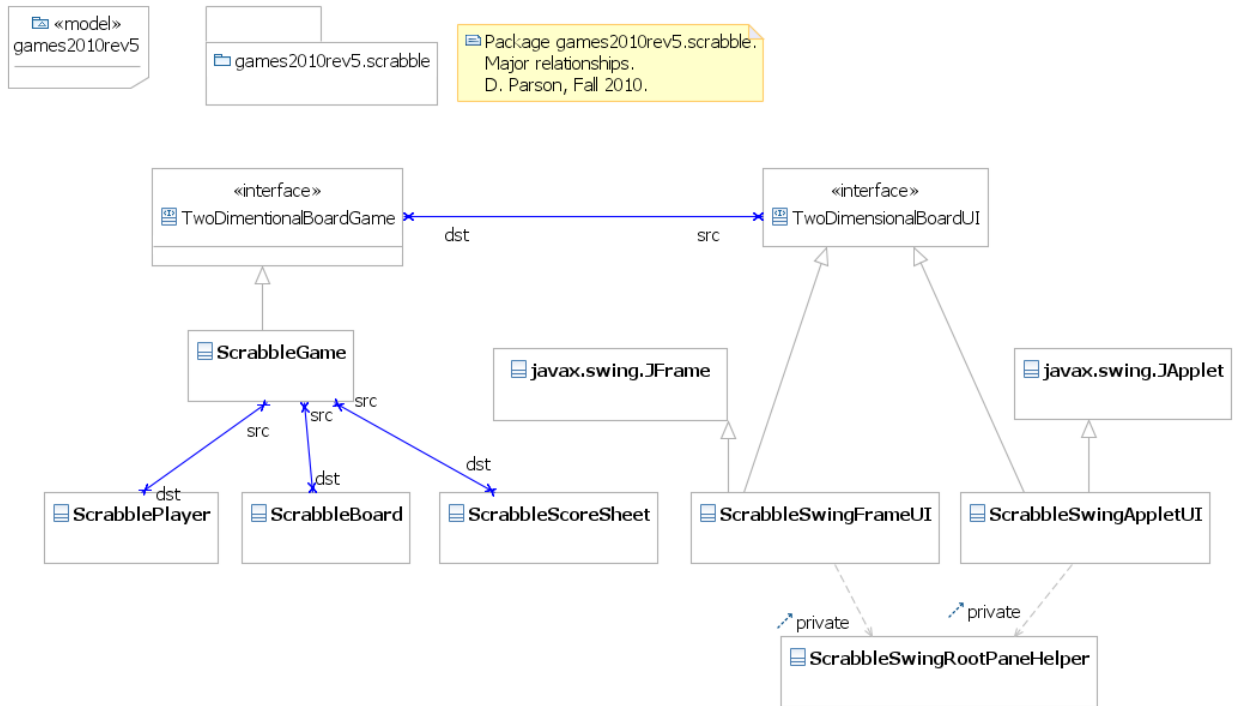


Figure 1, package games2010rev5.scrabble

Your assignment is to design the interface for a new class named ScrabbleAI and figure out where to fit it into the above package.

By *interface* in this case I do not mean a Java interface. I mean that you need to specify the set of Java methods, including parameters, return types and exceptions that they throw (if any), their Javadoc documentation, and any symbolic constants (if any) to be made available by class ScrabbleAI to any other class in package games2010rev5.scrabble.

For now we can assume that class ScrabbleAI needs to be visible only inside this package, so make its class declaration “package protected” by not declaring any of “public,” “protected” or “private” in the class declaration. See the declaration of class games2010rev5.scrabble.ScrabbleSwingRootPaneHelper for an example of a package-level “helper class.”

Make all of your method declarations **public** and symbolic constant declarations (if any) **public**, **static** and **final**. Our intent here is to establish only those methods that may be invoked by methods outside of class ScrabbleAI by other classes in games2010rev5.scrabble. These methods define its “public interface.” The bodies of these methods should be empty (literally “{}”) for now, so that you can compile the classes in order to check their signature syntax and Javadoc, and also so that do not actually implement anything. Please use my example Javadoc comments for guides. Class ScrabblePlayer has some example Javadoc comments.

You also need to update the above UML class diagram (you can use my UML2 gamestomidi2010rev5.scrabble.umlclass under models if you are using Eclipse UML2) to show all Associations into ScrabbleAI and out of ScrabbleAI to other classes (if any).

The class or classes that construct objects of class ScrabbleAI definitely must show an Association arrow going to ScrabbleAI, and one of your first design decisions is deciding which class or classes in this package are responsible for constructing an object of class ScrabbleAI and “knitting” that object into the system by supplying other objects in the package with references to the ScrabbleAI object and/or supplying the ScrabbleAI object with references to other objects from the package. We do not need to use the Factory Method or Abstract Factory Method patterns for constructing objects of class ScrabbleAI.

Class ScrabbleAI must implement interfaces `java.lang.Cloneable` and `java.io.Serializable` in order to support saving and restoring of game state, as I will go over in class. You do not need to write any code on behalf of these interfaces, but your code for class ScrabbleAI does need to declare that it implements these two interfaces.

The main two things you need to decide in specifying ScrabbleAI’s methods are the following.

- What data must be fed to an object of class ScrabbleAI so that it can keep track of the current state of the game?
- What data must an object of class ScrabbleAI supply when it proposes a move?

Those two questions intentionally imply two separate methods. You could conceivably specify one method to accept current game state in its parameters, and return a proposed move in its return value. Instead **I want you to specify two distinct methods**. The reason is that, although our implementation of ScrabbleAI is single-threaded this semester, we want to leave open the possibility of implementing a multi-threaded AI opponent that searches and plans during the time that other players are making their moves. Therefore we need two methods, one to accept game state updates, and another to propose moves for ScrabbleAI.

Once you know the data that ScrabbleAI needs for the first method and the data that it returns from the second, then you have your major clues about what class or classes need to associate with ScrabbleAI via UML associations. What class or classes **construct** it? What class or classes invoke its **state-updating method with appropriate parameters**? What class or classes invoke its **move-making method and use its return value to make its move**?

In supplying data to its **state-updating method**, I want you to play a game and see what data are available to you as a human player via the current GUI. You can start a new game simply by double-clicking file JavaLang/gamestomidi2010rev5/gamestomidi2010rev5.jar. The code base that you previously copied includes this executable JAR (Java archive) file, or you can invoke “gmake build” from within directory JavaLang/gamestomidi2010rev5 on bill. This compiled code is portable to any machine running JRE (Java Runtime Environment) version 1.6. **I want you to make sure that all data that is accessible to a human player of the current implementation of this software game is available via parameters to your state-updating method for ScrabbleAI.**

Your **goals** in completing this assignment should be to minimize coupling between classes and to avoid introducing any circular associations or dependencies among classes. Avoid any unnecessary complexity in creating inter-class associations. There are multiple classes with access to game state data. Some of that data take forms (data types) that can be used easily by ScrabbleAI. Use those forms. Also, some classes need to decide whose turn it is to move and when in order to invoke ScrabbleAI’s move-making method. Sequencing of players is a factor in deciding when to invoke ScrabbleAI’s move-making method and from what class or classes.

Finally, make sure that your design will allow game players to create more than one ScrabbleAI object during a game. A game must support anywhere from zero to four intelligent Scrabble players.

We will spend half of the class period on September 30 working on this project. I will take questions and then we will have a working session after the break. You need to have most of your investigation completed before you come to class. Please come to class with as much of this project completed as possible and with questions ready. The due date gives you one additional day to work.

How to turn it in: Create your file ScrabbleAI.java in directory JavaLang/games2010rev5 to include a package statement, empty method definitions as outlined above, and complete Javadoc for all methods and symbolic constants (if any). Run “**gmake build**” from directory JavaLang/games2010rev5 to make sure that your source file compiles, and run “**gmake Javadoc**” to ensure that your Javadoc comments are error free. I also want you to create a text file ScrabbleAI.txt in this directory that gives the reasons for your design decisions. Please

explain your decisions for methods, method parameters, return types, exceptions (if any), symbolic constants (if any), inter-class associations and dependencies. I also want you to turn in your revised copy of Figure 1 above showing how ScrabbleAI fits into the package. Thus there are three files in your assignment.

- ScrabbleAI.java
- ScrabbleAI.txt
- An updated class diagram for this package showing ScrabbleAI's relationships to other classes. Within ScrabbleAI.txt let me know the name of this class diagram file. If you use Eclipse UML, please have this be a PNG file. In other words, I should be able to view this file by double-clicking on it.

When you are ready to submit your assignment on October 1, run “**gmake turnitin**” from directory JavaLang/games2010rev5.