

## Java basic types, control constructs, arrays, methods, and library containers

Assignment 1 for CSC 243, Spring, 2010, Dr. Dale E. Parson

<http://faculty.kutztown.edu/parson/spring2010/CSC243Spring2010.html>

Assignment 1 is due 11:59 PM on Feb. 18. Use `gmake turnitin` to turn it in.

Perform the following steps to copy and inspect my initial code handout.

```
mkdir ~/JavaLang # Do this if you haven't already.
cp ~parson/JavaLang/games2010rev2csc243.zip ~/JavaLang
cd ~/JavaLang
/bin/unzip games2010rev2csc243.zip
cd ./games2010rev2
gmake clean testtuyi
```

This test should complete without any errors. It tests the operation of the Scrabble game from last semester in non-GUI, terminal command mode. There is an additional test that will fail initially, but that will succeed when you complete this assignment. Your code must compile and it must pass the following test as part of the assignment.

```
gmake testboard
```

The test fails initially because it needs the solution to your assignment.

This assignment uses the following directories.

Package `games2010rev2` hold the generic game framework that loads and runs a game.

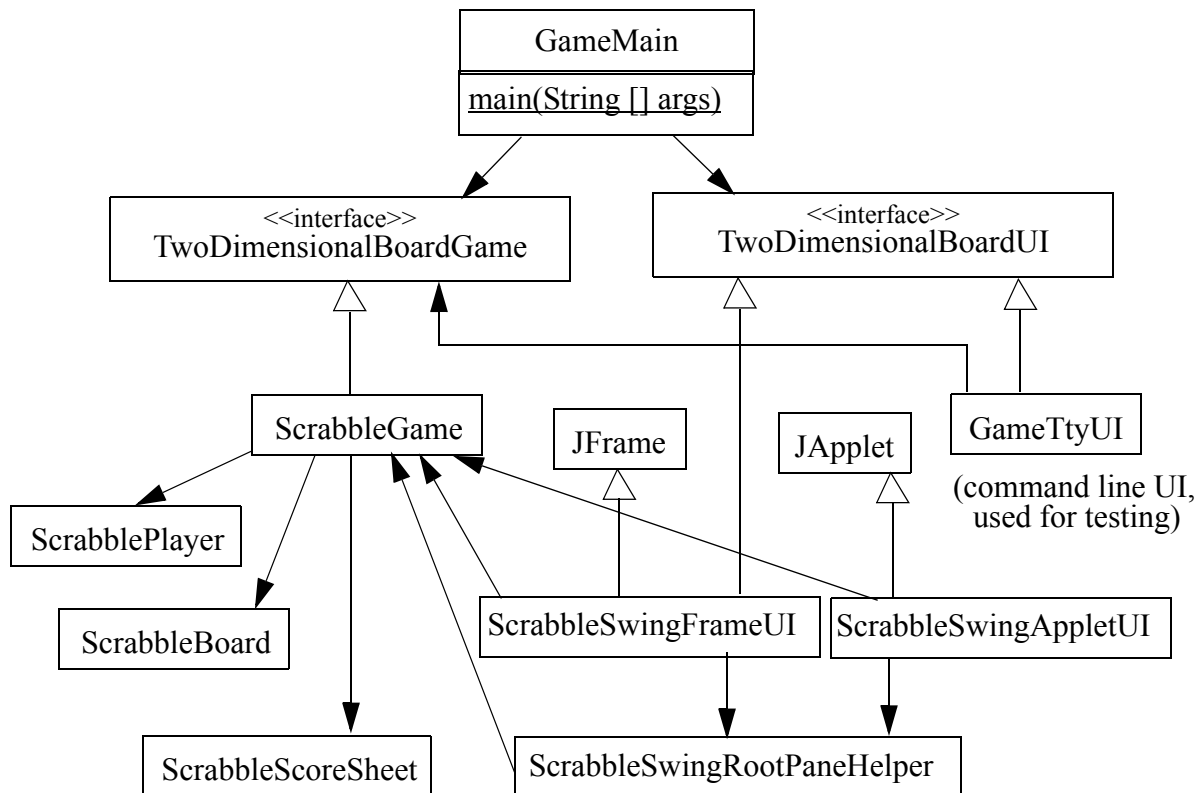
Package `games2010rev2/scrabble` holds a Scrabble plugin that the framework loads to play Scrabble.

Directory `games2010rev2/scrabble/pseudocode` holds file `ScrabbleBoard.cxx` that is pseudocode for your assignment. You must translate `ScrabbleBoard.cxx`'s logic to Java as explained below.

All of your source code additions go into file `games2010rev2/scrabble/ScrabbleBoard.java`. This file already exists. It performs validation of proposed moves in a Scrabble game, updates board state based on moves, and computes the score for a move. You will add code to this file in places denoted by an upper case `STUDENT` text tag. File `games2010rev2/pseudocode/ScrabbleBoard.cxx` gives the complete code logic for your solution, implemented in C++ and the C++ string, STL (Standard Template Library) and exception libraries. `ScrabbleBoard.cxx` compiles correctly using `g++`. In addition to `STUDENT` delineated work, `ScrabbleBoard.cxx` contains stubs for some of the fields that already are completed in `ScrabbleBoard.java`. You do not need to translate these completed parts to Java, but they do need to be present in `ScrabbleBoard.cxx` so that you can see how your `STUDENT` code uses them in Java.

**SKIM OVER THIS SECTION — IT IS NOT NEEDED FOR ASSIGNMENT #1.**

Figure 1 shows the complete class diagram for the final Java programming assignment of Fall, 2008 in UML (Unified Modeling Language) notation. This is the starting point for this semester's work. While all of the classes shown appear in the games2010rev2 and games2010rev2.scrabble packages, please focus on class ScrabbleBoard for this assignment. It is self-contained. We will restrict our focus to the STUDENT portions of ScrabbleBoard, and discuss this class diagram in more detail for the next assignment.



**Figure 1: Class diagram for packages games09 and games09.scrabble.**

Student work goes into class ScrabbleBoard.

Test “gmake testboard” which runs in non-graphical terminal (tty) mode puts the board into the configuration shown in Figure 2. Test “gmake testboard” invokes ScrabbleBoard’s main using the following command line invocations. System.in comes from file testboard.txt.

```
java games2010rev2.scrabble.ScrabbleBoard < testboard2010.txt > junk
grep -v TIME_DUMP < junk > testboard2010.out
/bin/rm junk
diff testboard2010.out testboard2010.ref > testboard2010.dif
java games2010rev2.scrabble.ScrabbleBoard english.0.no_jet.txt < testboard2010.txt > junk
2>&1
grep -v TIME_DUMP < junk > testboarddict2010.out
```

```

/bin/rm junk
diff testboarddict2010.out testboarddict2010.ref > testboarddict2010.dif
gmake: *** [testboard] Error

```

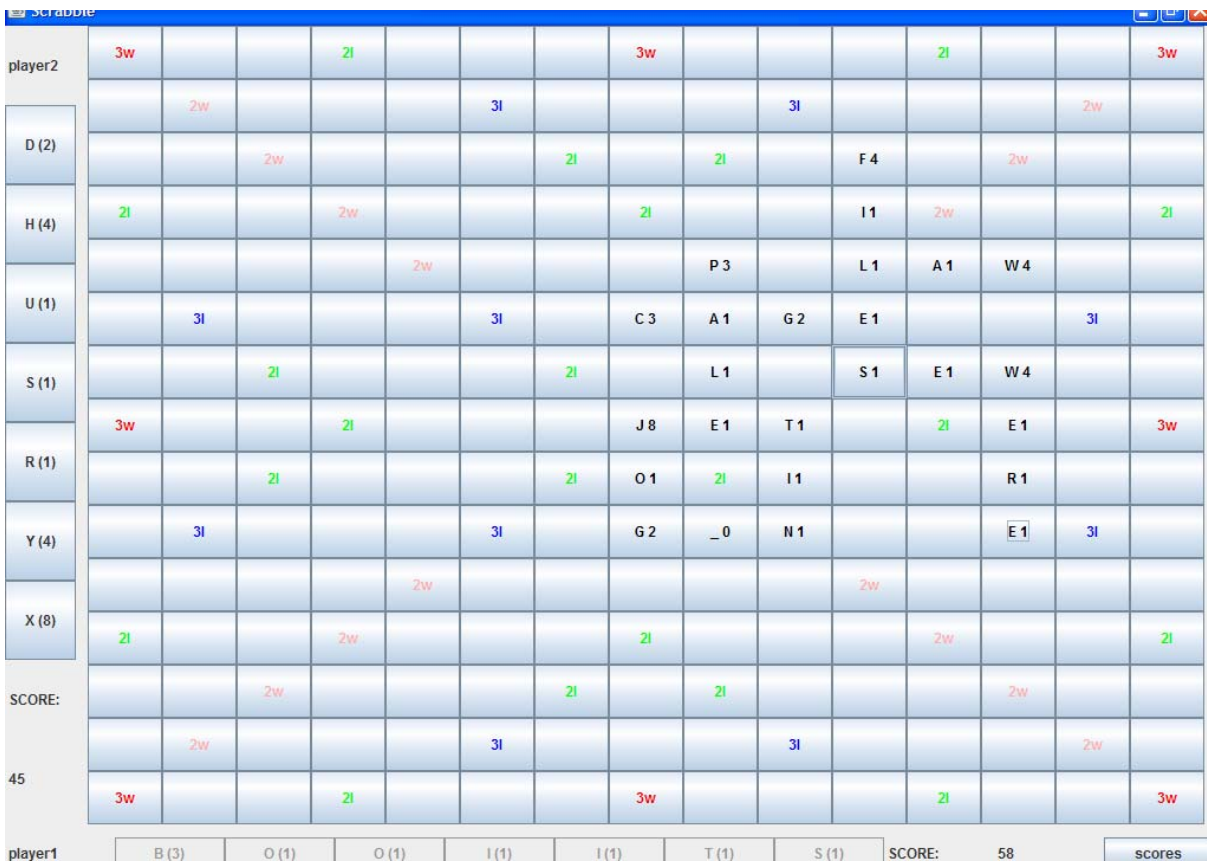


Figure 2: Scrabble GUI after 10 moves in file testboard09.txt.

Symbolic coordinates are A through O for the columns, 1 through 15 for the rows, with A at the left and 1 at the bottom. Location H8 is the center of the board.

Test “gmake testttyui” tests an entire game (instead of just ScrabbleBoard’s main() test driver) with this invocation. It puts the board into the same configuration as “gmake testboard.”

Test “gmake testgui” tests the graphical user interface with the following command. You must either run this on bill with X11 enabled or on a PC with Java 1.6 installed. Eclipse is optional.

```

java games2010rev2.GameMain games2010rev2.scrabble.ScrabbleGame
games2010rev2.scrabble.ScrabbleSwingFrameUI 4 (all in 1 line)

```

The parameters to the java JVM for “gmake testttyui” are as follows.

```

games2010rev2.GameMain      name of the class with the static main() method.
games2010rev2.scrabble.ScrabbleGame name of the game plugin to load and play

```

<b>games2010rev2.GameTtyUI</b>	name of the user interface plugin class
<b>2</b>	number of players, 2 to 4
<b>2</b>	a <i>seed</i> for a random number generator

ScrabbleGame uses a pseudo-random number generator to select order of play and to select tiles for players. The optional *seed* command line argument, set at 2 above, causes the pseudo-random number generator to generate the same set of numbers every time the program runs. A predictable sequence allows repetition of test results.

### **HERE ARE INSTRUCTIONS SPECIFIC TO YOUR ASSIGNMENT.**

**Here are the STUDENT instruction comments for your assignment from games2010rev2.scrabble/ScrabbleBoard.java.**

```

package games2010rev2.scrabble ;
import games2010rev2.* ;
import java.util.LinkedList ;
import java.util.HashSet ;
import java.util.Scanner ;
// STUDENT 1 (2010): Add an import statement for java.io.File.
// You need this class in order to read the dictionary of valid words.

private static final int OK = 0, BADFILE = 1 ;
/** Main is a test driver used for implementation testing only. */
public static void main(String [] args) {
    int status = OK ;
    Scanner linescanner = new Scanner(System.in);
    ScrabbleBoard board = new ScrabbleBoard(ScrabbleGame.weightOfTiles);
    while (linescanner.hasNextLine()) {
        String line = linescanner.nextLine();
        String [] fields = line.split(":");
        if (fields.length != 3) {
            System.err.println("invalid test file line: " + line);
            status = BADFILE ;
            continue ;
        }
        try {
            board.putWord(fields[0],fields[1],fields[2]);
        } catch (GameException gex) {
            System.err.println("Game Exception on data " + line
                + ": " + gex.getMessage());
            gex.printStackTrace();
            status = BADFILE ;
            continue ;
        }
    }
}

```

```

String [] query ;
Long [] timequery ;
try {
    System.out.println("\nFIFO DUMP\n");
    query = board.getWordList("FIFO");
    timequery = board.getTiming(true);
    for (int i = 0 ; i < query.length ; i++) {
        System.out.println(query[i] + "\nTIME_DUMP: " + timequery[i]);
    }
    System.out.println("\nLIFO DUMP\n");
    query = board.getWordList("LIFO");
    timequery = board.getTiming(false);
    for (int i = 0 ; i < query.length ; i++) {
        System.out.println(query[i] + "\nTIME_DUMP: " + timequery[i]);
    }
    System.out.println("\nH8\n");
    query = board.getWordList("H8");
    for (int i = 0 ; i < query.length ; i++) {
        System.out.println(query[i]);
    }
    System.out.println("\nH8:DEPTH\n");
    query = board.getWordList("H8:DEPTH");
    for (int i = 0 ; i < query.length ; i++) {
        System.out.println(query[i]);
    }
    System.out.println("\nH8:BREADTH\n");
    query = board.getWordList("H8:BREADTH");
    for (int i = 0 ; i < query.length ; i++) {
        System.out.println(query[i]);
    }
    // STUDENT 2 (Spring 2010): Currently this Scrabble game does not
    // validate words on the board against a dictionary.
    // Neither does this main function use its args parameter.
    // Insert code here to check whether there is at least 1 command
    // line argument in args. If there is, invoke function
    // "validateSpelling", which you will write below, with 2
    // arguments: the "query" array of Strings returned by the final
    // call to board.getWordList("H8:BREADTH") above, and
    // the first command line argument to this main function.
    // This latter String is the name of a file holding a dictionary.
    // See comments for function validateSpelling below.
} catch (GameException gxx) {
    System.err.println("Game Exception on dump: " + gxx.getMessage());
    status = BADFILE ;
}
System.exit(status);

```

```
}
```

**// STUDENT 2 (Spring 2010): Write the code for private function**

```
// validateSpelling that takes 2 parameters and returns void.
// The first parameter "wordarray" is an array of String objects
// that will look similar to this:
//   JET:H8:ACROSS
//   JOG:H8:DOWN
//   PALE:I11:DOWN
//   TIN:J8:DOWN
//   GiN:H6:ACROSS
//   CAGE:H10:ACROSS
//   FILES:K13:DOWN
//   LAW:K11:ACROSS
//   SEW:K9:ACROSS
//   WERE:M9:DOWN
// The second parameter "dictname" is a String that holds a file name.
// Your function validateSpelling must do the following steps:
// 0. Initialize an int "status" variable to OK.
// 1. Declare a Scanner in local variable "dictscanner" like this:
//   Scanner dictscanner = null ;
// 2. Construct a Scanner object for input File dictname like this:
//   dictscanner = new Scanner(new File(dictname));
// The compiler WILL GIVE YOU AN ERROR MESSAGES if you do not
// construct this Scanner object inside of a try clause, because
// the attempt to open file dictname may throw
// java.io.FileNotFoundException if the named file does not exist.
// Therefore, this line must be inside a "try" clause, and the
// following "catch" clause must catch java.io.FileNotFoundException.
// If you catch this exception, print an error to System.err and
// then exit the program with a status of BADFILE.
// 3. Construct a java.util.HashSet (of Strings) object like this:
//   HashSet<String> dictionary = new HashSet<String>();
// Then use your Scanner to iterate through all lines in
// dictname, a line at a time, and insert UPPER CASE copies of every
// line of dictname into HashSet dictionary. You can assume that there
// is one word per line in file dictname; it may be in lower case.
// Library operations that are useful:
//   3a. Function toUpperCase() in java.lang.String.
//   3b. Function add() in HashSet.
//   3c. Function split() in String (you will need this).
//   3d. Function contains() i HashSet (you will need this).
// 4. Iterate through the Strings in parameter wordarray, one at a time,
//   and for each do this:
//   4a. Use the String "split()" function to separate an element in
```

```

// wordarray into an array of three Strings. Every word in
// wordarray looks like "FILES:K13:DOWN" so you must split
// using ":" as an argument, and then use element [0] of the
// resulting 3-element array. It holds the word such as "FILES".
// There are example calls to String.split in this source file.
// 4b. Check to see whether the HashSet dictionary contains the
// UPPER CASE copy of this word. If it does not, print out an
// error message "STRING not found in dictionary." where STRING is
// the word from wordarray, and record the exit status as BADFILE.
// 5. After checking all words in wordarray against the dictionary,
// if status is not OK, then exit the process with that status.
// Otherwise return from this function.
// See PSEUDOCODE in games2010rev2/scrabble/pseudocode/ScrabbleBoard.cxx.
// See ~parson/JavaLang/countargs/CountArgString.java for line scanning.
// See ~parson/JavaLang/sortdemo/singlesort.java for array of ints.

```

**Here is the pseudocode from games2010rev2/scrabble/pseudocode/ScrabbleBoard.cxx.**

```

/** ScrabbleBoard.cxx

```

C++ pseudocode for Java assignment 1 of CSC 243, Spring, 2010.  
The fields and methods below have counterparts in project  
file ScrabbleBoard.java. This is not a complete class definition,  
but it does compile.

Dr. Dale E. Parson, Spring, 2010, CSC 243.

```

**/

```

```

// STL container classes have counterparts in package java.util.
// See cplusplus.com for on-line documentation on these.
#include <set> // STL equivalent to java.util.HashSet
#include <string> // Equivalent to java.lang.String
#include <iostream> // Use a java.util.Scanner and java.io.File.
#include <fstream>
using namespace std ;

static int OK = 0 ; // exit status when all is OK
static int BADFILE = 1 ; // exit status on a file error

class ScrabbleBoard {
private:
    static void validateSpelling(string wordarray[],
        int wordarray_length, // number of elements in wordarray[]
        string dictname) {
        int status = OK ;
        ifstream dictscanner(dictname.c_str(), ifstream::in);

```

```

if (dictscanner.fail()) { // catch a java.io.FileNotFoundException.
    cerr << "Dictionary file " + dictname + " not found.";
    exit(BADFILE);
}
set<string> dictionary ;
while (! dictscanner.eof()) { // While the file has another line.
    string line ;
    dictscanner >> line ; // assuming 1 string per line
    if (! dictscanner.eof()) { // C++ sets eof after input operation.
        dictionary.insert(line);
    }
}
for (int i = 0 ; i < wordarray_length ; i++) {
    // JAVA: String [] fields = wordarray[i].split(":");
    extern string * wordarray_i__split(string); // Pretend that C++ has this function.
    string *fields = wordarray_i__split(":");
    extern string toUpperCase(string); // member of java.lang.String (pretend for C++)
    if (dictionary.find(toUpperCase(fields[0])) == dictionary.end()) {
        // dictionary does not contain the upper case word
        cerr << fields[0] + " not found in dictionary." ;
        status = BADFILE ;
    }
}
if (status != OK) {
    exit(status);
}
}
};

```

When you have completed the assignment and “**gmake clean testboard**” works correctly from the games2010rev2/ directory, then from that directory enter **gmake turnitin**.

It will prompt you to hit Enter (Carriage Return) if you are sure, then it will bundle up the files and ship them to me. If you do not get an error message, then it worked. If you later make changes before the due date you can **gmake turnitin** again from the same directory, and it will over-write the prior submission. **Make sure that you turn this in by the end of February 18.**