Project 2

Attached Files
CITS1001Proj2.zip (42.622 KB)

CITS1001 Project 2, Semester 2 2022

If you have any questions about any aspect of the project, submit them to *help1001*.

Project Rules

- Submission deadline: 12.00 **pm (noon) Wednesday 19 October 2022**
- Submit via ***cssubmit***
- Value: 15% of CITS1001
- Project work to be done **in pairs**
- **Both students in a pair must submit the project to cssubmit**
- **Make sure both of your student numbers are in the submission .java files!**
- **Do not share code with other students outside your pair**. Be careful not to share code accidentally with other students via *help1001*, and make sure your GitHub repository is private (if you use one).
- **Do not ask for help on StackOverflow or similar**. You may only discuss the project with the teaching staff for CITS1001.

The project task is to construct a Java program containing your solution to the following problem. You must submit your program via *cssubmit*. No other method of submission is allowed. **Only submit the two .java files. No .class files, .zips, or others.**

You are expected to have read and understood the UWA Policy on Academic Conduct. In accordance with this policy, you may discuss with other students the general principles required to understand this project, but the work you submit must be the result of your own effort.

You must submit your project before the submission deadline above. UWA's Policy on Late Submission will apply.

Project Overview

Your task is to build on Wordle from Project 1. You will be implementing a simple artificial intelligence (AI) that can play Wordle, and also a class that analyses the performance of this AI.

Project Materials

Download the folder *CITS1001Proj2.zip* which is attached above. This folder contains the following.

1. Five Java Classes, one JUnit Test, and one Text File.

The Five Java Classes

A program skeleton is given with five pre-defined classes. You will just need to compete "TODO" sections inside some of the methods of these classes. You are not allowed to change any field variables; the given field variables are sufficient to complete the operations. You are also not allowed to change method or class definition. You should not add any new classes or methods.

WordleAI: A class implementing an AI algorithm for Wordle. Note that the class contains only static methods. You need to implement the following methods: *guessContains, isConsistent, isAllStars, findWord*

WordleAIAnalyser: A class for analysing the results of WordleAI. You need to implement the following methods/constructors: *WordleAIAnalyser, runExperiment, runExperimentsWithAllWords, runExperimentsWithWordsBetween, getUnsolvedWords, getGuessLetterFrequency, getNumGuessesFrequency, makeHistogram*

WordleExperimentResult: You **MUST NOT** modify this class in any way. Stores the results of experiments in WordleAIAnalyser.

WordleGame: You **MUST NOT** modify this class in any way. Implements Wordle, but slightly different from the Project 1 class.

WordleDictionary: You **MUST NOT** modify this class in any way. Implements the dictionary, but slightly different from the Project 1 class.

The One JUnit Test

WordleAITest: You **MUST NOT** modify this class in any way. It tests all the methods in WordleAI that you need to implement. If you pass all tests, you are (almost certainly) going to get full marks for correctness (but maybe not everything else).

The Text File

There is a single .txt file.

words.txt: You **MUST NOT** modify this text file in any way. It contains a list of words the WordleDictionary reads from. If you modify this file, some classes and tests may not work as expected.

Project Instructions

It is suggested to approach the project using the following steps:

1. Download *CITS1001Proj2.zip* (attached above)
2. Read over all the classes.
3. Observe both classes come with many helpful comments and hints. Read all of these.
4. Write both your student numbers at the top of WordleAI.java and WordleAIAnalyser.java where it says @author.
5. Observe the TODOs have numbers which explain the order that is best to implement the methods in.
6. After completing **WordleAI** the JUnit test can be used to check correctness.

7. After completing **WordleAIAnalyser**, see the comments on the printHistogram method.
8. There is no JUnit test for **WordleAIAnalyser.** You are encouraged to write your own to ensure correctness. We have a hidden JUnit test that we will test your code with. You do not need to submit any test you write, it is only for your own purposes to ensure correctness.
9. Submit ONLY your WordleAI.java and WordleAIAnalyser.java files to cssubmit. See the later section for details.

Project Management Tips

Before starting the project, students are expected to have

- studied the lectures up to week 10.
- completed the assigned labs.
- read and understood the whole of this project description,
- read and understood all relevant UWA policies.

Submit any questions about any of this to *help1001*.

It is recommended that you tackle the project tasks in the order indicated; that you compile frequently; and that you test and run the code after completing each method, to ensure your code behaves as you think it ought, and does not fail any tests. If you are stuck on a method, it is often a good idea to look in the lecture material or in the text for an example method which is similar in some way to the one that is confusing you.

You can gain good marks even if you do not complete all the methods, so long as the code you have written compiles and runs. But if you submit a large body of code that does not compile or that crashes often, then few marks can be awarded.

Hints and tips about the various methods may be uploaded here from time to time. Whenever that happens, the document version number at the top of the page will be updated.

Project Submission

Submit exactly two files: the ***WordleAI.java*** and ***WordleAIAnalyser.java*** source files by the submission deadline above via *cssubmit*. Do not submit anything else. No other method of submission is allowed.

- **Both students in a pair must submit to cssubmit**. A student who does not submit anything will get zero marks.
- **Make sure you put BOTH of your student numbers in WordleAI.java and WordleAIAnalyser.java**. There is an @author section at the top where you should fill in your student numbers.
- **If you do not submit exactly the same files, you may both get zero marks!**

The file names, class names, and method signatures in your submitted code must match the original specifications exactly. The marking process for correctness is automated, and any changes you make which break this can mean you get **zero marks**. If your code cannot be compiled with the original JUnit test cases provided, your submission **may get zero marks**.

Common mistakes are to submit .class files in place of .java files, or to submit the test classes in place of your code. If you do one of these (or something similar) then you will be due for any applicable late penalty up to the time you re-submit. *cssubmit* makes it easy to check your files after you have submitted them - do it!

Project Assessment

Your submission will be assessed on

- completeness: how many of the methods you have written;
- correctness: whether your methods implement the specifications exactly;
- clarity: whether your code is well-constructed and clear to read.

JUnit testing classes are provided to help you check your program before submission, but the correctness of your program remains your responsibility.

Completeness and Correctness guidelines (/20)

The marks available for each constructor and method are as follows. These numbers do not necessarily correlate with expected difficulty. They are designed so that everyone has a good chance of achieving a passing grade, but getting full marks is hard.

- **WordleAI**: *guessContains (1 mark), isConsistent (4 marks), isAllStars (1 mark), findWord (4 marks)*
- **WordleAIAnalyser**: *WordleAIAnalyser (1 mark), runExperiment (1 mark), runExperimentsWithAllWords (1 mark), runExperimentsWithWordsBetween (1 mark), getUnsolvedWords (1 marks), getGuessLetterFrequency (1 mark), getNumGuessesFrequency (1 mark), makeHistogram (3 marks)*

Methods will be assessed for correctness independently, so e.g. even if your guessWord is faulty, other methods like isInSecretWord or getWordsWithLength can get full marks.
Completeness and correctness will be evaluated (mostly) automatically using JUnit tests. You should, therefore, try to ensure the following.

- The code passes the tests as per the specification.
- Partial marks may be awarded for a method is that is partially correct.
- Validation should be **exactly** what is required by the assignment specification. Incorrectly rejecting a parameter will be penalized; do not add extra validation that is not asked for.
- Do not print things to System.out or System.err that are not asked for. Use the BlueJ debugger rather than println statements for debugging your code.
- Do not change the given file names or class names, or modify the signatures of existing methods.

Rubric for methods worth 1 mark:

- The method passes all tests including hidden tests (+1 marks)

Rubric for methods worth 3 marks:

- The method passes all tests including hidden tests (+3 marks)
- The method is wrong on hidden tests. However, it contains only minor bugs. (+2 marks)
- The method compiles and contains an attempt to solve the problem (+1 mark)

Rubric for methods worth 4 marks:

- The method passes all tests including hidden tests (+4 marks)
- It is not possible to get (+3 marks) for this question
- The method passes all provided tests BUT is wrong on hidden tests (+2 marks)
- The method compiles and contains an attempt to solve the problem (+1 mark)

Clarity guidelines (/5)

- Keep code as simple as possible for the job it is required to do.
- Use appropriate programming constructs for all implementations.
- Do not add fields to the classes.
- Give appropriate names to all variables.
- Lay out code neatly and with appropriate indentation, with lines no longer than eighty characters.
- If a method is particularly complex, add a brief comment explaining your strategy; but otherwise, do not comment code unnecessarily.

Rubric for code clarity:

- Follows all above guidelines (+5 marks)
- Follows almost all above guidelines but is significantly lacking in ONE category (+4 marks) (e.g. Good code but has no comments)
- Follows most above guidelines but is lacking in a few categories (+3 marks) (e.g. Good code but with incorrect indentation and no comments)
- Follows at least a few guidelines (+2 marks) (e.g. Has correct indentation but adds a field and has hard to understand variable names)
- Follows at least one guideline (+1 mark) (e.g. Code adds no additional fields but is otherwise hard to read)

Help!

The quickest way to get help with the project (unless you are sitting in a lab session) is via *help1001*. You can ask questions, browse previous questions and answers, and discuss all sorts of topics with both staff and other students.

Please read and follow these guidelines.

- Do not post project code on *help1001*. For obvious reasons, this behaviour undermines the entire assessment process: as such you will be liable for punishment under the University's Policy on Academic Conduct, even if the behaviour is unintentional.
- Before you start a new topic, check first to see if your question has already been asked and answered in an earlier topic. This will be the fastest way to get an answer!
- When you start a new topic, give it a meaningful subject. This will help all students to search and to make use of information posted on the page.
- Feel free to post answers to other students' questions. And don't be afraid to rely on other students' answers: we read everything on *help1001* and we will correct anything inaccurate or incomplete that may be posted from time to time.
- Be civil. Speak to others as you would want them to speak to you. Remember that when you post anonymously, this just means that your name is not displayed on the page; your identity is still recorded on our systems. Poor behaviour will not be tolerated.
- Do not post project code on *help1001*, or anywhere else online. This one is worth saying twice.

The project will also be a big topic for discussion in the labs and workshops. This is fine of course, but again be careful not to share code accidentally with other students.