CST2110 Resit Assessment Part 1

Deadline for submission 16:00, Friday 8th July, 2022

Please read all the assignment specification carefully first, before asking your tutor any questions.

General information

You are required to submit your work via the dedicated resit assignment link in the module myUnihub space by the specified deadline. This link will 'timeout' at the submission deadline. Your work will not be accepted as an email attachment if you miss this deadline. Therefore, you are strongly advised to allow plenty of time to upload your work prior to the deadline.

Submission should comprise a single 'ZIP' file. This file should contain a separate, cleaned[1], NetBeans project for the programming task described below. The work will be compiled and run in a Windows environment, i.e., the same configuration as the University networked labs and it is strongly advised that you test your work using the same configuration prior to cleaning and submission.

Your submission (ZIP file) must also include a completed declaration of authenticity using the form provided in the module myUnihub space. Your work will not be marked if you do not complete and submit the declaration.

The task described below requires your Java program to open and load words from an external file.

You must not, under any circumstances, include a hard-coded file path (within your program) to the file location i.e., a path that is only applicable to your own personal computer configuration. The file must be accessed *relative* to the NetBeans project folder. When viewing the project folder in Windows Explorer, the data file should be located as illustrated by Figure 1 below.

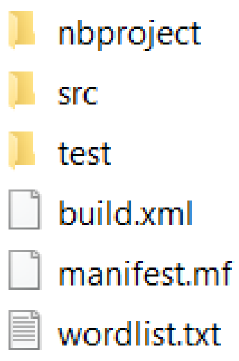

Figure 1: Location of the data file when viewed in Windows Explorer

Accordingly, if you select the 'Files' tab for the given NetBeans project, you will see the location of the external file relative to the project files, as illustrated by Figure 2 below.

[1] In the NetBeans project navigator window, right-click on the project and select 'clean' from the drop-down menu. This will remove .class files and reduce the project file size for submission.

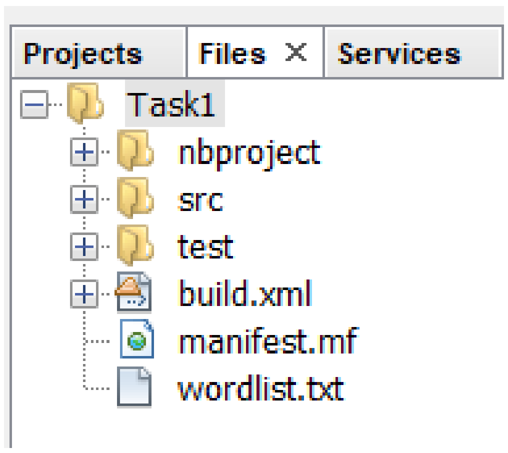

Figure 2: Location of the data file when viewed in NetBeans (select 'Files')

With the data file located in the correct place (i.e., the 'root' of the specific NetBeans project), then the following lines of code will correctly return a relative path to the data file.

```
String fileLocation = System.getProperty("user.dir");

String dataPath = fileLocation + File.separator + "wordlist.txt";
```

In order that no errors are made with respect to this structure, a NetBeans project template has been provided (so there really is no excuse!). If these instructions are not adhered to, and your solution includes a hard-coded path, a significant number of marks will be deducted.

Task (25 marks)

A NetBeans project template has been provided for you with the required data file located relative to the NetBeans project root folder. The data file contains a list of words to be used in a simple 'word guessing' game. The data file is called *wordlist.txt* and is provided as a *read-only* file that should not be altered.

Your task is to write a Java 8 program (as a NetBeans project) that selects one of the words provided in *wordlist.txt* file to be a 'secret word' and presents a console interaction for the User to guess the secret word in the style of a simple guessing game as described below.

This game must be entirely text-based, operating at the NetBeans console. This is a requirement of the task. Any submission that does not adhere to this requirement will receive zero marks.

Instructions

The program should randomly select one of the words provided in *wordlist.txt* file and present a console interaction for the User to guess the word. There should be a maximum of 10 guesses for any single guessing round of the game.

Examples of the required console output and interaction are provided in the Appendices to this document. Appendix A1 lists the console output when a User successfully guesses the word before the maximum number of guesses have been used. Following each guess, if the guessed letter exists in the secret word, then the display is updated to reflect the position(s) of that letter within the word. If the User guesses a letter that is not in the secret word, then that letter is added to a list of incorrect guesses, which is presented to the User after each guess so that s/he can keep track of previous incorrect guesses (if the User repeats a previous guess, then the console should simply print a message to this effect and ask the User to try again).

At the end of the round, the program should display the secret word to the User (in upper-case) and indicate if the User won or lost the round. The program should allow the User to either play again (without exiting the program) or exit the guessing game.

Appendix A2 illustrates a scenario in which the User fails to guess the secret word within the maximum number of guesses.

At any point in the guessing game, the User should be offered the option to 'give up', and this scenario is illustrated in Appendix A3.

Cheat feature

To make it easier for the tutor to assess your program, you are required to program a small 'cheat' feature into your program which enables the tutor to view the secret word prior to when a new game commences. To facilitate this, you should provide a simple Boolean flag (variable) in the main method. The default value is false (i.e., the User cannot see the hidden secret word), but when set to true (by the tutor) the secret word is displayed when the game starts. As illustrated below:

```
public class ApplicationRunner {

    public static void main(String[] args) {

        String dataFile = System.getProperty("user.dir") +
                                   File.separator + "wordlist.txt";

        boolean displaySecretWord = false;
```

```
          . . .
      }

  }
```

Failure to include the Boolean variable and the subsequent display of the secret word when it is set to true will result in a significant mark penalty.

Assessment

An assessment rubric is provided in the module handbook.

Your solution will be assessed according to robustness, functionality, and the correctness of program output. In addition, the assessment of this task will consider program modularity, formatting, and code style. In other words, credit will be gained for good use of methods, neat console-based text formatting, appropriate use of Java naming conventions, and appropriate code commenting. Note that you are only required to provide inline code commenting (not 'Javadoc' commenting). Code commenting should focus on explaining the key complexities of any methods and algorithms that you have used.

All grades are provisional and subject to adjustment during moderation of the overall module assessment which takes place at the end of the year with external examiner approval.

Appendix A1

Example console output for Task 2: User successfully guesses the word within allowed number of guesses.

```
Word Guessing Game

Play (1) or Exit (0) > 1

_ _ _ _ _ _ _
0 wrong guesses so far [ ]

Have a guess (lower case letter or * to give up)

> e

_ _ _ _ _ _ _
1 wrong guesses so far [ e ]

Have a guess (lower case letter or * to give up)

> a

_ _ _ a _ _ a

1 wrong guesses so far [ e ]

Have a guess (lower case letter or * to give up)

> i

_ i _ a _ _ a

1 wrong guesses so far [ e ]

Have a guess (lower case letter or * to give up)

> t

_ i _ a _ _ a

2 wrong guesses so far [ e t ]

Have a guess (lower case letter or * to give up)

> s

_ i _ a _ _ a

3 wrong guesses so far [ e s t ]

Have a guess (lower case letter or * to give up)

> o

_ i _ a _ _ a

4 wrong guesses so far [ e o s t ]

Have a guess (lower case letter or * to give up)

> l

_ i _ a _ _ a

5 wrong guesses so far [ e l o s t ]

Have a guess (lower case letter or * to give up)
```

```
> n

_ i _ a n _ a

5 wrong guesses so far [ e l o s t ]

Have a guess (lower case letter or * to give up)

> p

p i _ a n _ a

5 wrong guesses so far [ e l o s t ]

Have a guess (lower case letter or * to give up)

> r

p i r a n _ a

5 wrong guesses so far [ e l o s t ]

Have a guess (lower case letter or * to give up)

> h

p i r a n h a

5 wrong guesses so far [ e l o s t ]


The hidden word was PIRANHA

You Win! :-)


Play (1) or Exit (0) > 0
```

Appendix A2

Example console output for Task 2: User fails to guess the word within allowed number of guesses

```
Word Guessing Game

Play (1) or Exit (0) > 1

_ _ _ _ _ _ _
0 wrong guesses so far [ ]

Have a guess (lower case letter or * to give up)

> e

_ _ _ _ _ _ _
1 wrong guesses so far [ e ]

Have a guess (lower case letter or * to give up)

> a

_ _ _ _ _ a _
1 wrong guesses so far [ e ]

Have a guess (lower case letter or * to give up)

> i

_ _ _ _ _ a _
2 wrong guesses so far [ e i ]

Have a guess (lower case letter or * to give up)

> o

_ o _ _ _ a _
2 wrong guesses so far [ e i ]

Have a guess (lower case letter or * to give up)

> u

_ o _ _ _ a _
3 wrong guesses so far [ e i u ]

Have a guess (lower case letter or * to give up)

> t

_ o _ _ _ a _
4 wrong guesses so far [ e i t u ]

Have a guess (lower case letter or * to give up)

> s

_ o _ _ _ a _
5 wrong guesses so far [ e i s t u ]

Have a guess (lower case letter or * to give up)
```

```
> m

_ o _ _ _ a _

6 wrong guesses so far [ e i m s t u ]

Have a guess (lower case letter or * to give up)

> n

_ o _ _ _ a _

7 wrong guesses so far [ e i m n s t u ]

Have a guess (lower case letter or * to give up)

> f

_ o _ _ _ a _

8 wrong guesses so far [ e f i m n s t u ]

Have a guess (lower case letter or * to give up)

> l

l o _ _ _ a _

8 wrong guesses so far [ e f i m n s t u ]

Have a guess (lower case letter or * to give up)

> g

l o _ _ _ a _

9 wrong guesses so far [ e f g i m n s t u ]

Have a guess (lower case letter or * to give up)

> k

l o _ k _ a _

9 wrong guesses so far [ e f g i m n s t u ]

Have a guess (lower case letter or * to give up)

> q

l o _ k _ a _

10 wrong guesses so far [ e f g i m n q s t u ]


The hidden word was LOCKJAW

You lose :-(


Play (1) or Exit (0) > 0
```

Appendix A3

Example console output for Task 2: User gives up after a few guesses

```
Word Guessing Game

Play (1) or Exit (0) > 1

_ _ _ _ _ _ _ _ _
0 wrong guesses so far [ ]

Have a guess (lower case letter or * to give up)

> w

_ _ _ _ _ _ _ _ _
1 wrong guesses so far [ w ]

Have a guess (lower case letter or * to give up)

> g

_ _ _ _ _ _ _ _ _
2 wrong guesses so far [ g w ]

Have a guess (lower case letter or * to give up)

> k

_ _ _ _ _ _ _ _ _
3 wrong guesses so far [ g k w ]

Have a guess (lower case letter or * to give up)

> *

You gave up!

_ _ _ _ _ _ _ _ _
3 wrong guesses so far [ g k w ]


The hidden word was CAMPERVAN

You lose :-(


Play (1) or Exit (0) > 0
```