La Trobe University
Department of Computer Science & Information Technology
CSE1IOO/CSE4IOO Assignment Specification
Semester 2, 2019

Objectives: the aim of this assignment is to practise analysing a problem in an object-oriented manner, and designe and implement an object-oriented solution using JAVA language.

Due Date: 10:00am Monday October 14th, 2019.

1 General Information

General information about the assignment is specified in this section.

1.1 Assessment

This assignment contributes 30% of the final assessment for the subject.

1.2 Submission Instruction

Submit the electronic copy of your assignment via the 'Online Assignment Submission' system at http://students.cs.latrobe.edu.au/student-tools/online-assign-submit/.

Alternately, you can also submit directly through the latcs8 server. In the Putty window, type:

```
submit IOO <filename>
```

Where `<filename>` is the name of the file you want to submit. You should run this for each file that you intend to submit.

NOTE 1: While you are free to develop the code for this assignment on any operating system, your solution must run on the latcs8 system.
NOTE 2: Please DO NOT zip your source code.
NOTE 3: You should only submit your .java files.
NOTE 4: The submission folder will only become available 3 days before the due date.

Submission after the deadline will incur a penalty of 5% of the final assignment mark per day. If you have encountered difficulties that lead to late submission or no submission, you should apply for special consideration. No assignment is accepted after 4 days' delay.

1.3 Academic Integrity

This is an individual assignment. When submitting the assignment, students are required to submit their own work only. La Trobe University treats plagiarism seriously. When detected, penalties are strictly imposed.

Further information can be found on https://www.latrobe.edu.au/students/admin/academic-integrity

1.4 Programming Language

The program is required to be developed in JAVA.

1.5 Marking Scheme Overview

The assignment consists of two parts:

(1) The core tasks, as shown in sections 3.1 – 3.6
(2) An additional task in section 3.7 (Task 3.7) This is a mandatory task for CSE4IOO students. For CSE1IOO students, this task is optional. A maximum of 10% bonus marks will be awarded if a CSE1IOO student completes or partially completes the task.

Assignment marking is based on the following guidelines

(1) *Implementation (Execution of code)* (80%) (Do all parts of the program execute correctly? Does the program behave according to requirements of the assignment?)
Note that your program must satisfy the requirement 3.6. Up to 40% of mark deduction will be made for submitting a program with a different data structure. This also means any direct or indirect decedent classes of `Collection` interface in `JAVA API`, such as `LinkedList<E>`, `ArrayList<E>`, `Vector<E>` or `HashSet<E>` etc are not allowed in the assignment.
(2) *Program Design and Structure* (15%) (Does the program solve the problem in a well-designed manner?)
(3) *Layout and Documentation* (5%) (Does the code follow the Coding Standard?)

1.6 Execution Test

The execution test for the assignment is conducted during the lab session in Week 12 (*i.e.* between 21st and 25th October). During the test, your tutor will run the program with you while marking it. The test will be conducted from the submission area, not from student accounts. Please ensure that your code runs on latcs8 before submission.

2 Problem Description

WordLink is an English vocabulary game for two players. Players are required to present English words in turns, and the first character of the word must be the same as the last character of the previous word. For example, while player ***A*** and player ***B*** are playing the game, if ***A*** presents the word "grass", then ***B*** has to give a word starting with 's' which is the last character of "grass". If ***B*** presents "sunny", then ***A*** needs to supply a word starting with 'y'. A word cannot be used twice in one game. The game continues till one of the players loses. There are four scenarios in which a player loses the game. These are

(1) the player cannot find a word to continue the game.
(2) the player has presented a word which starts with a different character.
(3) the player has supplied a repeated word.
(4) the spelling of the word is incorrect

In any case, if one player is lost, then the other player wins the game. A number of examples of the game are listed as below.

Example 1
(Player ***A***) "grass" – (Player ***B***) "sunny" – (***A***) "yellow" – (***B***) "world" – (***A***) "dog" – (***B***) "grass"
The game is ended and Player ***A*** wins. This is because "grass" is used twice.

Example 2
(Player ***A***) "grass" – (Player ***B***) "sunny" – (***A***) "yellow" – (***B***) "world" – (***A***) "dog" – (***B***) "out"
The game is ended and Player ***A*** wins. This is because player B is expected to present a word starting with 'g', therefore "out" is not a proper choice.

Example 3
(Player ***A***) "grass" – (Player ***B***) "sunny" – (***A***) "yellow" – (***B***) "world" – (***A***) "doog"
The game is ended and Player ***B*** wins. This is because "doog" is incorrectly spelt.

In this assignment, you are required to build a program that plays WordLink with a kindergarten pupil. The pupil is player ***A*** and your program acts as player ***B***.

A text file (`dictionary.txt`) containing a set of English words will be provided. You can assume that `dictionary.txt` contains all English words that a kindergarten pupil may know. Words to be used in playing the game must be selected from the file.

Each game should start by letting player ***A*** enter a word. Your program then needs to validate the word (*ie.* if the word is included in `dictionary.txt`). If the word is invalid, then the game is terminated and your program (player ***B***) wins; otherwise your program needs to search the dictionary to find and enter a proper word. Your program will then let player ***A*** to type a word to continue the game. This can go back and forth a number of times till either your program or player ***A*** wins.

3 Functional and Non-functional Requirements

Your program must satisfy the following functional and non-functional requirements.

3.1 Menu Driven

The program should be menu driven. It displays the menu displayed as Figure 1 at the start and waits for the player to choose a function.

```
WordLink

A. set the difficulty level
B. display the dictionary
C. insert a word to the dictionary
D. play the game
E. exit

Select a function from the menu:
```

Figure 1 menu

If A is selected, it allows the player to set a difficulty level. Details are described in 3.2.
If B is selected, it displays all words in the dictionary as described in 3.3.
If C is selected, it inserts a new word into the dictionary as described in 3.4.
If D is selected, it starts playing the game. Refer to 3.5 for details.
If E is selected, it saves all changes to `dictionary.txt`, and terminates the program execution.

3.2 Set the difficulty level

Your program should provide two difficulty levels for the game - level 1 and level 2. `dictionary.txt` contains two sets of words – level 1 and level 2 words. When playing at level 2, all words in `dictionary.txt` will be used, while level 1 games use only level 1 words. Consequently, level 2 games are more difficult than level 1. At the start of your program execution, the difficulty level is 1 by default.

If the player chooses to set the difficulty level, your program should prompt the player and wait for an integer (1 or 2) input. The difficulty level is then set up.

A sample of `dictionary.txt` is displayed in figure 2.

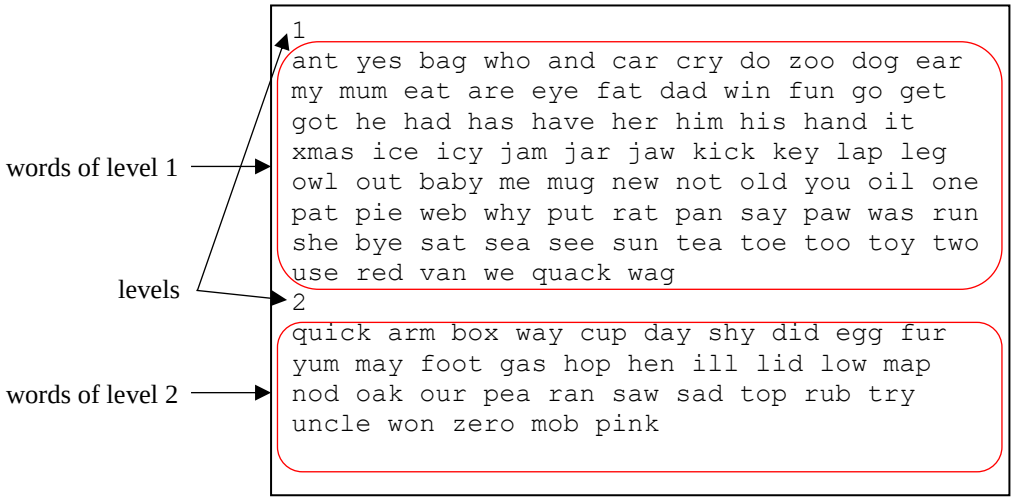

Figure 2 sample of `dictionary.txt`

3.3 Display the dictionary

If this function is selected, your program displays on the monitor all words in the dictionary (including those being added). It displays words level by level with lower level at the front. Within each level, words are displayed alphabetically in an ascending order. They should be displayed 7 words in a line and 5 lines for a screen. The player can press any key to display the next screen.

3.4 Insert a word to the dictionary

The player can insert a new word into the dictionary. To insert a new word, the player needs to provide the word and the difficulty level. Your program must check to ensure the absence of the word before adding. If the word is existing, then insertion cannot be performed and the player should be informed.

3.5 Play the game

A game starts by your program prompting the player to enter a word. After a word is entered, your program checks if the word exists in the dictionary. If it isn't, then the player loses and the game is over. If it is, then your program searches the dictionary and selects the first proper word to continue the game. For instance, if "`sad`", "`sat`", "`saw`", "`say`", "`sea`", "`see`", "`she`", "`shy`", "`sun`" are all available for selection, your program chooses the first word which is "`sad`".

3.6 The dictionary

After your program is started, it should read from `dictionary.txt` to create the dictionary. The dictionary in your program must be an array of linked lists shown in Figure 3. Each node represents a word and its level. Words must be sorted alphabetically in an ascending order on the linked list. If a new word is added, then the word with its level must be inserted to the linked list at a proper position.

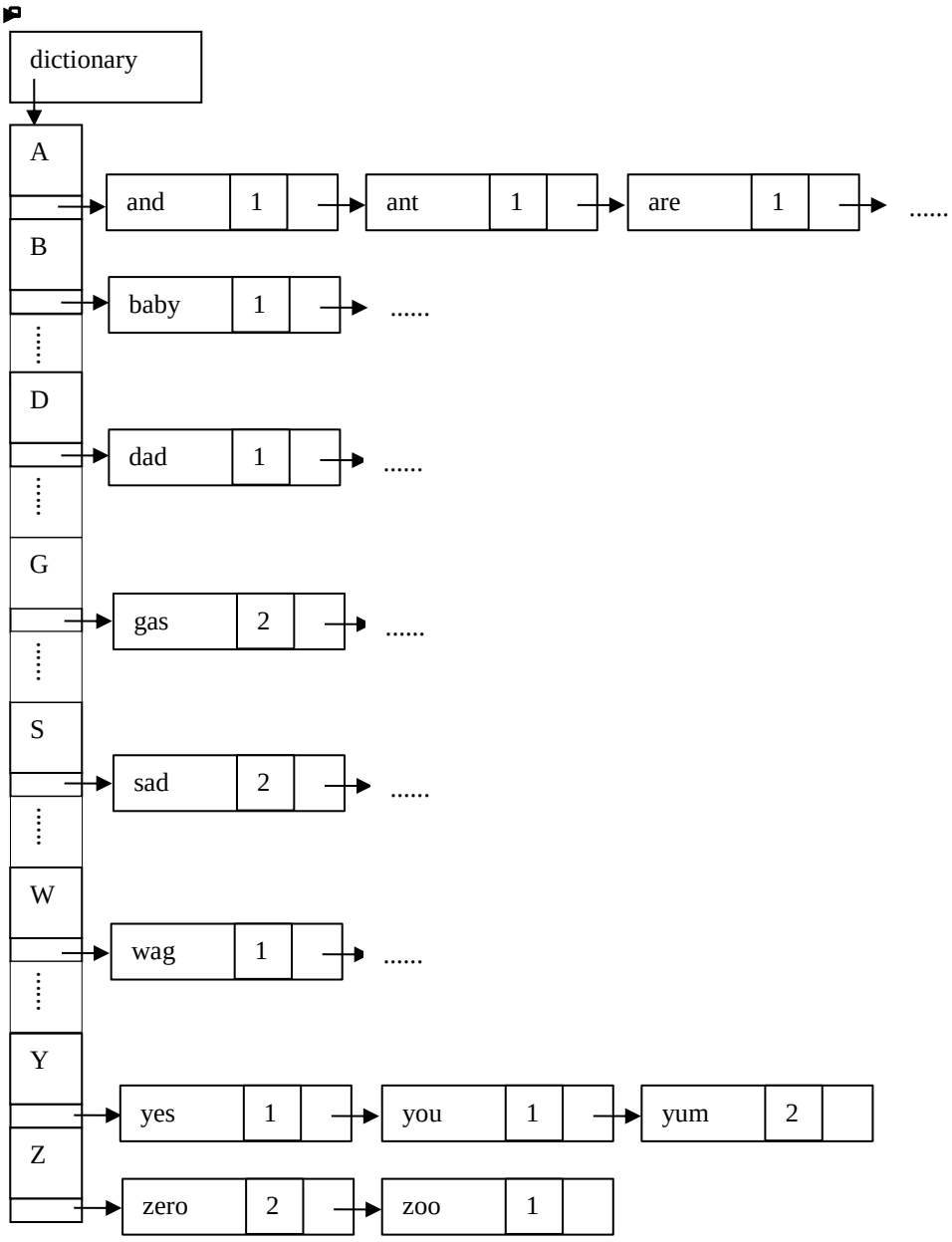

Figure 3 Structure of the dictionary

3.7 Bonus Task (Mandatory for CSE4IOO students, optional for CSE1IOO)

Instead of only being able to play against the computer, add a menu option for a two-player game and implement its functionality. A sample run of this is at the end of section 6.

4 Code Segments

Below are code segments that **must** be included in your program.

```
public class DictionaryNode {
//object of the class represents a single word
      protected String word; // word to be stored
      private int level; // level of the word
      private DictionaryNode next;
      public DictionaryNode(String _word, int _level) {
      //add your implementation for the constructor
      }
      //add any other attributes or methods if needed
}

public class ListOfNodes {
//object of the class represents a linked list of words starting
//with a specific character.
      private DictionaryNode head = null; //head of the linked list
      //add any other attributes or methods if needed
}

public class Dictionary {
//object of the class represents the whole dictionery
      private ListOfNodes[] data;
      //add any other attributes or methods if needed
}
```

5 Program Development

The following is a suggested breakdown for completing this assignment:

Task 1 Creating the Menu
Create the menu display, collect user's input and write a method stub for each menu option. Ensure that the correct function is called.

Task 2 File Handling
In Task 2, you need to consider all issues related to reading the text file (`dictionary.txt`) to obtain words and their levels. You can simply display contents read from the text file to ensure file reading is correctly conducted.

Task 3 Defining Classes
In this task, implement major classes for the assignment. This includes the necessary menu and interactions with the player for choosing options etc.

Task 4 Linked Lists
Implement classes required for linked lists and dictionary. Implement methods (such as insertInOrder, search etc) which are operations associated with the list and any other methods. Thoroughly test your linked list before integrating it into your program.

6 Sample of Execution

A sample of the program execution is as follows.

```
% java WordLink

   WordLink

      A set the difficulty level
      B display the dictionary
      C insert a word to the dictionary
      D play the game
      E exit

   Select a function from the menu: A


   Set the difficulty level

   The current difficulty level is 1. Type the new level: 2
   The difficulty level has now been set as 2.


   WordLink

      A set the difficulty level
      B display the dictionary
      C insert a word to the dictionary
      D play the game
      E exit

   Select a function from the menu: B


   Display the dictionary

Level 1
and   ant   are   baby  bag   bye   car
cry   dad   do    dog   ear   eat   eye
fat   fun   get   go    got   had   hand
has   have  he    her   him   his   ice
icy   it    jam   jar   jaw   key   kick

press a key to continue ...

lap   leg   me    mug   mum   my    new
not   oil   old   one   out   owl   pan
pat   paw   pie   put   quack rat   red
run   sat   say   sea   see   she   sun
tea   toe   too   toy   two   use   van

press a key to continue ...
```

```
wag   was   we    web   who   why   win
xmas  yes   you   zoo

press a key to continue ...


Level 2
arm   box   cup   day   did   egg   foot
fur   gas   hen   hop   lid   ill   low
map   may   mob   nod   oak   our   pea
pink  quick ran   rub   sad   saw   shy
top   try   uncle way   won   yum   zero

press a key to continue ...


   WordLink

      A set the difficulty level
      B display the dictionary
      C insert a word to the dictionary
      D play the game
      E exit

   Select a function from the menu: C

Insert a word to the dictionary

Enter the word: wear
Difficulty level: 1

"wear" is inserted.


   WordLink

      A set the difficulty level
      B display the dictionary
      C insert a word to the dictionary
      D play the game
      E exit

   Select a function from the menu: C

Insert a word to the dictionary

Enter the word: quack
Difficulty level: 2

"quack" exists in the dictionary. Insertion aborted.
```

```
   WordLink

      A set the difficulty level
      B display the dictionary
      C insert a word to the dictionary
      D play the game
      E exit

   Select a function from the menu: B


   Display the dictionary

Level 1
and   ant   are   baby  bag   bye   car
cry   dad   do    dog   ear   eat   eye
fat   fun   get   go    got   had   hand
has   have  he    her   him   his   ice
icy   it    jam   jar   jaw   key   kick

press a key to continue ...


lap   leg   me    mug   mum   my    new
not   oil   old   one   out   owl   pan
pat   paw   pie   put   quack rat   red
run   sat   say   sea   see   she   sun
tea   toe   too   toy   two   use   van

press a key to continue ...


wag   was   we    wear  web   who   why
win   xmas  yes   you   zoo

press a key to continue ...


Level 2
arm   box   cup   day   did   egg   foot
fur   gas   hen   hop   lid   ill   low
map   may   mob   nod   oak   our   pea
pink  quick ran   rub   sad   saw   shy
top   try   uncle way   won   yum   zero

press a key to continue ...
```

```
   WordLink

      A set the difficulty level
      B display the dictionary
      C insert a word to the dictionary
      D play the game
      E exit

   Select a function from the menu: D

play the game (Level 2)

Enter a word: tea
tea – and - day
tea – and – day – yes - sun
tea – and – day – yes – sun – new - what

“what” doesn’t exist in the dictionary. You didn’t win.



   WordLink

      A set the difficulty level
      B display the dictionary
      C insert a word to the dictionary
      D play the game
      E exit

   Select a function from the menu: D

play the game (Level 2)

Enter a word: may
may – yes - say
may – yes – say – you - uncle
may – yes – say – you – uncle – ear - red
may – yes – say – you – uncle – ear – red – dad - day
may – yes – say – you – uncle – ear – red – dad – day – yum - my

Well done! You win.



   WordLink

      A set the difficulty level
      B display the dictionary
      C insert a word to the dictionary
      D play the game
      E exit

   Select a function from the menu: E

Updating dictionary.txt ... Bye
```

A sample of the program execution containing the additional two-player task is as follows.

```
% java WordLink

WordLink

        A set the difficulty level
        B display the dictionary
        C insert a word to the dictionary
        D play the game (1 player)
        E exit
        F play the game (2 players)

Select a function from the menu: f

Play the game (2 Players - Level 1)

P1: Enter a word: has
P2:  - has - see
P1:  - has - see - ear
P2:  - has - see - ear - run
P1:  - has - see - ear - run - now
"now" isn't in the level 1 dictionary!

--Player two wins!--


WordLink

        A set the difficulty level
        B display the dictionary
        C insert a word to the dictionary
        D play the game (1 player)
        E exit
        F play the game (2 players)

Select a function from the menu: F

Play the game (2 Players - Level 1)

P1: Enter a word: wag
P2:  - wag - go
P1:  - wag - go - oil
P2:  - wag - go - oil - line
"line" isn't in the level 1 dictionary!

--Player one wins!--


WordLink

        A set the difficulty level
        B display the dictionary
        C insert a word to the dictionary
        D play the game (1 player)
        E exit
        F play the game (2 players)

Select a function from the menu: e

Updating dictionary.txt ... Bye
```