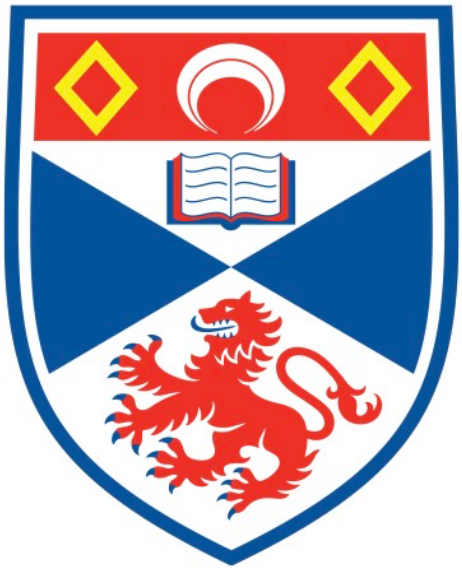
Personal report

Hangman

220032093



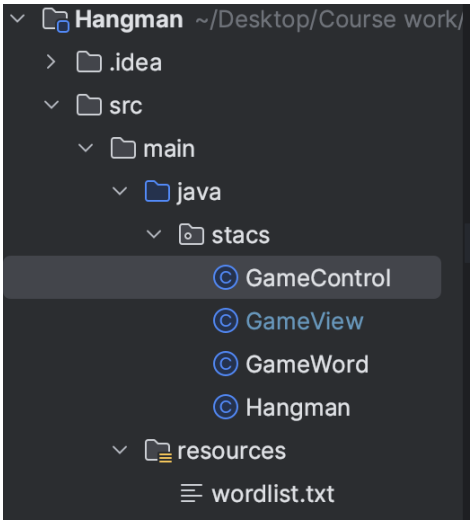
University of St. Andrews
CS5031- Software Engineering Practice
Academic Year of 2024, semester 2 – Assessment 1

Upload date February 12, 2024

An overview of Hangman implementation:

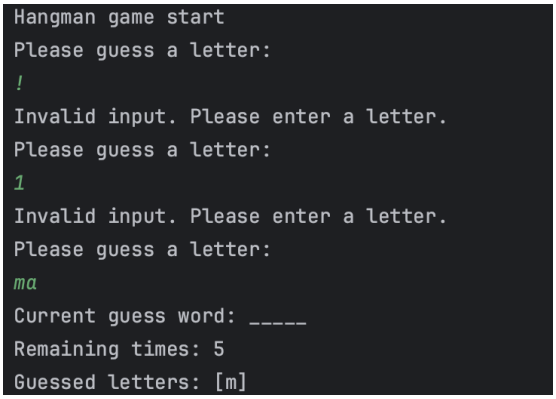
1. Hangman game code classification:

- **GameControl Class：** Mainly responsible for processing the logic of the game, including whether the letters guessed in a single time are correct, processing the number of word guesses, recording the guessed words, and judging whether the remaining times of the game match the guessed words, etc.
- **GameView Class：** The status information of the game displayed in the console includes Hangman's drawing logic, the words that the player has guessed, the words that the player has guessed correctly, the remaining number of guesses, and other prompt information when guessing incorrectly.
- **GameWord Class：** Responsible for randomly picking out a word from the given wordlist folder for players to guess.
- **Hangman Class：** Store Main method(The entrance of a program)

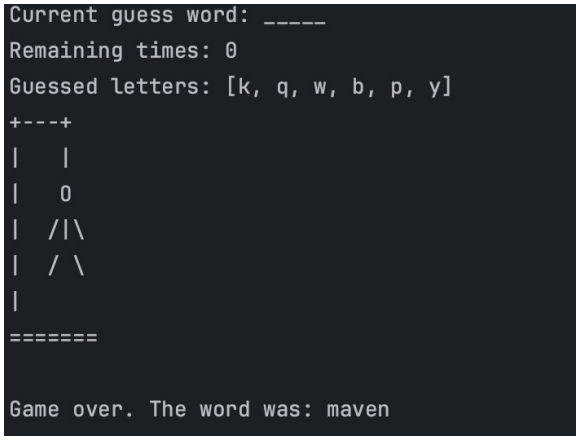


2. Interpretation of gameplay：

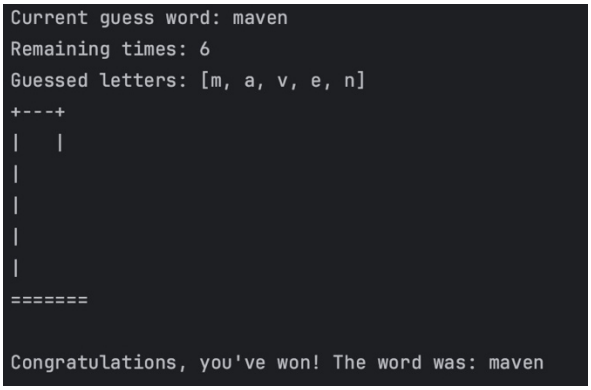
After running the Main method in Hangman, the player will first enter the game interface. The interface will prompt the player to enter a letter first. If the input is not letters but other illegal characters, the page will prompt the player that the input is invalid and they must guess again. If multiple letters are entered, the game will take the first letter entered as the guessed letter for this round of the game.



Regardless of whether the guess is correct or incorrect, each letter guessed will be recorded in "Guessed letters:". If the letters in the word are guessed correctly, the underline after "Current guess word:" will be replaced by the guessed word. If player guess wrong, the "Remaining times" displayed on the interface will be reduced by one. At the same time, for every wrong guess, the system will draw a part of Hangman's body.



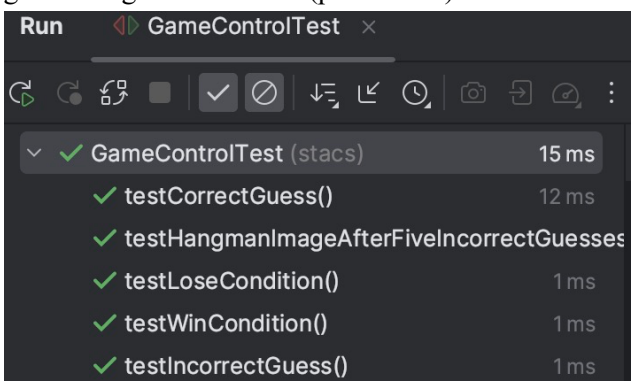
At the end of the game, if Hangman's body has not been fully drawn and the player guesses the letters of all the words correctly, the game interface will display "You've won" and the word to be guessed.



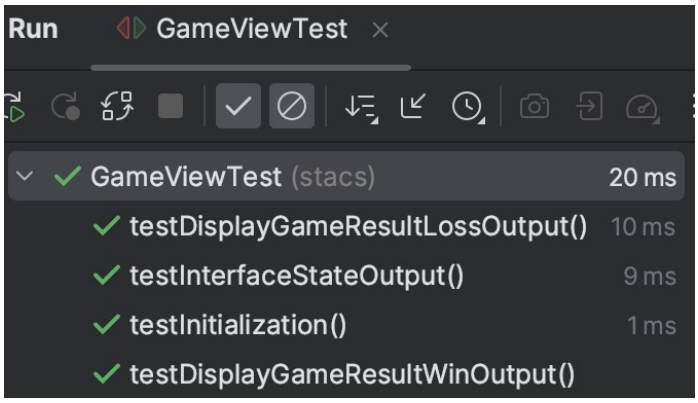
A description of test-driven approach：

1. Test code classification：

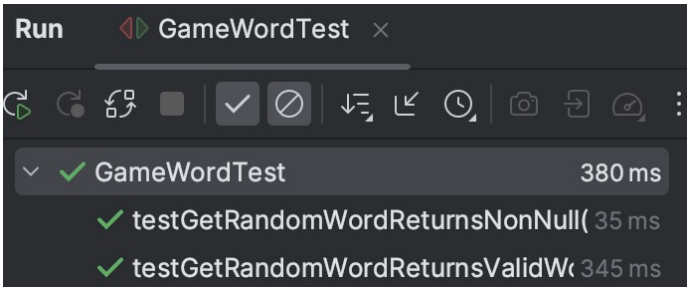
(1) Verify that the logic of the game is correct (pass 5 of 5)



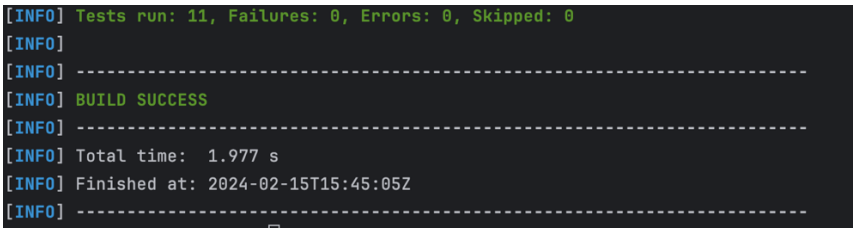
(2) GameViewTest： Test whether the interface display is normal (pass 4 of 4)



(3) GameWordTest： Test whether the randomly selected words are correct (pass 2 of 2)



All tests pass (11/11)



2. Test code interpretation:

2.1 GameControlTest Class：

（1）testCorrectGuess：

If the letter is guessed correctly, the method returns the correct boolean value, the word is updated correctly and the word is added to alreadyGuessedLetters.

（2）testIncorrectGuess：

If the letter is guessed wrong, can the method return the correct boolean value, check if the remaining times are reduced and the wrong word is added to alreadyGuessedLetters.

（3）testWinCondition：

Test your winning conditions by guessing all the correct letters.

（4）testLoseCondition：

Try playing with all the guesses possible to test the failing condition.

（5）testHangmanImageAfterFiveIncorrectGuesses：

Test whether the Hangman images after the fifth wrong guess are equal.

2.2 GameViewTest Class：

（1）testInitialization：

Check whether the game is initialized correctly.

（2）testInterfaceStateOutput：

Check whether the game's output interface is normal.

（3）testDisplayGameResultWinOutput：

When a player wins the game, whether the interface information output correct or not.

（4）testDisplayGameResultLossOutput：

When a player loses the game, whether the information on the output interface correct or not.

2.3 GameWordTest 类：

（1）testGetRandomWordReturnsNonNull：

Call the getRandom method in GameWord to see if the method can always return a non-empty word.

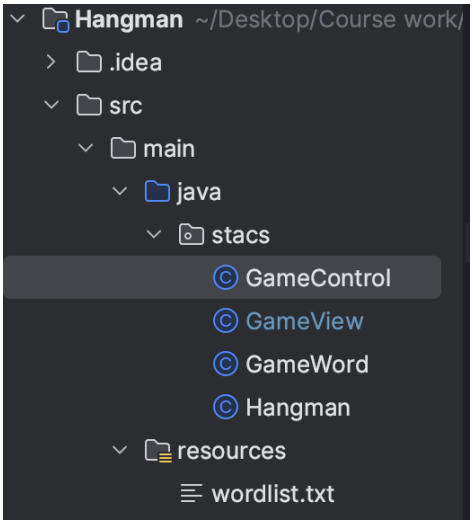
（2）testGetRandomWordReturnsValidWord：

Check whether getRandom can return a different word each time.

Refactoring approach throughout development：

1.Code classification:

Categorize code according to different functions.



2. Method optimization:

Split a single method into multiple methods for calling to improve the readability of the code and reduce the coupling of the code.

```java
public boolean guessResult(char guessedLetter) {
    guessedLetter = Character.toLowerCase(guessedLetter);// Tra

    boolean isCorrect = updateCurrentGuessWord(guessedLetter);

    updateRemainingTimes(isCorrect); // Choose whether you need

    recordGuessedLetter(guessedLetter);// Record the letters th

    return isCorrect; // Return the guessed result.
}
```

```java
private boolean updateCurrentGuessWord(char guessedLetter) {
    boolean isCorrect = false;
    for (int j = 0; j < currentGuessWord.length(); j++) {
        if (guessWord.charAt(j) == guessedLetter) {
            currentGuessWord.setCharAt(j, guessedLetter);
            isCorrect = true;
        }
    }
    return isCorrect;
}
```

```java
private void updateRemainingTimes(boolean isCorrect) {
    if (!isCorrect) {
        remainingTimes--;
    }
}
```

```java
private void recordGuessedLetter(char guessedLetter) {
    if (!alreadyGuessedLetters.contains(guessedLetter)) {
        alreadyGuessedLetters.add(guessedLetter);
    }
}
```

3. Code encapsulation:

Make the properties private and external classes can only access them through the public methods of the class.

```java
public class GameControl {
    4 usages
    private final String guessWord; //Reserving the word that user need to guess
    5 usages
    private final StringBuilder currentGuessWord; //Reserving the current state
    3 usages
    private final List<Character> alreadyGuessedLetters = new ArrayList<>(); //R
    4 usages
    private int remainingTimes = 6; //Reserving current remaining times.
```

Set updateCurrentGuessWord(), updateRemainingTimes(), recordGuessedLetter() as private and can only be accessed in inner classes:

```java
private boolean updateCurrentGuessWord(char guessedLetter) {
    boolean isCorrect = false;
    for (int j = 0; j < currentGuessWord.length(); j++) {
        if (guessWord.charAt(j) == guessedLetter) {
            currentGuessWord.setCharAt(j, guessedLetter);
            isCorrect = true;
        }
    }
    return isCorrect;
}
```

```java
private void updateRemainingTimes(boolean isCorrect) {
    if (!isCorrect) {
        remainingTimes--;
    }
}
```

```java
private void recordGuessedLetter(char guessedLetter) {
    if (!alreadyGuessedLetters.contains(guessedLetter)) {
        alreadyGuessedLetters.add(guessedLetter);
    }
}
```

4. Code comments:

This coursework uses the Maven configuration plug-in to generate Javadoc, and each method/class in the code has been commented in detail.

Package stacs

package stacs

Classes

Class	Description
GameControl	The GameControl class is responsible for managing the state and logic of the Hangman game.
GameView	The GameView use to show the game's information of state.
GameWord	This class use to generate the random word from word list file.
Hangman	