Améliorations du jeu

Temps de lecture : 3 minutes

Aller plus loin

Dans la leçon précédente, le jeu du Pendu fonctionne et nous pouvons y jouer. Cependant, il y a quelques améliorations que l'on peut apporter pour rendre le programme plus robuste et plus agréable à utiliser :

1. Vérifier que le joueur rentre bien un seul caractère lors de la saisie.
2. Afficher au joueur les lettres qu'il a déjà saisi
3. A la fin d'une partie demander au joueur s'il veut rejouer. Si oui, recommencer une partie.
4. Documenter avec la `javadoc` au moins les bouts de code `public` du programme. C'est une bonne habitude à prendre dès maintenant qui sera très utile dans votre carrière de développeurs dès lors que vous travaillerez avec d'autres développeurs.

Proposition de solution

Voici une proposition de solution :

Classe `Main`

```
import com.dyma.game.GuessGame;

import java.util.Random;
import java.util.Scanner;

/**
 * Class of the entrypoint of the Guess Game.
 */
public class Main {

    /**
     * Entry point of the Guess Game. Contains the main al
```

```
gorithm of the game.
     */
    public static void main(String[] args) {
        final var random = new Random();
        final var words = "abuser crottes fleches continen
tal babiole etoile bougie coup coeur malade".split(" ");
        final var lifePoints = 10;
        var wordToGuess = words[random.nextInt(words.lengt
h)];
        var game = new GuessGame(wordToGuess, lifePoints);

        System.out.println("Début du jeu.");

        while(true) {
            System.out.println(game);
            final var letter = scanLetter("Entrez une lett
re : ");

            game.guessLetter(letter);
            if (game.isLost()) {
                System.out.println("Perdu !");
            }
            if (game.isWon()) {
                System.out.println("Gagné !");
            }
            if (game.isWon() || game.isLost()) {
                System.out.println(game);
                var replayAnswer = scanLetter("Rejouer ?
(y, Y, o, O)");
                if (replayAnswer == 'y' || replayAnswer ==
'Y' || replayAnswer == 'o' || replayAnswer == 'O') {
                    wordToGuess = words[random.nextInt(wor
ds.length-1)];
                    game = new GuessGame(wordToGuess, life
Points);
                } else {
                    break;
                }
            }
        }
    }

    private static char scanLetter(String question) {
        final var scanner = new Scanner(System.in);
        Character letter = null;
        do {
            System.out.println(question);
            var input = scanner.nextLine();
            if (input.length() == 1) {
                letter = input.charAt(0);
            }
        } while (letter == null);
        return letter;
    }

}
```

Classe GuessGame

```
package com.dyma.game;

import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;
import java.util.Set;

/**
 * Class responsible of representing the Guess Game. Provi
des methods to :
 * - validate if the game is won or lost
 * - validate if a given letter is considered discovered o
r not in the secret word
 */
public class GuessGame {

    /**
     * Stores the secret word that the player wants to dis
cover
     */
    private final List<Character> secretWord = new ArrayLi
st<>();
    /**
     * Stores the remaining number of life points.
     */
    private int lifePoints;
    /**
     * Stores letters discovered by the player. '_' stored
for not discovered letters.
     */
    private final List<Character> guessWord = new ArrayLis
t<>();
    /**
     * Stores letters that the player has used to try to d
iscover the secret word.
     */
    private final Set<Character> guessedLetters = new Hash
Set<>();

    /**
     * Build a Guess Game object.
     * @param secretWord the secret word the player has to
discover.
     * @param lifePoints the number of retries allowed to
discover the secret word.
     */
    public GuessGame(String secretWord, int lifePoints) {
        for (char c : secretWord.toCharArray()) {
            this.secretWord.add(c);
        }
        this.lifePoints = lifePoints;
        for (int index = 0; index < secretWord.length(); i
ndex++) {
            this.guessWord.add('_');
        }
    }

    /**
     * Algorithm which verifies if a char given by the pla
yer is discovered in the secret word.
     * @param letter The letter to validate in `secretWord
` and `guessWord`.
     */
    public void guessLetter(char letter) {
        var isGoodLetter = secretWord.contains(letter) &&
!guessWord.contains(letter);
        guessedLetters.add(letter);
        if (isGoodLetter) {
            var index = 0;
            for (char c : secretWord) {
                if (c == letter) {
                    guessWord.set(index, c);
                }
                index++;
            }
        } else {
            lifePoints -= 1;
        }
    }

    /**
     * Check if the game is lost.
     * @return boolean true if the game is lost, false oth
erwise.
     */
    public boolean isLost() {
        return lifePoints <= 0;
    }

    /**
     * Check if the game is won.
     * @return boolean true is the game is won, false othe
rwise.
     */
    public boolean isWon() {
        return !guessWord.contains('_');
    }

    @Override
    public String toString() {
        return "mot à deviner : " + guessWord +
                " | points de vie : " + lifePoints +
                " | lettres essayées : " + guessedLetters;
    }
}
```