

Terminal Assessment

2020/May

***Java Words Game***

***( joc Fazan )***

***Romanian denomination***

**Software**

**Development**

***Lecturer:***

*Adriana E. Chis*

***Author:***

*Tudor George Pascu*

***Student number:***

*X19209690*

***“Java Words Game” is a turn-based game, where the players take turns to***

***provide one word at a time according to the rules of the game.***

* **“*Java Words Game*” game rules:**

1. **The game is played between two players, both human users.**
2. **Each player begin the game with the same number of lives and with 0 for each one.**
3. **First turn starts with a *random letter and a random player*, which has to play that letter; in our case, the computer makes the selection.**
4. **First stage of the game consists in that the randomly selected player has to write down in the program, a word that start with the randomly selected letter, at the beginning at the game.**
5. **The next player has to also input a word, but this time starting with the with the *last two letters* of the last word provided by the other player.**
6. **All the words has to be formed from at least 3 letters, and to be correctly wrote in English language, as it will be passed through specific input validation.**
7. **The game contains a limited 1024 English word which will be the base of the words used in the game.**
8. **Flowingly, each player will had to add a new word starting with the *last two letter from the last word player by the last player*.**
9. **If any player doesn´t know any word starting with a determinate sequence of two letters , then that player will lose a live, and the game will start from the beginning.**
10. **Each time when a correct word will be played, by input by any player, that player will receive *a concrete number of points*, described in a specific rule.**
11. **The main objective of the game is not to lose the lives that one has, and therefor introduce as many words as possible. If this rule is not followed the player who firstly waste his lives, will lose the game *and the one with lives remaining will win the game*.**
12. **At the end of the game the players will see a summary with their remaining lives and total gathered points along the game.**

* ***Specific rule for students whose student identification penultimate numbers is 9:***

**The player receives the same amount of points as the number of characters in the word when the word contains no vowels; otherwise 1 point.**

**Example:** **“gym” – 3 points**

**"rhythm" – 6 points**

**“song” – 1 point**

* ***Specific approach for students whose student identification antepenultimate numbers is 6 (pair):***

**The players must be asked at the beginning of the application how many games they would like to play, and allow the players to play that amount of games.**

* **Description of the input, main processing and output (IPO):**

**Here I am going to describe the IPO processes that take place in the game:**

* **Outside the game (Can be left empty, default player 1 / 2 will be used): Input the name of Player 1 and Player 2**

|  |  |  |
| --- | --- | --- |
| **INPUT** | **PROCESS** | **OUTPUT** |
| Inputted Character from the user | Save the Input in a players variable | Print the Inputted name in players  variable |

I added a customization of the default Player 1 and 2, before beginning the game: The player will be asked to enter his name and that will be used in the game as being one of the players name. The name is not required. It will appear empty but still will be properly described as player 1 or 2.

* **Inside the game: Main game –first word:**

|  |  |  |
| --- | --- | --- |
| **INPUT** | **PROCESS** | **OUTPUT** |
| Number of times the player wants to play:  **(Take integer)**  Take a string inputted by the user, by Scanner method.  **(Take strings)** | Introduce the input in a variable, in the main loop and play the x times the player has inputted it.  **Check if the input** is numerical and bigger than 0.  Validate the word in different ways:  1. The String **starts** with the **same character** as selected per the random function and converted in a String so that can be used in a String Variable**(Compare strings)** | The loop will repeat as per the players input and the finish  **(Repeat loop until condition is fulfilled)** |

Continuation:

|  |  |  |
| --- | --- | --- |
| **INPUT** | **PROCESS** | **OUTPUT** |
| Take a string inputted by the user, by Scanner method.  **(Take strings)** | 2. The introduced word in form of a String **is identical**, no errors, or gibberish text, non-numerical content; to one of the words, Strings in the file “Limited Vocabulary.java” which means that the String is a valid English word for the game. **(Compare Strings)**  3. If the input is equal to “-”  Then will be translated as being an error and will take to the deduction of a live from the lives given at the start of the game. **(Compare Strings)**  4. If the String introduced is not found in the Array of words in the “Limited Vocabulary.java**”. (Compare Strings)** | 1. If the word is confirmed the loop will change the turn and prompt a new word to be introduced by the other player. **(Repeat main loop, introduce new condition in the initial loop)**  2. **Calculate points** as per special rule above specified, **assign points** to variable type int.  3. **Print the points** related to each player“-”.**(Print message)**  3. Print the remaining stamina (lives) after deducing one live for the player who introduced an “-”.**(Print message)**  Start a new round from the beginning, with the random letter, etc **(restart loop with new score)**  4. print again message for user input**(Print input message)** |

* **Inside the game: Main game –following words after first one:**

|  |  |  |
| --- | --- | --- |
| **INPUT** | **PROCESS** | **OUTPUT** |
| Take a string inputted by the user, by Scanner method.  **(Take strings)** | 1. The String starts with the **same last two characters** that contained the last introduced String word. **(Compare strings)**  2. The introduced word in form of a String **is identical**, no errors, or gibberish text, non-numerical content; to one of the words, Strings in the file “Limited Vocabulary.java” which means that the String is a valid English word for the game. **(Compare Strings)**  3. If the input is equal to “-”  Then will be translated as being an error and will take to the deduction of a live from the lives given at the start of the game. **(Compare Strings)**  4. If the String introduced is not found in the Array of words in the “Limited Vocabulary.java**”. (Compare Strings)** | 1. If the word is confirmed the loop will change the turn and prompt a new word to be introduced by the other player. **(Repeat main loop, introduce new condition in the initial loop)**  2. **Calculate points** as per special rule above specified, **assign points** to variable type int.  3. **Print the points** related to each player“-”.**(Print message)**  3. Print the remaining stamina (lives) after deducing one live for the player who introduced an “-”.**(Print message)**  Start a new round from the beginning, with the random letter, etc **(restart loop with new score)**  4. print again message for user input**(Print input message)** |