Testing and Debugging

# Section 1. Issues found in Word.java

|  |  |  |
| --- | --- | --- |
| **Location (e.g. constructor, method name)** | **Describe the fault** | **Describe the fix** |
| public char[] getLetters() | This error did not permit the normal use of the software because the return statement of this method was an entire array containing all the possible letters of the alphabet: LETTERS that since it is in upper case letters, it is a constant. | This method should return the array this. letters containing as chars every single letter of the word that is going to be displayed on the board. |
| private static Map<Character, Integer> scoreMap() | The for loop start with the (int i) variable initialized as 1 instead of zero causing the missing iteration of the first element in the array because arrays start with the zero index. | This was fixed by setting the starting value at 0 so the for loop created to read all the members of the LETTERS array containing all the letters of the alphabet start reading it from the zero index, without skipping the letter “a”. |
| public static final char[] LETTERS | In the char array, there is the letter “n” twice instead of the letter “m” and the letter “x” is instead of the letter “y” | I fixed the problem removing one of the double letters replacing it with an “m” and changing the wrong letter with the right one. |
| private static final int[] SCORES | While testing the code, I realized that the score of the letter “m” was worth 1 even if according to the specification it should be 3. | As written in the specification every letter has a specific score so in the SCORES array the 13th number, corresponding to the value of the letter m should be changed at 3. |
| public boolean startsWith(char[] prefix) | This method had a Boolean as return statement that did not follow the logical procedure. The boolean was false as result of an if statement containing an equality. | There are two possible ways to fix this error.  It is possible to set the return statement of the quality as true, (so switching the return true with the return false on the line below). Another possible fix is changing the equality inside the if statement (==) in an inequality (!=). |

# Section 2. Issues found in Dictionary.java

|  |  |  |
| --- | --- | --- |
| **Location (e.g. constructor, method name)** | **Describe the fault** | **Describe the fix** |
| public boolean isValid(Word word) | The if statement checked if the map contained as a key, the first letter of the word that the user is trying to insert. So if the user inserted a word that begins with a letter that is on the map even if the word itself is not a value, it returned true. | This fault was fixed by adding to the check of the key value, another statement that checks if the word that the user would insert is present in the map as a value. |
| public boolean isValid(Word word) | In this method the if statement returned the same value that it is going to verify. | This was not an error, but an “area of bad practice”, and it would be better a return true statement instead of any repetition of code. |
| private static final String fileName | In Java, the naming convention says that constant values should be written in upper case.  The string fileName even if it is a constant it is in lower-case. | To fix this fault, the reference to dictionary.txt should be changed in an uppercase constant, so from fileName to FILE\_NAME conforming to the convention. |
| private Set<Word> readWordsFromFile() | Due to the previous change in order to respect the naming convention, the FILE\_NAME constant is written in the same way in other methods too, so this variable must be changed throughout the class. | This was fixed by changing it in order to match the convention. |
| public Map<Character, Set<Word>> dictionary; | This variable should be private in order to avoid to be accessed from outside the class. | This fault was fixed by changing the field Map dictionary as private instead of public to be used only inside the class. |

# Section 3. Issues found in Position.java

|  |  |  |
| --- | --- | --- |
| **Location (e.g. constructor, method name)** | **Describe the fault** | **Describe the fix** |
| public int getY() | Getter and setter are conventions to access private fields and variables instead of making them public.  This method as the name says it should be used to return “y” value but it returned the value of x (return x). | This fault was fixed by changing the return statement of the method in return y instead of return x. |
| public Position(int x, int y) | In the constructor, there is a mismatching between the value of the parameter (x) and the field of the current object (this.y). | To fix this error, there should be an equality between the parameter and the field in order to make this.y equal to y. |
| Headline of the class | Software sometimes is created and edited by more than one programmer. In this class, all the specification about the author of the code and about what is the aim of this class is missing. So even for a tester it could be difficult to work on it without any reference. | To fix this inaccuracy, there should be at the beginning of the class some lines giving some information about the class and the author such as: /\*\* \* Class to represent a Position @author Jennifer Warrender  \*/. |

# Section 4. Issues found in WordGame.java

|  |  |  |
| --- | --- | --- |
| **Location (e.g. constructor, method name)** | **Describe the fault** | **Describe the fix** |
| public int placeWord(Word w, Position p, boolean h) | This method was checking if there is space to place the word or if the word is in the dictionary (is a valid word). The fault is in the if statement because it was checking if the first was true or the second was true instead of both at the same time so for example if there was enough space but the word was not in the dictionary, the program displayed it the same. | This fault was fixed by changing the “or” (||) into an “and” (&&). |
| public boolean roomForWord(Word w, Position p, boolean h) | This method creates a new position for every letter of the word moving it with a for loop horizontally with p.getX() + the i variable of the for loop or vertically.  The error was in the vertical movement because the new position was the same of the horizontal one.  The same code lines in the placeWord method were right. | In order to display a word along the y-axis, the new position should be created increasing the p.getY() according to the for loop instead of returning the same value of the x-axis. |
| public boolean roomForWord(Word w, Position p, boolean h) | If the word in the board took more spaces to be displayed, if for example the user would display the word “france” at the end of the board at the coordinates (15,15) there was displayed only the first letter. | To avoid this inconvenient, there should a couple of if statements (if((p.getX() + i)>15){return false;}  if((p.getY() + i)>15){return false;}) or in a more efficient way a method in order to return false and delete the word instead of adding a single letter and adding points to the player.  In order to add other code lines, there should be added brackets too. |
| public String toString() | In the to string method, the method that returns a string representation of the board, there were two for loops the first was along the x-axis and the second one along the y-axes. The fault printed out the word along the wrong axis. | To fix this fault, the x in the for loop should be switched with the y, in order to print them in the right order, otherwise if the user would print a word horizontally, the program printed it vertically. |
| public int placeWord(Word w, Position p, boolean h) | This method gives points for each word inserted in the board but there was a fault because the if statement returned the same amount of points if the word was in the dictionary and there were empty spaces or even if the word was not placed. | If the statement is false, instead of returning the score as it would if the word was right, the method should return 0. |

# Section 5. Issues found in UserInterface.java

|  |  |  |
| --- | --- | --- |
| **Location (e.g. constructor, method name)** | **Describe the fault** | **Describe the fix** |
| private void printMenu() | In this method there were two print statements containing the word “statistics” misspelt.  The number 6 was misspelt “Statitics" and the number 7 “Satistics". | To fix this fault I changed the print statements that the software prints if the user chose from the menu the option number 6 and number 7. |
| public void run() | In this method inside the while loop after the switch, there was the method calling itself as if it was recursive.  That causes many errors such as the possibility for the user to choose zero from the menu or the board printing all the words to the console. | This fault was fixed by deleting it in order to run the rest of the code in this method. |
| private void placeVertical() | In this method, in order to add the score of the word inserted, the software pass as values, the word itself, the position and a boolean used in wordgame class.  If this boolean is true, the code places the word in horizontal but if it is false, the word is placed in vertical.  In this method there was a fault because the boolean was set as true as it is in placeHorizontal. | This was fixed by changing the boolean from true to false in order to place the word vertically. |
| private void saveStatsToFile() | While running this method, all the statistics of the game are saved in a file.  The order of the information saved is wrong because the reading method reads maxScore first then noOfGames but these values were saved in the opposite order. | This fault was fixed by switching the variable that saves the number of games with the one that saves the score in order to save first the maximum score. |
| private void clearPlayerStats() | This method that should set all the points to zero, was clearing all the player statistics except the current score. | This was fixed by adding to the method another code line to set the currentScore to 0. |
| public void run() | In this method a break statement was missing in the case 6 of the switch inside the while loop. | Adding a break statement in the case 6, the software can show the player statistics to the user on the console. |
| private String getStringInput() | The software was not case sensitive, so the user could not add word writing them in uppercase letters. | To let the user write words in uppercase, there should be a .toLowerCase() in the return statement of this method when the software receives the input from the keyboard. |

# Section 6. Brief report on your testing and debugging strategies (approx. 1-2 pgs.)

In order to test this game, first I tried to use it so I started playing with it to gain knowledge of the internal workings of the code.

Trying to execute as much code as I could and covering all the possible combination, when I attempted to insert a word, the aim of this game is to make more points as possible writing words in the board, I received the NullPointerException error. This error is thrown when the code attempts to use a reference to an object that points to no location (null), in fact, if I tried to print the board using Junit it gave me null as return.

My IDE warned that there were some errors in the class where the main method is.

While trying all the option of the menu, I noticed that the zero was not working so I thought that the while loop could not work properly then, I tried to force to software entering in it setting the choice variable as 1 from the main method and not inside the run method itself.

Following what learned during lectures, I started looking for some bugs there, and the first one I found out was that a method was constantly calling itself in a loop and that did not permit to run some lines of code.

Before commenting it out, I placed some print statement to look what part of the code was running and what part was not and I found out that the while loop was working perfectly and what causes some of the problems was this self-loop.

Investigating the code in order to understand how the software was working, I noticed that there was a break statement missing in one of the cases in the switch.

While testing the structure of the software using the white-box method I realized comparing methods name and variables produced that there were some logical errors such as the two methods that should return position in the x-axis and y-axis but they both returned the same position along the x-axis e.g. this.y = x. and getY(), then return x or another example is a boolean that after an equality returned false.

In the Word class, a fault that did not permit the right running of the software was a method that returned the constant “LETTERS” instead of the right object written in lowercase “letters”. This error was caused by a bad way of naming variables and fields because they were written in a too similar way causing confusion between them.

While using the white-box testing method I found also some areas of bad practice, for example, there were some spelling error and some part of the code did not respect the specification such as the matching between the game value of a letter and the one that it should have been or for example some repetition of code that could be reduced in a more efficient way by creating a new method.

To do this testing and debugging assessment I understood how useful comments are because I have spent time understanding how some code works only because there was no explanation or it was poor. As said during the lectures, in industry respecting the given specification is fundamental and code is often produced or tested by more than one person and not giving the right information about the program or not respecting the specification could cause some faults such as runtime errors or in non-functional aspects too. In some classes, constants were written in lower case, despite the naming convention says that constant values should be written in uppercase or the general information about the class and the author of the software were missing.

Testing is also a good way of revising some function: I revised how maps work to clarify my knowledge and that helped me to fix the null pointer exception error.

However, that was not enough; even if the board displayed the word, that was not placed in the right coordinates and not vertically or horizontally as it should have been and a feature of the game, the one that displays all the word starting with the selected letters did still not work.

Therefore, I tried the black-box testing method using JUnit thinking that simplifying the program as much as possible and using only the variables and the methods needed I could find a way to fix these new errors.

I attempted to use JUnit trying inputs and seeing if outputs are as expected but there were too many errors throughout the classes because so many values were shared between them and that did not permit a right and clear use of the tool.

So, at first it was a failure but at least, while testing with JUnit, the tool showed me some faults, for example, a loop to iterate every single element of the array which index started from 1 instead of 0 causing the failure of reading the first element.

The black-box method also highlighted that the software was not case sensitive, so if the user tried to insert a word using some uppercase letters, the game threw some errors. To fix this kind of error, I added the toLowerCase method to the return statement in order to give the user more freedom of choice to play the game.