Please enter your name here:

Please enter your Academic Declaration here:

Description of the coursework: ScraeBBKle

Logistics

On Moodle, behind the "Assessment" tile, you will find information about submission deadlines. You will also find the starter code for this coursework assignment as a `.zip` file that contains a "bare Git repository" (i.e., a Git repository that does not include a working copy). Download and unzip this file to your computer. Then clone this repository from your file system using a tool of your choice (e.g., Git on the command line, IntelliJ to start a new project, Eclipse, NetBeans, VSCode, ...). After cloning, you will have a local repository with a working copy. Write your code and make commits to your local repository as usual.

Submission of your work is via a Moodle dropbox. You will need to submit your local repository as a `.zip` file. To this end, make a `.zip` file of the directory (folder) with your working copy and the `.git/` subdirectory (subfolder). Make sure that the `.zip` file also contains the `.git/` subdirectory!

For marking, we will download and unzip your submission. We will then clone the content of your `.git/` subdirectory and assess the resulting local repository with its working copy.

Hint: before you submit, follow the above marking steps on your own computer and check whether you can compile and run the game from the resulting local repository.

Overview

This coursework is about writing a game called *ScraeBBKle*. It is inspired by the board game Scrabble, although it has certain differences with the standard Scrabble game. The game rules have been inspired by the ones available here for the Scrabble game.

To reduce your workload, ScraeBBKle has some simplifications over Scrabble, but also some generalisations. The game will be played between two players, Player 1 and Player 2. Each of the two players may either be a **human player** or be a **computer player**.

In our setting, a board has `M` columns and `N` rows (where `M` and `N` are both integer numbers), so it has a total of `M*N` squares. The value `M` must be at least 7 and at most 26, and `N` must be at least `10` and at most 99. (We set these limits to avoid visualisation/printing problems for large boards.) We also require that a board has at least 192 squares, so `M*N >= 192` must hold. (For comparison, a standard Scrabble board has `M = N = 15` with a total of `15*15 = 225` squares.)

The goal of the game is to populate the board with a grid of tiles, similar to a crossword, so that all words on the board are from a pre-defined word list. Both players have a rack of up to 7 tiles which is refilled from a tile bag after every move. Each tile has a letter and a numerical value.

When the game is started, the user is asked whether to load a specific board from the file system or whether to use the default board. The default board is inspired by the Scrabble board. It is provided in the file `resources/defaultBoard.txt`. Then the user is asked whether Player 1 is supposed to be a human player or a computer player, and then the user is asked the same question about Player 2.

Finally, the user is asked whether to play an open game or a closed game. The only difference is that in an open game, the player who makes the next move also gets to see the other player's tiles.

After that, both players get 7 randomly chosen tiles from the tile bag. The user is then shown the (currently empty) board. Player 1 plays the first move.

Notation and Symbols

The columns will be designated by lower-case letter characters from a to z and the rows by numbers from 1 to 99. The leftmost column is a and the top row is 1.

Gameplay

1. The first move is made by Player 1, by combining two or more of their letters to a word or by passing their turn. The word is placed on the board to read either to the right or downwards, with one letter on the **start square**, a designated square of the board. Like in Scrabble, diagonal words are not possible in ScraeBBKle. Whenever a tile is placed on a square, the letter and the value of the tile replace the square on the displayed board.

2. The player that plays the first word must use the start square for one of the letters of the word.

3. The game computes the score resulting from the move. As long as the tile bag is not empty, the player who just made a move will have their tile rack topped up with tiles taken from the tile bag so that the player's rack has 7 tiles again.

4. The next move is Player 2's. From now on, Player 1 and Player 2 take turns with their moves until one player has no more tiles on their rack or no more moves are possible.

5. The player whose turn it is adds one or more letters to the letters already on the board to form a new word. All letters played in a move must be placed in one row to the right or downwards and contribute to a new word. It is allowed to skip occupied positions (for example, one may extend `NO` to `SNOW` by adding a `S` at the beginning and at the same time a `W` at the end). In ScraeBBKle, every move may add only *one occurrence* of a new or changed word on the board. In contrast to the original Scrabble game, it is not allowed to place a word at right angles to a word already on the board unless there is an overlap of the word resulting from the move with the earlier word on the board, nor to place a complete word parallel immediately next to a word already played. (Later in this document, we will see some examples to clarify further.)

6. Once a tile has been used in a move, its position on the board stays unchanged.

7. There are special tiles, called *wildcards* or *blanks*, where the letter is initially not given, but can be chosen by the player as needed during a move. In our game, the choice is entered as a small letter. As soon as the choice has been made for a given wildcard, it stays fixed for the rest of the play. The value of a wildcard in ScraeBBKle is always 8. When a wildcard has been played, its letter used for display on the board is the chosen small letter - the choice of letter for this specific wildcard then lasts for the whole game.
8. A player is allowed to pass the current turn (i.e., not make a move and allow the next player to continue).
9. In ScraeBBKle, it is not allowed to exchange tiles from the tile rack with tiles from the tile bag.
10. Moves must always lead to words that are in the game's word list. (This is checked by the game, and invalid moves are rejected by the game. The word list can be found in the file `resources/wordlist.txt`.)
11. The game ends when the tile bag is empty and one of the player has an empty tile rack. The game also ends if both players pass twice in a row.

Scoring

1. The score for each move is calculated as the sum of the letter values in the word created or modified in the move, plus the extra points obtained (or lost!) from tiles placed on *premium squares*.
2. **Premium letter squares** have an integer number of at least -9 and at most 99 as a factor (specifically, 0 or negative values are possible). When a tile is placed on a premium letter square, the score for the tile is its value multiplied by the factor of the premium letter square. A premium letter square has the shape " x." (so with a leading space) if the factor x is a single character and the shape "xy." if the factor xy has two characters.
3. **Premium word squares** also have an integer number of at least -9 and at most 99 as a factor (specifically, 0 or negative values are possible). When a move places a tile on a premium word square, the factor of the premium word square will be multiplied with the score obtained for the word otherwise. If a move uses several premium word squares, the effect is cumulative (for example, when we use a premium word square with factor 4 and a second premium word square with factor 5 in the same move, the resulting factor for the word score would be `4*5 = 20`). Premium word squares are applied only *after* premium letter squares. A premium word square has the shape " x!" (so with a leading space) if the factor x is a single character and the shape "xy!" if the factor xy has two characters.
4. A square cannot be at the same time both a premium letter square and a premium word square. There can also be squares that are not premium letter squares or premium word squares. Such squares are displayed as " . " (i.e., a space, then a dot, then a space).
5. Letter and word premium squares are applied only in a single move. As soon as they have been covered by a tile, in later moves this tile will count at its face value (i.e., the score will not be affected by the premium formerly obtained from covering the tile).
6. Woo-hoo! If a player manages to play all 7 tiles in one move, they are awarded an

extra score of 60 points in ScraeBBKle. This extra score is added only after all the other calculations for the current move are done (so also with a premium word square involved in the move, the player would still get only 60 extra score points).

7. At the end of the game, at least one player will have unplayed tiles. Each player's score is reduced by the sum of the values of their own unplayed tiles.

Winning player

The player who has a higher score at the end of the game wins. If the scores are equal, the game is declared a draw.

Initial tile bag

ScraeBBKle is played with a fixed initial tile bag. It contains the individual tiles with the following quantities and values (which are slightly different from Scrabble):

8 x [A1]
2 x [B3]
2 x [C4]
4 x [D2]
9 x [E2]
3 x [F4]
4 x [G3]
3 x [H4]
9 x [I1]
1 x [J11]
2 x [K6]
4 x [L1]
2 x [M3]
7 x [N1]
7 x [O1]
2 x [P3]
1 x [Q12]
6 x [R1]
4 x [S1]
5 x [T1]
5 x [U1]
2 x [V4]
2 x [W4]
1 x [X9]
2 x [Y5]
1 x [Z9]

Here, the line

8 x [A1]

means: "there are 8 tiles with the letter A and the value 1"

Moreover, there are two wildcards "[_8]" of value 8 in the initial tile bag.

Example play

For example, after starting the game, the user may have the following interaction with the game:

```
============                     ============
============ S c r a e B B K l e ============
============                     ============

Would you like to _l_oad a board or use the _d_efault board?
Please enter your choice (l/d): d
Is Player 1 a _h_uman player or a _c_omputer player?
Please enter your choice (h/c): h
Is Player 2 a _h_uman player or a _c_omputer player?
Please enter your choice (h/c): c
Would you like to play an _o_pen or a _c_losed game?
Please enter your choice (o/c): o

    a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p

 1   3.-3! .  .  2. .  .  .  2! .  .  . -4. .  . -2!  1
 2   . 12! .  .  2. .  .  .  3! .  .  .  2. .  .  8!  2
 3   .  .  2! .  .  .  3. .  .  .  3. .  .  .  2! .   3
 4   .  .  .  2! .  .  .  2. .  2. .  .  .  2! .  .   4
 5   .  2. .  .  2! .  .  .  2. .  .  .  2! .  .  2.  5
 6   .  .  .  .  .  2! .  .  .  .  .  0! .  .  .  .   6
 7   .  .  3. .  .  .  3. .  .  .  3. .  .  .  3. .   7
 8   .  .  .  2. .  .  .  2. .  2. .  .  .  2. .  .   8
 9   .  .  3. .  .  .  3. .  .  .  3. .  .  .  3. .   9
10   .  .  .  .  .  2! .  .  .  .  .  2! .  .  .  .  10
11   .  2. .  . -1! .  .  .  0. .  .  . -1! .  .  2. 11
12   .  9! .  .  2. .  .  .  3! .  .  .  2. .  . 16! 12
13   .  .  .  .  2! .  .  4. .  5. .  .  .  2! .  .  13
14   .  .  .  3. .  .  . 42. .  1! .  .  .  2. .  .  14

    a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p

Start position: d7
OPEN GAME: Player 2's tiles:
OPEN GAME: [N1], [Z9], [H4], [N1], [T1], [F4], [O1]
It's your turn, Player 1! Your tiles:
[E2], [E2], [D2], [N1], [I1], [D2], [L1]
Please enter your move in the format: "word,square" (without the quotes)
For example, for suitable tile rack and board configuration, a downward move
could be "HI,f4" and a rightward move could be "HI,4f".

In the word, upper-case letters are standard tiles and lower-case letters
are wildcards.
Entering "," passes the turn.
DINED,d4
The move is:    Letters: DINED at position d4
Player 1 score: 20
Player 2 score: 0

    a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p
 1  3.-3! .  .  2. .  .  .  2! .  .  . -4. .  . -2!  1
 2  . 12! .  .  2. .  .  .  3! .  .  .  2. .  .  8!  2
 3  .  .  2! .  .  .  3. .  .  .  3. .  .  .  2! .   3
 4  .  .  . D2  .  .  .  2. .  2. .  .  .  2! .  .   4
 5  .  2. . I1  2! .  .  .  2. .  .  .  2! .  .  2.  5
 6  .  .  . N1  .  2! .  .  .  .  .  0! .  .  .  .   6
 7  .  .  3.E2  .  .  3. .  .  .  3. .  .  .  3. .   7
 8  .  .  . D2  .  .  .  2. .  2. .  .  .  2. .  .   8
 9  .  .  3. .  .  .  3. .  .  .  3. .  .  .  3. .   9
10  .  .  .  .  .  2! .  .  .  .  .  2! .  .  .  .  10
11  .  2. .  . -1! .  .  .  0. .  .  . -1! .  .  2. 11
12  .  9! .  .  2. .  .  .  3! .  .  .  2. .  . 16! 12
13  .  .  .  .  2! .  .  4. .  5. .  .  .  2! .  .  13
14  .  .  .  3. .  .  . 42. .  1! .  .  .  2. .  .  14

    a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p

Start position: d7
OPEN GAME: Player 1's tiles:
OPEN GAME: [E2], [L1], [R1], [R1], [E2], [V4], [O1]
It's your turn, Player 2!
The move is:    Letters: TNZON at position 7c
Player 1 score: 20
Player 2 score: 19

    a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p

 1  3.-3! .  .  2. .  .  .  2! .  .  . -4. .  . -2!  1
 2  . 12! .  .  2. .  .  .  3! .  .  .  2. .  .  8!  2
 3  .  .  2! .  .  .  3. .  .  .  3. .  .  .  2! .   3
 4  .  .  . D2  .  .  .  2. .  2. .  .  .  2! .  .   4
 5  .  2. . I1  2! .  .  .  2. .  .  .  2! .  .  2.  5
 6  .  .  . N1  .  2! .  .  .  .  .  0! .  .  .  .   6
 7  .  . T1 E2 N1 Z9 O1 N1  .  .  3. .  .  .  3. .   7
 8  .  .  . D2  .  .  .  2. .  2. .  .  .  2. .  .   8
 9  .  .  3. .  .  .  3. .  .  .  3. .  .  .  3. .   9
10  .  .  .  .  .  2! .  .  .  .  .  2! .  .  .  .  10
11  .  2. .  . -1! .  .  .  0. .  .  . -1! .  .  2. 11
12  .  9! .  .  2. .  .  .  3! .  .  .  2. .  . 16! 12
13  .  .  .  .  2! .  .  4. .  5. .  .  .  2! .  .  13
14  .  .  .  3. .  .  . 42. .  1! .  .  .  2. .  .  14

    a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p

Start position: d7
OPEN GAME: Player 2's tiles:
OPEN GAME: [H4], [F4], [I1], [L1], [A1], [S1], [P3]
It's your turn, Player 1! Your tiles:
[E2], [L1], [R1], [R1], [E2], [V4], [O1]
```

```
Please enter your move in the format: "word,square" (without the quotes)
For example, for suitable tile rack and board configuration, a downward move
could be "HI,f4" and a rightward move could be "HI,4f".

In the word, upper-case letters are standard tiles and lower-case letters
are wildcards.
Entering "," passes the turn.
O,e6
The board does not permit word O at position e6. Please try again.
Please enter your move in the format: "word,square" (without the quotes)
For example, for suitable tile rack and board configuration, a downward move
could be "HI,f4" and a rightward move could be "HI,4f".

In the word, upper-case letters are standard tiles and lower-case letters
are wildcards.
Entering "," passes the turn.
OR,8h
The board does not permit word OR at position 8h. Please try again.
Please enter your move in the format: "word,square" (without the quotes)
For example, for suitable tile rack and board configuration, a downward move
could be "HI,f4" and a rightward move could be "HI,4f".

In the word, upper-case letters are standard tiles and lower-case letters
are wildcards.
Entering "," passes the turn.
OVE,4e
The move is:    Letters: OVE at position 4e
Player 1 score: 29
Player 2 score: 19

    a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p

 1  3.-3! .  .  2. .  .  .  2! .  .  . -4. .  . -2!  1
 2  . 12! .  .  2. .  .  .  3! .  .  .  2. .  .  8!  2
 3  .  .  2! .  .  .  3. .  .  .  3. .  .  .  2! .   3
 4  .  .  . D2 O1 V4 E2  2. .  2. .  .  .  2! .  .   4
 5  .  2. . I1  2! .  .  .  2. .  .  .  2! .  .  2.  5
 6  .  .  . N1  .  2! .  .  .  .  .  0! .  .  .  .   6
 7  .  . T1 E2 N1 Z9 O1 N1  .  .  3. .  .  .  3. .   7
 8  .  .  . D2  .  .  .  2. .  2. .  .  .  2. .  .   8
 9  .  .  3. .  .  .  3. .  .  .  3. .  .  .  3. .   9
10  .  .  .  .  .  2! .  .  .  .  .  2! .  .  .  . 10
11  .  2. .  . -1! .  .  .  0. .  .  . -1! .  .  2. 11
12  .  9! .  .  2. .  .  .  3! .  .  .  2. .  . 16! 12
13  .  .  .  .  2! .  .  4. .  5. .  .  .  2! .  . 13
14  .  .  .  3. .  .  . 42. .  1! .  .  .  2. .  . 14

    a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p

Start position: d7
```

```
OPEN GAME: Player 1's tiles:
OPEN GAME: [L1], [R1], [R1], [E2], [T1], [C4], [O1]
It's your turn, Player 2!
The move is:    Letters: PAFISH at position h5
Player 1 score: 29
Player 2 score: 38

    a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p

 1  3.-3! .  .  2. .  .  .  2! .  .  . -4. .  . -2!  1
 2  . 12! .  .  2. .  .  .  3! .  .  .  2. .  .  8!  2
 3  .  .  2! .  .  .  3. .  .  .  3. .  .  .  2! .   3
 4  .  .  . D2 O1 V4 E2  2. .  2. .  .  .  2! .  .   4
 5  .  2. . I1  2! .  . P3  2. .  .  .  2! .  .  2.  5
 6  .  .  . N1  .  2! . A1  .  .  .  0! .  .  .  .   6
 7  .  . T1 E2 N1 Z9 O1 N1  .  .  3. .  .  .  3. .   7
 8  .  .  . D2  .  .  . F4  .  2. .  .  .  2. .  .   8
 9  .  .  3. .  .  .  3.I1  .  .  3. .  .  .  3. .   9
10  .  .  .  .  .  2! . S1  .  .  .  2! .  .  .  .  10
11  .  2. .  . -1! .  . H4  0. .  .  .  . -1! .  .  2. 11
12  .  9! .  .  2. .  .  .  3! .  .  .  2. .  . 16! 12
13  .  .  .  .  2! .  .  4. .  5. .  .  .  2! .  .  13
14  .  .  .  3. .  .  . 42. .  1! .  .  .  2. .  .  14

    a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p

Start position: d7
OPEN GAME: Player 2's tiles:
OPEN GAME: [L1], [U1], [S1], [I1], [_8], [N1], [N1]
It's your turn, Player 1! Your tiles:
[L1], [R1], [R1], [E2], [T1], [C4], [O1]
Please enter your move in the format: "word,square" (without the quotes)
For example, for suitable tile rack and board configuration, a downward move
could be "HI,f4" and a rightward move could be "HI,4f".

In the word, upper-case letters are standard tiles and lower-case letters
are wildcards.
Entering "," passes the turn.
RET,10f
The move is:    Letters: RET at position 10f
Player 1 score: 39
Player 2 score: 38

    a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p

 1  3.-3! .  .  2. .  .  .  2! .  .  . -4. .  . -2!  1
 2  . 12! .  .  2. .  .  .  3! .  .  .  2. .  .  8!  2
 3  .  .  2! .  .  .  3. .  .  .  3. .  .  .  2! .   3
 4  .  .  . D2 O1 V4 E2  2. .  2. .  .  .  2! .  .   4
 5  .  2. . I1  2! .  . P3  2. .  .  .  2! .  .  2.  5
 6  .  .  . N1  .  2! . A1  .  .  .  0! .  .  .  .   6
 7  .  . T1 E2 N1 Z9 O1 N1  .  .  3. .  .  .  3. .   7
 8  .  .  . D2  .  .  . F4  .  2. .  .  .  2. .  .   8
 9  .  .  3. .  .  .  3.I1  .  .  3. .  .  .  3. .   9
10  .  .  .  .  .  R1 E2 S1 T1  .  .  2! .  .  .  .  10
11  .  2. .  .  -1! .  . H4  0. .  .  .  -1! .  .  2. 11
12  .  9! .  .  2. .  .  .  3! .  .  .  2. .  .  16! 12
13  .  .  .  .  2! .  .  4. .  5. .  .  .  2! .  .  13
14  .  .  .  3. .  .  .  42. .  1! .  .  .  2. .  .  14

    a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p

Start position: d7
OPEN GAME: Player 1's tiles:
OPEN GAME: [L1], [R1], [C4], [O1], [A1], [S1], [I1]
It's your turn, Player 2!
The move is:    Letters: INNULaS at position 5i
Player 1 score: 39
Player 2 score: 134

    a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p

 1  3.-3! .  .  2. .  .  .  2! .  .  .  -4. .  .  -2!  1
 2  . 12! .  .  2. .  .  .  3! .  .  .  2. .  .  8!  2
 3  .  .  2! .  .  .  3. .  .  .  3. .  .  .  2! .   3
 4  .  .  . D2 O1 V4 E2  2. .  2. .  .  .  2! .  .   4
 5  .  2. . I1  2! .  . P3 I1 N1 N1 U1 L1 a8 S1  2.  5
 6  .  .  . N1  .  2! . A1  .  .  .  0! .  .  .  .   6
 7  .  . T1 E2 N1 Z9 O1 N1  .  .  3. .  .  .  3. .   7
 8  .  .  . D2  .  .  . F4  .  2. .  .  .  2. .  .   8
 9  .  .  3. .  .  .  3.I1  .  .  3. .  .  .  3. .   9
10  .  .  .  .  .  R1 E2 S1 T1  .  .  2! .  .  .  .  10
11  .  2. .  .  -1! .  . H4  0. .  .  .  -1! .  .  2. 11
12  .  9! .  .  2. .  .  .  3! .  .  .  2. .  .  16! 12
13  .  .  .  .  2! .  .  4. .  5. .  .  .  2! .  .  13
14  .  .  .  3. .  .  .  42. .  1! .  .  .  2. .  .  14

    a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p

Start position: d7
OPEN GAME: Player 2's tiles:
OPEN GAME: [I1], [T1], [G3], [A1], [I1], [W4], [M3]
It's your turn, Player 1! Your tiles:
[L1], [R1], [C4], [O1], [A1], [S1], [I1]
Please enter your move in the format: "word,square" (without the quotes)
For example, for suitable tile rack and board configuration, a downward move
could be "HI,f4" and a rightward move could be "HI,4f".

In the word, upper-case letters are standard tiles and lower-case letters
are wildcards.
```

```
Entering "," passes the turn.
OR,k10
The board does not permit word OR at position k10. Please try again.
Please enter your move in the format: "word,square" (without the quotes)
For example, for suitable tile rack and board configuration, a downward move
could be "HI,f4" and a rightward move could be "HI,4f".

In the word, upper-case letters are standard tiles and lower-case letters
are wildcards.
Entering "," passes the turn.
,
The move is:    Pass Move!
OPEN GAME: Player 1's tiles:
OPEN GAME: [L1], [R1], [C4], [O1], [A1], [S1], [I1]
It's your turn, Player 2!
The move is:    Letters: TAWIG at position j1
Player 1 score: 39
Player 2 score: 146

    a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p

 1  3.-3! .  .  2. .  .  .  2!T1  .  . -4. .  . -2!  1
 2  . 12! .  .  2. .  .  .  3!A1  .  .  2. .  .  8!  2
 3  .  .  2! .  .  .  3. .  . W4  3. .  .  .  2! .   3
 4  .  .  . D2 O1 V4 E2  2. . I1  .  .  .  2! .  .   4
 5  .  2. . I1  2! .  . P3 I1 N1 N1 U1 L1 a8 S1  2.  5
 6  .  .  . N1  .  2! . A1  . G3  .  0! .  .  .  .   6
 7  .  . T1 E2 N1 Z9 O1 N1  .  .  3. .  .  .  3. .   7
 8  .  .  . D2  .  .  . F4  .  2. .  .  .  2. .  .   8
 9  .  .  3. .  .  .  3.I1  .  .  3. .  .  .  3. .   9
10  .  .  .  .  . R1 E2 S1 T1  .  .  2! .  .  .  .  10
11  .  2. .  . -1! .  . H4  0. .  .  . -1! .  .  2. 11
12  .  9! .  .  2. .  .  .  3! .  .  .  2. .  . 16! 12
13  .  .  .  .  2! .  .  4. .  5. .  .  .  2! .  .  13
14  .  .  .  3. .  .  . 42. .  1! .  .  .  2. .  .  14

    a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p

Start position: d7
OPEN GAME: Player 2's tiles:
OPEN GAME: [I1], [M3], [H4], [U1], [_8], [O1], [L1]
It's your turn, Player 1! Your tiles:
[L1], [R1], [C4], [O1], [A1], [S1], [I1]
Please enter your move in the format: "word,square" (without the quotes)
For example, for suitable tile rack and board configuration, a downward move
could be "HI,f4" and a rightward move could be "HI,4f".

In the word, upper-case letters are standard tiles and lower-case letters
are wildcards.
Entering "," passes the turn.
```

```
qwerty
Illegal move format
Please enter your move in the format: "word,square" (without the quotes)
For example, for suitable tile rack and board configuration, a downward move
could be "HI,f4" and a rightward move could be "HI,4f".

In the word, upper-case letters are standard tiles and lower-case letters
are wildcards.
Entering "," passes the turn.
CAL,m2
The move is:    Letters: CAL at position m2
Player 1 score: 50
Player 2 score: 146

    a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p

 1  3.-3! .  .  2. .  .  .  .  2!T1  .  .  -4. .  .  -2!  1
 2  . 12! .  .  2. .  .  .  .  3!A1  .  .  C4  .  .  8!  2
 3  .  .  2! .  .  .  3. .  .  W4  3. .  A1  .  2! .   3
 4  .  .  . D2 O1 V4 E2  2. . I1  .  . L1  2! .  .   4
 5  .  2. . I1  2! .  . P3 I1 N1 N1 U1 L1 a8 S1  2.  5
 6  .  .  . N1  .  2! . A1  . G3  .  0! .  .  .  .   6
 7  .  . T1 E2 N1 Z9 O1 N1  .  .  3. .  .  .  3. .   7
 8  .  .  . D2  .  .  . F4  .  2. .  .  .  2. .  .   8
 9  .  .  3. .  .  .  3.I1  .  .  3. .  .  .  3. .   9
10  .  .  .  .  . R1 E2 S1 T1  .  .  2! .  .  .  .  10
11  .  2. .  . -1! .  . H4  0. .  .  . -1! .  .  2. 11
12  .  9! .  .  2. .  .  .  3! .  .  .  2. .  . 16! 12
13  .  .  .  .  2! .  .  4. .  5. .  .  .  2! .  .  13
14  .  .  .  3. .  .  . 42. .  1! .  .  .  2. .  .  14

    a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p

Start position: d7
OPEN GAME: Player 1's tiles:
OPEN GAME: [R1], [O1], [S1], [I1], [L1], [T1], [S1]
It's your turn, Player 2!
The move is:    Letters: cHOLIUM at position o6
Player 1 score: 50
Player 2 score: 236

    a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p

 1  3.-3! .  .  2. .  .  .  .  2!T1  .  .  -4. .  .  -2!  1
 2  . 12! .  .  2. .  .  .  .  3!A1  .  .  C4  .  .  8!  2
 3  .  .  2! .  .  .  3. .  .  W4  3. .  A1  .  2! .   3
 4  .  .  . D2 O1 V4 E2  2. . I1  .  . L1  2! .  .   4
 5  .  2. . I1  2! .  . P3 I1 N1 N1 U1 L1 a8 S1  2.  5
 6  .  .  . N1  .  2! . A1  . G3  .  0! .  . c8  .   6
 7  .  . T1 E2 N1 Z9 O1 N1  .  .  3. .  .  . H4  .   7
 8  .  .  . D2  .  .  . F4  .  2. .  .  .  2.O1  .   8
 9  .  .  3. .  .  .  3.I1  .  .  3. .  .  . L1  .   9
10  .  .  .  .  .  . R1 E2 S1 T1  .  .  2! .  . I1  .  10
11  .  2. .  . -1! .  . H4  0. .  .  . -1! . U1  2. 11
12  .  9! .  .  2. .  .  .  3! .  .  .  2. . M3 16! 12
13  .  .  .  .  2! .  .  4. .  5. .  .  .  2! .  .  13
14  .  .  .  3. .  .  . 42. .  1! .  .  .  2. .  .  14

    a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p

Start position: d7
OPEN GAME: Player 2's tiles:
OPEN GAME: [R1], [K6], [G3], [F4], [K6], [G3], [H4]
It's your turn, Player 1! Your tiles:
[R1], [O1], [S1], [I1], [L1], [T1], [S1]
Please enter your move in the format: "word,square" (without the quotes)
For example, for suitable tile rack and board configuration, a downward move
could be "HI,f4" and a rightward move could be "HI,4f".

In the word, upper-case letters are standard tiles and lower-case letters
are wildcards.
Entering "," passes the turn.
ROS,f9
The board does not permit word ROS at position f9. Please try again.
Please enter your move in the format: "word,square" (without the quotes)
For example, for suitable tile rack and board configuration, a downward move
could be "HI,f4" and a rightward move could be "HI,4f".

In the word, upper-case letters are standard tiles and lower-case letters
are wildcards.
Entering "," passes the turn.
IT,8i
The move is:    Letters: IT at position 8i
Player 1 score: 57
Player 2 score: 236

    a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p

 1  3.-3! .  .  2. .  .  .  2!T1  .  . -4. .  . -2!  1
 2  . 12! .  .  2. .  .  .  3!A1  .  . C4  .  .  8!  2
 3  .  .  2! .  .  .  3. .  . W4  3. . A1  .  2! .   3
 4  .  .  . D2 O1 V4 E2  2. . I1  .  . L1  2! .  .   4
 5  .  2. . I1  2! .  . P3 I1 N1 N1 U1 L1 a8 S1  2.  5
 6  .  .  . N1  .  2! . A1  . G3  .  0! .  . c8  .   6
 7  .  . T1 E2 N1 Z9 O1 N1  .  .  3. .  .  . H4  .   7
 8  .  .  . D2  .  .  . F4 I1 T1  .  .  .  2.O1  .   8
 9  .  .  3. .  .  .  3.I1  .  .  3. .  .  . L1  .   9
10  .  .  .  .  .  . R1 E2 S1 T1  .  .  2! .  . I1  .  10
11  .  2. .  . -1! .  . H4  0. .  .  . -1! . U1  2. 11
12  .  9! .  .  2. .  .  .  3! .  .  .  2. . M3 16! 12
```

```
13  .  .  .  .  2! .  .  4. .  5. .  .  .  2! .  .  13
14  .  .  .  3. .  .  .  42. .  1! .  .  .  2. .  .  14

    a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p

Start position: d7
OPEN GAME: Player 1's tiles:
OPEN GAME: [R1], [O1], [S1], [L1], [S1], [D2], [J11]
It's your turn, Player 2!
The move is:    Letters: FRG at position 2h
Player 1 score: 57
Player 2 score: 263

    a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p

 1  3.-3! .  .  2. .  .  .  2!T1  .  . -4. .  . -2!  1
 2  . 12! .  .  2. .  . F4 R1 A1 G3  . C4  .  .  8!  2
 3  .  .  2! .  .  .  3. .  . W4  3. . A1  .  2! .   3
 4  .  .  . D2 O1 V4 E2  2. . I1  .  . L1  2! .  .   4
 5  .  2. . I1  2! .  . P3 I1 N1 N1 U1 L1 a8 S1  2.  5
 6  .  .  . N1  .  2! . A1  . G3  .  0! .  . c8  .   6
 7  .  . T1 E2 N1 Z9 O1 N1  .  .  3. .  .  . H4  .   7
 8  .  .  . D2  .  .  . F4 I1 T1  .  .  .  2.O1  .   8
 9  .  .  3. .  .  .  3.I1  .  .  3. .  .  . L1  .   9
10  .  .  .  .  . R1 E2 S1 T1  .  .  2! .  . I1  .  10
11  .  2. .  . -1! .  . H4  0. .  .  .  . -1! . U1  2. 11
12  .  9! .  .  2. .  .  .  3! .  .  .  2. . M3 16! 12
13  .  .  .  .  2! .  .  4. .  5. .  .  .  2! .  .  13
14  .  .  .  3. .  .  .  42. .  1! .  .  .  2. .  .  14

    a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p

Start position: d7
OPEN GAME: Player 2's tiles:
OPEN GAME: [K6], [K6], [G3], [H4], [E2], [R1], [M3]
It's your turn, Player 1! Your tiles:
[R1], [O1], [S1], [L1], [S1], [D2], [J11]
Please enter your move in the format: "word,square" (without the quotes)
For example, for suitable tile rack and board configuration, a downward move
could be "HI,f4" and a rightward move could be "HI,4f".

In the word, upper-case letters are standard tiles and lower-case letters
are wildcards.
Entering "," passes the turn.
```

The example play still goes on for a while until one of the players has emptied their tile rack. Look at the play carefully, it can help disambiguate special cases of the specification!

Note specifically that in ScraeBBKle, a move may never lead to more than one word of 2 or more letters being created on the board. (This is different from Scrabble.) The created word must always be in the same direction as the player's move and use all tiles of the move. This is why the move `O,e6` was not allowed in the example play: it would have created these two occurrences of words: `ON` at `e6` (in the direction of the move) and `NO` at `6d` (not in the direction of the move).

Moves

As indicated above, you will need to read *moves* for the human player.

The *pass move* is indicated by writing a line that consists of just one comma: `,`

A move to *play tiles* is indicated by writing the word spelled by the tiles that are played (excluding the tiles already on the board), then a comma, then the position with the direction in format `xy`:

- If `xy` has `x` as a column label and `y` as a row number, it means that the tiles are played downward, with `xy` as the "origin" of the move. An example is `d7` (where `x=d` and `y=7`) to indicate that the first of the played tiles should go to `d7` and the tiles should be played downward.
- If `xy` has `x` as a row number and `y` as a column label, it means that the tiles are played from left to right, with `yx` as the "origin" of the move. An example is `3g` (where `x=3` and `y=g`) to indicate that the first of the played tiles should go to `g3` and the tiles should be played from left to right.

For example, `OND,f8` says that from the position in column `f` and row `8` on the board, going down, a tile sequence corresponding to the letters `OND` should be played, where occupied squares are skipped. Another example is `VLuE,10b`, which says that from position `b10`, going right, a tile sequence corresponding to the letters `VLUE` should be played, where a wildcard should be used for the `U` on the board (indicated by the use of the lower-case letter `u`).

Board specification files

Recall that the game initially asks the user whether they want to load a board file.

We need to define a file format for ScraeBBKle board specifications that a (very enthusiastic) user may write with an external text editor of their choice and that are used also to represent the default board of the game. We use a plain-text format.

The **first line** of a file for a ScraeBBKle board stores an integer number `M` between 7 and 26. The **second line** of a file for a ScraeBBKle board stores an integer number `N` between `10` and 99. These number indicates the number of columns and rows, respectively, of our `M x N` ScraeBBKle board. The value of multiplying `M` and `N` must be at least 192. The **third line** of a file for a ScraeBBKle board stores the position of the start square, mentioning first the column with its label and then the row with its label (for example, `d7` could be a valid line).

After the third line, there are `N` further lines, with the value `N` read earlier. Each of these lines consists of exactly `M` of the following "tokens", each of which represents a square:

- `.` represents a standard square without premium.
- `[n]` represents a letter premium square, where `n` is an integer between `-9` and 99.
- <n> represents a word premium square, where `n` is an integer between `-9` and 99.

Note that in our file format, a premium square may need either 3 or 4 characters, and the representation of a premium square in our file format is *different from* the way it is displayed on the screen. Finally, our file format does not allow for spaces, so a standard square uses a single character.

See the file `resources/defaultBoard.txt` for an example. A file is *valid* if it is syntactically correct as specified above.

User requirements

> **Note: the requirements below are mandatory to follow. You will lose marks if your implementation does not meet these requirements**

In this coursework, you shall implement a Java program in which a user shall play ScraeBBKle described above. **Player 1 always makes the first move.**

Initiation

When the program is executed, it first prompts the user to provide a file name that stores a plain board configuration:

```
Would you like to _l_oad a board or use the _d_efault board?
Please enter your choice (l/d):
```

The user inputs `l` or `d`. Otherwise, the program states this and asks the user again for input. If the user inputs `d`, the default board is loaded from the file `resources/defaultBoard.txt`. If the user inputs `l`, the program asks for the file name:

```
Please enter the file name of the board:
```

Then the user enters the file name relative to the path `resources/`. If this file is not valid, the program states that and prompts to provide a file name again:

```
This is not a valid file. Please enter the file name of the board:
```

This continues until the user provides a valid file. If a valid file is provided, the board that it contains is used for the game.

Then the user is asked whether Player 1 shall be a human player or a computer player:

```
Is Player 1 a _h_uman player or a _c_omputer player?
Please enter your choice (h/c):
```

The user inputs h or c. Otherwise, the program states this and asks the user again for input. If the user inputs h, Player 1 will be a human player who makes their move inputs via the keyboard, and if the user inputs c, Player 1 will be a computer player and make their moves without user interaction.

Then the user is asked whether Player 2 shall be a human player or a computer player:

```
Is Player 2 a _h_uman player or a _c_omputer player?
Please enter your choice (h/c):
```

The user inputs h or c, with analogous handling of the input as for the choice of Player 1.

Finally, the user is asked whether to play an open game or a closed game:

```
Would you like to play an _o_pen or a _c_losed game?
Please enter your choice (o/c):
```

The user inputs `o` or `c`. Otherwise, the program states this and asks the user again for input. If the user inputs `o`, an open game will be played, and if the user inputs `c`, a closed game will be played. The only difference between an open game and a closed game is that before every move, the tiles on the other player's rack are displayed (the lines starting with `OPEN GAME:` in the example play).

After these initial questions to the user, the game begins.

Play moves

As we observe in the example play, Player 1 and Player 2 take turns with their moves. Player 1 makes the first move.

Before each move, the program prints the current score and the current board.

If it is a human player's turn, the program asks the human player for their move:

```
Please enter your move in the format: "word,square" (without the quotes)
For example, for suitable tile rack and board configuration, a downward move
could be "HI,f4" and a rightward move could be "HI,4f".

In the word, upper-case letters are standard tiles and lower-case letters
are wildcards.
Entering "," passes the turn.
```

If the entered move does not follow the requested format, the program states this and asks again for input of a move:

```
Illegal move format
Please enter your move in the format: "word,square" (without the quotes)
For example, for suitable tile rack and board configuration, a downward move
could be "HI,f4" and a rightward move could be "HI,4f".

In the word, upper-case letters are standard tiles and lower-case letters
are wildcards.
Entering "," passes the turn.
```

This continues until the user inputs a valid move. Then the updated board and score are displayed.

If it is a computer player's turn, the move is computed by the program based on the tiles in the computer player's tile rack and the current state of the board. The move is then printed, and the updated board and score are displayed.

If the condition for the end of the game is reached (see Rules above), this is stated, the final scores of both players are calculated and shown, and the winner is announced.

```
Game Over!
Player 1 scored 216 points.
Player 2 scored 203 points.
Player 1 wins!
```

(scores may vary). Instead of

```
Player 1 wins!
```

it may also read

```
Player 2 wins!
```

or

```
It's a draw!
```

Then the program terminates.

Computer player

Implement the computer player in such a way that it will *always* make a move with at least one tile whenever such a move is possible with the given tile rack, board state, and word list.

Hint: A possible strategy to achieve this goal could be the following:

1. For each number `n` of tiles that could be played from the current tile rack (between at most the number of tiles on the rack and at least 1), for each free position on the board, and for both directions, check whether it is possible to play `n` tiles there. (Often enough, the answer will be `false`, e.g., because no connection to the existing crossword on the board would be made. There is no need to check all the possible combinations of `n` tiles for a position and direction where they cannot be played anyway!)
2. For those cases of part (1) where it is indeed possible to play `n` tiles, go through all permutations of `n` tiles on your rack. If one of them leads to a valid move for the board and the word list, make the move.

Hint: Consider implementing first the other parts of the coursework project. While you are getting up and running, a computer player that always passes their move does the job.

Platform independence

Your code shall run on all Java platforms. In particular, two issues to look our for are:

1. Use relative instead of absolute paths. It is very unlikely that other users will have `C:\Users\jdoe` or `/home/jdoe` or `/Users/jdoe` on their computer.
2. On Windows, `\` is used as the path separator. On Unix-based platforms (Linux, Mac), `/` is used as the path separator. If you use `/` in a path, Java will transform it internally as needed for the user's operating system (including Windows), so there is no good reason to use `\` as path separator.

Implementation requirements

- The requirements below are mandatory to follow. **You will lose marks** if your implementation does not adhere to these requirements, **even in case your program runs with no errors**.
- Your implementation shall be in **Java 25**. Implementations that do not compile with Java 25 but do compile with Java 21 will also be accepted.
- The class from which the program is run shall be called `Main.java` and be in the package `pij.main`.
- Your program design shall employ object-oriented principles with suitable definitions of classes/interfaces and using packages, encapsulation, polymorphism, and inheritance as appropriate for this case.
- If your program uses classes or interfaces of the Java Collections Framework, the chosen classes and interfaces shall be appropriate to the task at hand (e.g., sometimes a `Set` is more appropriate, sometimes a `List` is more appropriate).

Validation

Your submission shall contain 20 (or more) JUnit 5 test cases that test the functionality of your implementation. The test cases should cover a broad range of scenarios.

Development and Submission

Your code shall be well-structured in terms of visual readability (use indentation, spacing, etc.) and non-redundancy (instead of duplicating long pieces of code, introduce your own methods). Follow sensible naming conventions. Your code shall be well documented using JavaDoc.

You shall access this coursework assignment via the repository with the starter code provided behind the *Assessment* tile of the Moodle website for the *Programming in Java* module.

You must work on the coursework in a *Git* repository. You must create commits, with messages, that describe the history of the development of your project. We expect **a rich commit history**. *You will lose marks if you do not meet this requirement*. The use of other features of Git, e.g., branches, is optional and does not contribute to your mark. We will mark the `master` branch of your repository.

Information about the release date and the submission deadline for this coursework assignment is available behind the *Assessment* tile of the Moodle website for the *Programming in Java* module.

Academic declaration

The file `README.md` shall contain your name in the first section. In the second section, you shall provide the following academic declaration:

“I have read and understood the sections of plagiarism in the College Policy on academic integrity and misconduct and confirm that the work is my own, with the work of others clearly acknowledged. I give my permission to submit my report to the plagiarism testing database that the College is using and test it using plagiarism detection software, search engines or meta-searching software.”

This refers to the following document: College Policy on academic integrity and misconduct

A submission without this declaration shall receive 0 marks.

The other sections (from "Description of the coursework: ScraeBBKle" onward) shall not be modified.

Hint: Downloading, unzipping, and cloning the provided repository from Moodle to your computer, then entering your name and the academic declaration into `README.md`, and then committing your changes could be a wonderful way to start your coursework assignment and check whether your Git set-up works as intended.

Submission

The files of the final version of your project must be submitted via Moodle. We require at least the following in your submission:

- `README.md` with your name and your academic declaration added in the first two sections
- `src/main/pij/main/Main.java` as the class in package `pij.main` from which your program is started
- `resources/defaultBoard.txt` as the file from which the default board is loaded (should not be changed for the submission)
- `resources/wordlist.txt` as the word list used by the game (should not be changed for the submission)

and any other files or directories that are needed for the program started from class `Main` in package `pij.main` to run, as well as the JUnit tests.

The above files will already be present in your initial repository. You will need to modify `README.md` and `src/main/pij/main/Main.java` and add further files as needed.

Additional Libraries

You can use *any standard Java libraries that are part of Java 25* in your implementation. Additionally, you will need the libraries for JUnit 5. Otherwise, no libraries are allowed (no Gson, no Mockito, ...).

Marking

We aim to determine your mark according to the following rubric:

- **User requirements** (system responds to all use scenarios, and all responses are as expected): weight 40%
- **Implementation requirements** (a sensible design has been chosen, and its implementation is correct): weight 30%
- **Validation** (appropriate variety of scenarios are verified, and tests for them are correctly written): weight 10%
- **Development style** (visual readability and well-structuredness, documentation, and rich commit history): weight 20%

Credits

The file `resources/wordlist.txt` is a slightly abridged version of the SOWPODS word list that is available for download here.