**The demo video, plagiarism statements and java code files are available at:**

<https://drive.google.com/drive/folders/1lSq48yuXFv3keNFaczyejBAoLSNbtdss?usp=sharing>

For running the code in eclipse IDE (Version: 2021-09 (4.21.0)), all the class files in the Code folder in the above link should be put in the default folder of a new java project and the Game.java file should be run as application.

**The code has been tested and JAR file created on the following java version:**

java version "16.0.2" 2021-07-20

Java(TM) SE Runtime Environment (build 16.0.2+7-67)

Java HotSpot(TM) 64-Bit Server VM (build 16.0.2+7-67, mixed mode, sharing)

**Notation to Show gameboard:**

@ - food  
^ \* \* \* - snake position ( ^ means front of snake ) ( ^ is used for showing that default movement will be upwards, < is used for left default movement, > is used for right default movements, d is used for default downwards movements. )  
  
**Sample Game board:**

. . . . . . . . . .

. @ . . . . . . . .

. . . . . . . . . .

. . . . . . ^ . . .

. . . . \* \* \* . . .

. . . . . . . . . .

. . . . . . . . . .

. . . . . . . . . .

. . . . . . . . . .

. . . . . . . . . .

**Controls of the game:**

Enter '<' for moving your snake LEFT

Enter '>' for moving your snake RIGHT

Enter '^' for moving your snake UP

Enter 'd' / ’D’ for moving your snake DOWN

**Available Board sizes:**

10x10

15x15

20x20

**Difficulty Levels and Scoring Mechanism:**

For each board size available we have three difficulty levels:

Easy: user gets 5 points for each food eaten in easy level and 10 seconds to input snake move in easy level.

Medium: user gets 10 points for each food eaten and 7 seconds to input snake move in medium level.

Hard: user gets 15 points for each food eaten and 4 seconds to input snake move in hard level.

|  |  |  |
| --- | --- | --- |
| Difficulty Level | Food Point | Time for input (in secs) |
| Easy | 5 | 10 |
| Medium | 10 | 7 |
| Hard | 15 | 4 |

For each player, three scores are maintained, score respective to board size 10x10, score respective to board size 15x15, score respective to board size 20x20.

As soon as the player signs up, all these scores are initialised to zero.

Suppose a player signs up and starts a game in board size 10x10, suppose the player plays in Easy level and 3 foods are eaten and the game ends. 5\*3=15 score points are added to the player's score respective to board size 10x10.

Suppose now the same player starts a new game in board size 10x10, suppose the player plays in medium level and 3 foods are eaten and the game ends. 10\*3=30 score points are added to the player's score respective to board size 10x10.

After signing up and playing above two games, the player’s score would be:

Score respective to board size 10x10: 45

Score respective to board size 15x15: 0 (as the player hasn’t played any game of boardsize 15 after signing up)

Score respective to board size 20x20: 0 (as the player hasn’t played any game of boardsize 15 after signing up)

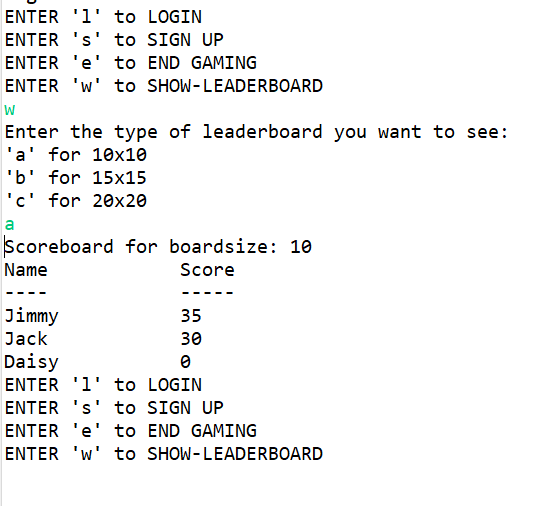
**Leaderboard:**

For displaying the leaderboard, the user is first required to enter the boardsize of which they want to see the leaderboard.

For 10x10, the players are sorted in descending order of their scores respective to board size 10x10. For any player who has not yet played any game with board size 10x10, default score 0 is considered for sorting and displaying of score.

For 15x15, the players are sorted in descending order of their scores respective to board size 15x15. For any player who has not yet played any game with board size 15x15, default score 0 is considered for sorting and displaying of score.

For 20x20, the players are sorted in descending order of their scores respective to board size 20x20. For any player who has not yet played any game with board size 20x20, default score 0 is considered for sorting and displaying of score.



**Note regarding inputs:** At each stage where the user is expected to input a single character (like the snake move or selecting board size, etc. as you will see while running the program) and if the user ends up inputting a multiple character string, only the first character of the trimmed inputted string is considered as input and the rest of the string is ignored.

All string inputs are trimmed before further usage in the program.

Inputting snake directions: while entering a move if the user enters “ d” as input, we have considered ‘d’ as the final input. If a user enters “ ” (a string of spaces as input or just hits the enter key), the snake would move in the default direction. Also if the user enters an input other than the characters of game controls, the snake would move in the default direction.

At choice points in the game other than snake directions: At every stage of the game the user is expected give a character input for selecting a choice, eg., inputting ‘s’ for signup, we have coded such that even if the user inputs ‘S’ instead of ‘s’, the same choice is made. (so the case (upper or lower) of the input does not impact the choice making) Here if the user inputs a string of spaces, nothing happens till the user inputs a string which contains characters apart from space, newline(default delimiters).

Some case-wise specifications:

**Inputting username:** the user is expected to enter a string which contains characters other than default delimiters. The program waits till such a string is entered. The spaces from the beginning and end of the inputted string are trimmed and the rest of the string is accepted as input. Here if the user inputs a string of spaces, nothing happens till the user inputs a string which contains characters apart from space, newline(default delimiters) also if the user inputs “ jack bey ” as their username, only the string “jack bey” is set as username. Here, the case of the input matters, the username “JACK” is considered different from the username “Jack”. Hence both users with these names can exist simultaneously.

**Limitation on Password:** while setting the password the program sets the password after trimming the inputted string from both sides. Eg. suppose the user enters “ 12 3 ” as their password. Then only the string “12 3” is set as the password of the user. Hence the spaces at the ends of the ends of the string are not included in the password. Similarly if the password of a particular user is “hello” and the user inputs string “ hello ” as their password, then the login will be successful as the code trims the input password string from both ends before checking for validity of password.

Note regarding snake direction:

If the default direction of the snake is ^ (UP) and the user enters d (DOWN) as its move, the snake ends up moving in the default direction itself. (as it is not possible for the snake to take a direct U turn).

If the default direction of the snake is > (RIGHT) and the user enters < (LEFT) as its move, the snake ends up moving in the default direction itself.

If the default direction of the snake is < (LEFT) and the user enters > (RIGHT) as its move, the snake ends up moving in the default direction itself.

If the default direction of the snake is d (DOWN) and the user enters ^ (UP) as its move, the snake ends up moving in the default direction itself.