# SE 3XA3: Module Interface Specification Scrabble Project

Team #214, The Trifecta
Kanakabha Choudhri, Choudhrk
Lucia Cristiano, Cristial
Raymond Tu, Tur1

April 6, 2020

This document is the Module Interface Specification of the Scrabble Project being done by Team Trifecta.

Table 1: **Revision History**

| Date | Version | Notes |
|---|---|---|
| 13/3/20 | ~~1.0~~ 0.0 | Had revision 0 finished. |
| 6/4/20 | 1.0 | Revision 1 complete. |

# Tile Module

## Module

Tile Type

## Uses

N/A

## Syntax

### Exported Constants

N/A

### Exported Types

Tile = tuple of (letter: String, score: $\mathbb{N}$)

### Exported Access Programs

| Routine name | In | Out | Exceptions |
|---|---|---|---|
| init | String | Tile | invalid_size |
| getLetter | | String | |
| getScore | | $\mathbb{N}$ | |

## Semantics

### State Variables

letter
score

### Environment Variables

None

### State Invariant

$0 < score \leq 10$

### Assumptions

N/A

### Access Routine Semantics

init($letter$):

- transition: $score := LETTER\_VALUES[letter]$
- output: None
- exception: ~~None~~ invalid_size

getLetter():

- transition: None
- output: letter
- exception: None

getScore():

- transition: None
- output: score
- exception: None

## Local Constants

$LETTER\_VALUES$ = tuple of ("$A$" : 1, "$B$" : 3, "$C$" : 3, "$D$" : 2, "$E$" : 1, "$F$" : 4, "$G$" : 2, "$H$" : 4, "$I$" : 1, "$J$" : 1, "$K$" : 5, "$L$" : 1, "$M$" : 3, "$N$" : 1, "$O$" : 1, "$P$" : 3, "$Q$" : 10, "$R$" : 1, "$S$" : 1, "$T$" : 1, "$U$" : 1, "$V$" : 4, "$W$" : 4, "$X$" : 8, "$Y$" : 4, "$Z$" : 10)

# Bag Module

## Module

Bag Type

## Uses

Tile

## Syntax

### Exported Constants

N/A

### Exported Types

Bag = list of Tiles

### Exported Access Programs

| Routine name | In | Out | Exceptions |
|---|---|---|---|
| init | | Bag | |
| addToBag | Tile, N | Bag | |
| initBag | | | |
| takeFromBag | | Tile | |
| getRemainingTiles | | N | |

## Semantics

### State Variables

Bag

### Environment Variables

None

**State Invariant**

$0 \leq |Bag| \leq 100$

**Assumptions**

N/A

**Access Routine Semantics**

init():

- transition: $Bag \rightarrow Bag$
- output: None
- exception: None

addToBag(Tile, n):

- transition: $Bag \rightarrow Bag + n * Tiles$
- output: None
- exception: None

initBag():

- transition: $Bag \rightarrow Bag + 10 * Tiles(A) + 2 * Tiles(B) + 2 * Tiles(C) + 4 * Tiles(D) + 12 * Tiles(E) + 2 * Tiles(F) + 3 * Tiles(G) + 2 * Tiles(H) + 9 * Tiles(I) + 1 * Tiles(J) + 1 * Tiles(K) + 4 * Tiles(L) + 2 * Tiles(M) + 6 * Tiles(N) + 8 * Tiles(O) + 2 * Tiles(P) + 1 * Tiles(Q) + 6 * Tiles(R) + 5 * Tiles(S) + 6 * Tiles(T) + 4 * Tiles(U) + 2 * Tiles(V) + 2 * Tiles(W) + 1 * Tiles(X) + 2 * Tiles(Y) + 1 * Tiles(Z)$
  Additionally shuffles the order of the letters.
- output: None
- exception: None

takeFromBag():

- transition: $|Bag| \rightarrow |Bag| - 1$
- output: $Bag(|Bag| - 1)$

- exception: None

getRemainingTiles():

- transition: None
- output: $|Bag|$
- exception: None

# Rack Module

## Module

Rack Type

## Uses

Bag

## Syntax

### Exported Constants

N/A

### Exported Types

Rack = set of Tiles where $t : Tile \in Bag$

### Exported Access Programs

| Routine name | In | Out | Exceptions |
|---|---|---|---|
| init | Bag | Rack | |
| addToRack | | | |
| initialize | | | |
| getRackStr | | String | |
| getRackArr | | Rack | |
| removeFromRack | Tile | | |
| getRackLength | | N | |
| replenishRack | | | |

## Semantics

### State Variables

rack
bag

**Environment Variables**

None

**State Invariant**

$0 < |rack| \leq 7$

**Assumptions**

N/A

**Access Routine Semantics**

init($Bag$):

- transition: $rack := \emptyset$
  $bag = Bag$
- output: None
- exception: None

addToRack():

- transition: $rack \rightarrow rack + t$
  where $t : Tile \in bag$
- output: None
- exception: None

initialize():

- transition: $rack \rightarrow rack + 7 * t$
  where $t : Tile \in bag$
- output: None
- exception: None

getRackStr():

- transition: None
- output: $r : Rack \rightarrow s : String$
  where r and s represent same set of characters.
- exception: None

getRackArr():

- transition: None
- output: rack
- exception: None

removeFromRack(tile):

- transition: $rack \rightarrow rack \setminus tile$
  where tile : Tile
- output: None
- exception: None

getRackLength():

- transition: None
- output: —rack—
- exception: None

replenishRack():

- transition: $rack \rightarrow rack + n * t$
  where $n : 7 - |rack|$
- output: None
- exception: None

# Player Module

## Module

Player Type

## Uses

Bag, Rack

## Syntax

### Exported Constants

N/A

### Exported Types

Player = tuple of $(rack : Rack, score : \mathbb{N})$

### Exported Access Programs

| Routine name | In | Out | Exceptions |
|---|---|---|---|
| init | Bag | Player | |
| getRackStr | | String | |
| getRackArr | | Rack | |
| increaseScore | $\mathbb{N}$ | | |
| getScore | | $\mathbb{N}$ | |

## Semantics

### State Variables

Score
Rack

### Environment Variables

None

**State Invariant**

N/A

**Assumptions**

N/A

**Access Routine Semantics**

init(Bag):

- transition: $Rack = t : Tile \in Bag$
  $score = 0$
- output: None
- exception: None

getRackStr():

- transition: None
- output: $r : Rack \rightarrow s : String$
  where r and s represent same set of characters.
- exception: None

getRackArr():

- transition: None
- output: Rack
- exception: None

increaseScore($increase$):

- transition: $score \rightarrow score + increase$
- output: None
- exception: None

getScore():

- transition: None
- output: score
- exception: None

# Board Module

## Module

Board Type

## Uses

N/A

## Syntax

### Exported Constants

N/A

### Exported Types

Board = $16 \times 16$ matrix of Tiles

### Exported Access Programs

| Routine name | In | Out | Exceptions |
|---|---|---|---|
| init | | Board | |
| getBoard | | Board | |
| updateBackBoard | N, N, String, String | | ValueError |

## Semantics

### State Variables

backBoard

### Environment Variables

None

### State Invariant

$|Board| = 256$

### Assumptions

N/A

### Access Routine Semantics

init():

- transition: $Board \rightarrow Board$
- output: None
- exception: None

getLetter():

- transition: None
- output: backBoard
- exception: None

updateBackBoard($row, column, direction, word$):

- transition: $Board \rightarrow Board + word$
  where first letter of word is added from Board[row][column] and the rest are added to row(right) or column(down) depending on direction.
- output: None
- exception: ~~None~~ ValueError

# EndTurn Module

## Module

## Uses

Tiles, Bag, Rack

## Syntax

### Exported Constants

N/A

### Exported Types

N/A

### Exported Access Programs

| Routine name | In | Out | Exceptions |
|---|---|---|---|
| updateFrontBoard | $\mathbb{N}, \mathbb{N}$, String, String | List | ValueError |
| removeTile | String, Rack | | |
| exchangeTile | String, Rack | | |
| calculateScore | $\mathbb{N}, \mathbb{N}$, String, String | N | ValueError |
| checkWinState | Rack, Rack, Bag | $\mathbb{B}$ | |

## Semantics

### State Variables

word_score

### Environment Variables

None

### State Invariant

N/A

### Assumptions

N/A

### Access Routine Semantics

updateFrontBoard($row, column, direction, word$):

- transition: Empty $List \rightarrow List$ of Tuples
- output: List of Tuples
- exception: ~~None~~ ValueError

removeTile($word, rack$):

- transition: $Rack \rightarrow Rack \setminus lettersinword$
  $Rack \setminus lettersinword \rightarrow (Rack \setminus lettersinword) + n$
  where n = letters in word.
- output: None
- exception: None

exchangeTile($word, rack$):

- transition: $Rack \rightarrow Rack$
- output: None
- exception: None

calculateScore($row, column, direction, word$):

- transition: $word\_score \rightarrow +(\forall letters \in word \cdot score)$
- output: word_score
- exception: ~~None~~ ValueError

checkWinState($rack1, rack2, bag$):

- transition: None
- output: $\mathbb{B}$
- exception: None

# WordChecks Module

## Module

Correct scrabble word check.

## Uses

N/A

## Syntax

### Exported Constants

N/A

### Exported Types

N/A

### Exported Access Programs

| Routine name | In | Out | Exceptions |
|---|---|---|---|
| checkRack | String, Rack | $\mathbb{B}$ | |
| checkInDict | String | $\mathbb{B}$ | |

## Semantics

### State Variables

N/A

### Environment Variables

~~N/A~~ Dictionary = dic.txt

### State Invariant

N/A

### Assumptions

N/A

### Access Routine Semantics

checkRack($word, rack$):

- transition: None
- output: $\mathbb{B}\ (\exists\ word \in rack \cdot true)$
- exception: None

checkInDict($word$):

- transition: None
- output: $\mathbb{B}\ (\exists\ word \in Dictionary \cdot true)$
- exception: None

# BoardChecks Module

## Uses

WordChecks

## Right Module

### Module

Right Direction Board Checks.

### Syntax

### Exported Constants

N/A

### Exported Types

N/A

### Exported Access Programs

| Routine name | In | Out | Exceptions |
|---|---|---|---|
| occupiedTiles | $\mathbb{N}, \mathbb{B}$, String, Board | $\mathbb{B}$ | |
| adjWordCheck | $\mathbb{N}, \mathbb{B}$, String, Board | $\mathbb{B}$ | |
| outOfBounds | $\mathbb{N}, \mathbb{B}$, String, Board | $\mathbb{B}$ | |
| placementCheck | $\mathbb{N}, \mathbb{B}$, String, Board, $\mathbb{N}$ | $\mathbb{B}$ | |
| rightCheck | $\mathbb{N}, \mathbb{B}$, String, Board, $\mathbb{N}$ | $\mathbb{B}$ | |

### Semantics

### State Variables

matches

### Environment Variables

N/A

**State Invariant**

N/A

**Assumptions**

N/A

**Access Routine Semantics**

occupiedTile($row, column, word, board$):

- transition: None
- output: $\mathbb{B}$ for whether a Tile is occupied or not.
- exception: None

adjWordCheck($row, column, word, board$):

- transition: None
- output: $\mathbb{B}$ if there are adjacent words that can be made with user's word placement.
- exception: None

outOfBounds($row, column, word, board$):

- transition: None
- output: $\mathbb{B}$ if word placement is out of the bounds of the board.
- exception: None

placementCheck($row, column, word, board, count$):

- transition: None
- output: $\mathbb{B}$ for the first word starting at tile $7 \times 7$.
- exception: None

rightCheck($row, column, word, board, count$):

- transition: None
- output: $\mathbb{B}$ for correct placement of word in the right direction using free tiles.
- exception: None

## Down Module

### Module

Down Direction Board Checks.

### Syntax

### Exported Constants

N/A

### Exported Types

N/A

### Exported Access Programs

| Routine name | In | Out | Exceptions |
|---|---|---|---|
| occupiedTiles | $\mathbb{N}, \mathbb{B}$, String, Board | $\mathbb{B}$ | |
| adjWordCheck | $\mathbb{N}, \mathbb{B}$, String, Board | $\mathbb{B}$ | |
| outOfBounds | $\mathbb{N}, \mathbb{B}$, String, Board | $\mathbb{B}$ | |
| placementCheck | $\mathbb{N}, \mathbb{B}$, String, Board, $\mathbb{N}$ | $\mathbb{B}$ | |
| downCheck | $\mathbb{N}, \mathbb{B}$, String, Board, $\mathbb{N}$ | $\mathbb{B}$ | |

### Semantics

### State Variables

N/A

### Environment Variables

N/A

### State Invariant

N/A

### Assumptions

N/A

**Access Routine Semantics**

occupiedTile($row, column, word, board$):

- transition: None
- output: $\mathbb{B}$ for whether a Tile is occupied or not.
- exception: None

adjWordCheck($row, column, word, board$):

- transition: None
- output: $\mathbb{B}$ if there are adjacent words that can be made with user's word placement.
- exception: None

outOfBounds($row, column, word, board$):

- transition: None
- output: $\mathbb{B}$ if word placement is out of the bounds of the board.
- exception: None

placementCheck($row, column, word, board, count$):

- transition: None
- output: $\mathbb{B}$ for the first word starting at tile $7 \times 7$.
- exception: None

downCheck($row, column, word, board, count$):

- transition: None
- output: $\mathbb{B}$ for correct placement of word in the down direction using free tiles.
- exception: None

# MainGame Module

## Uses

~~sys~~, tkinter, ~~Board, Bag, Player, Rack, Tile, BoardChecks, WordChecks, EndTurn~~, WidgetCreation, GameController

## FrontEndMain Module

### Module

Game introduction screens which take players information.

### Syntax

### Exported Constants

N/A

### Exported Types

N/A

### Exported Access Programs

| Routine name | In | Out | Exceptions |
|---|---|---|---|
| init | | tkinter Grid | |
| instructions | | tkinter Grid | |
| getPlayerName | | tkinter Grid | |

### Semantics

### State Variables

turn, player_1rack, player2_rack, roundCount

### Environment Variables

~~N/A~~ *GameWindow · Tkinter grid*

### State Invariant

N/A

### Assumptions

N/A

### Access Routine Semantics

init():

- transition: None
- output: ~~A tkinter grid~~ GameWindow displaying the introduction screen with options to start the game or read the instructions.
- exception: None

instructions():

- transition: None
- output: ~~A tkinter screen~~ GameWindow which lays out the rules of Scrabble.
- exception: None

getPlayerName():

- transition: None
- output: ~~A tkinter screen~~ GameWindow that asks for the two player names.
- exception: None

## BoardFrame Module

### Module

Creates the GUI of the playable scrabble board.

### Syntax

### Exported Constants

N/A

### Exported Types

N/A

### Exported Access Programs

| Routine name | In | Out | Exceptions |
|---|---|---|---|
| scrabbleBoard | tkinter root, tkinter frame, String, String | tkinter Grid | |

### Semantics

### State Variables

root, frame, player1name, player2name

### Environment Variables

*GameWindow · Tkinter grid*

### State Invariant

N/A

### Assumptions

N/A

### Access Routine Semantics

scrabbleBoard(root, frame, player1Name, player2Name):

- transition: None
- output: GameWindow displaying a functional scrabble board, input boxes for word, direction, starting row and column values, shared letters and letters to exchange, and current players turn and their score.
- exception: None

# ~~BoardFrame~~ GameController

## Module

~~Window with scrabble board that controls game play~~ Contains control for the back end logic of the Scrabble game based on user inputs.

## Uses

~~sys~~, tkinter, Board, Bag, Player, Rack, Tile, BoardChecks, WordChecks, EndTurn

## Syntax

### Exported Constants

N/A

### Exported Types

N/A

### Exported Access Programs

| Routine name | In | Out | Exceptions |
|---|---|---|---|
| updateGUI | List of Tile location | | |
| clearEntry | 6 Strings | | |
| skipTurn | 2 Strings | | |
| exchangeTiles | 3 Strings | | |
| scoreBoard | 3 Strings | | |
| completeTurn | 15 Strings | | |
| endChecks | 18 Strings | | |
| endMove | 17 Strings | | |
| updateLabelText | String | | |
| scrabbleBoard | 4 Strings | | |

### Semantics

### State Variables

turn, player_1rack, player2_rack, roundCount

**Environment Variables**

N/A

**State Invariant**

N/A

**Assumptions**

N/A

**Access Routine Semantics**

updateGUI($updateList$):

- transition: Updates board with a tuple of row and column per letter of inputted word.
- output: None
- exception: None

clearEntry($inputWordE, inputRowE, inputColE, inputDirE, inputWordSharedE, inputWordExchange$

- transition: Clears text boxes for game inputs.
- output: None
- exception: None

skipTurn($turnLabel, rackLabel$):

- transition: Skips players turn if the enter button is hit.
- output: None
- exception: None

exchangeTiles($exchangeTiles, label, turnLabel$):

- transition: Exchanges current rack tiles with tiles in the bag.
- output: None
- exception: None

scoreBoard($frame, score1Label, score2Label$):

- transition: Declares winner of game and their score.
- output: None
- exception: None

completeTurn($frame, word, row, col, dir, player, rackLabel, score1Label, score2Label, turnLabel, inputWordE, inputRowE, inputColE, inputDirE, validMoveL$):

- transition: Signifies the completion of a turn.
- output: None
- exception: None

endChecks($frame, word, row, col, dir, player, rackLabel, score1Label, score2Label, turnLabel, inputWordE, inputRowE, inputColE, inputDirE, inputWordSharedE, inputWordExchangeE, validMoveL, sharedLetters$):

- transition: Performs checks on input data from players turn.
- output: None
- exception: None

endMove($frame, word, row, col, dir, rackLabel, score1Label, score2Label, turnLabel, inputWordE, inputRowE, inputColE, inputDirE, inputWordSharedE, inputWordExchangeE, validMoveL, sharedLetters$):

- transition: Takes in user data from window text boxes.
- output: None
- exception: None

updateLabelText($label$):

- transition: Updates window components labels with string input label.
- output: None
- exception: None

scrabbleBoard($root, frame, player1Name, player2Name$):

- transition: Creates initial scrabble board after taking player's names.
- output: None
- exception: None

# WidgetCreation

## ~~BoardLabel~~ MakeLabel Module

### Module

Creates labels for various tkinter window components.

### Syntax

### Exported Constants

N/A

### Exported Types

N/A

### Exported Access Programs

| Routine name | In | Out | Exceptions |
|---|---|---|---|
| init | | tkinter Label | |

### Semantics

### State Variables

N/A

### Environment Variables

N/A

### State Invariant

N/A

### Assumptions

N/A

### Access Routine Semantics

init():

- transition: None
- output: A tkinter label to be attached to the various tkinter window components.
- exception: None

## ~~ColorButton~~ MakeButtons Module

### Module

~~Creates labels~~ Creates button objects for the various tkinter window components.

### Syntax

### Exported Constants

N/A

### Exported Types

N/A

### Exported Access Programs

| Routine name | In | Out | Exceptions |
|---|---|---|---|
| init | String, String, String, String, String | tkinter button | |
| configure | String, String | | |

### Semantics

**State Variables**

N/A

**Environment Variables**

N/A

**State Invariant**

N/A

**Assumptions**

N/A

**Access Routine Semantics**

init($frame, colour, row, column, text$):

- transition: None
- output: A tkinter button representing each tile on the board.
- exception: None

configure($attribute, text$):

- transition: Changes the tile button attributes label.
- output: None
- exception: None