

# **TCP2201 Key Collector Project**

Trimester 1, 2017/2018

**by Invisible Wing**

**Team Leader:**

Goh Kun Shun

017-285 9398

[kunshun225@gmail.com](mailto:kunshun225@gmail.com)

**Team members:**

Christopher Too Wei Bin

010-291 3385

[christophertwoweibin@gmail.com](mailto:christophertwoweibin@gmail.com)

Ng Jing Keong

017-212 5595

[james0523njik@gmail.com](mailto:james0523njik@gmail.com)

Ong Koon Hua

010-280 2826

[koonhua310@gmail.com](mailto:koonhua310@gmail.com)

# Instructions

To run the program from the command line, unzipped the zip file into any folder. Set the directory of the command line to the directory where the files are extracted to. The extracted files should include the following:

- KeyCollectorGame.java
- Tile.java
- Item.java
- Player.java
- Key.java
- Movement.java
- icons folder containing:
  - 1.gif
  - 2.gif
  - 3.gif
  - 4.gif
  - 5.gif
  - a.gif
  - b.gif
  - c.gif
  - d.gif
  - e.gif

In the command line, type in `"javac KeyCollectorGame.java"` and press enter to compile the .java files. Notice that the .class files will be created for each corresponding .java file. Then, type in `"java KeyCollectorGame"`. A windows will popup and the game starts!

# UML Class Diagram

Below are the UML class diagram for the classes we created, **Tile**, **Item**, **Player**, **Key**, **Movement** and **KeyCollectorGame**.

Tile
- x: int - y: int - walkable: Boolean - key: Key - player: Player
+ Tile( x: int, y: int ) + getTileX(): int + getTileY(): int + getPlayer(): Player + getKey(): Key + hasPlayer(): Boolean + hasKey(): Boolean + getWalkable(): Boolean + setPlayer( player: Player ): void + setKey( key: Key ): void + setWalkable( walkable: Boolean ): void + removePlayer(): void + attemptMove( target: Tile ): Boolean

Player
- x: int - y: int - key: Key - player: Player
+ Player( name: String, iconPath: String, x: int, y: int ) + setCoordinates( x: int, y: int ): void + getX(): int + getY(): int + clearKeys(): void + keysCollected(): String + numKeysCollected(): int + addKey( k: Key ): void

Item
- name: String - icon: ImageIcon # x: int # y: int # movement: Movement
+ Item( name: String, iconPath: String, movement: Movement ) + getName(): String + getIcon(): ImageIcon + getMovement(): Movement + setX( x: int ): void + setY( y: int ): void + addkey( k: Key ): void + getKey(): Key

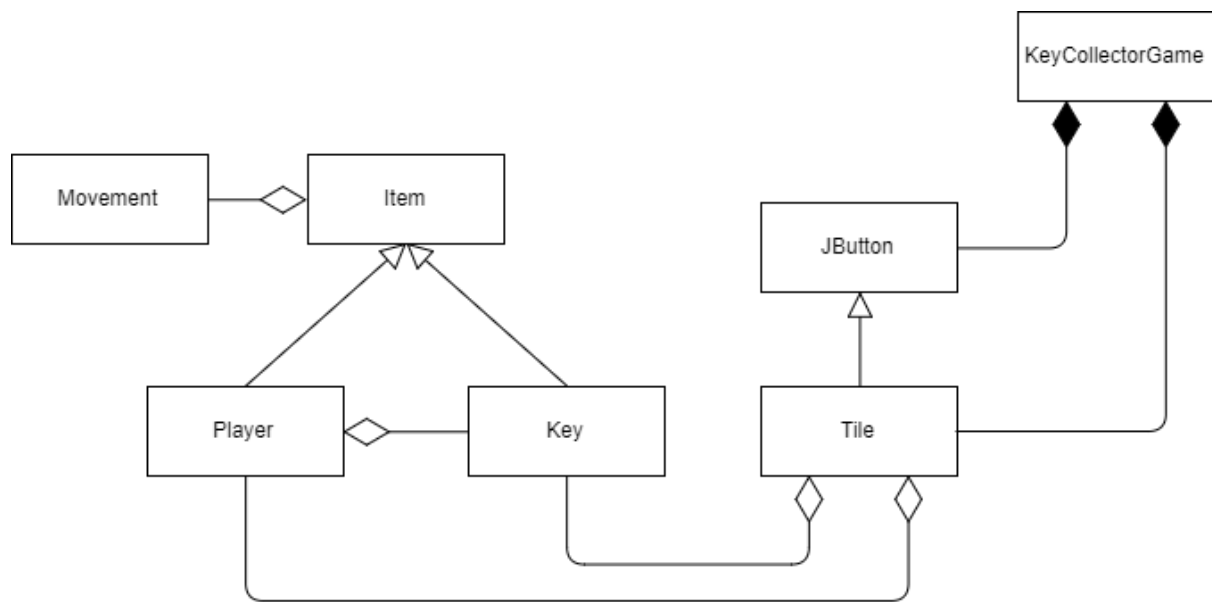
Movement
- horizontal: int - vertical: int - diagonal: int - mustSkip: Boolean
+ Movement( horizontal: int, vertical: int, diagonal: int, mustSkip: int ) + getHorizontal(): int + getVertical(): int + getDiagonal(): int + mustSkip: Boolean

Key
+ Key( name: String, iconPath: String, movement: Movement )

### KeyCollectorGame

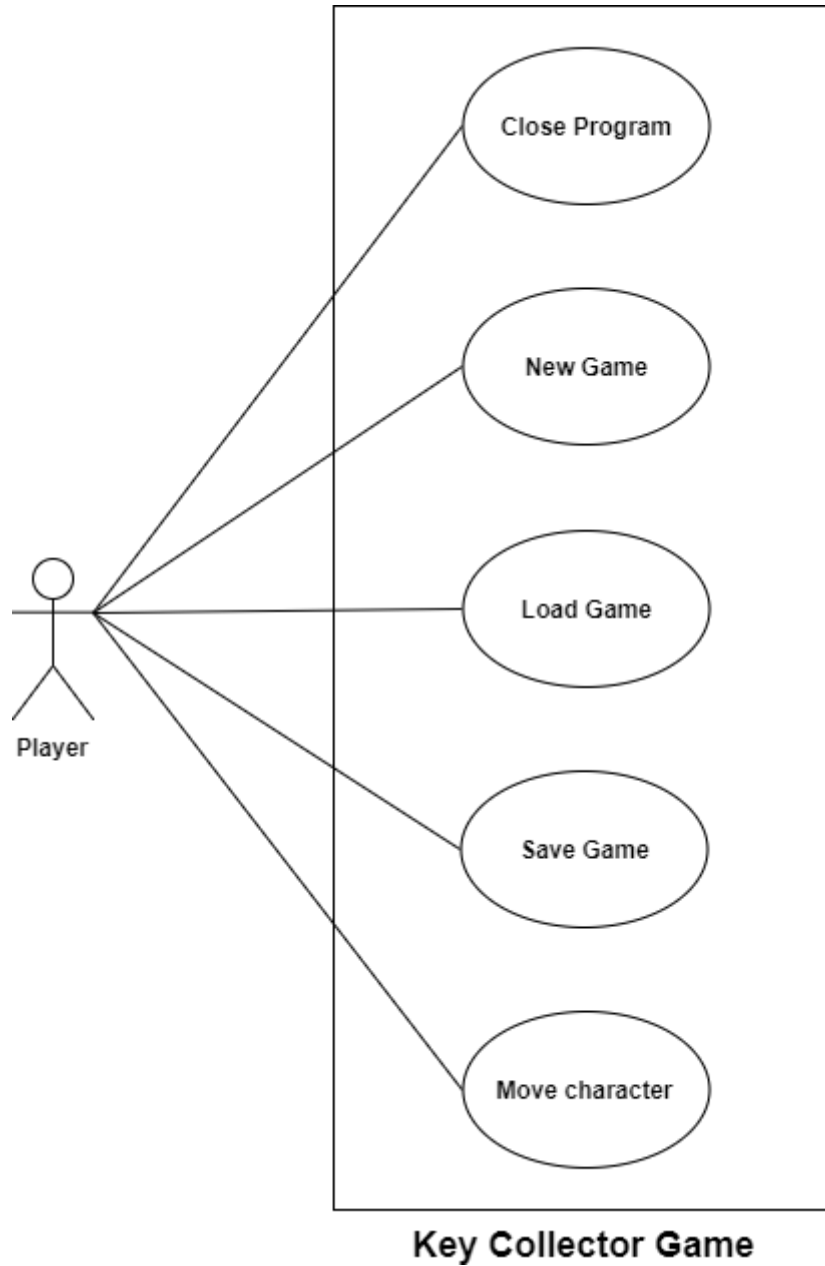
~ tiles: Tile [ ] [ ]  
~ players: Player[ ]  
~ keys: Key[ ]  
~ current\_player: Player  
~ current\_tile: Tile  
~ round: int  
~ gameOver: Boolean  
~ gameStatus: String  
~ statusPanel: JPanel  
~ statusLabel: javax.swing.JLabel  
~ actionPanel: JPanel  
~ menuPanel: JPanel  
~ actionLabel: javax.swing.JLabel

+ main: void  
+ KeyCollectorGame()  
- initialize(): void  
- randomPlaceKeys(): void  
- checkWalkable(): void  
- resetWalkable(): void  
- updateGameStatus(): void  
- updateActionLabel( s: String ): void  
- newGame(): void  
- clearBoard(): void  
- loadGame(): void  
- saveGame(): void  
+ actionPerformed(ae: ActionEvent): void



The figure above shows how the classes are related to each other. Every **Item** has a **Movement**. A **Player** and **Key** is an **Item**. A **Tile**, which inherits from the **JButton** can have a **Key** and a **Player**. A **Player** can have none to many **Keys**. The **KeyCollectorGame** is composed of **JButtons** and **Tiles**.

# Use Case Diagram



# Sequence Diagrams

