CSC301: Project
Team Subpar's Game Report:
System Design Report

Table of Contents

CRC Cards

Class name: Game

Parent class (if any): N/A

Subclasses (if any): N/A

Responsibilities:

* It manipulates the states of the game

* Tells the game when and what to render

Collaborators:

* GameObjectHandlerView class

* MapMaker

Class name: GameObjectHandlerView

Parent class (if any): N/A

Subclasses (if any): N/A

Responsibilities:

* It contains all the elements that will be drawn.

* Can tick and render through all those elements, when appropriate.

Collaborators:

* GameObject class

*MovableObject class

* Player class

* Wall class

Class name: GameObject

Parent class (if any): N/A

Subclasses (if any): MovableObject, Player, Walls

Responsibilities:

- An arbitrary rectangle that should know how to draw itself and update itself as needed.

Collaborators:

* Player class

* Wall class

* MovableObject class

* GameObjectHandlerView class

Class name: MapMaker

Parent class (if any): N/A

Subclasses (if any): N/A

Responsibilities:

* Initialize given objects

* Add those objects to the game

Collaborators:

* PlayerBuilder class

* MapReader class

* GameObjectHandlerView class

Class name: MapReader

Parent class (if any): N/A

Subclasses (if any): N/A

Responsibilities:

*Ability to read and choose a random file from a given directory and interpret a file and create objects accordingly

Collaborators:

* MapMaker class

* Player class

* Wall Class

Class name: PlayerBuilder

Parent class (if any): N/A

Subclasses (if any): N/A

Responsibilities:

* Creates a player given specific parameters

Collaborators:

* Player class

Class name: MovableObject

Parent class (if any): GameObject

Subclasses (if any): Player

Responsibilities:

* A GameObject that can move and has velocity

Collaborators:

* Player class

* GameObject

Class name: Player class

Parent class (if any): MovableObject

Subclasses (if any): N/A

Responsibilities:

* notifies the GameObject interface when a game “tick” is updated such as when the player is moving.

* detects walls and other players.

* collides with other objects.

Collaborators:
* GameObjectHandlerView class

* Wall class

* KeyHandler clas

Class name: Wall class

Parent class (if any): GameObject

Subclasses (if any): N/A

Responsibilities:

* walls follow a set number of rules, such as blocking a player or rebounding a shot.

* is used to create the map.

Collaborators:

* Player class

Class name: KeyHandler

Parent class (if any): N/A

Subclasses (if any): N/A

Responsibilities:

* Handles all the key presses and notifies all of it's observers that are looking for key strokes.

* Moves the player

Collaborators:

* Observable

* Player class

* MainMenu class

Class name: MenuMain

Parent class (if any): N/A

Subclasses (if any): N/A

Responsibilities:

* Guides the user to the game

* Moves the player

Collaborators:

* Observable

System Interaction

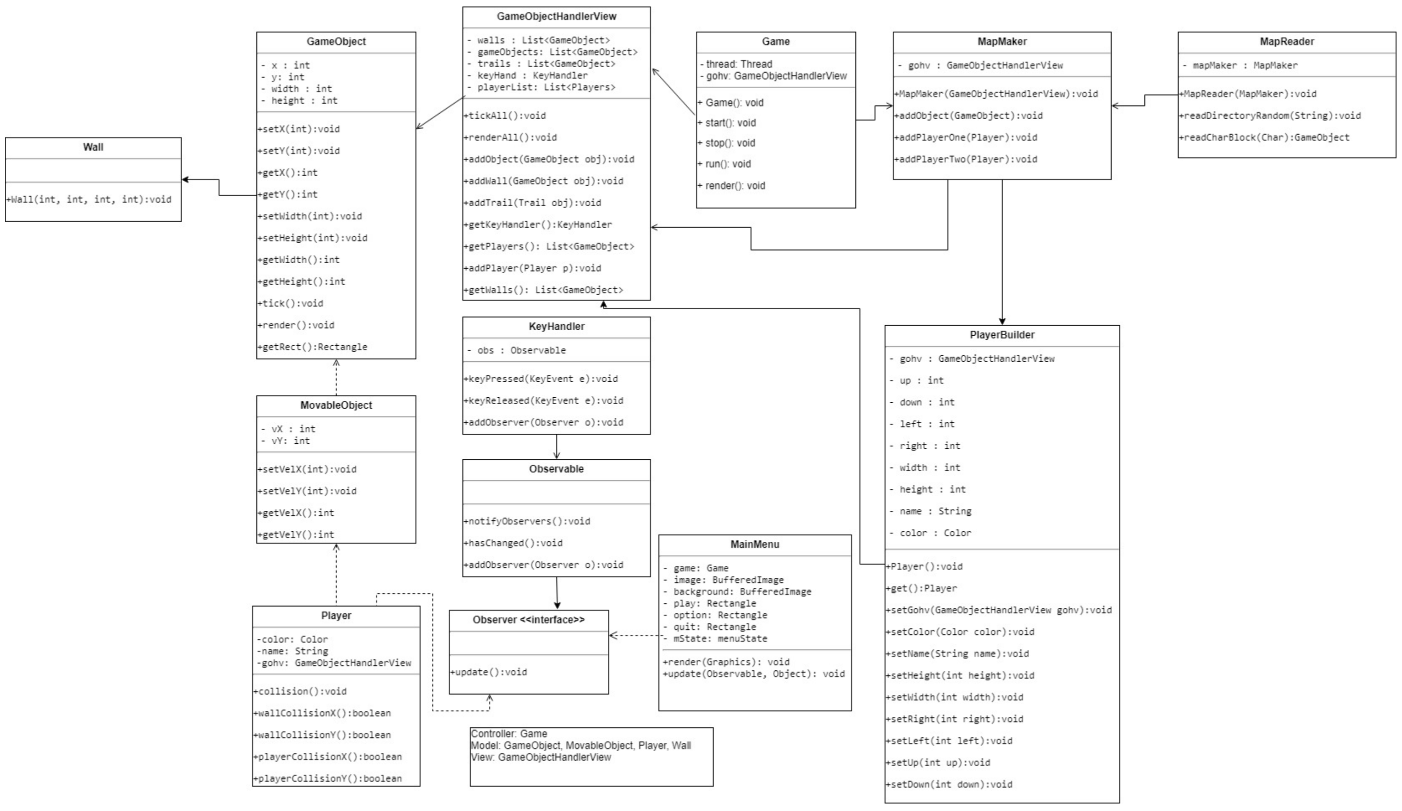
- The system should be able to run a .jar file, so they would need java of a version of 8.0.1110.14 or greater

----

System Decomposition

- Since the user input is whitelisted, the program can only accept user keystrokes that move the player avatar. Therefore, no exception handling should be needed for failure testing.

System Architecture


GameObjectHandlerView
- walls : List<GameObject>
- gameObjects: List<GameObject>
- trails : List<GameObject>
- keyHand : KeyHandler
- playerList: List<Players>
+tickAll():void
+renderAll():void
+addObject(GameObject obj):void
+addWall(GameObject obj):void
+addTrail(Trail obj):void
+getKeyHandler():KeyHandler
+getPlayers(): List<GameObject>
+addPlayer(Player p):void
+getWalls(): List<GameObject>
GameObject
- x : int
- y: int
- width : int
- height : int
+setX(int):void
+setY(int):void
+getX():int
+getY():int
+setWidth(int):void
+setHeight(int):void
+getWidth():int
+getHeight():int
+tick():void
+render():void
+getRect():Rectangle
Wall
+Wall(int, int, int, int):void
Game
- thread: Thread
- gohv: GameObjectHandlerView
+ Game(): void
+ start(): void
+ stop(): void
+ run(): void
+ render(): void
MapMaker
- gohv : GameObjectHandlerView
+MapMaker(GameObjectHandlerView):void
+addObject(GameObject):void
+addPlayerOne(Player):void
+addPlayerTwo(Player):void
MapReader
- mapMaker : MapMaker
+MapReader(MapMaker):void
+readDirectoryRandom(String):void
+readCharBlock(Char):GameObject
KeyHandler
- obs : Observable
+keyPressed(KeyEvent e):void
+keyReleased(KeyEvent e):void
+addObserver(Observer o):void
MovableObject
- vX : int
- vY: int
+setVelX(int):void
+setVelY(int):void
+getVelX():int
+getVelY():int
Observable
+notifyObservers():void
+hasChanged():void
+addObserver(Observer o):void
MainMenu
- game: Game
- image: BufferedImage
- background: BufferedImage
- play: Rectangle
- option: Rectangle
- quit: Rectangle
- mState: menuState
+render(Graphics): void
+update(Observable, Object): void
Player
-color: Color
-name: String
-gohv: GameObjectHandlerView
+collision():void
+wallCollisionX():boolean
+wallCollisionY():boolean
+playerCollisionX():boolean
+playerCollisionY():boolean
Observer <<interface>>
+update():void
Controller: Game
Model: GameObject, MovableObject, Player, Wall
View: GameObjectHandlerView
PlayerBuilder
- gohv : GameObjectHandlerView
- up : int
- down : int
- left : int
- right : int
- width : int
- height : int
- name : String
- color : Color
+Player():void
+get():Player
+setGohv(GameObjectHandlerView gohv):void
+setColor(Color color):void
+setName(String name):void
+setHeight(int height):void
+setWidth(int width):void
+setRight(int right):void
+setLeft(int left):void
+setUp(int up):void
+setDown(int down):void