

**VIETNAM NATIONAL UNIVERSITY
INTERNATIONAL UNIVERSITY**



**OBJECT-ORIENTED PROGRAMING
PROJECT REPORT**

Game project: Candy Crush

Language: Java 14

Course: OOP - Semester 1 (2020 - 2021) - Group 3

Due date: January, 7th 2021

GROUP 63:

Trần Nam Tuấn - ITITIU19230

Phan Hưng Thịnh - ITDSIU19055

Trương Gia Khang - ITDSIU19041

I. Introduction:

Candy Crush is one of the most popular games which is a free-to-play match-three puzzle video game released by King on April 12, 2012, for Facebook, and another version for iOS, Android, Windows Phone.

This report will present more details of development and implementation of a simple Candy-Crush-style game. The game is written completely in Java 14 with no additional library installed. Most elements are inherited and implemented from the Java AWT library.

II. Program description.

1. The rules.

The game rules are kept to be simple to account for the capability of the group members. The levels are locked until the last level is completed. Each level has a target point that the player needs to reach and a provided amount of moves that they can use. If the target point is reached with enough moves, the player win and the next level is unlocked. The level resets itself if the player loses.

Regarding the puzzle, candies can be matched if three candies or more are aligned in any direction. No special candy is implemented yet. The grid layout of each level can be change inside the level configuration to add or remove tiles.

2. Game design.

a) The main class

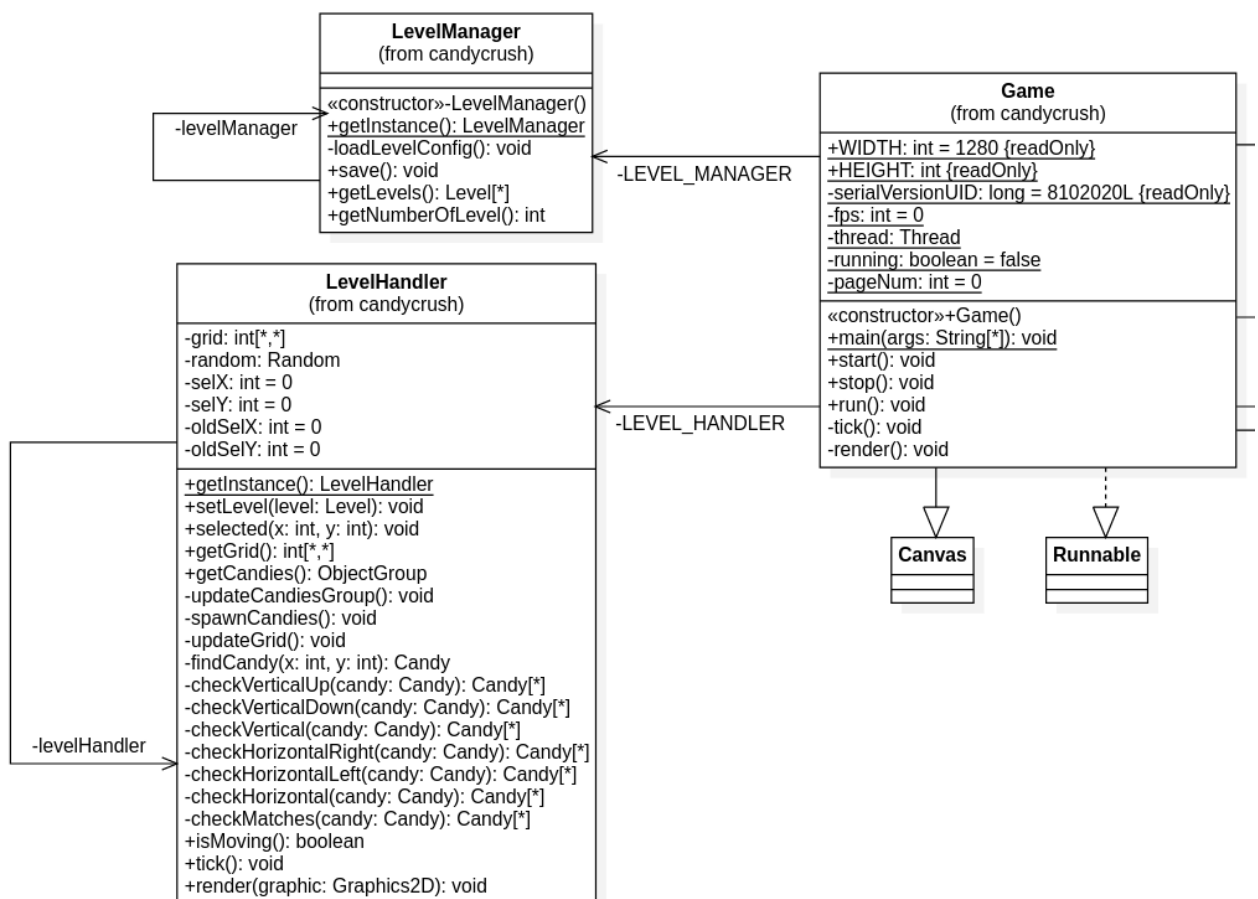
The game runs on the main class, Game, that extends the Canvas class from java AWT and implements the Runnable interface. A thread is created to run the program through the interface. Inside the Game class, there are five key method: start, run, tick, render, stop. A constructor is used to add elements to the game. The main method calls the constructor inside the class itself to start the program.

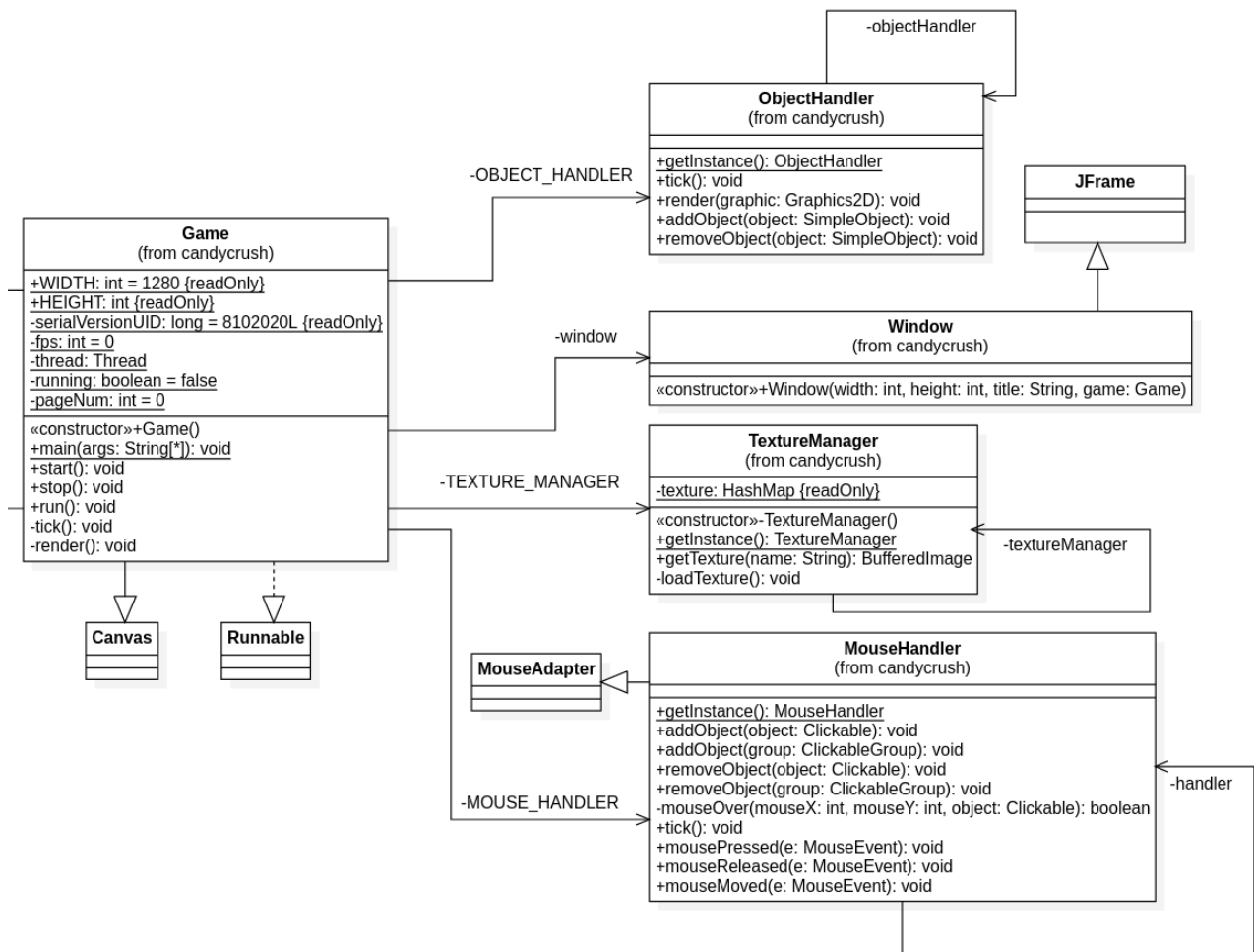
The start method starts the thread. This leads to the run being called afterward. A game loop inside the run method is written so that it can call tick and render 60 times a second, independent of frame rate. That is the game will update 60 time per seconds and render as much frame as possible. Stop method is called to stop the program.

The Windows class extends JFrame to create the window that contains the canvas (the Game class) and lets the render method draws elements. To keep the program simple and because of the AWT limitation, the window is set to not resizable. The window width is set to 1280 and the height is set to 960.

The program runs by executing three handlers: ObjectHandler, MouseHandler, LevelHandler; via the tick method inside Game. These handlers are responsible for executing and rendering components inside the Game constructor. Objects are added to these handlers to be process or removed to disable them.

Additionally, there are resource manager: TextureManager, LevelManager. These managers provide method for loading resources into memory and get them via Hash Map.





b) Objects.

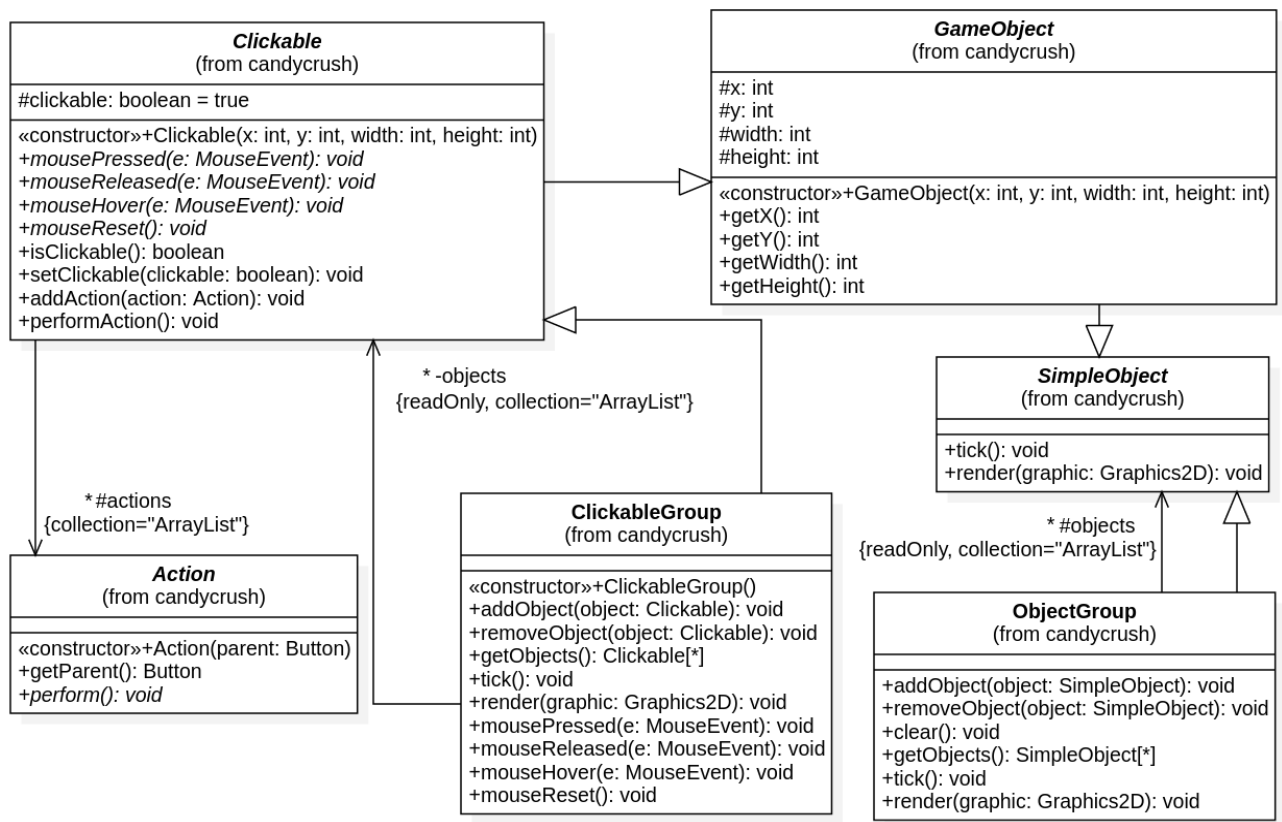
Objects are mainly considered to be instances of class that are added to the handlers, and extend the SimpleObject class.

SimpleObject is the base of most objects. It provided methods for executing and rendering to be call by handlers.

GameObject inherits SimpleObject and adds coordination, width, height as well as getter methods. In addition, ObjectGroup also inherits SimpleObject. This class has a private collection of SimpleObject (Array List) that is executing and rendering one by one via tick and render method.

Similarly, Clickable extends GameObject and added method that can be called by MouseHandler. It also as a collection of Action that will be executed when

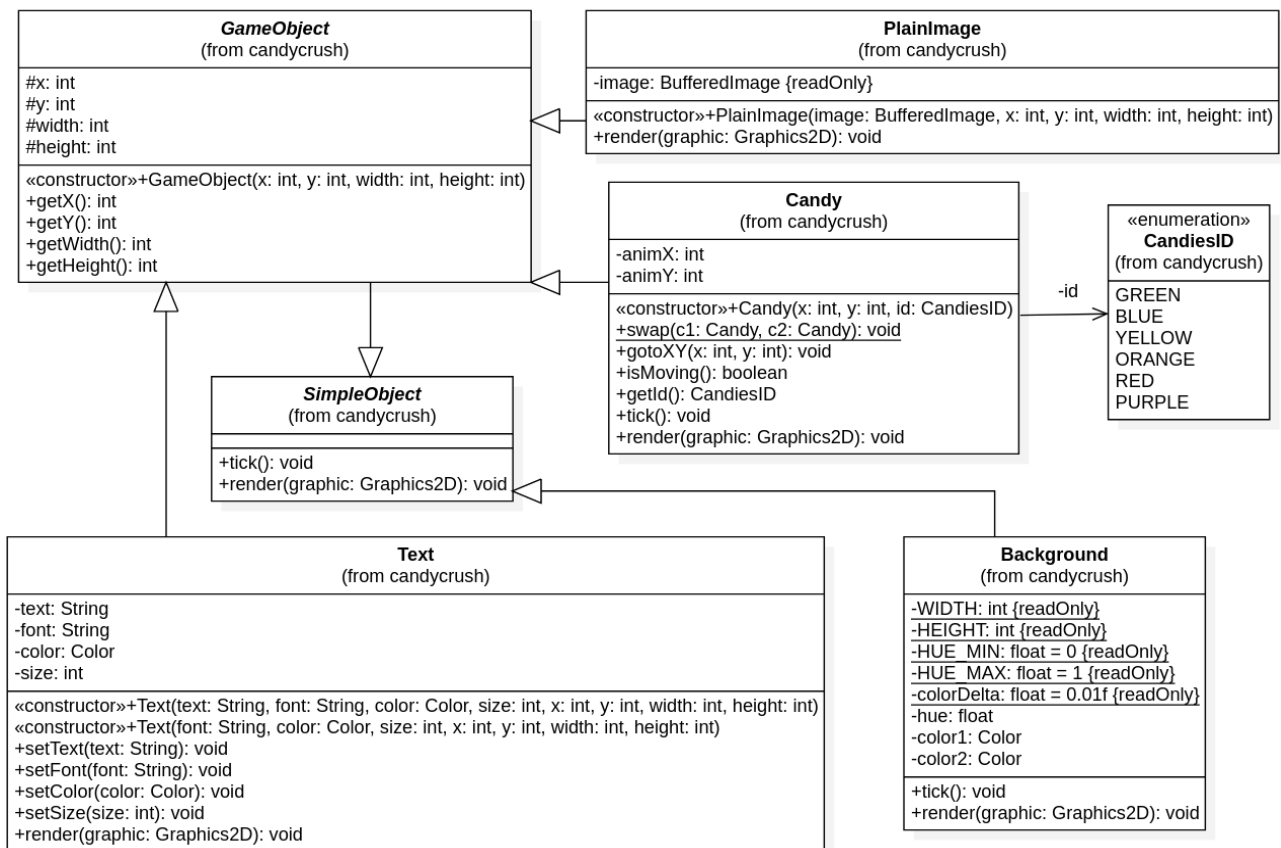
mouseReleased is called. ClickableGroup inherits Clickable and behaves like ObjectsGroup.



From these objects, UI components are constructed to display information on screen.

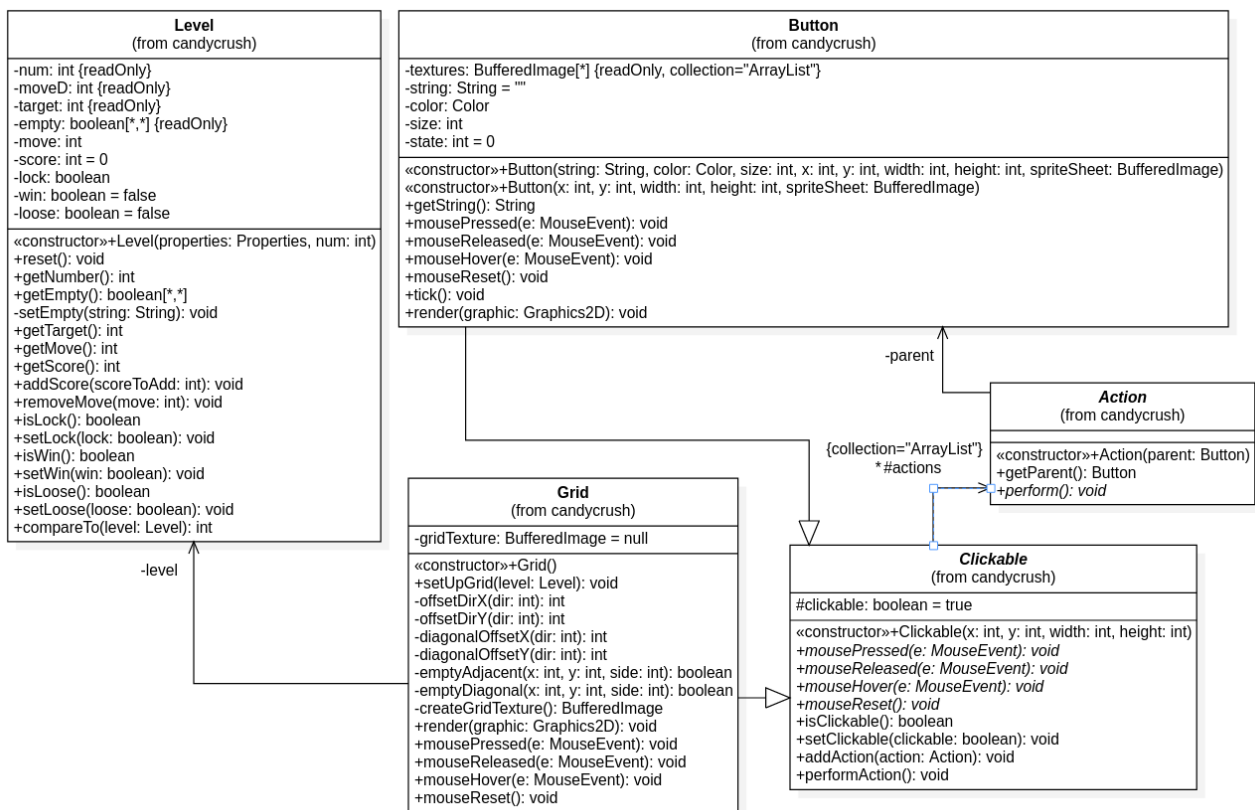
PlainImage and Text are simple as they are only carry out inanimated task. Background, however, has its tick and render overwritten to display a circular gradient of colors. The gradient change the hue value because it wrap around in the color wheel.

Candy, besides providing standard method for handlers, also has special method and attributes that aid in game logic. The class has two type of coordinate, a base coordinate inherited from **GameObject** and a coordinate for animation. This is designed to make match checking easier. However, this approach create a visual bug where candies will overlap in rendering when created. Attempt at fixing this issues created more complex problems that make it not worth the effort. CandiesID enumeration store specific information of each type of candies.



Button and grid both extend Clickable. The Button class has two constructor, one with label and the other with an empty one. An Action can be add to Button as a task to be perform when pressed. Texture for Button has three states that are loaded from a sprite sheet: state 0 for normal or untouched, state 1 for when the mouse is over it, and state 2 when it is pressed. The Grid class provided method for creating the texture for the grid. Since each level can have a custom grid, the class will base on the configuration files of each level, loaded from LevelManager, to construct a new grid texture each level. The texture is only generated once every level start to optimize for performance.

The Level is different from the other object classes. It is loaded by LevelManager and save to a collection. The constructor require a Properties object which is loaded form file form the LevelManager. This Properties object set the attributes like score, move, lock, win and lose state, target point. The empty attribute is an 2D array that contain the information of which tiles in the grid is disabled or voided.

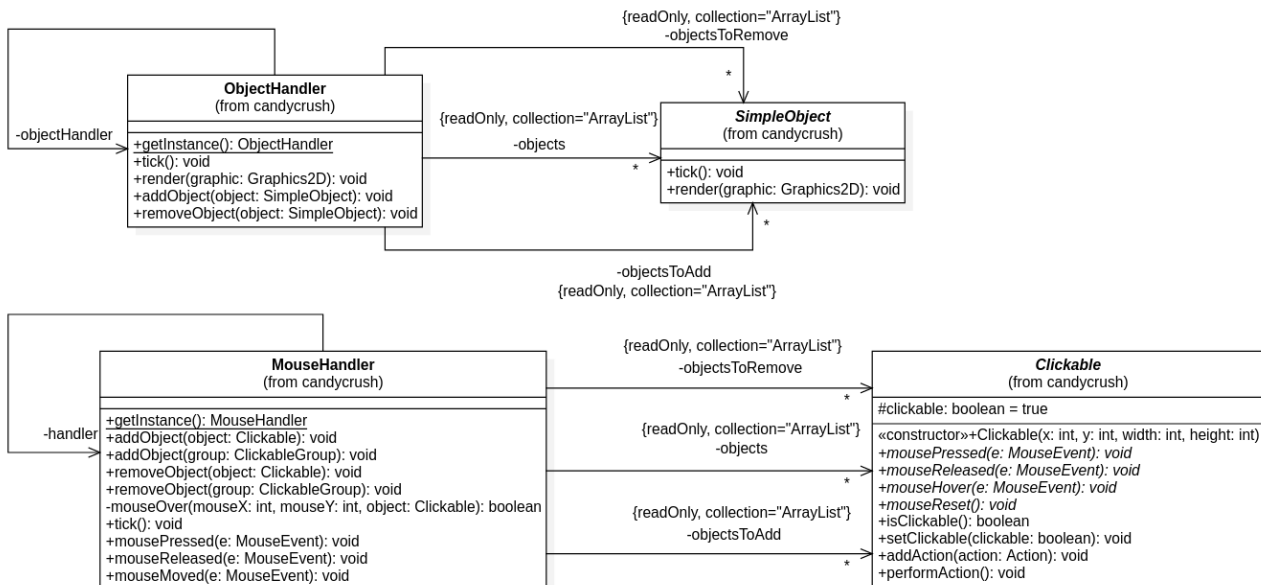


c) Handlers.

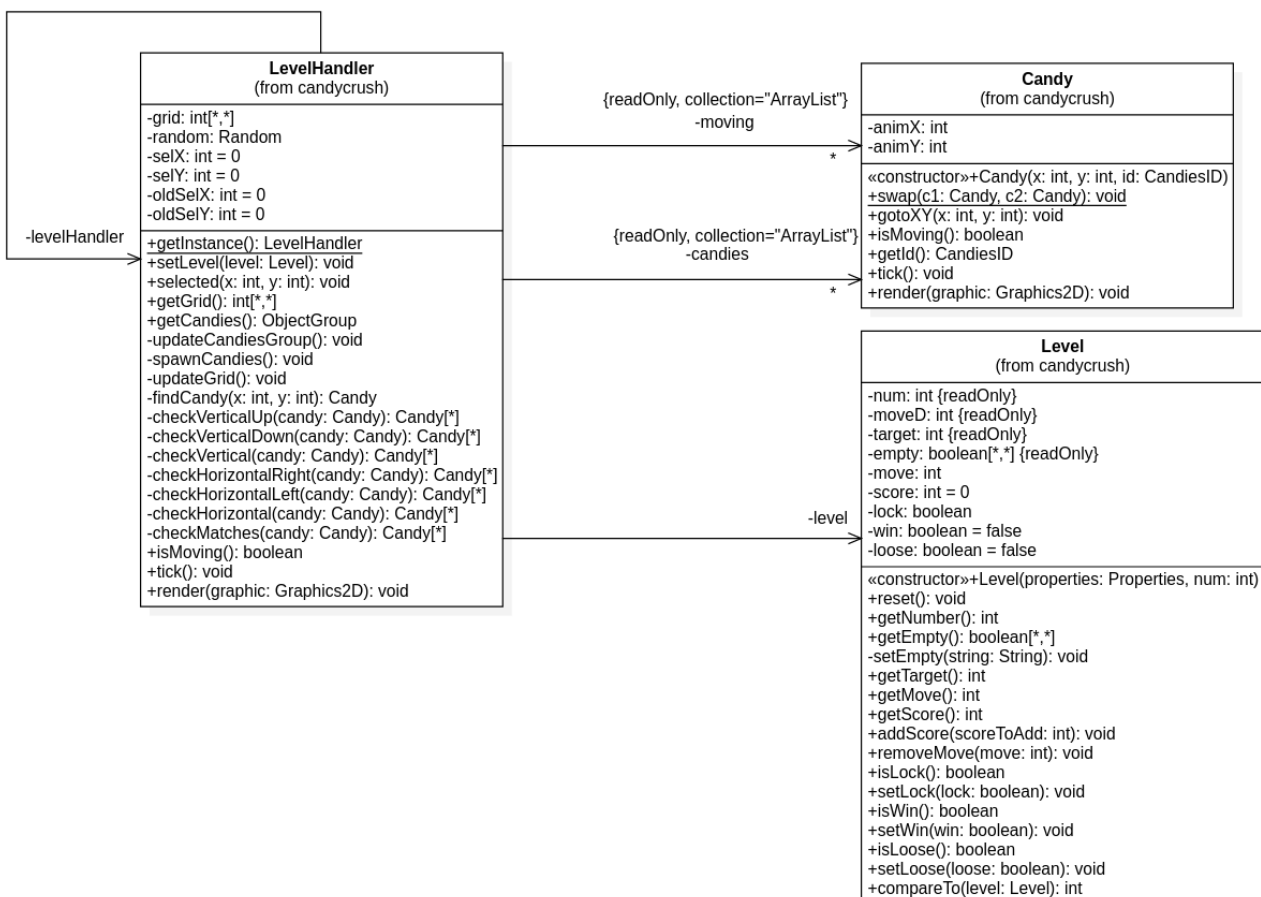
The handlers manage the objects in the program and is designed as singleton class which is declared static in Game class. This is because there should be only one of each type of handlers every time the program start. The act of storing object and call them one by one is also adapted from the Command/Observer pattern. There exists three handlers: ObjectHandler, MouseHandler, LevelHandler. These handlers are called in their respective method of the Game class via the game loop.

The ObjectHandler handles logic and rendering of objects. It contains collection of SimpleObject and this list can only be modified after every items' tick method has been called. Since ObjectGroup inherits SimpleObject, this make object management much easier because a group can be add to the handler.

The MouseHandler is similar to ObjectHandler and extends the MouseAdapter class so that it can be add as a mouse listener of Game (Java AWT Canvas). The mousePressed, mouseReleased and mouseMove methods are called by their respective behaviors when detected by the Canvas.



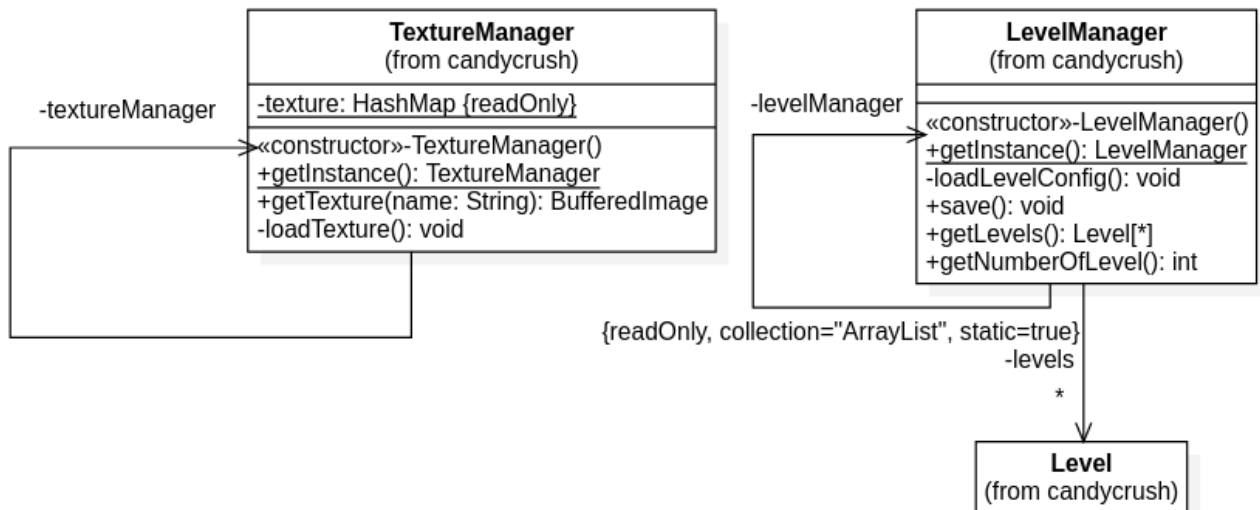
The LevelHandler handles most of the matching logic: spawning candies, moving and matching them. The handler first read information form Level, construct move-able point to a grid. Then, it create candies from a location hidden at the top of the grid.



These candies are then fall down into place. After waiting for the animation, the handler check for input taken by Grid class and move candies accordingly. The a whole grid check for matching in any direction of every candy is carried out. More than or equal to three candies is in a straight line the condition to match. The matched candies is deleted afterward and point is calculated. The level is win if the target is reach and has move above or equal 0, and fail if otherwise. Next level will be unlock if the current level is won. The current level will reset it self if fail.

d) Resource manager.

Resource manager consists of TextureManager for loading and storing texture, LevelManager for loading and saving level information. Both are designed as singletons.




III. Implementation difficulties:

Since the program are design from scratch in Java with no additional library or specialized library for designing game, the process become much harder because there is no predefined standard and efficient workflow.

IV. Conclusion.

While Candy Crush is a simple game concept, the approach of only using vanilla Java does not make the development process easy as it lacks specialized tools for designing game. If the project is to be remake, a gaming developing library or toolkit



will be use so that time can be spent more on perfecting the game play design rather than wasting on creating a base system.

V. References:

Wikihow.com: <https://www.wikihow.com/Play-Candy-Crush-Saga>

Wikipedia.com: https://en.wikipedia.org/wiki/Candy_Crush_Saga

Game loop explantion: <https://gamedev.stackexchange.com/questions/160329/java-game-loop-efficiency>

Smooth gradient background animation java:

<https://stackoverflow.com/questions/45603312/smooth-gradient-background-animation-java>