INFO1113 Inkball Assignment Report

520651281

OO Design Decisions:

App

This class is the main game controller and represents the core functionality (i.e., the main game window). It handles the drawing of game elements such as boards, balls, and player-drawn lines. It manages the level and game states, such as loading the next level when ending in a win, or restarting the same level, updating the score and timers. It coordinates the response to the user inputs such as mouse and keyboard events. The decision to centralize these responsibilities in App allowed for the streamlined management of the game states across levels.

Ball

This class manages movement and collision detection of balls, using helper methods to handle collisions, walls, squiggles, and game boundaries.

Hole, Spawner extends Tile

I decided to consider the “dynamic” tile types as “tiles”; that is, the tiles that have logic that is continuously updated throughout the frames. I implemented the abstract Tile class to serve as their base class. These classes, alongside the TimedTiles and ball class are considered GameObjects since they are “dynamic” elements in at least one non-trivial way, such as moving around (constantly updating position), or inducing change on other elements (holes absorbing nearby balls). The hole class contains logic to attract balls and determine whether a ball has been successfully captured based on matching colors. Having these individual class types allowed me to implement finer grain control over the specific behaviors. For instance, the holes needed to take up 4 tiles, so how I implemented its drawing (and finding its center) had to be slightly different.

Squiggle

This class implements the player-drawn lines by maintaining a list of points forming the squiggle, and provides methods to detect and handle collisions with balls. I decided to encapsulate the squiggle behavior within its own class so that I implement logic dealing with individual squiggles separately, such as only removing the particular squiggle that I have right clicked (which I place a pointer “currentSquiggle” on). Furthermore, each individual squiggle would be able to encapsulate and maintain its own set of points so that logic to do squiggle removal is simple.

UI Elements

This abstract class serves as the base for dynamic UI related elements. For my program, this ended up only consisting of messages, but could easily be extended to other elements such as buttons. I had the Message abstract class extending it, which outlines the basic structure specific message types

that are conditionally displayed at different points in the game, including the TimeUpMessage, and EndMessage, and PausedMessage.

Extension: Timed tiles

These tiles are instances of the class TiledTile extending Tile. The implementation is fairly simple, relying on overriding the updateAlpha() method to gradually decrease the tile's transparency until it becomes completely transparent, upon which the TimedTile instance is marked inactive to disable collisions. Implementing a dedicated class for these tiles allows me to easily extend their usage to other levels by simply placing a 'T' character in the respective level files, since logic handling the printing and initialization of the board is streamlined in the drawBoard() method of App.java.