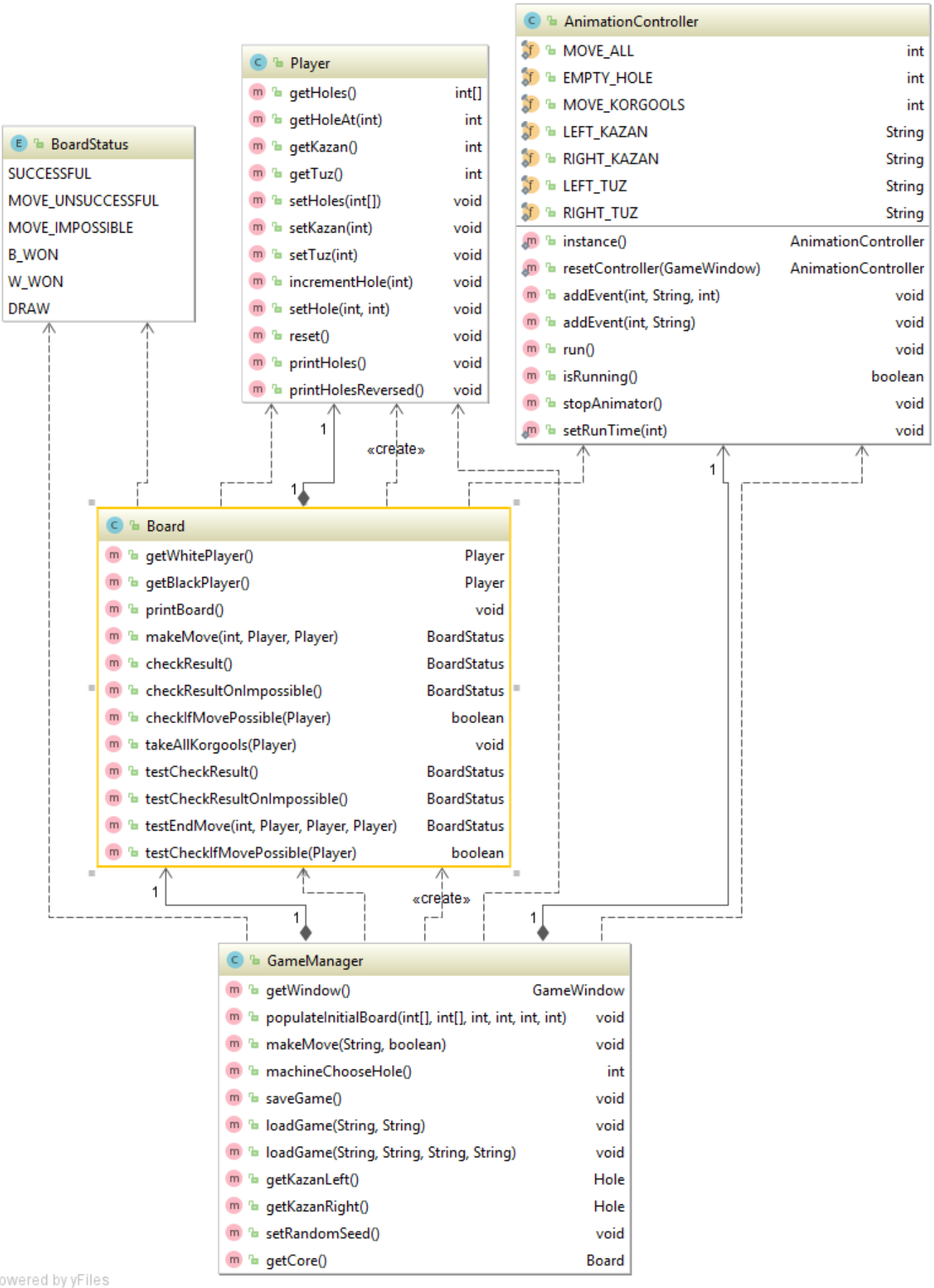
AGILE PROJECT – DESIGN DOCUMENT

Team Platypus

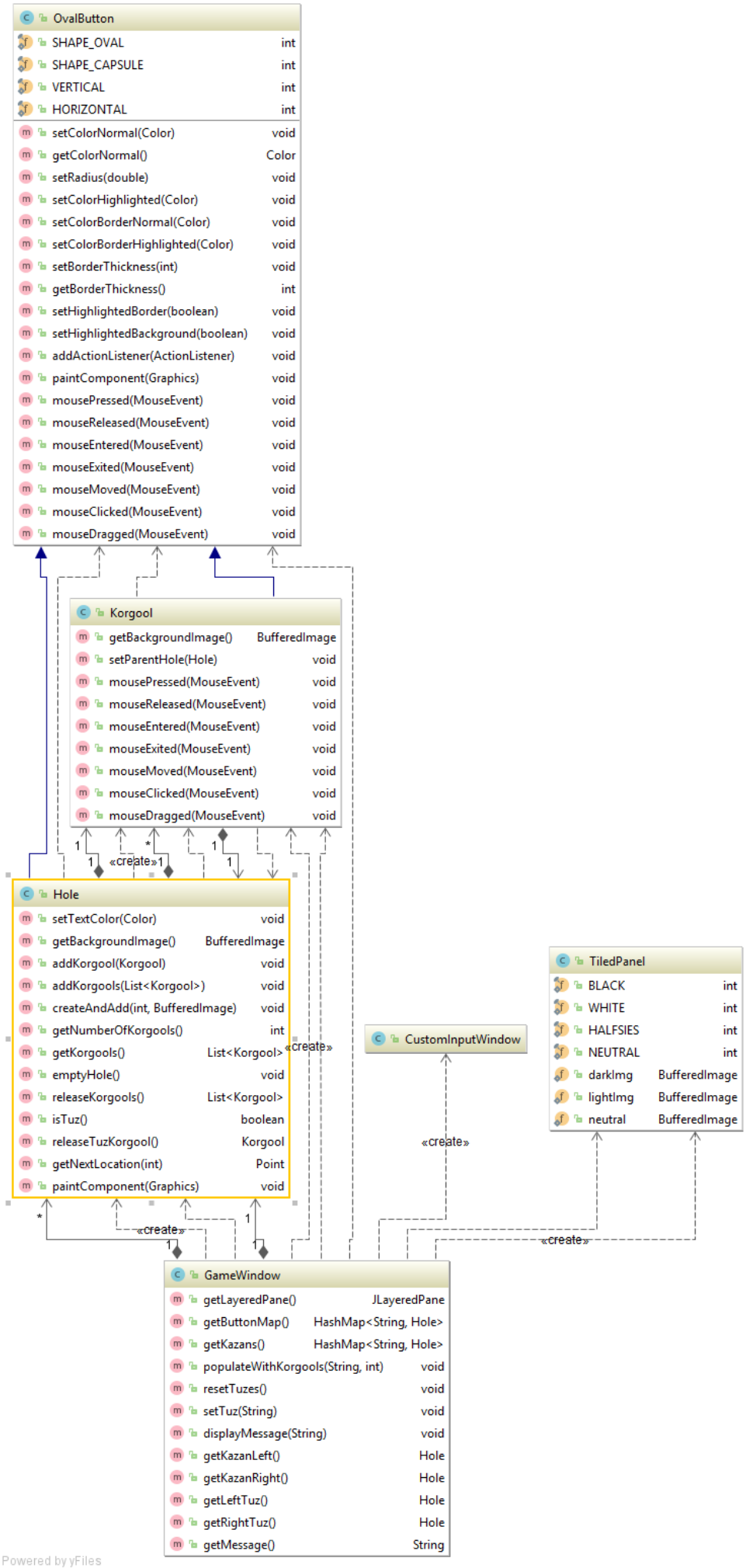
Class Diagrams

Package Logic



Package

GUI



High-Level Design Overview

GUI Package

OvalButton

When initially designing the holes, the team opted for an oval shape before finally settling on a capsule-like shape similar to the real-life game board. The OvalButton class, an extension of JButton, can therefore handle both of these shapes. Oval Buttons as objects have high customizability and can have custom border colours in addition to various backgrounds.

Hole

The Hole class extends OvalButton and provides the means of storing and retrieving korgools. It also handles the resizing of the korgools, and ensures they are always positioned within the border of their parent hole.

Korgool

The Korgool class also extend OvalButton. Korgool objects have no special function besides representing the korgool in game. It dispatches any events (such as a mouse click) to its parent hole.

GameWindow

The GameWindow class forms the foundation of the front-end. It is responsible for constructing the game board and all its various subcomponents. This class extends JFrame and therefore acts as the 'main window' for the game. GameWindow communicates with the GameManager class in the logic package in order to get information on the game state, which it then passes on to the various front-end components.

CustomInputWindow

The CustomInputWindow class is an extension of a JDialog. This screen allows for users to add their own custom input to the game. The window allows users to create a custom game scenario (valid within the rules of the game) using various numerical inputs to set hole and kazan values, along with using radio buttons to set the Tuz on either side. Once the user has confirmed a valid input, the class communicates with the GameManager class to update the backend and GameWindow is updated in turn with the new scenario.

Logic Package

GameManager

The GameManager class oversees connecting the backend of the project (Board), with the GUI (GameWindow). This class contains methods to populate the board in a default state (for example at the start of a new game) and in a custom state.

The core method within GameManger is *makeMove().* This method is called in the GameWindow class when a hole is clicked and oversees both updating backend values with the result of the move and additionally triggering display updates and animations within the GUI. After the user has made their move, the basic AI player selects a random hole (that is either not a Tuz or not empty) within which to make its next move.

If at any point one of the players is unable to make a move (all holes are empty), GameManager handles the distribution of korgools into the kazan and determines which player has won the game. The class also handles situations where a player has won normally.

A method to set the random seed of the AI player was created, in order to generate a predictable sequence of moves to be used in testing.

Board

The Board class represents the real-life, physical board in the game - it is the core logic of the application that is responsible for setting up and storing the state of the board of the game. Each instance of the Board class has 2 players that store the current state of each player's holes, tuzes and kazans. Each instance also stores information on which part of the board we are - white or black player's side.

The *makeMove()* method updates values in the holes of both players according to the game rules and triggers the appropriate animations. Then it calls an *endMove()* method that updates the tuzes and kazans, again according to the rules of the game.

The class also contains methods that check the result of the game after each move - if someone won or if there is a draw, or if there are no more moves to make, or if the game should continue.

Player

The Player class represents part of the board belonging to one player in the game. It stores number of korgools in each hole on the player's part of the board and in the player's kazan, and index of the hole on the opponent's board representing the hole that the player chose as their tuz.

This class contains methods to populate the entities with initial values, empty all the holes, increment or set the number of korgools in a hole, kazan or tuz to given value, in addition to reading their values.

BoardStatus

ENUM type class that stores all possible status of the game after each move.