Software Engineering Methods

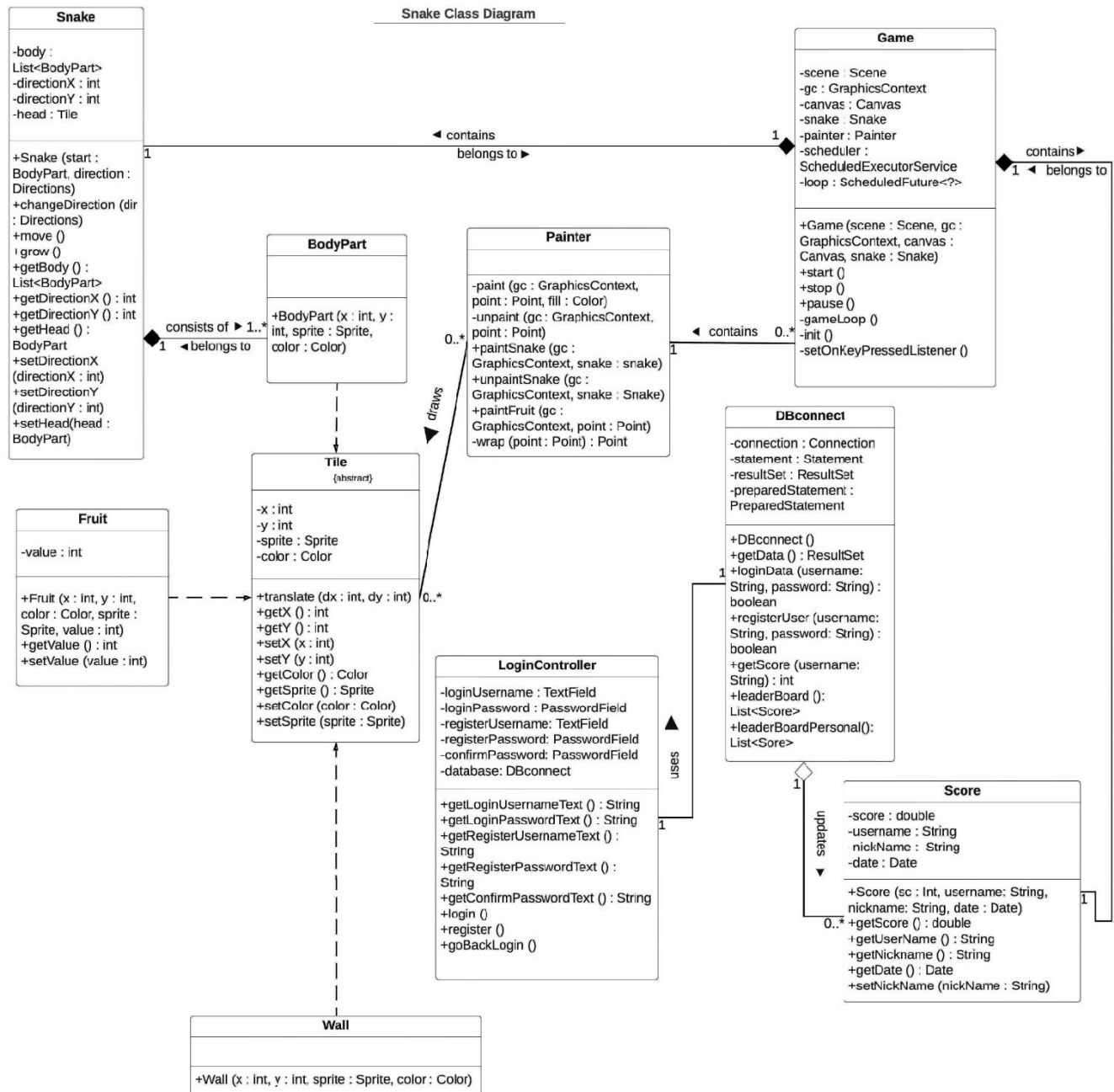# Assignment 2
Snake-Group-1

Table of Contents

Exercise 1) Modelling Class Diagrams

The main part of the game is the Game class. This class contains methods related to a single game on its own, such as pause, stop and start. The Snake class is used to create a snake and holds functionality such as move and change direction. Every game only has exactly one snake and a snake belongs to exactly one game. Every game has a Painter class which is used to draw (i.e. paint) snakes and fruits entities on the game screen.

BodyPart is extended from a general Tile class and is used by the Snake class. A tile represents a square with an x- and y-coordinate on an invisible grid which overlays the game screen. Fruit and Wall are two more subclasses of Tile.

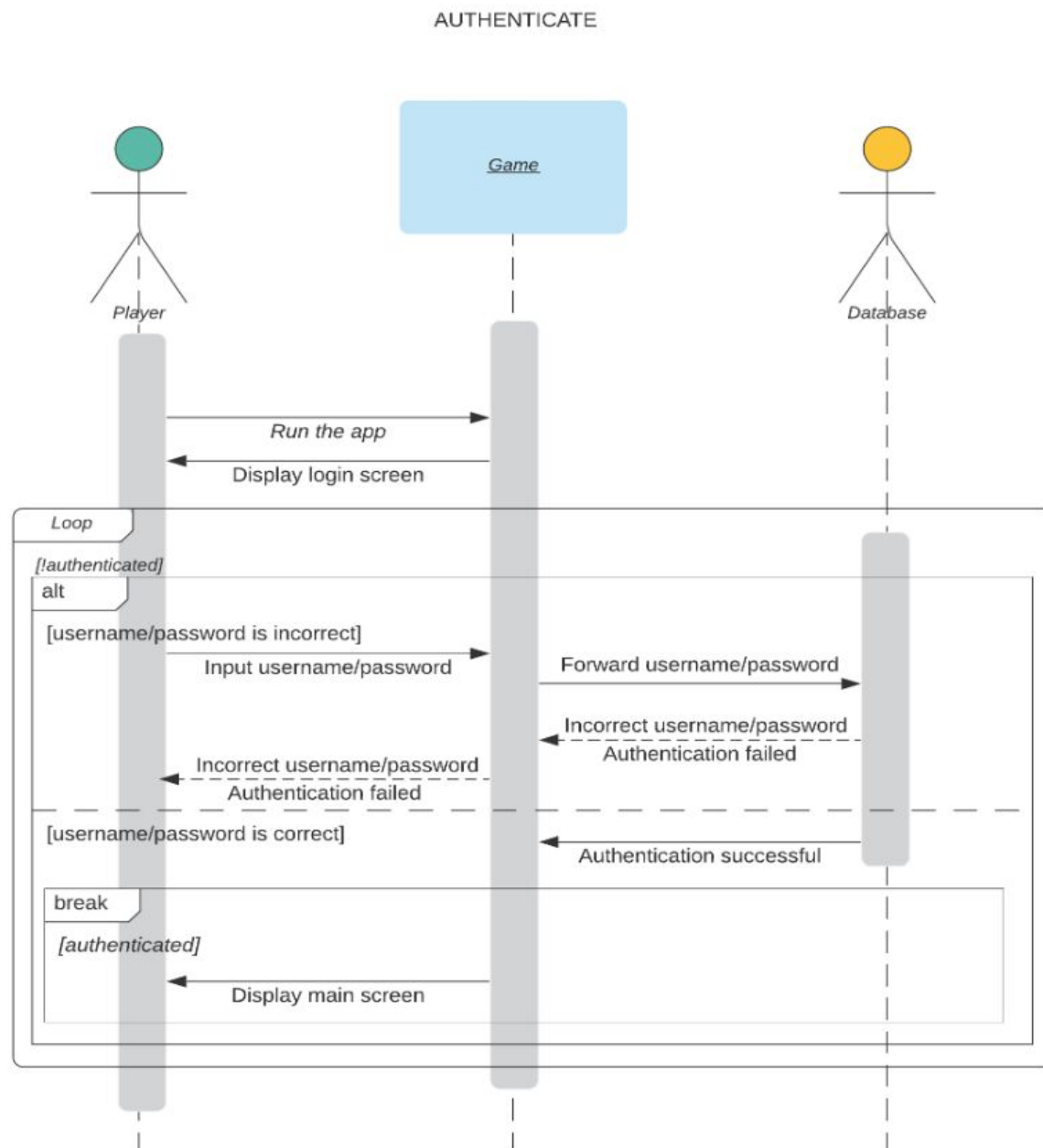DBConnect creates a connection with Database. This class handles creating accounts for new users, logging in existing users, storing the scores obtained per game and getting data that goes on the leaderboard. Score is a model class used to store the score of each game. This class sets the structure in which the score needs to be saved to the database.
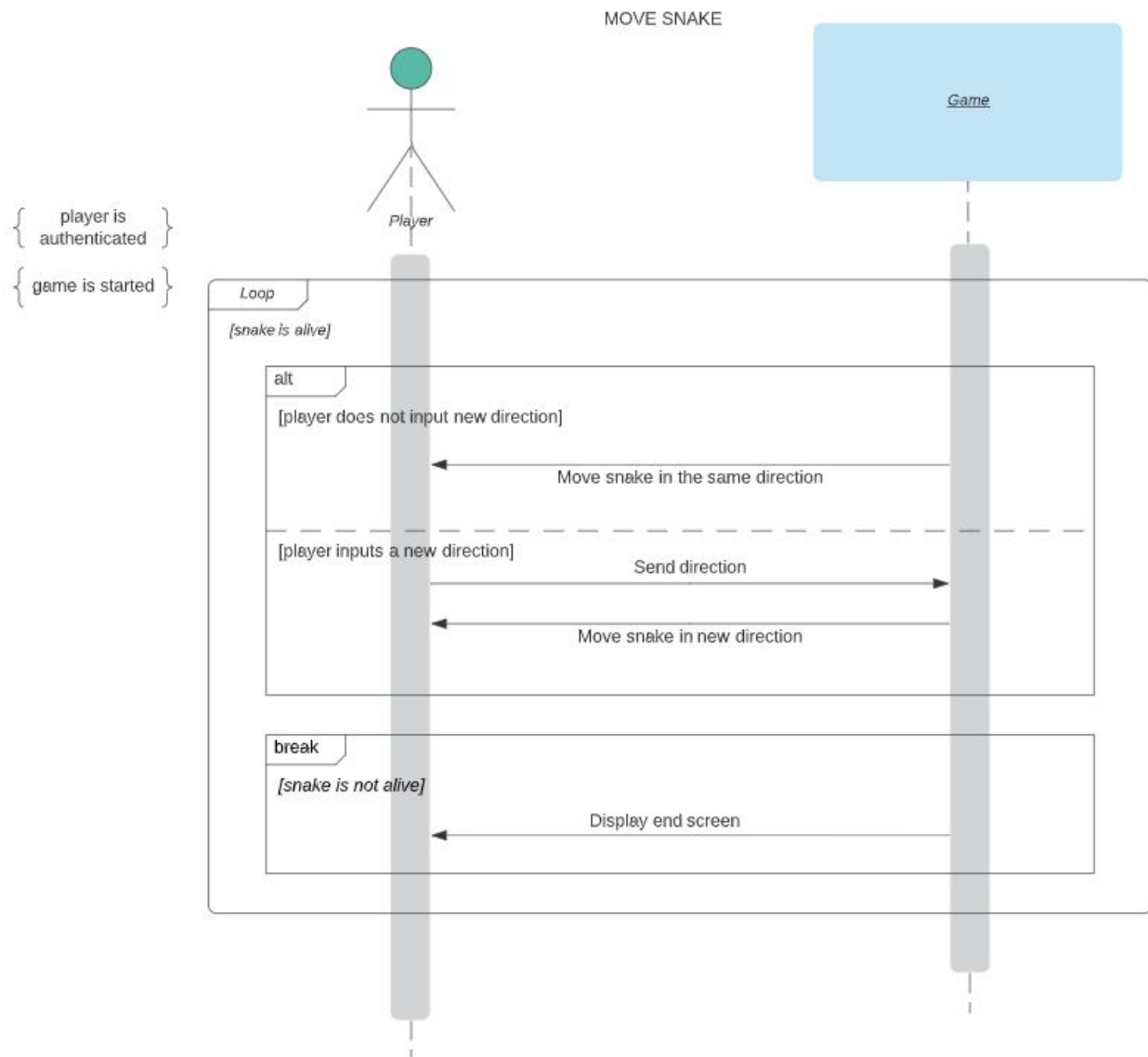
Link to the class diagram: https://bit.ly/2PcvH8W (higher quality diagram)

Snake

-body : List<BodyPart>
-directionX : int
-directionY : int
-head : Tile

+Snake (start : BodyPart, direction : Directions)
+changeDirection (dir : Directions)
+move ()
+grow ()
+getBody () : List<BodyPart>
+getDirectionX () : int
+getDirectionY () : int
+getHead () : BodyPart
+setDirectionX (directionX : int)
+setDirectionY (directionY : int)
+setHead(head : BodyPart)

◄ contains
belongs to ►

Game

-scene : Scene
-gc : GraphicsContext
-canvas : Canvas
-snake : Snake
-painter : Painter
-scheduler : ScheduledExecutorService
-loop : ScheduledFuture<?>

+Game (scene : Scene, gc : GraphicsContext, canvas : Canvas, snake : Snake)
+start ()
+stop ()
+pause ()
-gameLoop ()
-init ()
-setOnKeyPressedListener ()

contains►
belongs to

BodyPart

+BodyPart (x : int, y : int, sprite : Sprite, color : Color)

consists of ► 1..*
◄ belongs to

Painter

-paint (gc : GraphicsContext, point : Point, fill : Color)
-unpaint (gc : GraphicsContext, point : Point)
+paintSnake (gc : GraphicsContext, snake : snake)
+unpaintSnake (gc : GraphicsContext, snake : Snake)
+paintFruit (gc : GraphicsContext, point : Point)
-wrap (point : Point) : Point

◄ contains   0..*

draws

Tile
{abstract}

-x : int
-y : int
-sprite : Sprite
-color : Color

+translate (dx : int, dy : int)
+getX () : int
+getY () : int
+setX (x : int)
+setY (y : int)
+getColor () : Color
+getSprite () : Sprite
+setColor (color : Color)
+setSprite (sprite : Sprite)

Fruit

-value : int

+Fruit (x : int, y : int, color : Color, sprite : Sprite, value : int)
+getValue () : int
+setValue (value : int)

DBconnect

-connection : Connection
-statement : Statement
-resultSet : ResultSet
-preparedStatement : PreparedStatement

+DBconnect ()
+getData () : ResultSet
+loginData (username: String, password: String) : boolean
+registerUser (username: String, password: String) : boolean
+getScore (username: String) : int
+leaderBoard (): List<Score>
+leaderBoardPersonal(): List<Sore>

uses

LoginController

-loginUsername : TextField
-loginPassword : PasswordField
-registerUsername: TextField
-registerPassword: PasswordField
-confirmPassword: PasswordField
-database: DBconnect

+getLoginUsernameText () : String
+getLoginPasswordText () : String
+getRegisterUsernameText () : String
+getRegisterPasswordText () : String
+getConfirmPasswordText () : String
+login ()
+register ()
+goBackLogin ()

updates

Score

-score : double
-username : String
-nickName : String
-date : Date

+Score (sc : Int, username: String, nickname: String, date : Date)
+getScore () : double
+getUserName () : String
+getNickname () : String
+getDate () : Date
+setNickName (nickName : String)

Wall

+Wall (x : int, y : int, sprite : Sprite, color : Color)

Exercise 2) Modelling Sequence Diagrams

Use case 1: Authenticate

AUTHENTICATE

Use case 2: Move Snake

MOVE SNAKE

Game

{ player is authenticated }

Player

{ game is started }

Loop

[snake is alive]

alt

[player does not input new direction]

Move snake in the same direction

[player inputs a new direction]

Send direction

Move snake in new direction

break

[snake is not alive]

Display end screen

Use case 3: Start Game

START GAME

Player

player is
authenticated

Push "START" button

Start the game

Initialize score to 0

Render scene

Game