Project 6 - Project Update - Snakey 2D

Trevor Green, Amanda Smith, Connor Tribbett

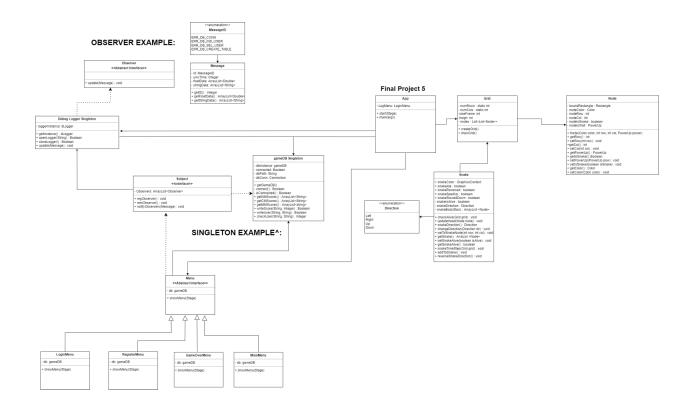Repo: https://github.com/trgr5899/Snakey-2D

Status Summary:

Our team consists of Trevor Green, Amanda Smith, and Connor Tribbett. For our semester project, we are developing a snake game called "Snakey 2D" in Java with JavaFX. So far, we have successfully gotten the game to launch, created a database, and created the beginnings of the snake game. Amanda worked on the login, registration, and game over menus. Connor worked on the database for login information and scores. Trevor made the main menu and the beginnings of the snake game. After implementing the menus, we realized that it would not be effective for them to be a template pattern because each one only requires one function, which varies for all of them. So far, we have implemented the observer and singleton patterns. The observer helps to debug our code and the singletons allow us to implement the database without accidentally creating multiple implementations.

Class Diagram:

Link:
https://drive.google.com/file/d/1KiBxPgXW0jRzdnPVwfM6Q7imhIPQxFNW/view?usp=sharing

OBSERVER EXAMPLE:

<<enumeration>>
MessageID
- ERR_DB_CONN
- ERR_DB_INS_USER
- ERR_DB_SEL_USER
- ERR_DB_CREATE_TABLE

Observer
<<Abstract Interface>>
+ update(Message) : void

Message
- id : MessageID
- unixTime : Integer
- floatData : ArrayList<Double>
- stringData : ArrayList<String>
+ getID() : Integer
+ getFloatData() : ArrayList<Double>
+ getStringData() : ArrayList<String>

Debug Logger Singleton
- loggerInstance : dLogger
+ getInstance() : dLogger
+ openLogger(String) : Boolean
+ closeLogger() : Boolean
+ update(Message) : void

Subject
<<Interface>>
- Observers : ArrayList<Observer>
+ regObserver() : void
+ remObserver() : void
+ notifyObservers(Message) : void

Final Project 5

App
- LogMenu : LoginMenu
+ start(Stage)
+ main(args)

Grid
- numRows : static int
- numCols : static int
- sizeFrame : int
- height : int
- nodes : List<List<Node>>
+ createGrid()
+ drawGrid()

Node
- boundRectangle : Rectangle
- nodeColor : Color
- nodeRow : int
- nodeCol : int
- nodeIsSnake : boolean
- nodeIsWall : PowerUp
+ Node(Color color, int row, int col, PowerUp power)
+ getRow() : int
+ setRow(int row) : void
+ getCol() : int
+ setCol(int col) : void
+ getPowerUp() : PowerUp
+ getIsSnake() : Boolean
+ setPowerUp(PowerUp pow) : void
+ setIsSnake(boolean isSnake) : void
+ getColor() : Color
+ setColor(Color color) : void

gameDB Singleton
- dbInstance: gameDB
- connected: Boolean
- dbPath: String
- dbConn: Connection
+ getGameDB()
- connect() : Boolean
- isConnected() : Boolean
+ getNMScores() : ArrayList<String>
+ getCMScores() : ArrayList<String>
+ getIMScores() : ArrayList<string>
+ writeScore(String, Integer) : Boolean
+ writeUser(String, String) : Boolean
+ checkUser(String, String) : Integer

Snake
+ snakeColor : GraphicsContext
+ snakeAte : boolean
+ snakeReversed : boolean
+ snakeSpedUp : boolean
+ snakeSlowedDown : boolean
- snakeIsAlive : boolean
- snakeDirection : Direction
- snakeBodyStack : ArrayList <Node>
+ checkAlive(Grid grid) : void
+ updateHead(Node node) : void
+ snakeDirection() : Direction
+ changeDirection(Direction dir) : void
+ setToSnakeNode(int row, int col) : void
+ getSnake() : ArrayList <Node>
+ getSnakeAlive(boolean isAlive) : void
+ getSnakeAlive() : boolean
+ snakeTimeStep(Grid grid) : void
+ addToSnake() : void
+ reverseSnakeDirection() : void

<<enumeration>>
Direction
Left
Right
Up
Down

SINGLETON EXAMPLE^:

Menu
<<Abstract Interface>>
- db : gameDB
+ showMenu(Stage)

LoginMenu
- db : gameDB
+ showMenu(Stage)

RegisterMenu
- db : gameDB
+ showMenu(Stage)

GameOverMenu
- db : gameDB
+ showMenu(Stage)

MainMenu
- db : gameDB
+ showMenu(Stage)

Plan for Next Iteration:

The plan for the next iteration is to finish the project. Which means finishing all of the game classes, powerup factory, and our command pattern. Basically in this iteration we layed out the basic framework for the game and now we need to finish implementing it. By the due date we plan on having the entire project finished, exactly how we laid it out in our project design for project 5. The rest of the work will probably take at least 20 hours if all 3 of us work on it. We should have plenty of time to finish. If we do not run into any unforeseen problems.