Read Me

This program simulates a Memory Game with a simple GUI. To run it, just run the application on an IDE.

The game has both GUI (actual game play) and console input interaction (for game setup and messages).

The game will first ask you to type player names, you can keep typing as many players as you want, if no players are typed, the game will default to one player.

Then, the user will be asked to choose between two game modes, if an invalid game mode is typed, regular play will be used.
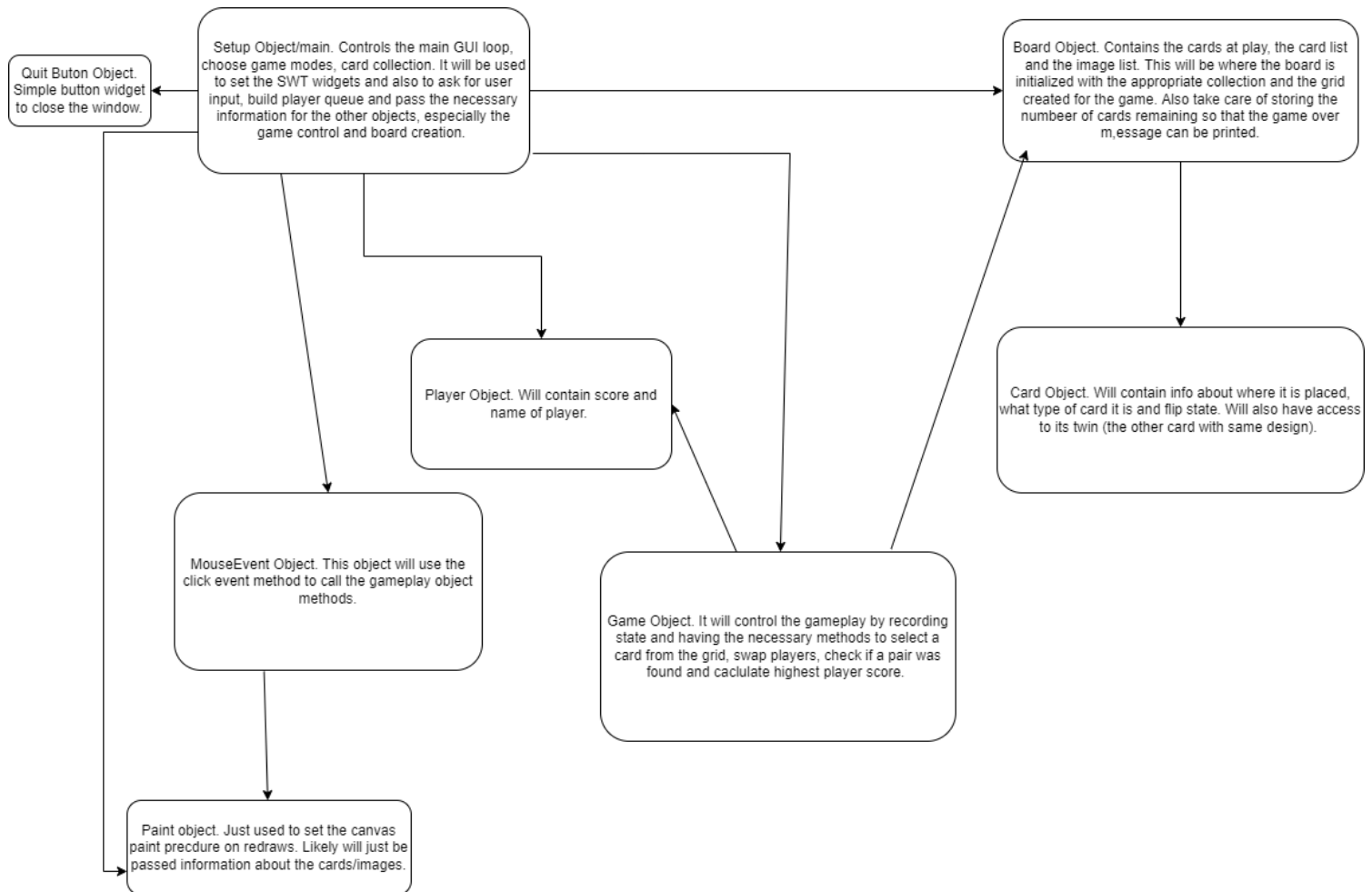
Finally, the user is asked to choose between two sets of cards, if an invalid set is typed, fruit cards will be used.

The game window will then pop open, the player will be able to choose two cards in their turn like a regular memory game. If the player finds a pair, the cards will stay face up. However, if a pair is not found, the player will need to click outside of the grid of cards to turn the cards back down. If the player chooses to just select or click a card, the game will not break, but the player will automatically select the card they just clicked, so it is recommended that the player clicks outside of the grid to reset it if they have not found a pair.

Object Oriented Analysis

Use case:

- When the game opens a prompt should be given for the number of players. The user will be able to type player names into input and the program will stop taking input and those typed players will be added to the player queue.
- Then, there will be a prompt to select the game mode, the two available options will be regular play (where your cards stay flipped and you go next if you get a correct pair) and one flip (where even if you get it right, it changes players turns).
- The next steps will be to either one of two collections.
- Then, the game will start and the board with the cards face down will pop up via GUI (may have GUI before that for the user prompts). The board will have ten pairs of cards face down (number may change, but for now hard coded 6 pairs regardless of number of players or game mode).
- Each player will then take turns, in each turn the player will select a card, it will flip to show what's in it. Then, the same player will select another card and it will flip. If the cards are a pair (match), then the cards will stay flipped and the player that guessed right will be awarded 10 points. Otherwise, both cards will flip back.
  - If the game mode is regular, if the player guessed correctly, then he will go again until he guesses wrong, which goes for the other players' turn.
  - If the game mode is one flip, regardless of whether the player got the matching pair or not, it will be the next player's turn.
- After all cards have been flipped, the game will end and the player with the highest score will be declared the winner.

Object Diagram:

Setup Object/main. Controls the main GUI loop, choose game modes, card collection. It will be used to set the SWT widgets and also to ask for user input, build player queue and pass the necessary information for the other objects, especially the game control and board creation.

Quit Buton Object. Simple button widget to close the window.

Board Object. Contains the cards at play, the card list and the image list. This will be where the board is initialized with the appropriate collection and the grid created for the game. Also take care of storing the numbeer of cards remaining so that the game over m,essage can be printed.

Player Object. Will contain score and name of player.

Card Object. Will contain info about where it is placed, what type of card it is and flip state. Will also have access to its twin (the other card with same design).

MouseEvent Object. This object will use the click event method to call the gameplay object methods.

Game Object. It will control the gameplay by recording state and having the necessary methods to select a card from the grid, swap players, check if a pair was found and caclulate highest player score.

Paint object. Just used to set the canvas paint precdure on redraws. Likely will just be passed information about the cards/images.

Object Oriented Design

Class Diagram:

ConcentrationGame
+ getPlayers(Scanner): Queue<Player>
+ selectGameMode(Scanner): int
+ getCollection(Scanner): int
+ initializeGrid(Display, GameBoard): List<Image>
+ createWidgets(Shell): void
+ setGameInteraction(Shell, GameBoard, Queue<Player>, Display, int): void
+ main(String[]): void

GameBoard
- cardList: List<Card>
- imgList: List<Image>
- blank: Image
- display: Display
- cardsAtPlay: int
- FRUITS: String[]
- ANIMALS: String[]

+ GameBoard(int, Display): Constructor
+ getCard(int): Card
+ size(): int
+ cardsAtPlay(): int
+ makePlay(): void
+ getImgData(): ImageData'
+ getBlank(): Image
+ setImg(Card): void
+ getImgList(): List<Image>
+ initializeCards(String[]): List<Card>
+ initializeGrid(Display, List<Card>): List<Image>

Player
- score: int
- name: String

+ Player(String)
+ addScore(): void
+ getScore(): int
+ getName(): String

CanvasMouseListener
- game: GamePlay
- imgList: List<Image>
- shell: Shell
- canvas: Canvas
- display: Display

+ CanvasMouseListener(Shell, Canvas, Display, List<Image>, GamePlay)
+ mouseDoubleClick(MouseEvent): void
+ mouseDown(MouseEvent): void
+ mouseUp(MouseEvent): void
- updateImgList(): void

ButtonSelectionListener
+ widgetSelected(SelectionEvent: void
+ widgetDefaultSelected(SelectionEvent): void

CanvasPaintListener
+ shell: Shell
+ imgList: List<Image>

+ CanvasPaintListener(Shell, List<Image>)
+ paintControl(PaintEvent): null

GamePlay
- gameCards: GameBoard
- mode: int
- clicks: int
- currCard: Card
- prevCard: Card
- player: Player
- players: Queue<Player>

+ GamePlay(int, Queue<Player>, GameBoard): Constructor
+ checkPair(Card, int): void
+ gamePlay(int, int): void
- printScores(): void
- swapPlayers(): void

Card
- flipped: boolean
- twin: Card
- image: String

+ Card(String): Constructor
+ isTwin(Card): boolean
+ setTwin(Card): void
+ getImage(): String
+ flip(): void
+ unflip(): void
+ isFlipped(): boolean