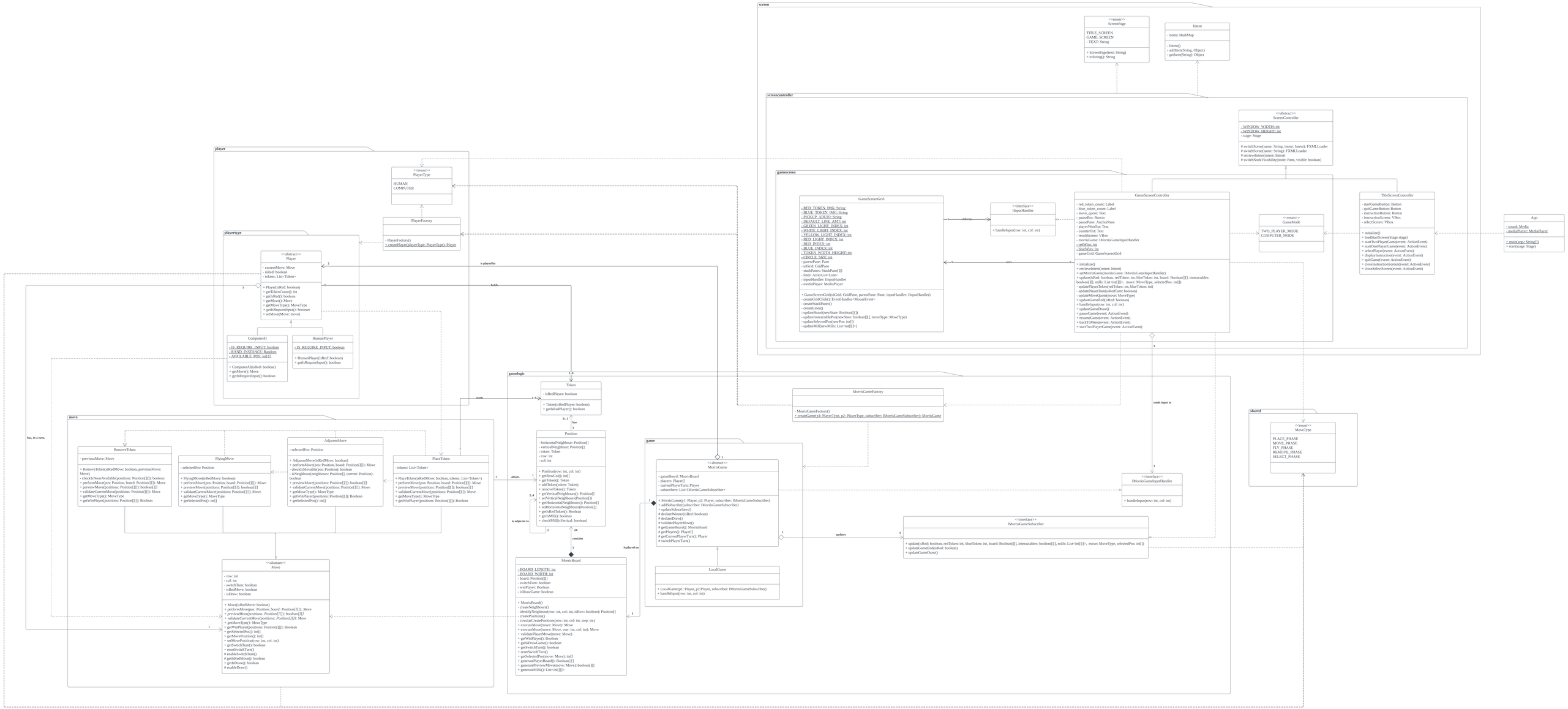
screen
<<enum>>
ScreenPage
TITLE_SCREEN
GAME_SCREEN
- TEXT: String
+ ScreenPage(text: String)
+ toString(): String
Intent
- items: HashMap
- Intent()
- addItem(String, Object)
- getItem(String): Object
screencontroller
<<abstract>>
ScreenController
- WINDOW_WIDTH: int
- WINDOW_HEIGHT: int
- stage: Stage
switchScene(name: String, intent: Intent): FXMLLoader
switchScene(name: String): FXMLLoader
retrieveIntent(intent: Intent)
switchNodeVisibility(node: Pane, visible: boolean)
gamescreen
GameScreenGrid
- RED_TOKEN_IMG: String
- BLUE_TOKEN_IMG: String
- PICKUP_AUDIO: String
- DEFAULT_LINE_ANT: int
- GREEN_LIGHT_INDEX: int
- WHITE_LIGHT_INDEX: int
- YELLOW_LIGHT_INDEX: int
- RED_LIGHT_INDEX: int
- RED_INDEX: int
- BLUE_INDEX: int
- TOKEN_WIDTH_HEIGHT: int
- CIRCLE_SIZE: int
- parentPane: Pane
- srcGrid: GridPane
- stackPanes: StackPane[][]
- lines: ArrayList<Line>
- inputHandler: IInputHandler
- mediaPlayer: MediaPlayer
+ GameScreenGrid(srcGrid: GridPane, parentPane: Pane, inputHandler: IInputHandler)
- createGridClick(): EventHandler<MouseEvent>
- createStackPanes()
- createLines()
- updateBoard(newState: Boolean[][])
- updateInteractablePos(newState: boolean[][], moveType: MoveType)
- updateSelectedPos(newPos: int[])
- updateMill(newMills: List<int[][]>)
<<interface>>
IInputHandler
+ handleInput(row: int, col: int)
inform
turn
GameScreenController
- red_token_count: Label
- blue_token_count: Label
- move_quote: Text
- passeBtn: Button
- passePane: AnchorPane
- playerWonTxt: Text
- counterTxt: Text
- resultScreen: VBox
- morrisGame: IMorrisGameInputHandler
- redWins: int
- blueWins: int
- gameGrid: GameScreenGrid
+ initialize()
+ retrieveIntent(intent: Intent)
+ setMorrisGame(morrisGame: IMorrisGameInputHandler)
+ update(isRed: boolean, redToken: int, blueToken: int, board: Boolean[][], interactables: boolean[][], mills: List<int[][]>, move: MoveType, selectedPos: int[])
+ updatePlayerToken(redToken: int, blueToken: int)
+ updatePlayerTurn(isRedTurn: boolean)
+ updateMoveQuote(move: MoveType)
+ updateGameEnd(isRed: boolean)
+ handleInput(row: int, col: int)
+ updateGameDraw()
+ pauseGame(event: ActionEvent)
+ resumeGame(event: ActionEvent)
+ backToMenu(event: ActionEvent)
+ startTwoPlayerGame(event: ActionEvent)
<<enum>>
GameMode
TWO_PLAYER_MODE
COMPUTER_MODE
TitleScreenController
- startGameButton: Button
- quitGameButton: Button
- instructionButton: Button
- instructionScreen: VBox
- selectScreen: VBox
+ initialize()
+ loadStartScreen(Stage stage)
+ startTwoPlayerGame(event: ActionEvent)
+ startOnePlayerGame(event: ActionEvent)
+ selectPlayer(event: ActionEvent)
+ displayInstruction(event: ActionEvent)
+ quitGame(event: ActionEvent)
+ closeInstructionScreen(event: ActionEvent)
+ closeSelectScreen(event: ActionEvent)
App
- sound: Media
- mediaPlayer: MediaPlayer
+ main(args: String[])
+ start(stage: Stage)
sends input to
player
<<enum>>
PlayerType
HUMAN
COMPUTER
PlayerFactory
- PlayerFactory()
+ createPlayer(playerType: PlayerType): Player
playertype
<<abstract>>
Player
- currentMove: Move
- isRed: boolean
- tokens: List<Token>
+ Player(isRed: boolean)
+ getTokenCount(): int
+ getIsRed(): boolean
+ getMove(): Move
+ getMoveType(): MoveType
+ getIsRequireInput(): boolean
+ setMove(Move: move)
ComputerAI
- IS_REQUIRE_INPUT: boolean
- RAND_INSTANCE: Random
- AVAILABLE_POS: int[][]
+ ComputerAI(isRed: boolean)
+ getMove(): Move
+ getIsRequireInput(): boolean
HumanPlayer
- IS_REQUIRE_INPUT: boolean
+ HumanPlayer(isRed: boolean)
+ getIsRequireInput(): boolean
is played by
has
both
gamelogic
Token
- isRedPlayer: boolean
+ Token(isRedPlayer: boolean)
+ getIsRedPlayer(): boolean
Position
- horizontalNeighbour: Position[]
- verticalNeighbour: Position[]
- token: Token
- row: int
- col: int
+ Position(row: int, col: int)
+ getRowCol(): int[]
+ getToken(): Token
+ addToken(token: Token)
+ removeToken(): Token
+ getVerticalNeighbours(): Position[]
+ setVerticalNeighbours(Position[])
+ getHorizontalNeighbours(): Position[]
+ setHorizontalNeighbours(Position[])
+ getIsRedToken(): Boolean
+ getIsMill(): boolean
+ checkMill(isVertical: boolean)
affects
is adjacent to
contains
MorrisBoard
- BOARD_LENGTH: int
- BOARD_WIDTH: int
- board: Position[][]
- switchTurn: boolean
- winPlayer: Boolean
- isDrawGame: boolean
+ MorrisBoard()
- createNeighbours()
- identifyNeighbour(row: int, col: int, isRow: boolean): Position[]
- createPositions()
- circularCreatePositions(row: int, col: int, step: int)
+ executeMove(move: Move): Move
+ executeMove(move: Move, row: int, col: int): Move
+ validatePlayerMove(move: Move)
+ getWinPlayer(): Boolean
+ getIsDrawGame(): boolean
+ getSwitchTurn(): boolean
+ resetSwitchTurn()
+ getSelectedPos(move: Move): int[]
+ generatePlayerBoard(): Boolean[][]
+ generatePreviewMove(move: Move): boolean[][]
+ generateMills(): List<int[][]>
is played on
game
<<abstract>>
MorrisGame
- gameBoard: MorrisBoard
- players: Player[]
- currentPlayerTurn: Player
- subscribers: List<IMorrisGameSubscriber>
+ MorrisGame(p1: Player, p2: Player, subscriber: IMorrisGameSubscriber)
+ addSubscriber(subscriber: IMorrisGameSubscriber)
+ updateSubscribers()
# declareWinner(isRed: boolean)
# declareDraw()
# validatePlayerMove()
# getGameBoard(): MorrisBoard
# getPlayers(): Player[]
# getCurrentPlayerTurn(): Player
# switchPlayerTurn()
LocalGame
+ LocalGame(p1: Player, p2:Player, subscriber: IMorrisGameSubscriber)
+ handleInput(row: int, col: int)
updates
<<interface>>
IMorrisGameSubscriber
+ update(isRed: boolean, redToken: int, blueToken: int, board: Boolean[][], interactables: boolean[][], mills: List<int[][]>, move: MoveType, selectedPos: int[])
+ updateGameEnd(isRed: boolean)
+ updateGameDraw()
<<interface>>
IMorrisGameInputHandler
+ handleInput(row: int, col: int)
MorrisGameFactory
- MorrisGameFactory()
+ createGame(p1: PlayerType, p2: PlayerType, subscriber: IMorrisGameSubscriber): MorrisGame
move
RemoveToken
- previousMove: Move
+ RemoveToken(isRedMove: boolean, previousMove: Move)
- checkAbNoneAvailable(positions: Position[][]): boolean
+ performMove(pos: Position, board: Position[][]): Move
+ previewMove(positions: Position[][]): boolean[][]
+ validateCurrentMove(positions: Position[][]): Move
+ getMoveType(): MoveType
+ getWinPlayer(positions: Position[][]): Boolean
FlyingMove
- selectedPos: Position
+ FlyingMove(isRedMove: boolean)
+ performMove(pos: Position, board: Position[][]): Move
+ previewMove(positions: Position[][]): boolean[][]
+ validateCurrentMove(positions: Position[][]): Move
+ getMoveType(): MoveType
+ getSelectedPos(): int[]
AdjacentMove
- selectedPos: Position
+ AdjacentMove(isRedMove: boolean)
+ performMove(pos: Position, board: Position[][]): Move
- checkAbMovable(pos: Position): boolean
- isNeighbour(neighbours: Position[], current: Position): boolean
+ previewMove(positions: Position[][]): boolean[][]
+ validateCurrentMove(positions: Position[][]): Move
+ getMoveType(): MoveType
+ getWinPlayer(positions: Position[][]): Boolean
+ getSelectedPos(): int[]
PlaceToken
- tokens: List<Token>
+ PlaceToken(isRedMove: boolean, tokens: List<Token>)
+ performMove(pos: Position, board: Position[][]): Move
+ previewMove(positions: Position[][]): boolean[][]
+ validateCurrentMove(positions: Position[][]): Move
+ getMoveType(): MoveType
+ getWinPlayer(positions: Position[][]): Boolean
<<abstract>>
Move
- row: int
- col: int
- switchTurn: boolean
- isRedMove: boolean
- isDraw: boolean
+ Move(isRedMove: boolean)
+ performMove(pos: Position, board: Position[][]): Move
+ previewMove(positions: Position[][]): boolean[][]
+ validateCurrentMove(positions: Position[][]): Move
+ getMoveType(): MoveType
+ getWinPlayer(positions: Position[][]): Boolean
+ getSelectedPos(): int[]
+ getMovePosition(): int[]
+ setMovePosition(row: int, col: int)
+ getSwitchTurn(): boolean
+ resetSwitchTurn()
# enableSwitchTurn()
# getIsRedMove(): boolean
# getIsDraw(): boolean
# enableDraw()
has, in a turn,
shared
<<enum>>
MoveType
PLACE_PHASE
MOVE_PHASE
FLY_PHASE
REMOVE_PHASE
SELECT_PHASE