Project Title:
31 - Settlers of Catan

Team Members:
Chris Hitte
Brandon Boylan-Peck
Gene Zhang

Part 5 - Final Report

I. Implemented Features

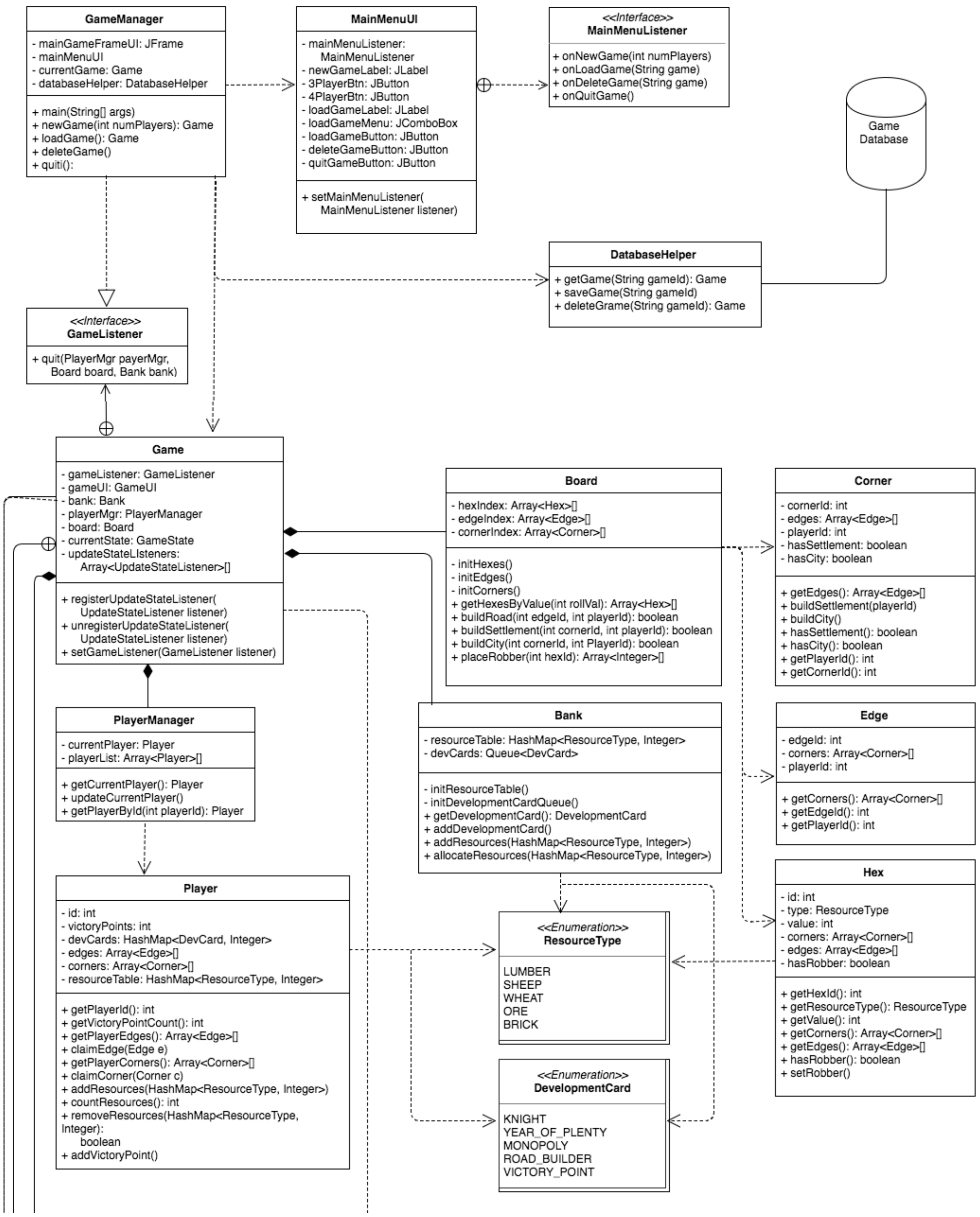
Use Case ID	Use Case Title
UC-01	Start New Game
UC-02	Setup Initial Pieces
UC-04	Start Turn (& Distribute Resources)
UC-09	Purchase Settlement
UC-10	Purchase Road
UC-11	Purchase City
UC-13	Place Road
UC-14	Place City
UC-15	Place Settlement

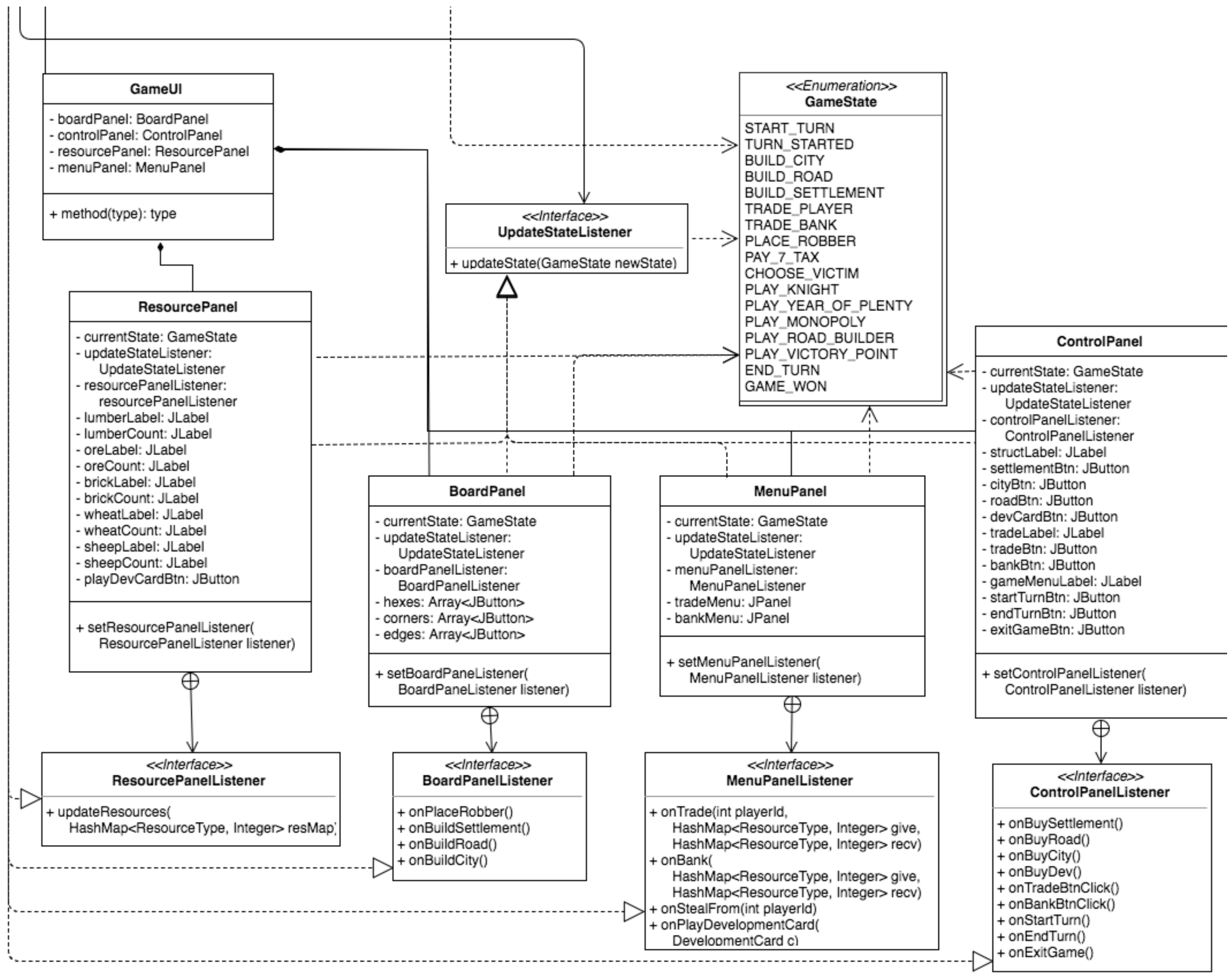
II. Features Not Yet Implemented

Use Case ID	Use Case Title
UC-03	Load Game
UC-05	Trade Resources w/ Players
UC-06	Trade Resources w/ Bank
UC-07	Make Offer
UC-08	Finish Trade
UC-12	Purchase Development Card
UC-16	Play 'Knight' Card
UC-17	Play 'Road Builder' Card
UC-18	Play 'Monopoly' Card
UC-19	Play 'Year of Plenty' Card
UC-20	Play 'Victory Point' Card

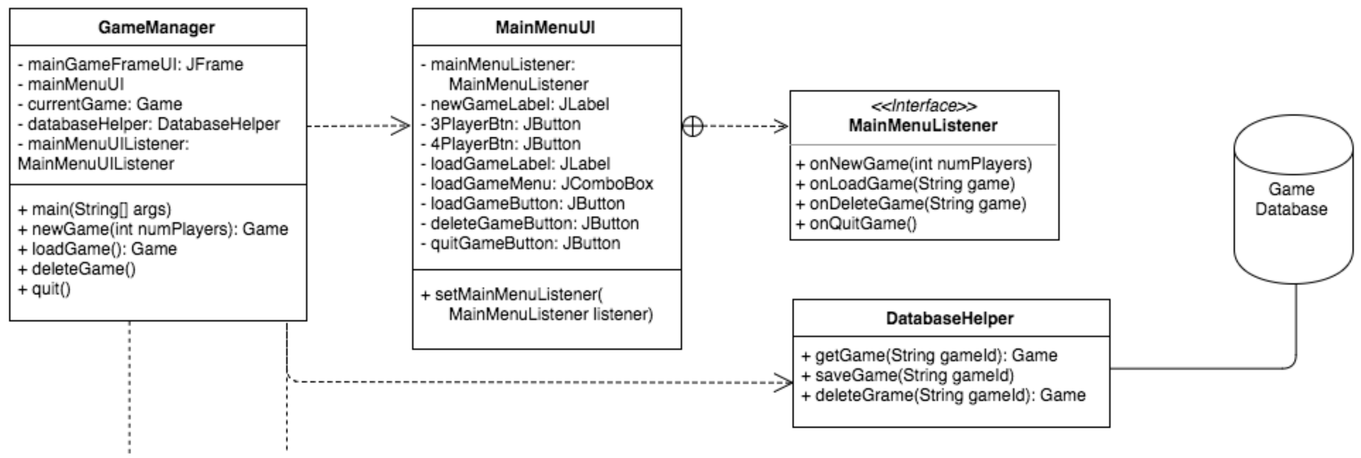
III. Class Diagrams

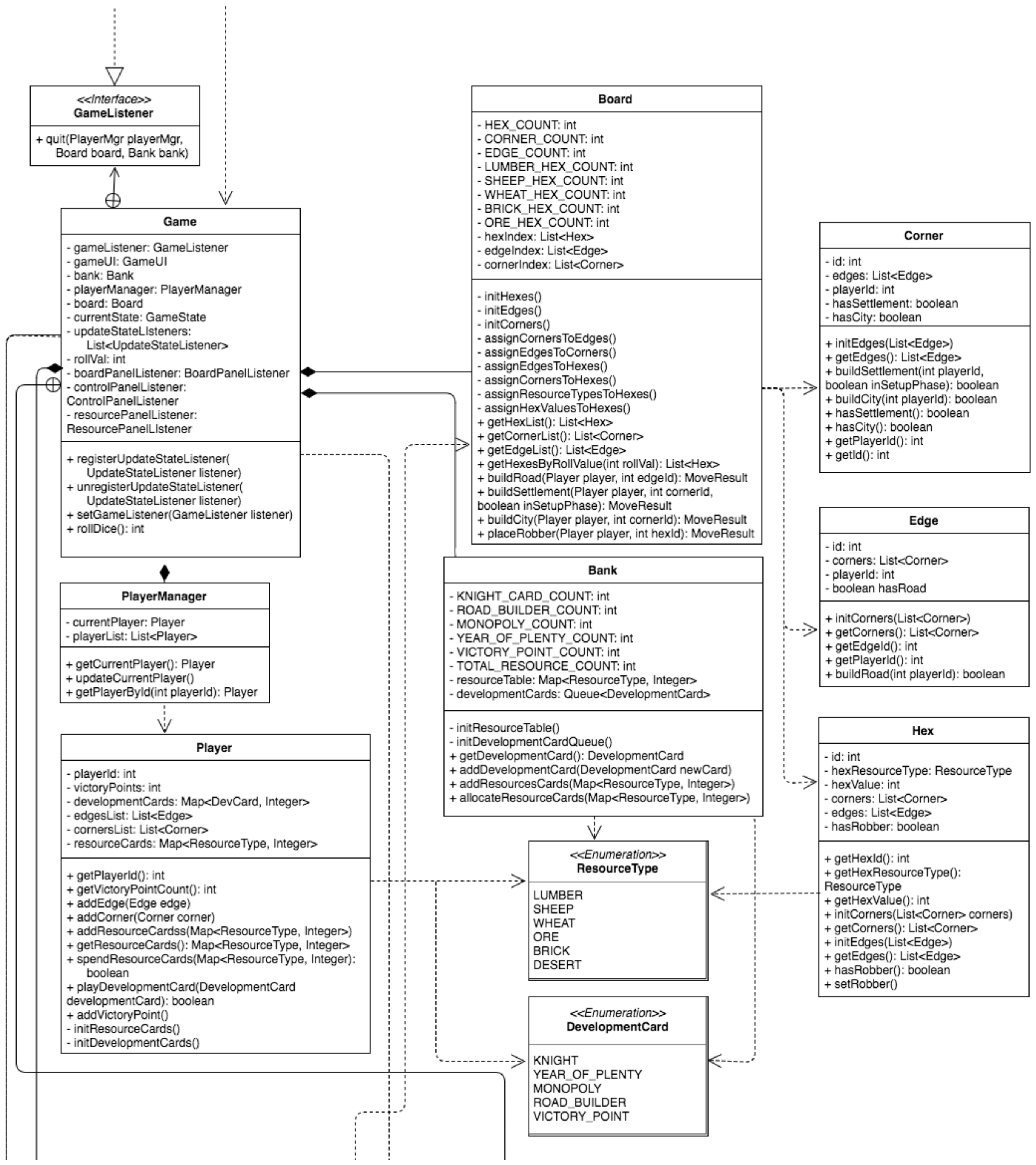
Part 2 Class Diagram


GameManager
- mainGameFrameUI: JFrame
- mainMenuUI
- currentGame: Game
- databaseHelper: DatabaseHelper
+ main(String[] args)
+ newGame(int numPlayers): Game
+ loadGame(): Game
+ deleteGame()
+ quiti():
MainMenuUI
- mainMenuListener: MainMenuListener
- newGameLabel: JLabel
- 3PlayerBtn: JButton
- 4PlayerBtn: JButton
- loadGameLabel: JLabel
- loadGameMenu: JComboBox
- loadGameButton: JButton
- deleteGameButton: JButton
- quitGameButton: JButton
+ setMainMenuListener(MainMenuListener listener)
<<Interface>>
MainMenuListener
+ onNewGame(int numPlayers)
+ onLoadGame(String game)
+ onDeleteGame(String game)
+ onQuitGame()
Game Database
DatabaseHelper
+ getGame(String gameId): Game
+ saveGame(String gameId)
+ deleteGrame(String gameId): Game
<<Interface>>
GameListener
+ quit(PlayerMgr payerMgr, Board board, Bank bank)
Game
- gameListener: GameListener
- gameUI: GameUI
- bank: Bank
- playerMgr: PlayerManager
- board: Board
- currentState: GameState
- updateStateLIsteners: Array<UpdateStateListener>[]
+ registerUpdateStateListener(UpdateStateListener listener)
+ unregisterUpdateStateListener(UpdateStateListener listener)
+ setGameListener(GameListener listener)
Board
- hexIndex: Array<Hex>[]
- edgeIndex: Array<Edge>[]
- cornerIndex: Array<Corner>[]
- initHexes()
- initEdges()
- initCorners()
+ getHexesByValue(int rollVal): Array<Hex>[]
+ buildRoad(int edgeId, int playerId): boolean
+ buildSettlement(int cornerId, int playerId): boolean
+ buildCity(int cornerId, int PlayerId): boolean
+ placeRobber(int hexId): Array<Integer>[]
Corner
- cornerId: int
- edges: Array<Edge>[]
- playerId: int
- hasSettlement: boolean
- hasCity: boolean
+ getEdges(): Array<Edge>[]
+ buildSettlement(playerId)
+ buildCity()
+ hasSettlement(): boolean
+ hasCity(): boolean
+ getPlayerId(): int
+ getCornerId(): int
PlayerManager
- currentPlayer: Player
- playerList: Array<Player>[]
+ getCurrentPlayer(): Player
+ updateCurrentPlayer()
+ getPlayerById(int playerId): Player
Bank
- resourceTable: HashMap<ResourceType, Integer>
- devCards: Queue<DevCard>
- initResourceTable()
- initDevelopmentCardQueue()
+ getDevelopmentCard(): DevelopmentCard
+ addDevelopmentCard()
+ addResources(HashMap<ResourceType, Integer>)
+ allocateResources(HashMap<ResourceType, Integer>)
Edge
- edgeId: int
- corners: Array<Corner>[]
- playerId: int
+ getCorners(): Array<Corner>[]
+ getEdgeId(): int
+ getPlayerId(): int
Hex
- id: int
- type: ResourceType
- value: int
- corners: Array<Corner>[]
- edges: Array<Edge>[]
- hasRobber: boolean
+ getHexId(): int
+ getResourceType(): ResourceType
+ getValue(): int
+ getCorners(): Array<Corner>[]
+ getEdges(): Array<Edge>[]
+ hasRobber(): boolean
+ setRobber()
Player
- id: int
- victoryPoints: int
- devCards: HashMap<DevCard, Integer>
- edges: Array<Edge>[]
- corners: Array<Corner>[]
- resourceTable: HashMap<ResourceType, Integer>
+ getPlayerId(): int
+ getVictoryPointCount(): int
+ getPlayerEdges(): Array<Edge>[]
+ claimEdge(Edge e)
+ getPlayerCorners(): Array<Corner>[]
+ claimCorner(Corner c)
+ addResources(HashMap<ResourceType, Integer>)
+ countResources(): int
+ removeResources(HashMap<ResourceType, Integer): boolean
+ addVictoryPoint()
<<Enumeration>>
ResourceType
LUMBER
SHEEP
WHEAT
ORE
BRICK
<<Enumeration>>
DevelopmentCard
KNIGHT
YEAR_OF_PLENTY
MONOPOLY
ROAD_BUILDER
VICTORY_POINT

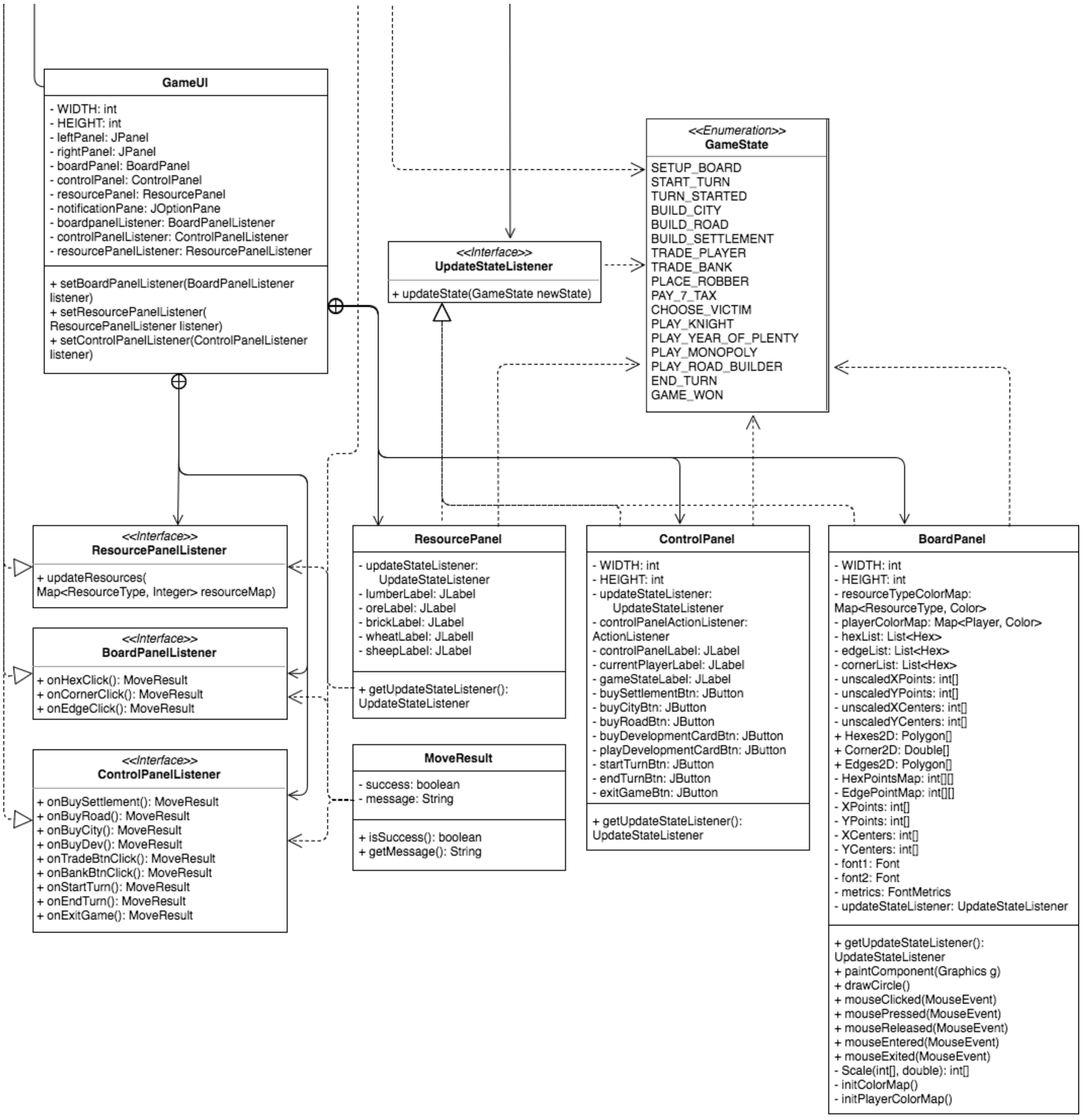


Final Class Diagram


GameManager
- mainGameFrameUI: JFrame
- mainMenuUI
- currentGame: Game
- databaseHelper: DatabaseHelper
- mainMenuUIListener: MainMenuUIListener
+ main(String[] args)
+ newGame(int numPlayers): Game
+ loadGame(): Game
+ deleteGame()
+ quit()
MainMenuUI
- mainMenuListener: MainMenuListener
- newGameLabel: JLabel
- 3PlayerBtn: JButton
- 4PlayerBtn: JButton
- loadGameLabel: JLabel
- loadGameMenu: JComboBox
- loadGameButton: JButton
- deleteGameButton: JButton
- quitGameButton: JButton
+ setMainMenuListener(MainMenuListener listener)
<<Interface>>
MainMenuListener
+ onNewGame(int numPlayers)
+ onLoadGame(String game)
+ onDeleteGame(String game)
+ onQuitGame()
Game Database
DatabaseHelper
+ getGame(String gameId): Game
+ saveGame(String gameId)
+ deleteGrame(String gameId): Game



<<Interface>>
GameListener
+ quit(PlayerMgr playerMgr, Board board, Bank bank)
Game
- gameListener: GameListener
- gameUI: GameUI
- bank: Bank
- playerManager: PlayerManager
- board: Board
- currentState: GameState
- updateStateLIsteners: List<UpdateStateListener>
- rollVal: int
- boardPanelListener: BoardPanelListener
- controlPanelListener: ControlPanelListener
- resourcePanelListener: ResourcePanelLIstener
+ registerUpdateStateListener(UpdateStateListener listener)
+ unregisterUpdateStateListener(UpdateStateListener listener)
+ setGameListener(GameListener listener)
+ rollDice(): int
Board
- HEX_COUNT: int
- CORNER_COUNT: int
- EDGE_COUNT: int
- LUMBER_HEX_COUNT: int
- SHEEP_HEX_COUNT: int
- WHEAT_HEX_COUNT: int
- BRICK_HEX_COUNT: int
- ORE_HEX_COUNT: int
- hexIndex: List<Hex>
- edgeIndex: List<Edge>
- cornerIndex: List<Corner>
- initHexes()
- initEdges()
- initCorners()
- assignCornersToEdges()
- assignEdgesToCorners()
- assignEdgesToHexes()
- assignCornersToHexes()
- assignResourceTypesToHexes()
- assignHexValuesToHexes()
+ getHexList(): List<Hex>
+ getCornerList(): List<Corner>
+ getEdgeList(): List<Edge>
+ getHexesByRollValue(int rollVal): List<Hex>
+ buildRoad(Player player, int edgeId): MoveResult
+ buildSettlement(Player player, int cornerId, boolean inSetupPhase): MoveResult
+ buildCity(Player player, int cornerId): MoveResult
+ placeRobber(Player player, int hexId): MoveResult
Corner
- id: int
- edges: List<Edge>
- playerId: int
- hasSettlement: boolean
- hasCity: boolean
+ initEdges(List<Edge>)
+ getEdges(): List<Edge>
+ buildSettlement(int playerId, boolean inSetupPhase): boolean
+ buildCity(int playerId): boolean
+ hasSettlement(): boolean
+ hasCity(): boolean
+ getPlayerId(): int
+ getId(): int
Edge
- id: int
- corners: List<Corner>
- playerId: int
- boolean hasRoad
+ initCorners(List<Corner>)
+ getCorners(): List<Corner>
+ getEdgeId(): int
+ getPlayerId(): int
+ buildRoad(int playerId): boolean
Hex
- id: int
- hexResourceType: ResourceType
- hexValue: int
- corners: List<Corner>
- edges: List<Edge>
- hasRobber: boolean
+ getHexId(): int
+ getHexResourceType(): ResourceType
+ getHexValue(): int
+ initCorners(List<Corner> corners)
+ getCorners(): List<Corner>
+ initEdges(List<Edge>)
+ getEdges(): List<Edge>
+ hasRobber(): boolean
+ setRobber()
PlayerManager
- currentPlayer: Player
- playerList: List<Player>
+ getCurrentPlayer(): Player
+ updateCurrentPlayer()
+ getPlayerById(int playerId): Player
Bank
- KNIGHT_CARD_COUNT: int
- ROAD_BUILDER_COUNT: int
- MONOPOLY_COUNT: int
- YEAR_OF_PLENTY_COUNT: int
- VICTORY_POINT_COUNT: int
- TOTAL_RESOURCE_COUNT: int
- resourceTable: Map<ResourceType, Integer>
- developmentCards: Queue<DevelopmentCard>
- initResourceTable()
- initDevelopmentCardQueue()
+ getDevelopmentCard(): DevelopmentCard
+ addDevelopmentCard(DevelopmentCard newCard)
+ addResourcesCards(Map<ResourceType, Integer>)
+ allocateResourceCards(Map<ResourceType, Integer>)
Player
- playerId: int
- victoryPoints: int
- developmentCards: Map<DevCard, Integer>
- edgesList: List<Edge>
- cornersList: List<Corner>
- resourceCards: Map<ResourceType, Integer>
+ getPlayerId(): int
+ getVictoryPointCount(): int
+ addEdge(Edge edge)
+ addCorner(Corner corner)
+ addResourceCardss(Map<ResourceType, Integer>)
+ getResourceCards(): Map<ResourceType, Integer>
+ spendResourceCards(Map<ResourceType, Integer): boolean
+ playDevelopmentCard(DevelopmentCard developmentCard): boolean
+ addVictoryPoint()
- initResourceCards()
- initDevelopmentCards()
<<Enumeration>>
ResourceType
LUMBER
SHEEP
WHEAT
ORE
BRICK
DESERT
<<Enumeration>>
DevelopmentCard
KNIGHT
YEAR_OF_PLENTY
MONOPOLY
ROAD_BUILDER
VICTORY_POINT


GameUI
- WIDTH: int
- HEIGHT: int
- leftPanel: JPanel
- rightPanel: JPanel
- boardPanel: BoardPanel
- controlPanel: ControlPanel
- resourcePanel: ResourcePanel
- notificationPane: JOptionPane
- boardpanelListener: BoardPanelListener
- controlPanelListener: ControlPanelListener
- resourcePanelListener: ResourcePanelListener
+ setBoardPanelListener(BoardPanelListener listener)
+ setResourcePanelListener(ResourcePanelListener listener)
+ setControlPanelListener(ControlPanelListener listener)
<<Enumeration>>
GameState
SETUP_BOARD
START_TURN
TURN_STARTED
BUILD_CITY
BUILD_ROAD
BUILD_SETTLEMENT
TRADE_PLAYER
TRADE_BANK
PLACE_ROBBER
PAY_7_TAX
CHOOSE_VICTIM
PLAY_KNIGHT
PLAY_YEAR_OF_PLENTY
PLAY_MONOPOLY
PLAY_ROAD_BUILDER
END_TURN
GAME_WON
<<Interface>>
UpdateStateListener
+ updateState(GameState newState)
<<Interface>>
ResourcePanelListener
+ updateResources(Map<ResourceType, Integer> resourceMap)
<<Interface>>
BoardPanelListener
+ onHexClick(): MoveResult
+ onCornerClick(): MoveResult
+ onEdgeClick(): MoveResult
<<Interface>>
ControlPanelListener
+ onBuySettlement(): MoveResult
+ onBuyRoad(): MoveResult
+ onBuyCity(): MoveResult
+ onBuyDev(): MoveResult
+ onTradeBtnClick(): MoveResult
+ onBankBtnClick(): MoveResult
+ onStartTurn(): MoveResult
+ onEndTurn(): MoveResult
+ onExitGame(): MoveResult
ResourcePanel
- updateStateListener: UpdateStateListener
- lumberLabel: JLabel
- oreLabel: JLabel
- brickLabel: JLabel
- wheatLabel: JLabell
- sheepLabel: JLabel
+ getUpdateStateListener(): UpdateStateListener
MoveResult
- success: boolean
- message: String
+ isSuccess(): boolean
+ getMessage(): String
ControlPanel
- WIDTH: int
- HEIGHT: int
- updateStateListener: UpdateStateListener
- controlPanelActionListener: ActionListener
- controlPanelLabel: JLabel
- currentPlayerLabel: JLabel
- gameStateLabel: JLabel
- buySettlementBtn: JButton
- buyCityBtn: JButton
- buyRoadBtn: JButton
- buyDevelopmentCardBtn: JButton
- playDevelopmentCardBtn: JButton
- startTurnBtn: JButton
- endTurnBtn: JButton
- exitGameBtn: JButton
+ getUpdateStateListener(): UpdateStateListener
BoardPanel
- WIDTH: int
- HEIGHT: int
- resourceTypeColorMap: Map<ResourceType, Color>
- playerColorMap: Map<Player, Color>
- hexList: List<Hex>
- edgeList: List<Hex>
- cornerList: List<Hex>
- unscaledXPoints: int[]
- unscaledYPoints: int[]
- unscaledXCenters: int[]
- unscaledYCenters: int[]
+ Hexes2D: Polygon[]
+ Corner2D: Double[]
+ Edges2D: Polygon[]
- HexPointsMap: int[][]
- EdgePointMap: int[][]
- XPoints: int[]
- YPoints: int[]
- XCenters: int[]
- YCenters: int[]
- font1: Font
- font2: Font
- metrics: FontMetrics
- updateStateListener: UpdateStateListener
+ getUpdateStateListener(): UpdateStateListener
+ paintComponent(Graphics g)
+ drawCircle()
+ mouseClicked(MouseEvent)
+ mousePressed(MouseEvent)
+ mouseReleased(MouseEvent)
+ mouseEntered(MouseEvent)
+ mouseExited(MouseEvent)
- Scale(int[], double): int[]
- initColorMap()
- initPlayerColorMap()

As you can see from the two Class Diagrams, there were not significant changes between Part 2 and the Final implementation. The biggest difference between the two diagrams is due to our group's lack of familiarity with Swing during Part 2; the UI components of our application were greatly simplified.

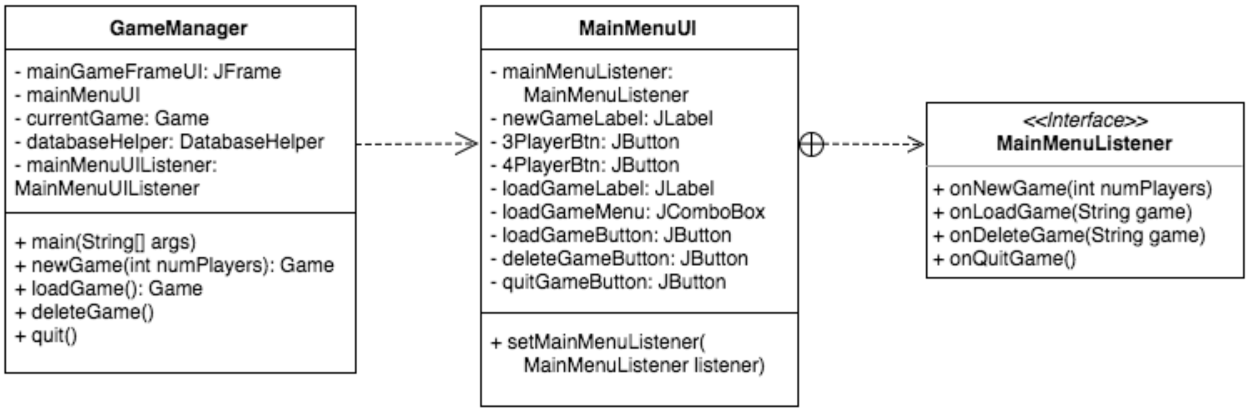
We credit our design's consistence entirely to the planning process in Part 2. All of us have had the experience of designing a project on the fly and all of the problems that come with it. However, by identifying the requirements, organizing them into use cases, and building a Class diagram that satisfied those use cases made implementing our overall project very easy. The early adaptation of the Observer design pattern also helped us out substantially. We will discuss this in more detail in the following section.

IV. Design Patterns

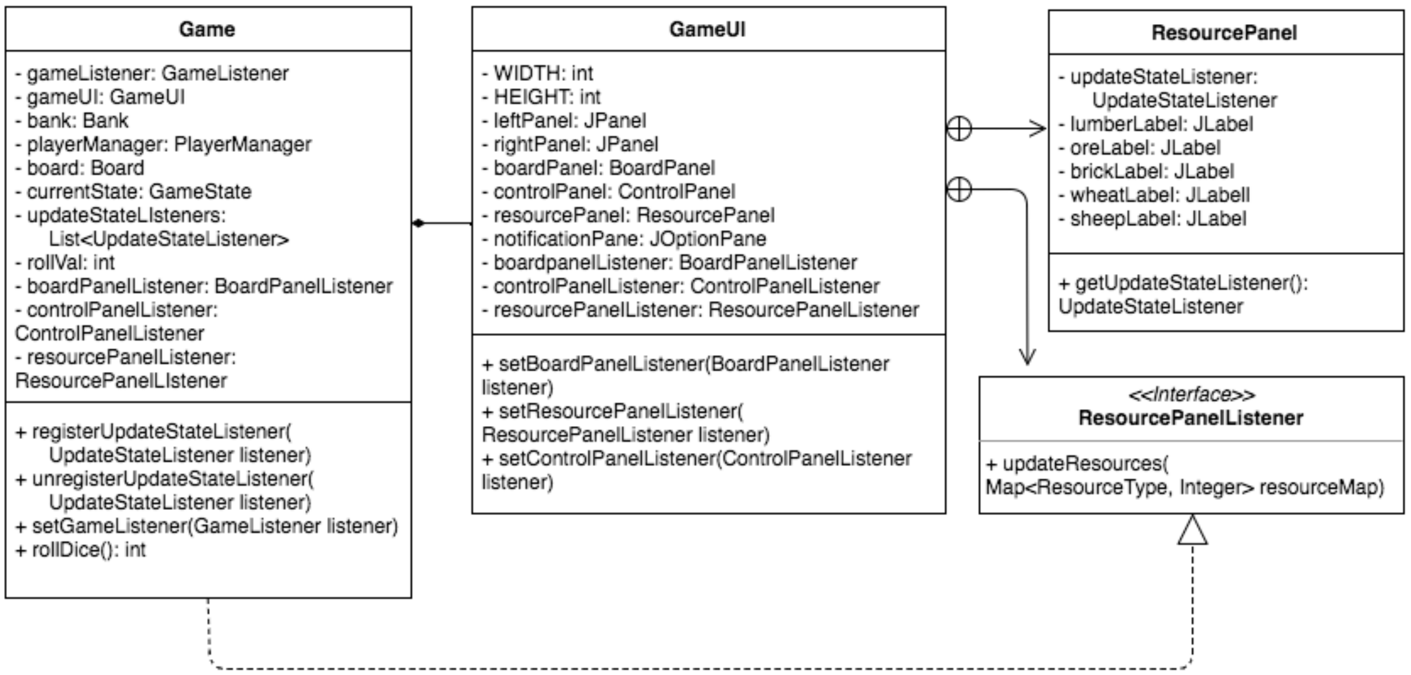
Our project made extensive use of the Observer design pattern. From the earliest stages of our planning, we knew that we our application would have to handle many distinct UI events. These included clicks on Corner, Edge, and Hex BoardPanel components, as well as buttons on the ControlPanel, and more. The Observer pattern seemed appropriate because of the that it can decouple a controller class from associated view classes.

As of our final submission, our application utilizes the Observer pattern in five places:

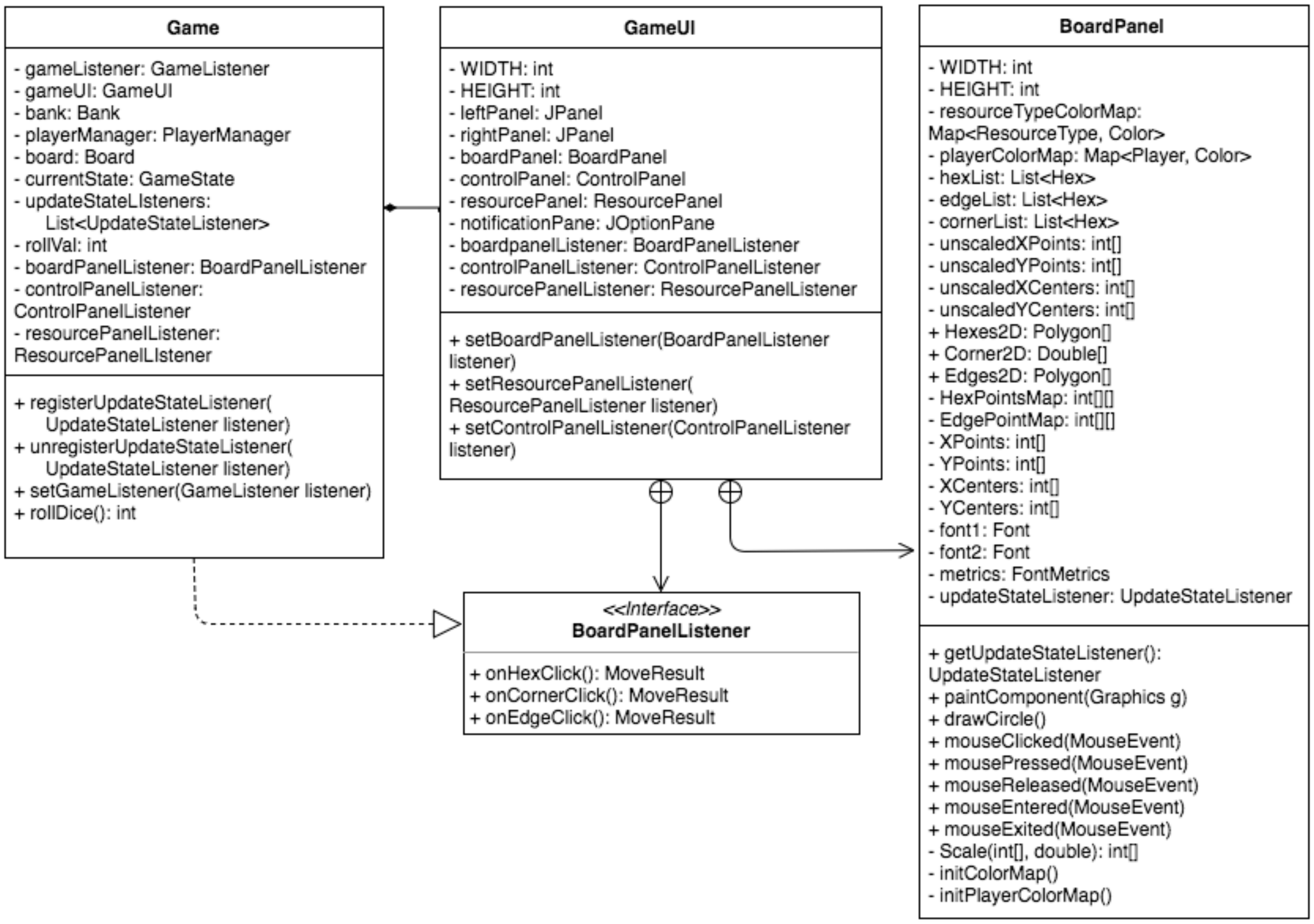
MainMenuUIListener


GameManager
- mainGameFrameUI: JFrame
- mainMenuUI
- currentGame: Game
- databaseHelper: DatabaseHelper
- mainMenuUIListener: MainMenuUIListener
+ main(String[] args)
+ newGame(int numPlayers): Game
+ loadGame(): Game
+ deleteGame()
+ quit()
MainMenuUI
- mainMenuListener: MainMenuListener
- newGameLabel: JLabel
- 3PlayerBtn: JButton
- 4PlayerBtn: JButton
- loadGameLabel: JLabel
- loadGameMenu: JComboBox
- loadGameButton: JButton
- deleteGameButton: JButton
- quitGameButton: JButton
+ setMainMenuListener(MainMenuListener listener)
<<Interface>>
MainMenuListener
+ onNewGame(int numPlayers)
+ onLoadGame(String game)
+ onDeleteGame(String game)
+ onQuitGame()

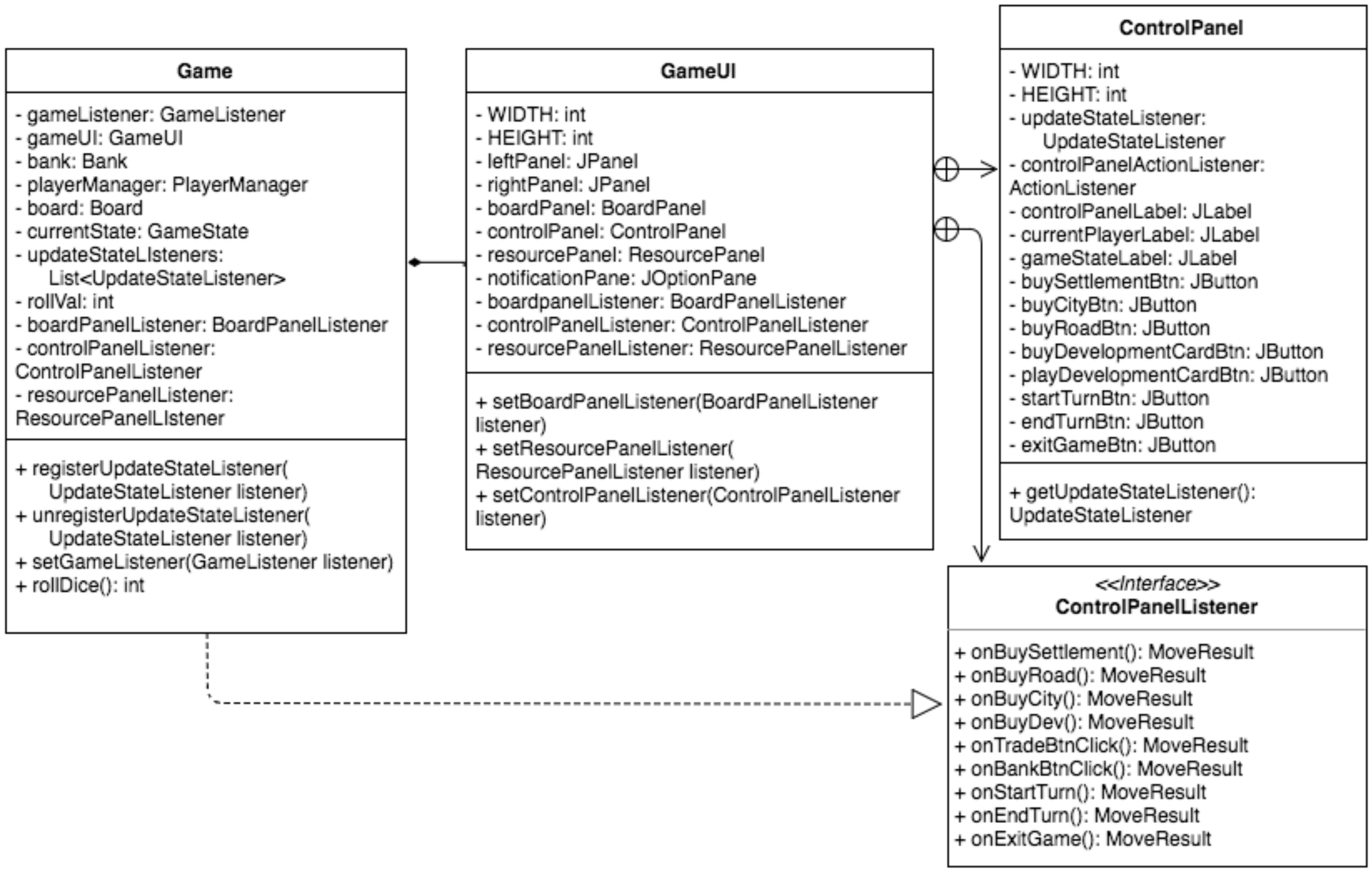
ResourcePanelListener


Game
- gameListener: GameListener
- gameUI: GameUI
- bank: Bank
- playerManager: PlayerManager
- board: Board
- currentState: GameState
- updateStateLIsteners: List<UpdateStateListener>
- rollVal: int
- boardPanelListener: BoardPanelListener
- controlPanelListener: ControlPanelListener
- resourcePanelListener: ResourcePanelLIstener
+ registerUpdateStateListener(UpdateStateListener listener)
+ unregisterUpdateStateListener(UpdateStateListener listener)
+ setGameListener(GameListener listener)
+ rollDice(): int
GameUI
- WIDTH: int
- HEIGHT: int
- leftPanel: JPanel
- rightPanel: JPanel
- boardPanel: BoardPanel
- controlPanel: ControlPanel
- resourcePanel: ResourcePanel
- notificationPane: JOptionPane
- boardpanelListener: BoardPanelListener
- controlPanelListener: ControlPanelListener
- resourcePanelListener: ResourcePanelListener
+ setBoardPanelListener(BoardPanelListener listener)
+ setResourcePanelListener(ResourcePanelListener listener)
+ setControlPanelListener(ControlPanelListener listener)
ResourcePanel
- updateStateListener: UpdateStateListener
- lumberLabel: JLabel
- oreLabel: JLabel
- brickLabel: JLabel
- wheatLabel: JLabell
- sheepLabel: JLabel
+ getUpdateStateListener(): UpdateStateListener
<<Interface>>
ResourcePanelListener
+ updateResources(Map<ResourceType, Integer> resourceMap)

BoardPanelListener


Game
- gameListener: GameListener
- gameUI: GameUI
- bank: Bank
- playerManager: PlayerManager
- board: Board
- currentState: GameState
- updateStateLIsteners: List<UpdateStateListener>
- rollVal: int
- boardPanelListener: BoardPanelListener
- controlPanelListener: ControlPanelListener
- resourcePanelListener: ResourcePanelLIstener
+ registerUpdateStateListener(UpdateStateListener listener)
+ unregisterUpdateStateListener(UpdateStateListener listener)
+ setGameListener(GameListener listener)
+ rollDice(): int
GameUI
- WIDTH: int
- HEIGHT: int
- leftPanel: JPanel
- rightPanel: JPanel
- boardPanel: BoardPanel
- controlPanel: ControlPanel
- resourcePanel: ResourcePanel
- notificationPane: JOptionPane
- boardpanelListener: BoardPanelListener
- controlPanelListener: ControlPanelListener
- resourcePanelListener: ResourcePanelListener
+ setBoardPanelListener(BoardPanelListener listener)
+ setResourcePanelListener(ResourcePanelListener listener)
+ setControlPanelListener(ControlPanelListener listener)
<<Interface>>
BoardPanelListener
+ onHexClick(): MoveResult
+ onCornerClick(): MoveResult
+ onEdgeClick(): MoveResult
BoardPanel
- WIDTH: int
- HEIGHT: int
- resourceTypeColorMap: Map<ResourceType, Color>
- playerColorMap: Map<Player, Color>
- hexList: List<Hex>
- edgeList: List<Hex>
- cornerList: List<Hex>
- unscaledXPoints: int[]
- unscaledYPoints: int[]
- unscaledXCenters: int[]
- unscaledYCenters: int[]
+ Hexes2D: Polygon[]
+ Corner2D: Double[]
+ Edges2D: Polygon[]
- HexPointsMap: int[][]
- EdgePointMap: int[][]
- XPoints: int[]
- YPoints: int[]
- XCenters: int[]
- YCenters: int[]
- font1: Font
- font2: Font
- metrics: FontMetrics
- updateStateListener: UpdateStateListener
+ getUpdateStateListener(): UpdateStateListener
+ paintComponent(Graphics g)
+ drawCircle()
+ mouseClicked(MouseEvent)
+ mousePressed(MouseEvent)
+ mouseReleased(MouseEvent)
+ mouseEntered(MouseEvent)
+ mouseExited(MouseEvent)
- Scale(int[], double): int[]
- initColorMap()
- initPlayerColorMap()

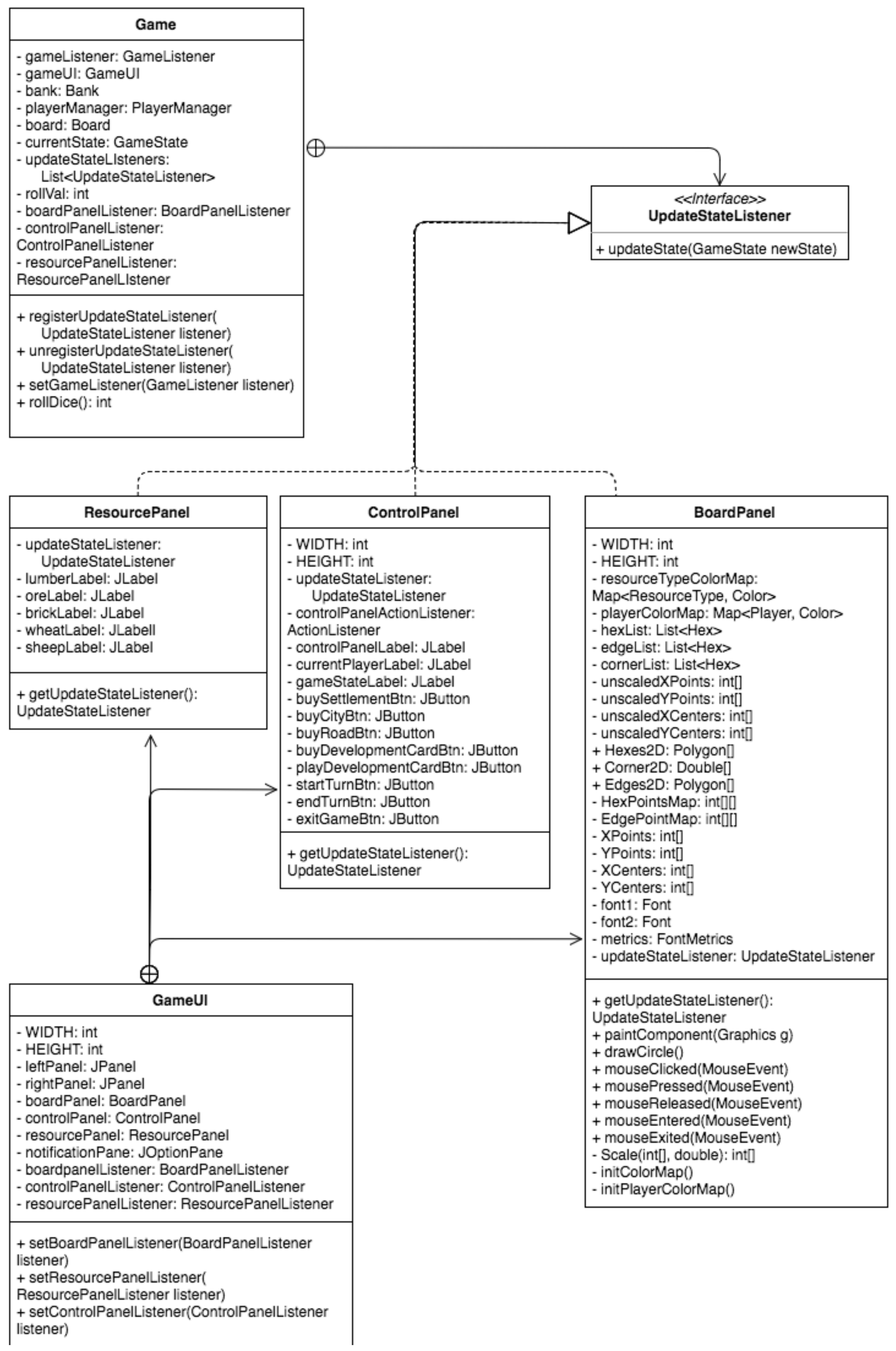
ControlPanelListener


Game
- gameListener: GameListener
- gameUI: GameUI
- bank: Bank
- playerManager: PlayerManager
- board: Board
- currentState: GameState
- updateStateLIsteners: List<UpdateStateListener>
- rollVal: int
- boardPanelListener: BoardPanelListener
- controlPanelListener: ControlPanelListener
- resourcePanelListener: ResourcePanelLIstener
+ registerUpdateStateListener(UpdateStateListener listener)
+ unregisterUpdateStateListener(UpdateStateListener listener)
+ setGameListener(GameListener listener)
+ rollDice(): int
GameUI
- WIDTH: int
- HEIGHT: int
- leftPanel: JPanel
- rightPanel: JPanel
- boardPanel: BoardPanel
- controlPanel: ControlPanel
- resourcePanel: ResourcePanel
- notificationPane: JOptionPane
- boardpanelListener: BoardPanelListener
- controlPanelListener: ControlPanelListener
- resourcePanelListener: ResourcePanelListener
+ setBoardPanelListener(BoardPanelListener listener)
+ setResourcePanelListener(ResourcePanelListener listener)
+ setControlPanelListener(ControlPanelListener listener)
ControlPanel
- WIDTH: int
- HEIGHT: int
- updateStateListener: UpdateStateListener
- controlPanelActionListener: ActionListener
- controlPanelLabel: JLabel
- currentPlayerLabel: JLabel
- gameStateLabel: JLabel
- buySettlementBtn: JButton
- buyCityBtn: JButton
- buyRoadBtn: JButton
- buyDevelopmentCardBtn: JButton
- playDevelopmentCardBtn: JButton
- startTurnBtn: JButton
- endTurnBtn: JButton
- exitGameBtn: JButton
+ getUpdateStateListener(): UpdateStateListener
<<Interface>>
ControlPanelListener
+ onBuySettlement(): MoveResult
+ onBuyRoad(): MoveResult
+ onBuyCity(): MoveResult
+ onBuyDev(): MoveResult
+ onTradeBtnClick(): MoveResult
+ onBankBtnClick(): MoveResult
+ onStartTurn(): MoveResult
+ onEndTurn(): MoveResult
+ onExitGame(): MoveResult

The first four diagrams follow a standard recipe of the Observer pattern wherein a class publishes an interface, and a controller class sets an implementation of that interface back on the publishing class. This approach allows UI components to update the various classes that maintain necessary game state without being passed an explicit reference to said classes. The only reference to the game state classes are in the Game class itself, where the interfaces are implemented. This completely decouples the UI components from the controller and model classes.

The exception to this paradigm is the fifth instance:

UpdateStateListener


Game
- gameListener: GameListener
- gameUI: GameUI
- bank: Bank
- playerManager: PlayerManager
- board: Board
- currentState: GameState
- updateStateLIsteners: List<UpdateStateListener>
- rollVal: int
- boardPanelListener: BoardPanelListener
- controlPanelListener: ControlPanelListener
- resourcePanelListener: ResourcePanelLIstener
+ registerUpdateStateListener(UpdateStateListener listener)
+ unregisterUpdateStateListener(UpdateStateListener listener)
+ setGameListener(GameListener listener)
+ rollDice(): int
<<Interface>>
UpdateStateListener
+ updateState(GameState newState)
ResourcePanel
- updateStateListener: UpdateStateListener
- lumberLabel: JLabel
- oreLabel: JLabel
- brickLabel: JLabel
- wheatLabel: JLabell
- sheepLabel: JLabel
+ getUpdateStateListener(): UpdateStateListener
ControlPanel
- WIDTH: int
- HEIGHT: int
- updateStateListener: UpdateStateListener
- controlPanelActionListener: ActionListener
- controlPanelLabel: JLabel
- currentPlayerLabel: JLabel
- gameStateLabel: JLabel
- buySettlementBtn: JButton
- buyCityBtn: JButton
- buyRoadBtn: JButton
- buyDevelopmentCardBtn: JButton
- playDevelopmentCardBtn: JButton
- startTurnBtn: JButton
- endTurnBtn: JButton
- exitGameBtn: JButton
+ getUpdateStateListener(): UpdateStateListener
BoardPanel
- WIDTH: int
- HEIGHT: int
- resourceTypeColorMap: Map<ResourceType, Color>
- playerColorMap: Map<Player, Color>
- hexList: List<Hex>
- edgeList: List<Hex>
- cornerList: List<Hex>
- unscaledXPoints: int[]
- unscaledYPoints: int[]
- unscaledXCenters: int[]
- unscaledYCenters: int[]
+ Hexes2D: Polygon[]
+ Corner2D: Double[]
+ Edges2D: Polygon[]
- HexPointsMap: int[][]
- EdgePointMap: int[][]
- XPoints: int[]
- YPoints: int[]
- XCenters: int[]
- YCenters: int[]
- font1: Font
- font2: Font
- metrics: FontMetrics
- updateStateListener: UpdateStateListener
+ getUpdateStateListener(): UpdateStateListener
+ paintComponent(Graphics g)
+ drawCircle()
+ mouseClicked(MouseEvent)
+ mousePressed(MouseEvent)
+ mouseReleased(MouseEvent)
+ mouseEntered(MouseEvent)
+ mouseExited(MouseEvent)
- Scale(int[], double): int[]
- initColorMap()
- initPlayerColorMap()
GameUI
- WIDTH: int
- HEIGHT: int
- leftPanel: JPanel
- rightPanel: JPanel
- boardPanel: BoardPanel
- controlPanel: ControlPanel
- resourcePanel: ResourcePanel
- notificationPane: JOptionPane
- boardpanelListener: BoardPanelListener
- controlPanelListener: ControlPanelListener
- resourcePanelListener: ResourcePanelListener
+ setBoardPanelListener(BoardPanelListener listener)
+ setResourcePanelListener(ResourcePanelListener listener)
+ setControlPanelListener(ControlPanelListener listener)

In this version of the Observer pattern, the ControlPanel, BoardPanel, and ResourcePanel classes each have an implementation of the UpdateStateListener interface published by the Game class itself. When the GameUI parent class is instantiated, it passes the UpdateStateListener instance from each of its inner UI classes to the registerUpdateStateListener(…) method on the Game class. This method adds the listener to an array of such UpdateStateListener implementations. When the Game class detects that the GameState has changed as a result of some UI event, it calls the updateState(…) method on each UpdateStateListener instance in its array. This alerts the various UI components to the current GameState and allows them to set their own state accordingly.

V. Design Conclusions

In our team, there was a very diverse set of experience in Software Development. We each have very different backgrounds, both in education and professionally. And to be honest, each of us began this project with doubts about the benefits that this design process would yield. Like many developers, we were each overly confident in our abilities, while simultaneously ignoring the fact that our typical experience implementing a project was rife with errors and setbacks.

After completing this project, we have each come to appreciate the process. Never before have we had a project come together so smoothly before. The process of identifying requirements, developing use cases, and designing a class diagram that satisfied each use case handled so many issues that would have thrown off our development process later on. Instead, implementing the outlined features felt simple and straight forward. Once we began writing code, everything came together quickly and efficiently.

Additionally, the design process also helped our team develop a unified vision for the project before a single line of code had been written. This was a hidden benefit that none of us expected. Often the hardest part of completing a group project is keeping every team member working towards the same goal. In our experience, this involves many rounds of refactoring code developed with different goals to work together. The process of planning our entire project out in the beginning spared us this frustration and again helped our development process immeasurably.

In conclusion, each of us are avid converts to the design process presented in this class. The benefits speak for themselves. We all feel that this project has been one of the most rewarding experiences we have had in our CS education, and we look forward to taking the lessons learned over the course of this semester, and we plan to take this process with us into our group projects and eventually industry projects in the future.