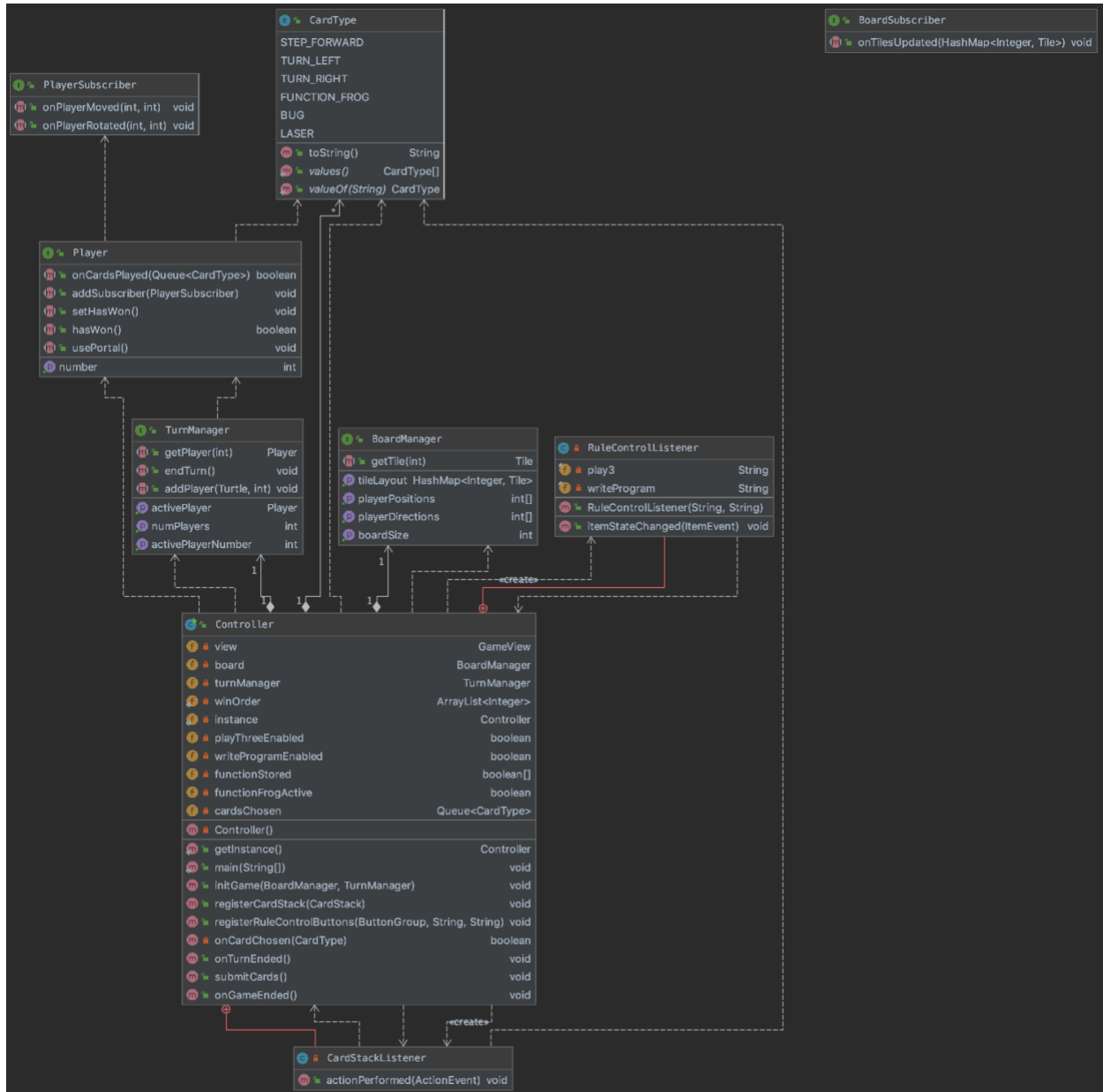# COMP 3721 - Milestone 5 Documentation
By: Sam Chambré and Eamon McCarron

The most prevalent changes are the use of the Factory and Observer design patterns. To handle all the new types of tiles, we used the factory pattern. This makes it easy to generate tiles whose type can be decided at runtime. The observer pattern was used to improve our model-view-controller setup. As such, the view subscribes to the model via the controller and the controller updates the view when state changes are made in the model.
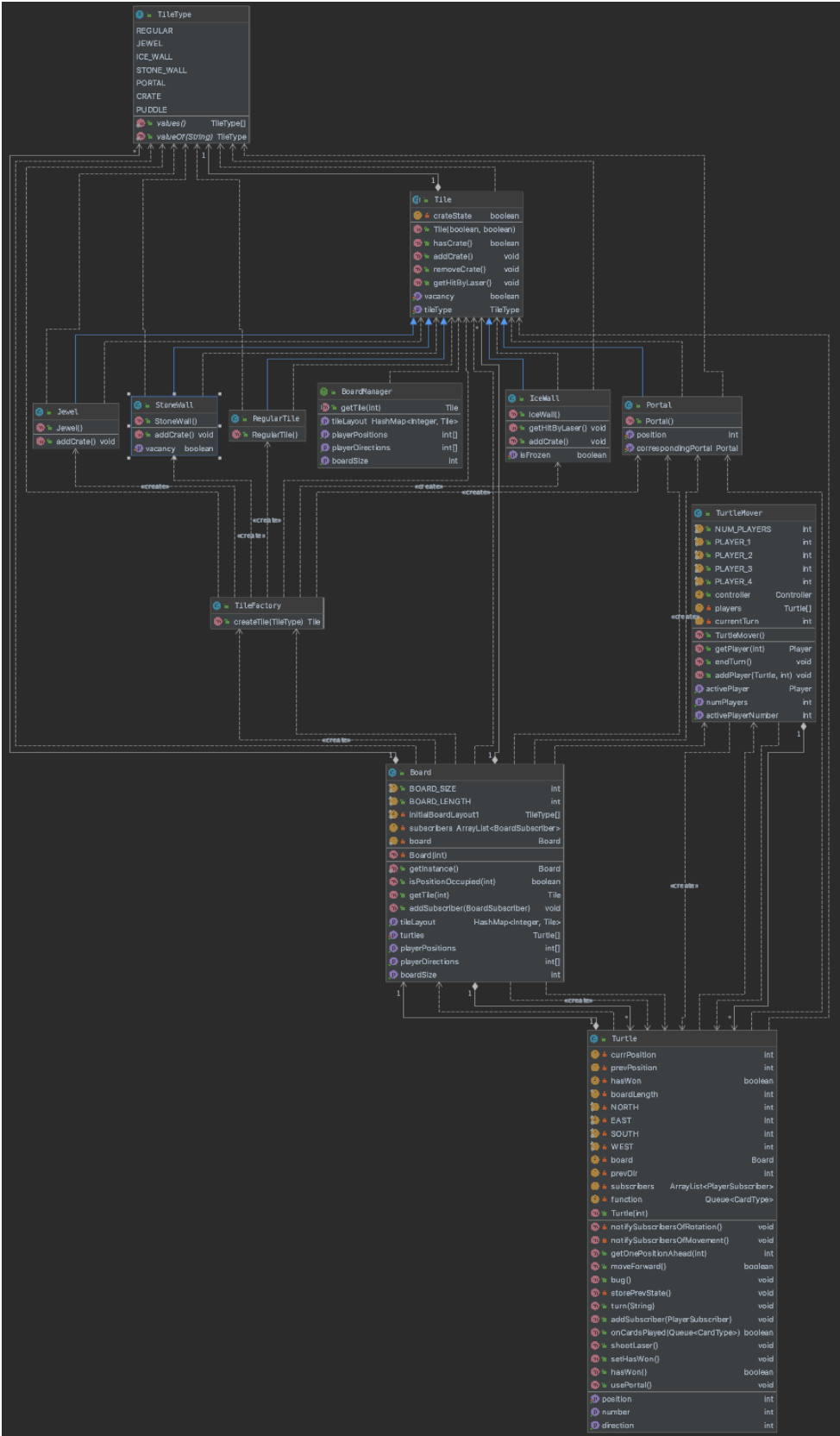
The only other design pattern used is Singleton. We made the Board and Controller be singletons to ensure that there is never a second instance of the object being used.

It should be noted that although Crate is a TileType, there is not class for Crate. We instead decided that each tile should have a boolean variable hasCrate since a crate could appear on top of a melted Ice Wall.
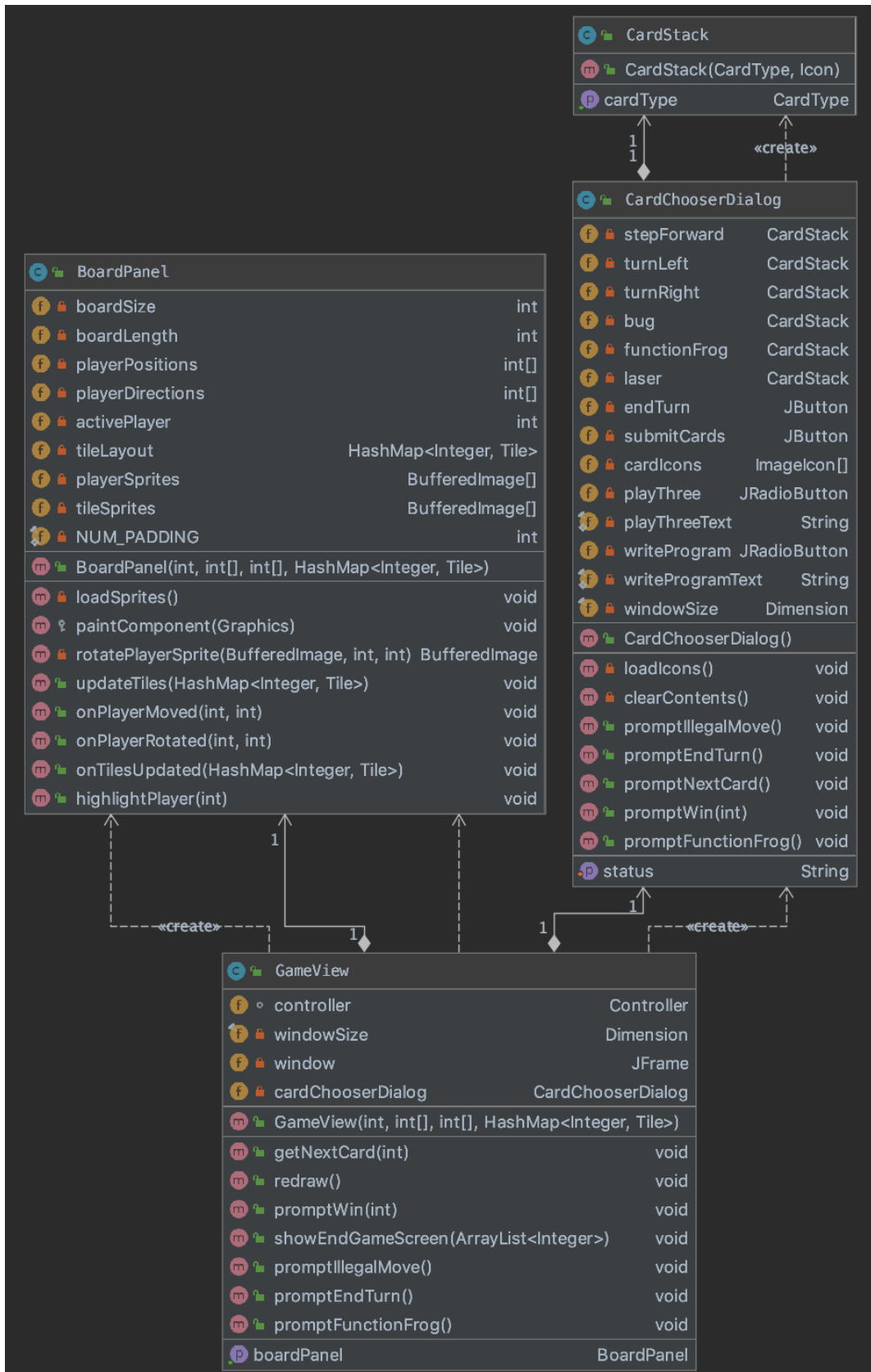
Controller

«interface» PlayerSubscriber
- onPlayerMoved(int, int) : void
- onPlayerRotated(int, int) : void

«enumeration» CardType
- STEP_FORWARD
- TURN_LEFT
- TURN_RIGHT
- FUNCTION_FROG
- BUG
- LASER
- toString() : String
- values() : CardType[]
- valueOf(String) : CardType

«interface» BoardSubscriber
- onTilesUpdated(HashMap<Integer, Tile>) : void

«interface» Player
- onCardsPlayed(Queue<CardType>) : boolean
- addSubscriber(PlayerSubscriber) : void
- setHasWon() : void
- hasWon() : boolean
- usePortal() : void
- number : int

«interface» TurnManager
- getPlayer(int) : Player
- endTurn() : void
- addPlayer(Turtle, int) : void
- activePlayer : Player
- numPlayers : int
- activePlayerNumber : int

«interface» BoardManager
- getTile(int) : Tile
- tileLayout : HashMap<Integer, Tile>
- playerPositions : int[]
- playerDirections : int[]
- boardSize : int

«class» RuleControlListener
- play3 : String
- writeProgram : String
- RuleControlListener(String, String)
- itemStateChanged(ItemEvent) : void

«class» Controller
- view : GameView
- board : BoardManager
- turnManager : TurnManager
- winOrder : ArrayList<Integer>
- instance : Controller
- playThreeEnabled : boolean
- writeProgramEnabled : boolean
- functionStored : boolean[]
- functionFrogActive : boolean
- cardsChosen : Queue<CardType>
- Controller()
- getInstance() : Controller
- main(String[]) : void
- initGame(BoardManager, TurnManager) : void
- registerCardStack(CardStack) : void
- registerRuleControlButtons(ButtonGroup, String, String) : void
- onCardChosen(CardType) : boolean
- onTurnEnded() : void
- submitCards() : void
- onGameEnded() : void

«create»

1 1 1

«class» CardStackListener
- actionPerformed(ActionEvent) : void

«creates»

Model

TileType «enumeration»
- REGULAR
- JEWEL
- ICE_WALL
- STONE_WALL
- PORTAL
- CRATE
- PUDDLE
- values() : TileType[]
- valueOf(String) : TileType

Tile
- crateState : boolean
- Tile(boolean, boolean)
- hasCrate() : boolean
- addCrate() : void
- removeCrate() : void
- getHitByLaser() : void
- vacancy : boolean
- tileType : TileType

Jewel
- Jewel()
- addCrate() : void

StoneWall
- StoneWall()
- addCrate() : void
- vacancy : boolean

RegularTile
- RegularTile()

BoardManager
- getTile(int) : Tile
- tileLayout : HashMap<Integer, Tile>
- playerPositions : int[]
- playerDirections : int[]
- boardSize : int

IceWall
- IceWall()
- getHitByLaser() : void
- addCrate() : void
- isFrozen : boolean

Portal
- Portal()
- position : int
- correspondingPortal : Portal

TileFactory
- createTile(TileType) : Tile

TurtleMover
- NUM_PLAYERS : int
- PLAYER_1 : int
- PLAYER_2 : int
- PLAYER_3 : int
- PLAYER_4 : int
- controller : Controller
- players : Turtle[]
- currentTurn : int
- TurtleMover()
- getPlayer(int) : Player
- endTurn() : void
- addPlayer(Turtle, int) : void
- activePlayer : Player
- numPlayers : int
- activePlayerNumber : int

Board
- BOARD_SIZE : int
- BOARD_LENGTH : int
- InitialBoardLayout1 : TileType[]
- subscribers : ArrayList<BoardSubscriber>
- board : Board
- Board(int)
- getInstance() : Board
- isPositionOccupied(int) : boolean
- getTile(int) : Tile
- addSubscriber(BoardSubscriber) : void
- tileLayout : HashMap<Integer, Tile>
- turtles : Turtle[]
- playerPositions : int[]
- playerDirections : int[]
- boardSize : int

Turtle
- currPosition : int
- prevPosition : int
- hasWon : boolean
- boardLength : int
- NORTH : int
- EAST : int
- SOUTH : int
- WEST : int
- board : Board
- prevDir : int
- subscribers : ArrayList<PlayerSubscriber>
- function : Queue<CardType>
- Turtle(int)
- notifySubscribersOfRotation() : void
- notifySubscribersOfMovement() : void
- getOnePositionAhead(int) : int
- moveForward() : boolean
- bug() : void
- storePrevState() : void
- turn(String) : void
- addSubscriber(PlayerSubscriber) : void
- onCardsPlayed(Queue<CardType>) : boolean
- shootLaser() : void
- setHasWon() : void
- hasWon() : boolean
- usePortal() : void
- position : int
- number : int
- direction : int

«create»

View

CardStack

Ⓜ 🔒	CardStack(CardType, Icon)	
Ⓟ	cardType	CardType

CardChooserDialog

Ⓕ 🔒	stepForward	CardStack
Ⓕ 🔒	turnLeft	CardStack
Ⓕ 🔒	turnRight	CardStack
Ⓕ 🔒	bug	CardStack
Ⓕ 🔒	functionFrog	CardStack
Ⓕ 🔒	laser	CardStack
Ⓕ 🔒	endTurn	JButton
Ⓕ 🔒	submitCards	JButton
Ⓕ 🔒	cardIcons	ImageIcon[]
Ⓕ 🔒	playThree	JRadioButton
Ⓕ 🔒	playThreeText	String
Ⓕ 🔒	writeProgram	JRadioButton
Ⓕ 🔒	writeProgramText	String
Ⓕ 🔒	windowSize	Dimension
Ⓜ 🔒	CardChooserDialog()	
Ⓜ 🔒	loadIcons()	void
Ⓜ 🔒	clearContents()	void
Ⓜ 🔒	promptIllegalMove()	void
Ⓜ 🔒	promptEndTurn()	void
Ⓜ 🔒	promptNextCard()	void
Ⓜ 🔒	promptWin(int)	void
Ⓜ 🔒	promptFunctionFrog()	void
Ⓟ	status	String

BoardPanel

Ⓕ 🔒	boardSize	int
Ⓕ 🔒	boardLength	int
Ⓕ 🔒	playerPositions	int[]
Ⓕ 🔒	playerDirections	int[]
Ⓕ 🔒	activePlayer	int
Ⓕ 🔒	tileLayout	HashMap<Integer, Tile>
Ⓕ 🔒	playerSprites	BufferedImage[]
Ⓕ 🔒	tileSprites	BufferedImage[]
Ⓕ 🔒	NUM_PADDING	int
Ⓜ 🔒	BoardPanel(int, int[], int[], HashMap<Integer, Tile>)	
Ⓜ 🔒	loadSprites()	void
Ⓜ	paintComponent(Graphics)	void
Ⓜ 🔒	rotatePlayerSprite(BufferedImage, int, int)	BufferedImage
Ⓜ 🔒	updateTiles(HashMap<Integer, Tile>)	void
Ⓜ 🔒	onPlayerMoved(int, int)	void
Ⓜ 🔒	onPlayerRotated(int, int)	void
Ⓜ 🔒	onTilesUpdated(HashMap<Integer, Tile>)	void
Ⓜ 🔒	highlightPlayer(int)	void

GameView

Ⓕ ○	controller	Controller
Ⓕ 🔒	windowSize	Dimension
Ⓕ 🔒	window	JFrame
Ⓕ 🔒	cardChooserDialog	CardChooserDialog
Ⓜ 🔒	GameView(int, int[], int[], HashMap<Integer, Tile>)	
Ⓜ 🔒	getNextCard(int)	void
Ⓜ 🔒	redraw()	void
Ⓜ 🔒	promptWin(int)	void
Ⓜ 🔒	showEndGameScreen(ArrayList<Integer>)	void
Ⓜ 🔒	promptIllegalMove()	void
Ⓜ 🔒	promptEndTurn()	void
Ⓜ 🔒	promptFunctionFrog()	void
Ⓟ	boardPanel	BoardPanel

«create»
1
1
«create»
1
1
1
1
1

ID:	StartGame
Title:	The game starts
Description:	The board, turtles, and jewels are initialized. It is Player 1's turn.
Primary Actor:	Player 1
Preconditions:	N/A
Postconditions:	The game has started
Main Success Scenario:	The action on corresponding to the card played is performed.
Extensions:	In a future version of the game, obstacles could be initialized too.
Frequency of Use:	Once per game.
Status:	Implementation
Owner:	Sam Chambré and Eamon McCarron
Priority:	High

ID:	CardPlaying
Title:	Player plays a card
Description:	After a player plays a card the turtle mover will perform the card's move on the corresponding turtle (if the move is legal). The player has the option to hit the bug card and redo their move. If they don't use the bug card, they end their turn. It becomes the next player's turn.
Primary Actor:	The player.
Preconditions:	It's the player's turn.
Postconditions:	The game's state has been updated accordingly and the player's turn is over.
Main Success Scenario:	The action on corresponding to the card played is performed.

Extensions:	A player could attempt to make an illegal move (i.e. trying to move forward when this would send you off the board or attempting to move to a tile occupied by an immovable obstacle).
Frequency of Use:	Once per turn.
Status:	Implementation
Owner:	Sam Chambré and Eamon McCarron
Priority:	High

ID:	UpdateView
Title:	Update the view
Description:	After a player plays a valid card, the controller tells the view to update itself with the new game state.
Primary Actor:	The player.
Preconditions:	A player has the option to play a card.
Postconditions:	The view has been updated to display the change that the card played made.
Main Success Scenario:	The controller tells the view to update.
Extensions:	If an invalid card is played, the view will not have anything to update.
Frequency of Use:	Once per valid card played.
Status:	Implementation
Owner:	Sam Chambré and Eamon McCarron
Priority:	High

ID:	WinGame
Title:	A gem is reached
Description:	On their turn, a player plays a move forward card which brings their turtle from an adjacent tile onto a jewel. They have now won.
Primary Actor:	The player.
Preconditions:	The player's turtle is adjacent to a jewel and playing the move forward card will move the turtle onto a jewel.
Postconditions:	The player's turtle is on the jewel and they have won this round.
Main Success Scenario:	The rest of the players continue until they have won too.
Extensions:	If the player was the last person to move onto a jewel, the game is over.
Frequency of Use:	Once per player per game.
Status:	Planning phase (pre implementation).
Owner:	Sam Chambré and Eamon McCarron
Priority:	High

ID:	EndGame
Title:	The last gem is reached
Description:	On their turn, the last player still in the game plays a move forward card which brings their turtle from an adjacent tile onto a jewel. They have now won.
Primary Actor:	The player.
Preconditions:	The player's turtle is adjacent to a jewel and playing the move forward card will move the turtle onto a jewel.
Postconditions:	The player's turtle is on the jewel and they have won this round. The game is now over.
Main Success Scenario:	The game ends.
Extensions:	N/A
Frequency of Use:	Once per game.

Status:	Implementation
Owner:	Sam Chambré and Eamon McCarron
Priority:	High

ID:	PushCrate
Title:	The turtle attempts to push a crate as they move forward.
Description:	When a player selects to move forward and there's a crate in front of them, the crate will move forward as well if the space in front of it is vacant. If not, the crate won't move, and neither will the turtle.
Primary Actor:	The player.
Preconditions:	The player's turtle is adjacent to a crate and playing the move forward card would move the turtle to where the crate is.
Postconditions:	The player's turtle and the crate have moved forward or neither have moved.
Main Success Scenario:	The turtle and crate move
Extensions:	In a future game we could have other movable objects.
Frequency of Use:	Possibility every time a turtle tries to move forward
Status:	Implementation
Owner:	Sam Chambré and Eamon McCarron
Priority:	High

ID:	UsePortal
Title:	The turtle uses a portal
Description:	When a player selects to move forward and there's a portal in front of them, the turtle will be moved to this portal's corresponding portal's location
Primary Actor:	The player.
Preconditions:	The player's turtle is adjacent to a portal and playing the move forward card would move the turtle to where the portal is.
Postconditions:	The player's turtle is moved to the corresponding portal's location.
Main Success Scenario:	The turtle is moved to the new location
Extensions:	If a turtle is already on the corresponding portal, the turtle cannot move onto the portal in front of them.
Frequency of Use:	Possibility every time a turtle tries to move forward.
Status:	Implementation
Owner:	Sam Chambré and Eamon McCarron
Priority:	High

ID:	PlayMultipleCards
Title:	Multiple cards played on the same turn
Description:	On their turn, a player uses play three, write program, or the function frog to play multiple cards on their turn.
Primary Actor:	The player.
Preconditions:	It is the player's turn and they have selected one of the options to play multiple cards.
Postconditions:	The player's turtle has performed the actions on all the cards.
Main Success Scenario:	The turtle performs all the actions
Extensions:	It's possible that the sequence of actions will not be legal.
Frequency of Use:	Once per turn if option is selected

Status:	Implementation
Owner:	Sam Chambré and Eamon McCarron
Priority:	High