

# Diagramma delle classi in UML progetto Juno

Venturini Daniele e D'annibale Luigi

## Indice

[View](#)

[View.Animation](#)

[View.Elements](#)

[View.Pages](#)

[View.Pages.ProfilePanels](#)

[Model](#)

[Model.Players](#)

[Model.Cards](#)

[Model.Rules](#)

[Controller](#)

[Controller.Utilities](#)

# package src.View

## View

### Utils

+ Utils():  
+ toBufferedImage(Image): BufferedImage  
+ getImage(String): Image  
+ applyQualityRenderingHints(Graphics2D): void  
+ rotateImage(BufferedImage, double): BufferedImage  
+ getBufferedImage(String): BufferedImage?

<<package>>

**package View.Animations**

<<package>>

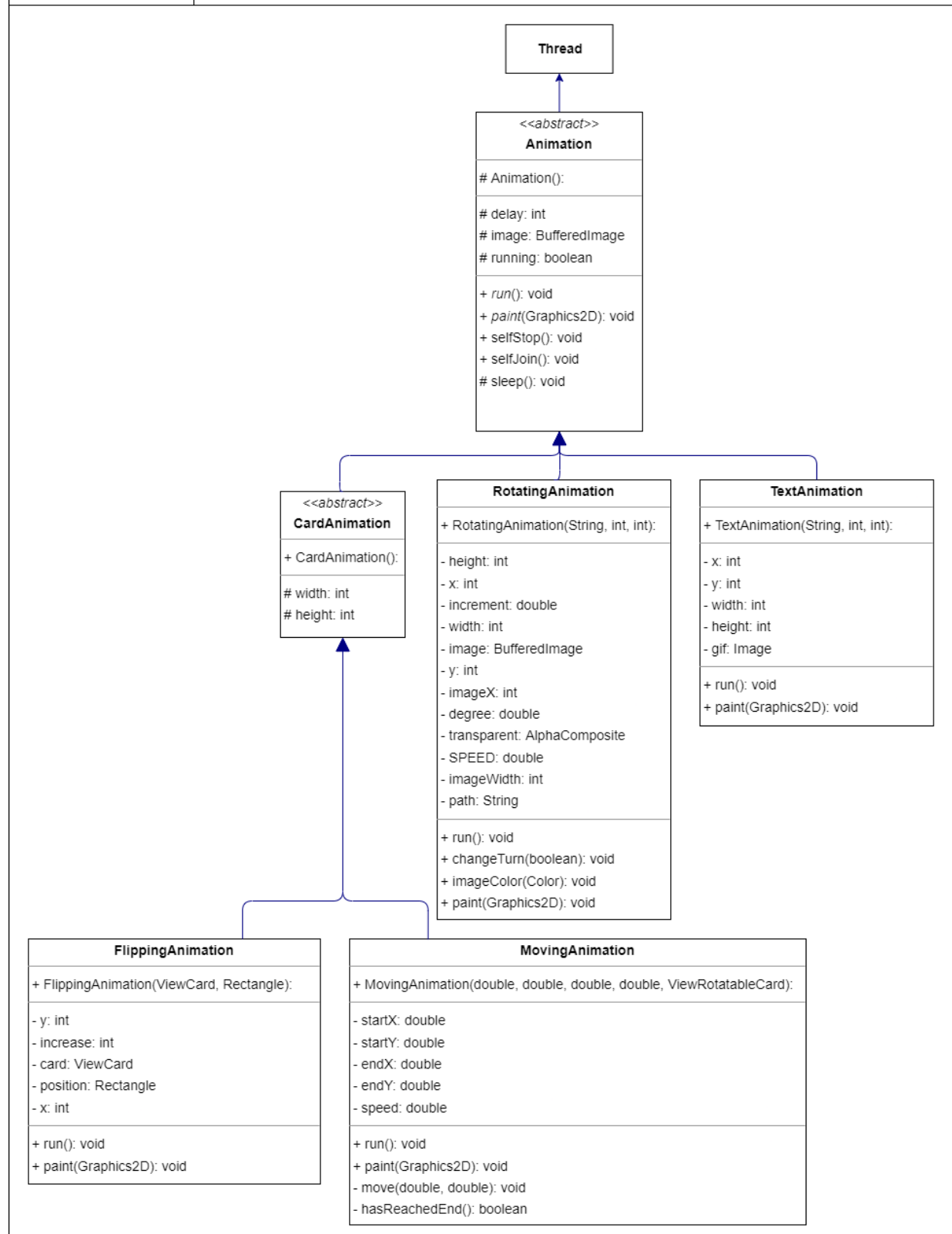
**package View.Elements**

<<package>>

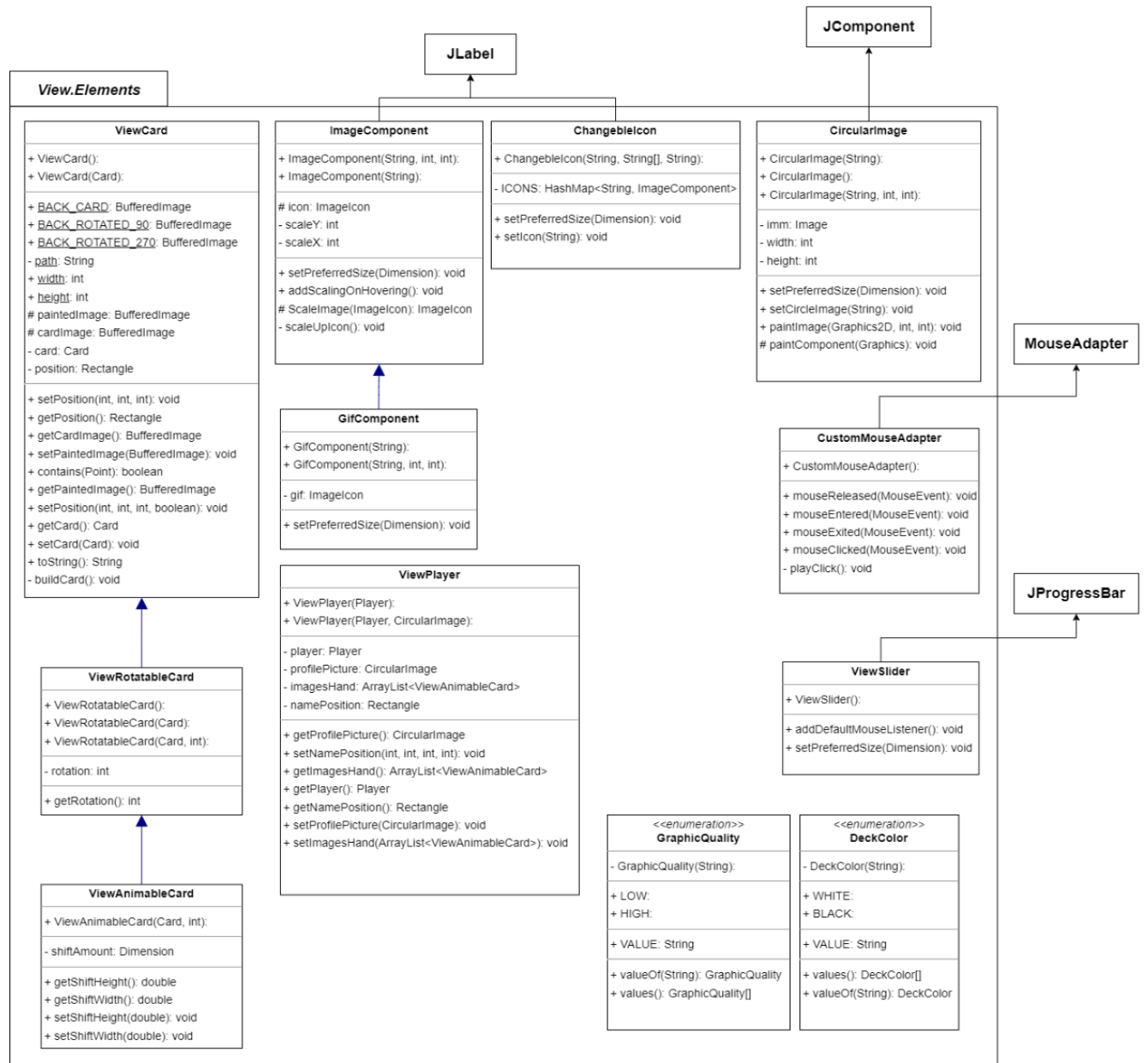
**package View.Pages**

# package src.View.Animation

View.Animations



# package src.View.Elements



# package src.View.Pages

## View.Pages

GamePanel
+ GamePanel(ViewPlayer[]):
- maxCardsHeight: int
- ladder: List<Player>
- titleFont: Font
- continuePosition: Rectangle
- transparent: AlphaComposite
- ladderHeight: int
- rotatingAnimation: RotatingAnimation
- deckY: int
- gameRunning: boolean
- discard: ViewCard
- deckX: int
- gameThread: Thread
- fontNames: Font
- skipTurnPosition: Rectangle
- animations: ArrayList<Animation>
- currentPlayerBackground: Color
- centerX: int
- aiRunning: boolean
- ladderY: int
- movingAnimation: MovingAnimation
- a: BasicStroke
- viewPlayers: ViewPlayer[]
- playerBackground: Color
- discardX: int
- lastCard: Card
- currentViewPlayer: ViewPlayer
- continueString: String
- discardY: int
- imagePath: String
- deckSize: int
- ladderX: int
- ladderThickness: int
- deck: ViewCard
- centerY: int
- unoPosition: Rectangle
- green: Color
- players: Player[]
- flipAnimation: FlippingAnimation
- ladderWidth: int
- maxCardsWidth: int
- aiThread: Thread
- ladderFont: Font
- currentState: State
+ pauseGame(): void
+ exposedAnimation(): Animation
+ animateCardsOnHovering(MouseEvent): void
+ choseColorByUser(): Color
+ flipCardAnimation(ViewCard): Animation
+ drawCardAnimation(ViewPlayer, ViewRotatableCard): Animation
+ update(Observable, Object): void
+ animationRunning(Animation): boolean
+ stopTimer(): void
+ calculateState(): State
+ createCards(): void
+ playCardAnimation(ViewRotatableCard): Animation
+ resumeGame(): void
+ getContinuePosition(): Rectangle
+ shoutUnoAnimation(Player): void
+ getCurrentState(): State
+ getDeck(): ViewCard
+ getViewPlayers(): ViewPlayer[]
+ getUnoPosition(): Rectangle
+ getSkipTurnPosition(): Rectangle
+ chosePlayerToSwap(): Player
+ getPlayers(): Player[]
# paintComponent(Graphics): void
- asyncAITurn(UnoGameTable): void
- randomSleep(int, int): void
- initializeComponents(): void
- repaintView(): void
- getCardsWidth(ViewPlayer, int): int
- drawHorizontalHand(ViewPlayer, Graphics2D, int): void
- drawVerticalHand(ViewPlayer, Graphics2D, int): void
- drawNames(ViewPlayer, int, int, Graphics2D, int, int): void

<<package>>  
package View.Pages.ProfilePanels

SettingsPanel
+ SettingsPanel():
- pathImages: String
- effectsLabel: ChangebleIcon
- qualityCombo: JComboBox<GraphicQuality>
- green: Color
- whiteDeck: DeckRectangle
- darkDeck: DeckRectangle
- qualityLabel: ChangebleIcon
- saveButton: GifComponent
- effectsViewSlider: ViewSlider
- musicViewSlider: ViewSlider
- closeButton: GifComponent
- gameOver: ImageComponent
- deckStyleLabel: ImageComponent
- font: Font
- musicLabel: ChangebleIcon
+ getWhiteDeck(): DeckRectangle
+ getEffectsVolumeSlider(): ViewSlider
+ getDarkDeck(): DeckRectangle
+ getMusicVolumeSlider(): ViewSlider
+ getQualityLabel(): ChangebleIcon
+ getEffectsLabel(): ChangebleIcon
+ getMusicLabel(): ChangebleIcon
+ getQualityCombo(): JComboBox<GraphicQuality>
+ getCloseButton(): JLabel
+ getSaveButton(): JLabel
+ getGameOverButton(): JLabel
# paintComponent(Graphics): void
- initializeComponents(): void

ResizablePanel
+ ResizablePanel(int, int, int):
# panelHeight: int
# panelWidth: int
+ getPreferredSize(): Dimension
# resizeComponents(): void

StartingMenuPanel
+ StartingMenuPanel():
- darkBlue: Color
- startGameLabel: ImageComponent
- settingLabel: ImageComponent
- angle: int
- lightBlue: Color
- offset: int
- quitLabel: ImageComponent
- pathImages: String
+ getGameChoiceIcon(): ImageComponent
+ getSettingIcon(): ImageComponent
+ getQuitIcon(): ImageComponent
# paintComponent(Graphics): void
- initializeComponents(): void

MainFrame
+ MainFrame():
+ IMAGE_PATH: String
- pathImages: String
- dimension: Dimension
- background: Image
- gameBackground: JPanel
- mainBackground: JPanel
- settingBackground: JPanel
- gbc: GridBagConstraints
+ getGameBackground(): JPanel
+ addProfilePanel(JPanel): void
+ addCenteredPanels(JPanel): void
+ getDimension(): Dimension

<<enumeration>> Cards
- Cards():
+ GAME:
+ SETTINGS:
+ MAIN:
+ values(): Cards
+ toString(): String
+ valueOf(String): Cards

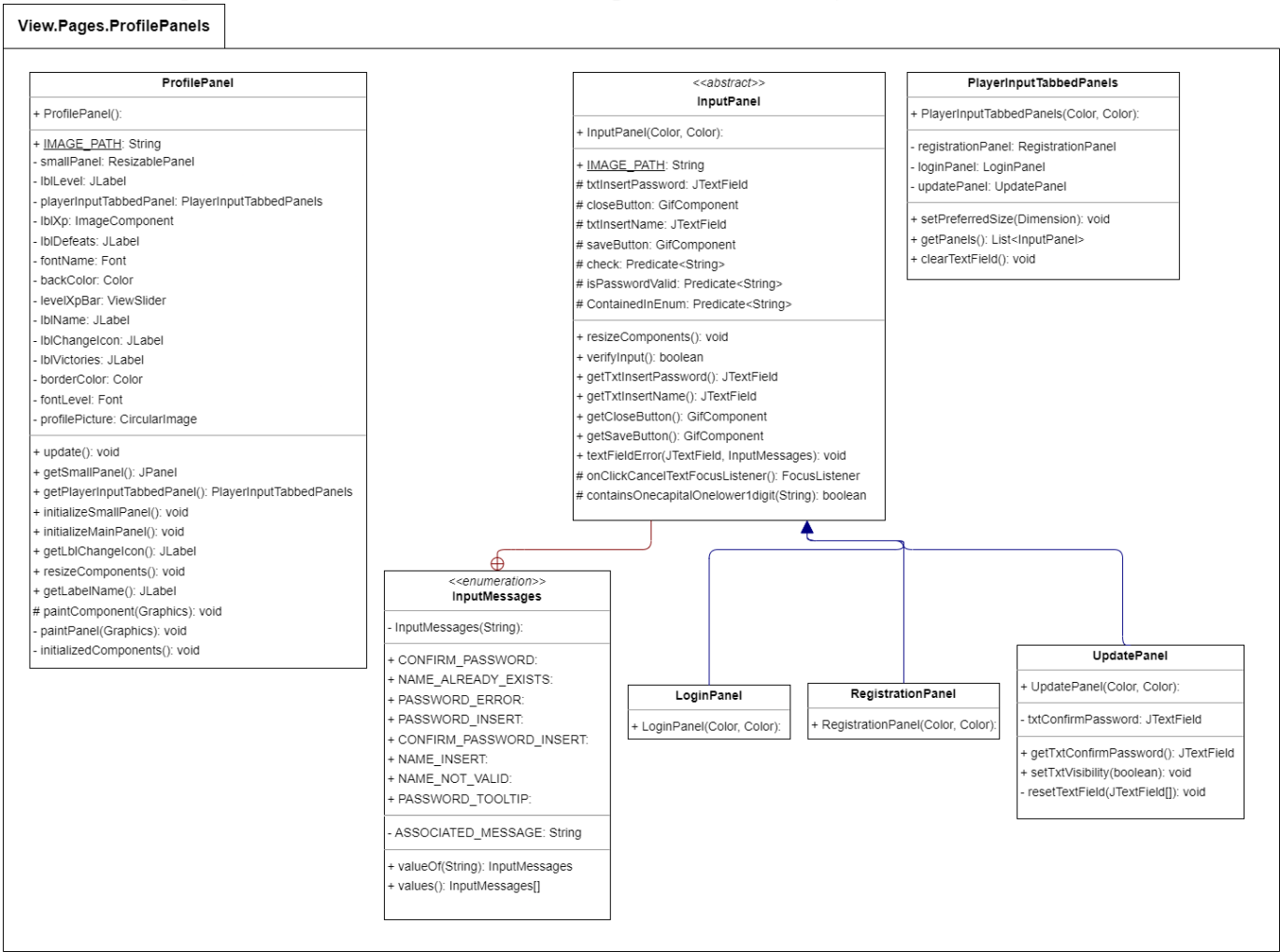
JComponent

DeckRectangle
+ DeckRectangle(String, String):
- radius: int
- width: int
- height: int
- title: String
- image: BufferedImage
- deckRectangleBorder: Color
- deckColor: DeckColor
- paintBackground: boolean
+ setPaintBackground(boolean): void
+ getDeckColor(): DeckColor
+ setPreferredSize(Dimension): void
# paintComponent(Graphics): void

<<enumeration>> State
- State():
+ GAME_PAUSED:
+ PLAYER_TURN:
+ AI_TURN:
+ MATCH_WIN:
+ WIN:
+ values(): State[]
+ valueOf(String): State

GameChoicePanel
+ GameChoicePanel():
- gameModes: GifComponent[]
- infosHeight: int
- infos: BufferedImage[]
- title: ImageComponent
- pathImages: String
- infosWidth: int
- currentInfo: int
- infoLabels: ImageComponent[]
- back: ImageComponent
+ getSevenoGame(): GifComponent
+ getInfoLabels(): JLabel[]
+ getMemeGame(): GifComponent
+ setCurrentInfo(int): void
+ getBack(): ImageComponent
+ getBasicGame(): GifComponent
+ paint(Graphics): void
+ getGameModes(): GifComponent[]
# paintComponent(Graphics): void
- initializeComponents(): void

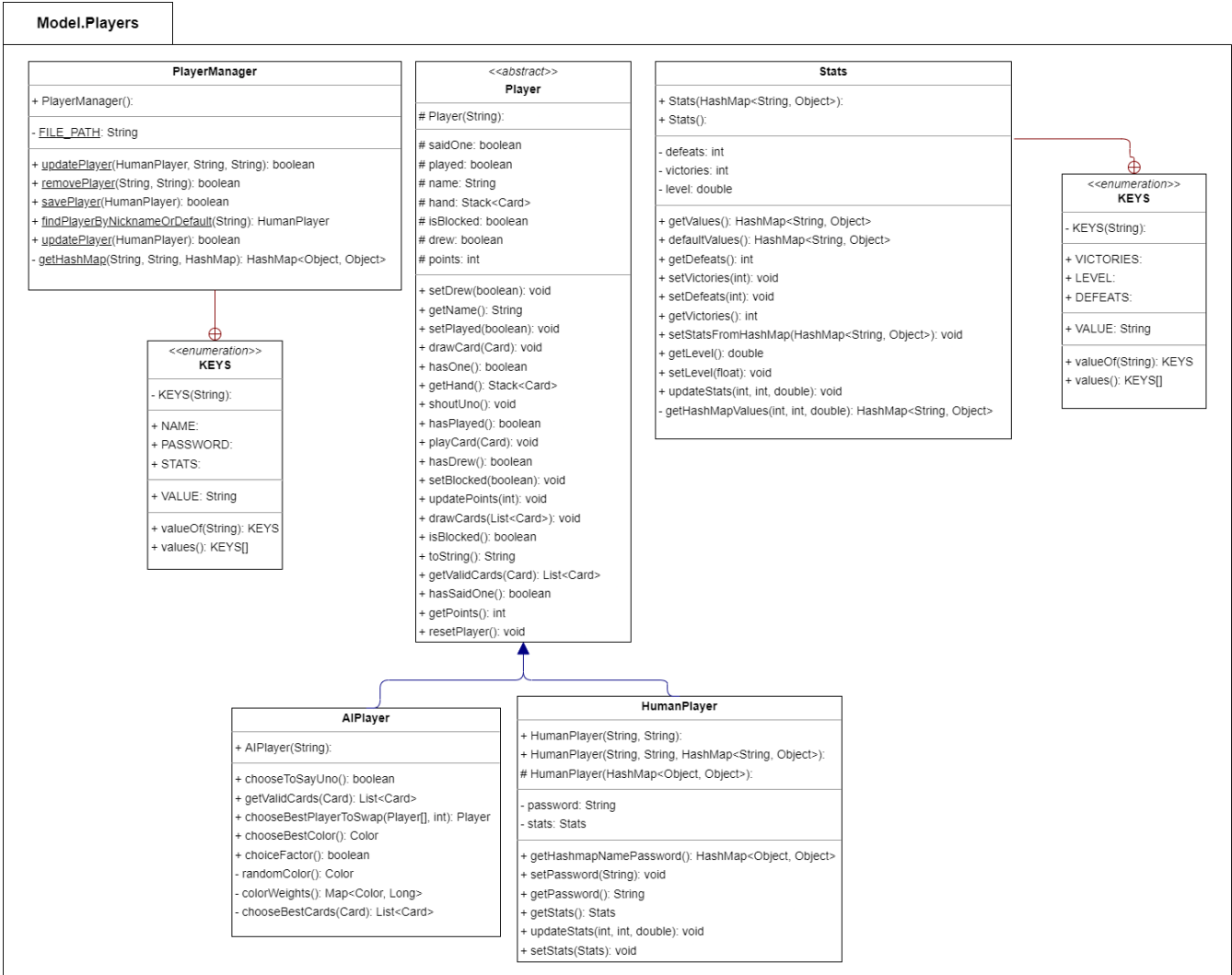
# package src.View.Pages.ProfilePanels



# package src.Model

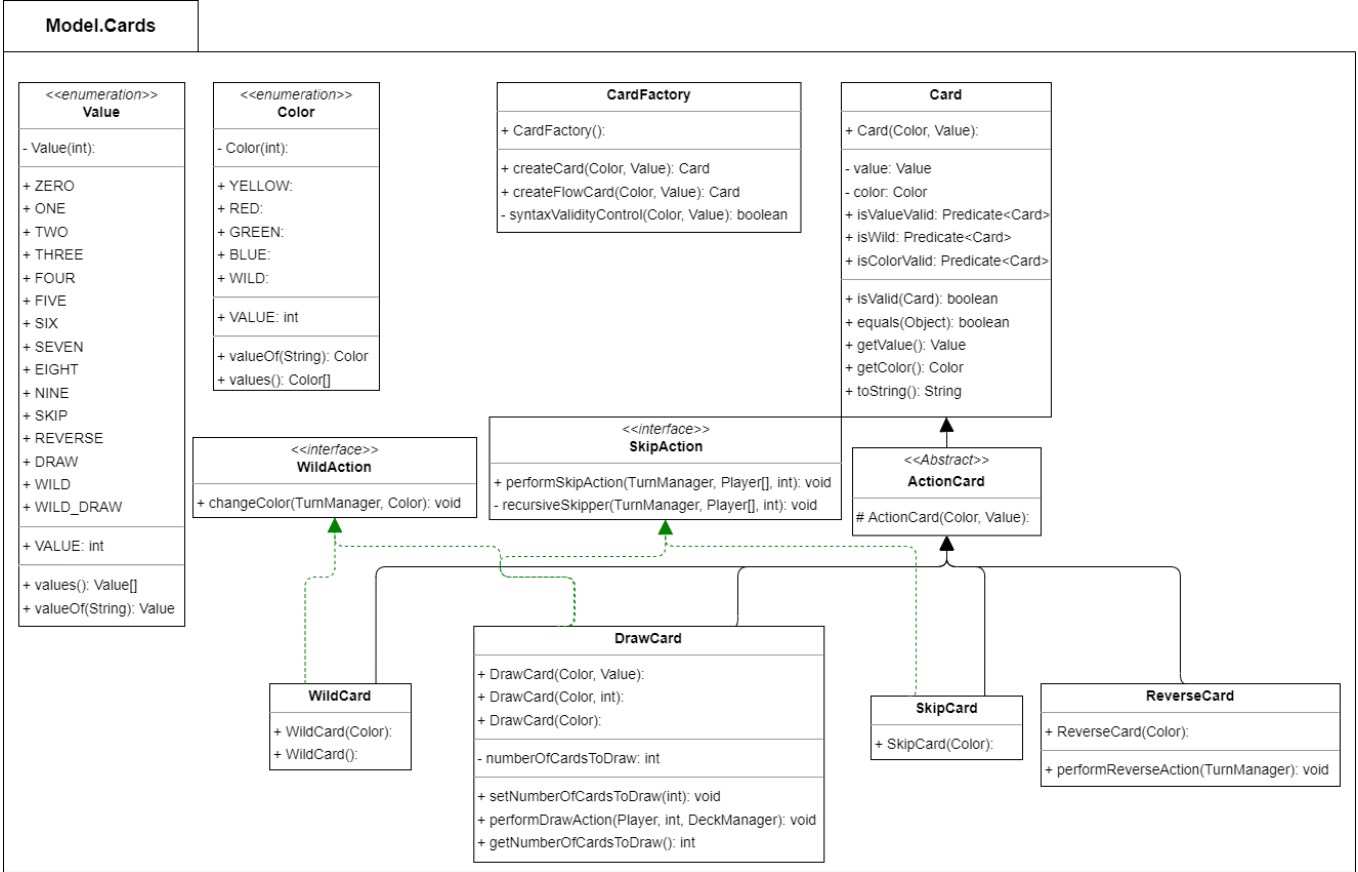
Model			
<div>&lt;&lt;package&gt;&gt; package Model.Players</div> <div>&lt;&lt;package&gt;&gt; package Model.Cards</div> <div>&lt;&lt;package&gt;&gt; package Model.Rules</div>	<div>DeckManager</div> <div>+ DeckManager(HashMap&lt;Value, Integer&gt;); + DeckManager();  + CLASSIC_RULES_CARD_DISTRIBUTION: HashMap&lt;Value, Integer&gt; - deck: Stack&lt;Card&gt; - discards: Stack&lt;Card&gt;  + peekDeck(): Card + peekDiscards(): Card + draw(): Card + draw(int): ArrayList&lt;Card&gt; + pushDiscards(Card): void + getDeck(): Stack&lt;Card&gt; + size(): int + shuffle(): void - re_shuffle(): void + addManyCards(Color, Value, int): void + createDeck(HashMap&lt;Value, Integer&gt;): void</div>	<div>UnoGameTable</div> <div>+ UnoGameTable(Player[], UnoGameRules);  - turnManager: TurnManager - deckManager: DeckManager - win: boolean - ruleManager: UnoGameRules - players: Player[]  + passTurn(): void + currentPlayerIndex(): int + antiClockwiseTurn(): boolean + getLastCard(): Card + checkWin(Player): boolean + playCard(Card): ActionPerformResult + getPlayers(): Player[] + performFirstCard(Options): ActionPerformResult + isExposable(int): boolean + cardActionPerformance(Options): ActionPerformResult + hasWin(): boolean + currentPlayer(): Player + getDeck(): DeckManager + getTurnManager(): TurnManager + drawCard(Player): void + peekNextCard(): Card + checkGameWin(Player): boolean + startGame(): ActionPerformResult + getCurrentPlayerPlayableCards(): List&lt;Card&gt; + expose(Player): void + getOptions(): OptionsBuilder - updateObservers(): void</div>	<div>TurnManager</div> <div>+ TurnManager(Card);  - lastCardPlayed: Card - numberOfPlayers: int - increase: int - player: int  + next(int): int + getLastCardPlayed(): Card + antiClockwiseTurn(): boolean + passTurn(): void + next(): int + previous(int): int + reverseTurn(): void + previous(): int + updateLastCardPlayed(Value, Color): void + getPlayer(): int + updateLastCardPlayed(Card): void + setPlayer(int): void</div>

# package src.Model.Players

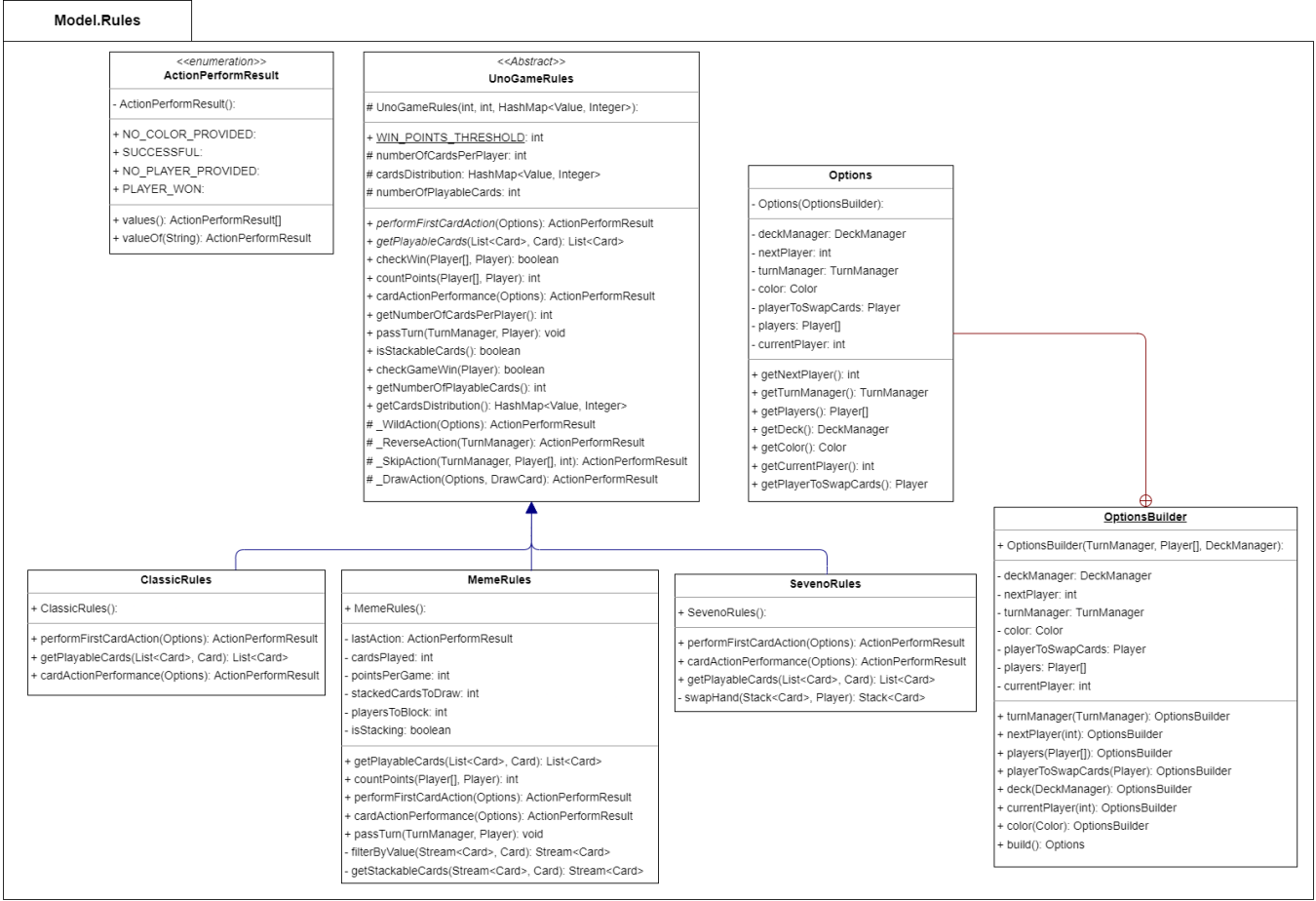




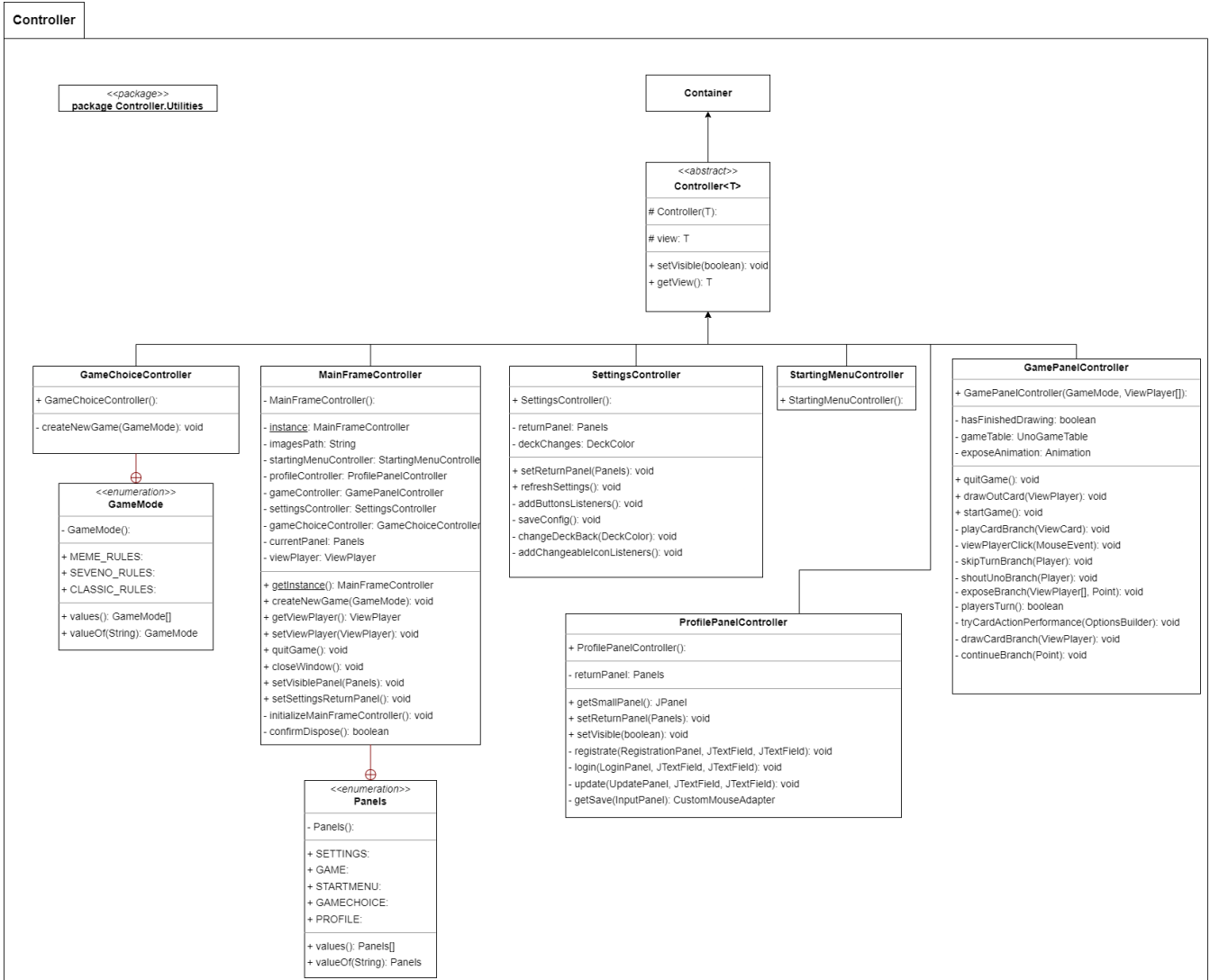
# package src.Model.Cards



# package src.Model.Rules



# package src.Controller



# package src.Controller.Utilities

Controller.Utilities

<<enumeration>> Musics	
- Musics():	
+ CALM_BACKGROUND:	⊕
+ values(): Musics[]	
+ valueOf(String): Musics	

<<enumeration>> Effect	
- Effect():	
+ ERROR:	
+ WIN:	
+ GREEN:	
+ NOT_VALID:	
+ LOSS:	
+ AUDIO_TEST:	
+ DRAW:	
+ REVERSE:	
+ YELLOW:	⊕
+ PLAY:	
+ RED:	
+ SKIP:	
+ FLIP:	
+ BLUE:	
+ WILD_DRAW:	
+ UNO:	
+ CLICK:	
+ DRAW_CARD:	
+ valueOf(String): Effect	
+ values(): Effect[]	

AudioManager	
- AudioManager():	
- instance: AudioManager	
- effectsTrack: Clip	
- audioTrack: Clip	
- convert: Function<Integer, Float>	
- pathAudio: String	
- folder: String	
+ getInstance(): AudioManager	
+ setAudio(Musics): void	
+ stopEffect(): void	
+ setEffect(Effct): void	
+ setEffect(Effct, int): void	
+ setFolder(String): void	
- getPathAudio(String, String): String	
- getCommonAudioFile(String): File	
- getSpecificAudioFile(String): File	
- getAudioFile(String): File	

DataAccessManager	
+ DataAccessManager():	
- PLAYER_CONFIG_JSON: String	
- INIT_JSON: String	
+ getModelProfile(String): HumanPlayer	
+ loadInitOrDefault(): boolean	
+ updateProfile(HumanPlayer, String, String): boolean	
+ saveModelProfile(HumanPlayer): boolean	
+ savePlayerConfig(HumanPlayer): boolean	
+ saveInit(HumanPlayer): boolean	
+ loadPlayerProfile(String): boolean	
+ updatePlayerConfig(HumanPlayer, String): boolean	
+ saveProfile(HumanPlayer): boolean	
+ updateModelProfile(String, String): boolean	
- loadConfig(HashMap<Object, Object>): void	
- loadInit(): boolean	

Config	
+ Config():	
+ DEFAULT_ICON_PATH: String	
+ scalingPercentage: double	
+ loggedPlayer: HumanPlayer	
+ graphicQuality: GraphicQuality	
+ effectsVolume: int	
+ deckStyle: DeckColor	
+ musicVolume: int	
+ savedIconPath: String	
+ setHashMap(HashMap<Object, Object>): void	
+ assignDefaultValues(): void	
+ refreshScalingPercentage(): void	
+ getSettings(): HashMap<Object, Object>	
+ storeConfig(): void	
+ getPlayerConfig(HumanPlayer): HashMap<Object, Object>	
+ getDefaultSettings(): HashMap<Object, Object>	
+ getHashMap(int, int, DeckColor, GraphicQuality, String): HashMap<Object, Object>	

<<enumeration>> KEYS	
- KEYS(String):	
+ EFFECTS_VOLUME:	
+ MUSIC_VOLUME:	
+ LOGGED_PLAYER:	
+ DECK_STYLE:	
+ CONFIG:	⊕
+ SAVED_ICON_PATH:	
+ GRAPHIC_QUALITY:	
+ VALUE: String	
+ valueOf(String): KEYS	
+ values(): KEYS[]	

JsonFileManager	
+ JsonFileManager():	
+ writeJson(List<HashMap<Object, Object>>, String): boolean	
+ readJson(String): ArrayList<HashMap<Object, Object>>	
+ overWriteJson(List<HashMap<Object, Object>>, String): boolean	
- read(String): JSONArray	
- write(JSONArray, String): boolean	
- jsonToHashMap(JSONObject): HashMap<Object, Object>	