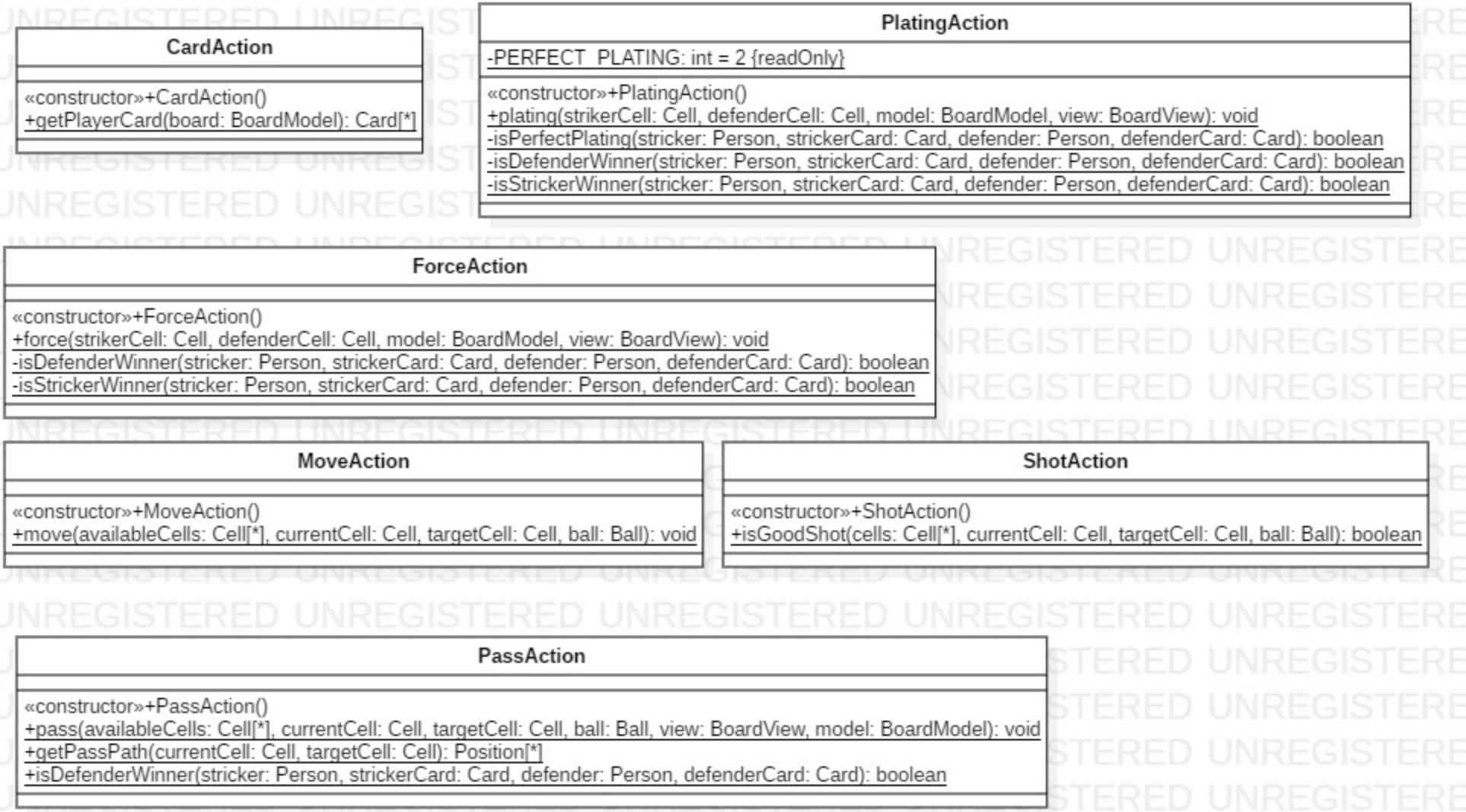
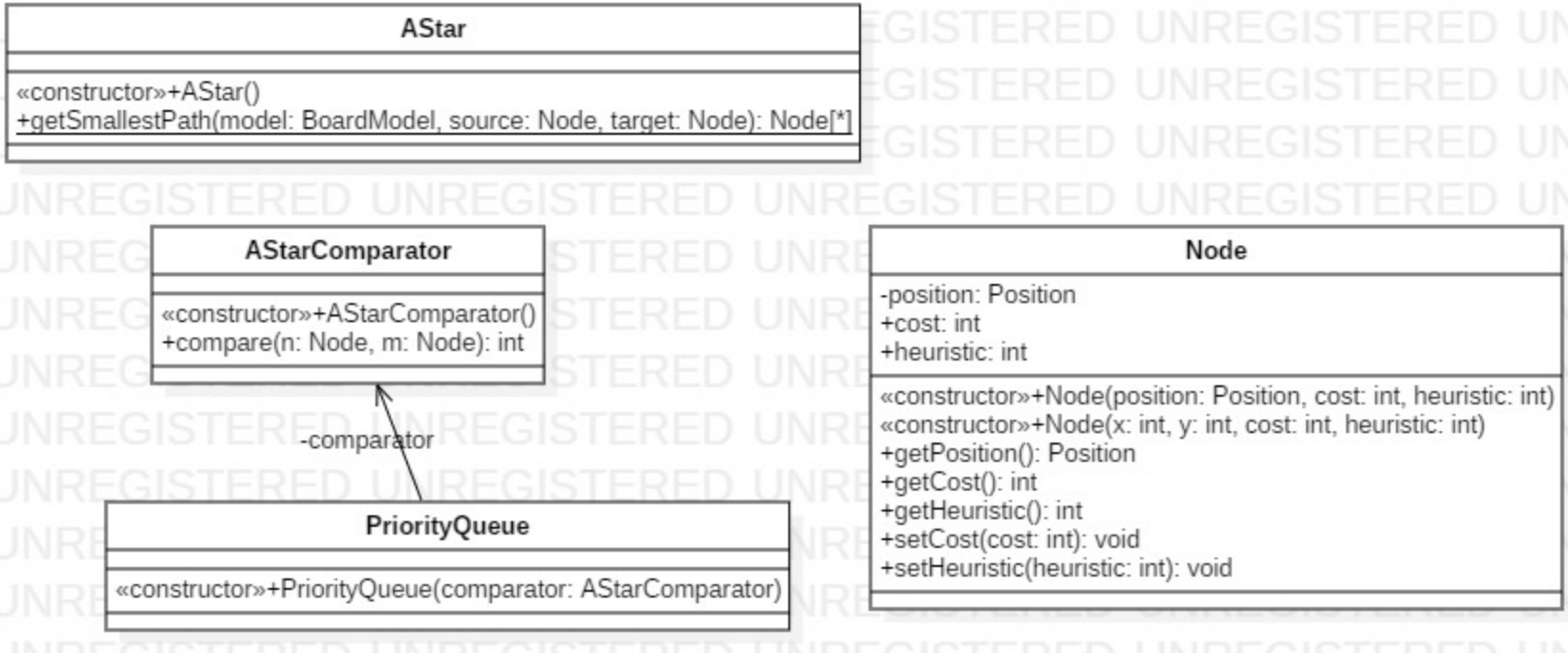
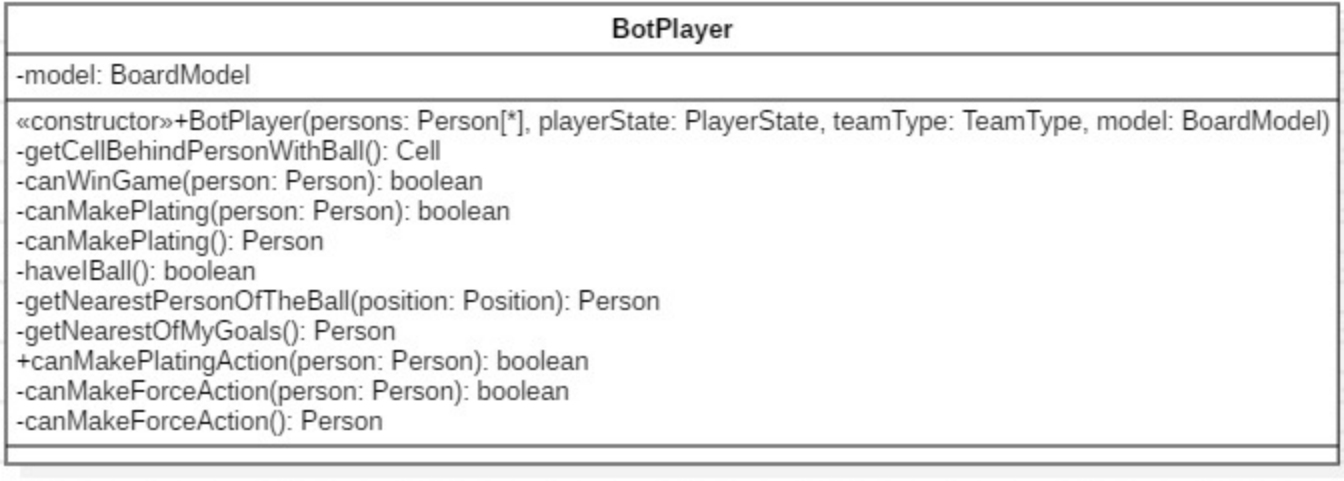
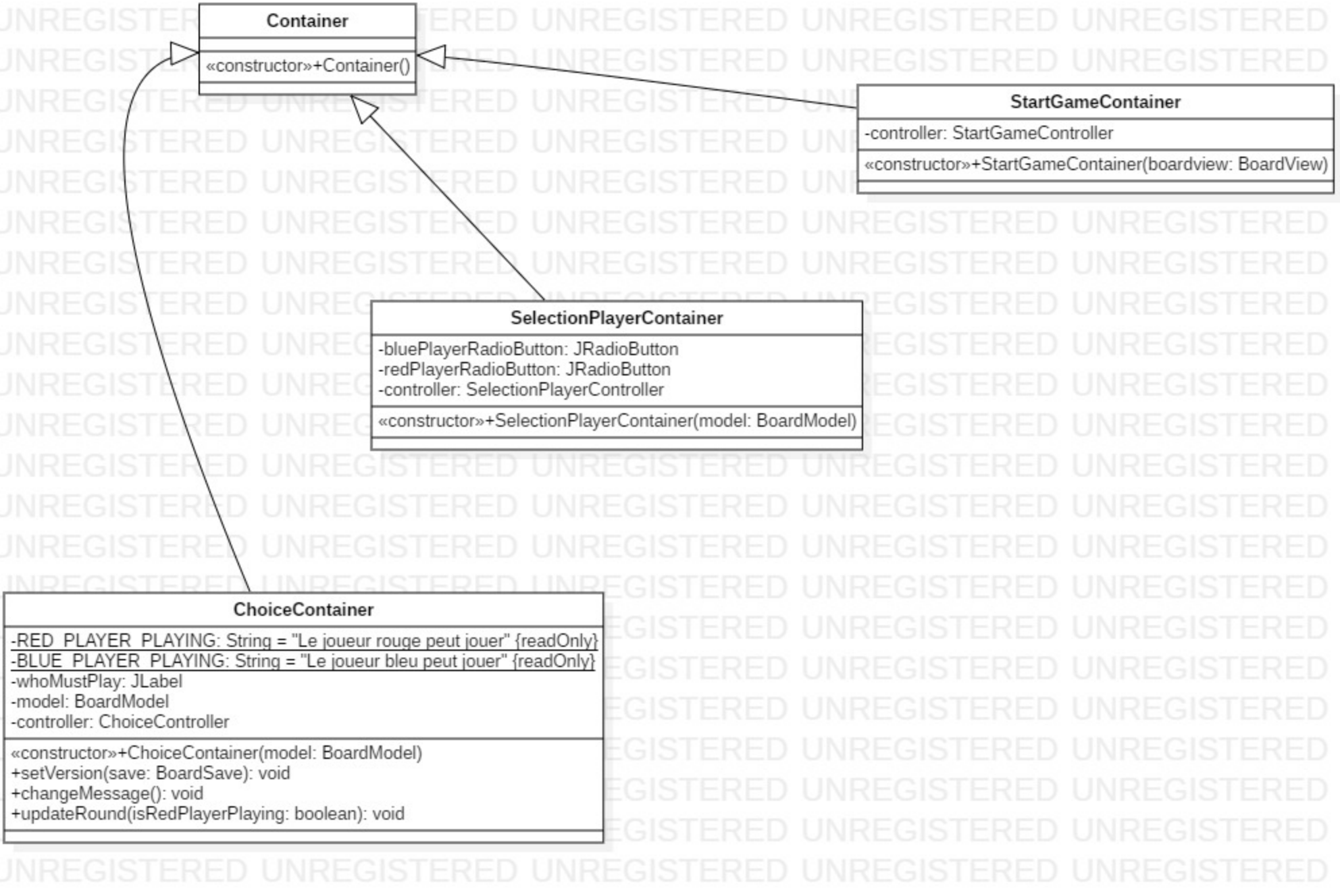
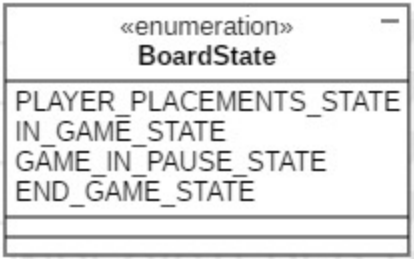

CardAction
«constructor»+CardAction()
+getPlayerCard(board: BoardModel): Card[*]
PlatingAction
-PERFECT_PLATING: int = 2 {readOnly}
«constructor»+PlatingAction()
+plating(strikerCell: Cell, defenderCell: Cell, model: BoardModel, view: BoardView): void
-isPerfectPlating(stricker: Person, strickerCard: Card, defender: Person, defenderCard: Card): boolean
-isDefenderWinner(stricker: Person, strickerCard: Card, defender: Person, defenderCard: Card): boolean
-isStrickerWinner(stricker: Person, strickerCard: Card, defender: Person, defenderCard: Card): boolean
ForceAction
«constructor»+ForceAction()
+force(strikerCell: Cell, defenderCell: Cell, model: BoardModel, view: BoardView): void
-isDefenderWinner(stricker: Person, strickerCard: Card, defender: Person, defenderCard: Card): boolean
-isStrickerWinner(stricker: Person, strickerCard: Card, defender: Person, defenderCard: Card): boolean
MoveAction
«constructor»+MoveAction()
+move(availableCells: Cell[*], currentCell: Cell, targetCell: Cell, ball: Ball): void
ShotAction
«constructor»+ShotAction()
+isGoodShot(cells: Cell[*], currentCell: Cell, targetCell: Cell, ball: Ball): boolean
PassAction
«constructor»+PassAction()
+pass(availableCells: Cell[*], currentCell: Cell, targetCell: Cell, ball: Ball, view: BoardView, model: BoardModel): void
+getPassPath(currentCell: Cell, targetCell: Cell): Position[*]
+isDefenderWinner(stricker: Person, strickerCard: Card, defender: Person, defenderCard: Card): boolean

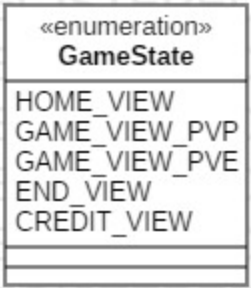

AStar
«constructor»+AStar()
+getSmallestPath(model: BoardModel, source: Node, target: Node): Node[*]
AStarComparator
«constructor»+AStarComparator()
+compare(n: Node, m: Node): int
-comparator
PriorityQueue
«constructor»+PriorityQueue(comparator: AStarComparator)
Node
-position: Position
+cost: int
+heuristic: int
«constructor»+Node(position: Position, cost: int, heuristic: int)
«constructor»+Node(x: int, y: int, cost: int, heuristic: int)
+getPosition(): Position
+getCost(): int
+getHeuristic(): int
+setCost(cost: int): void
+setHeuristic(heuristic: int): void

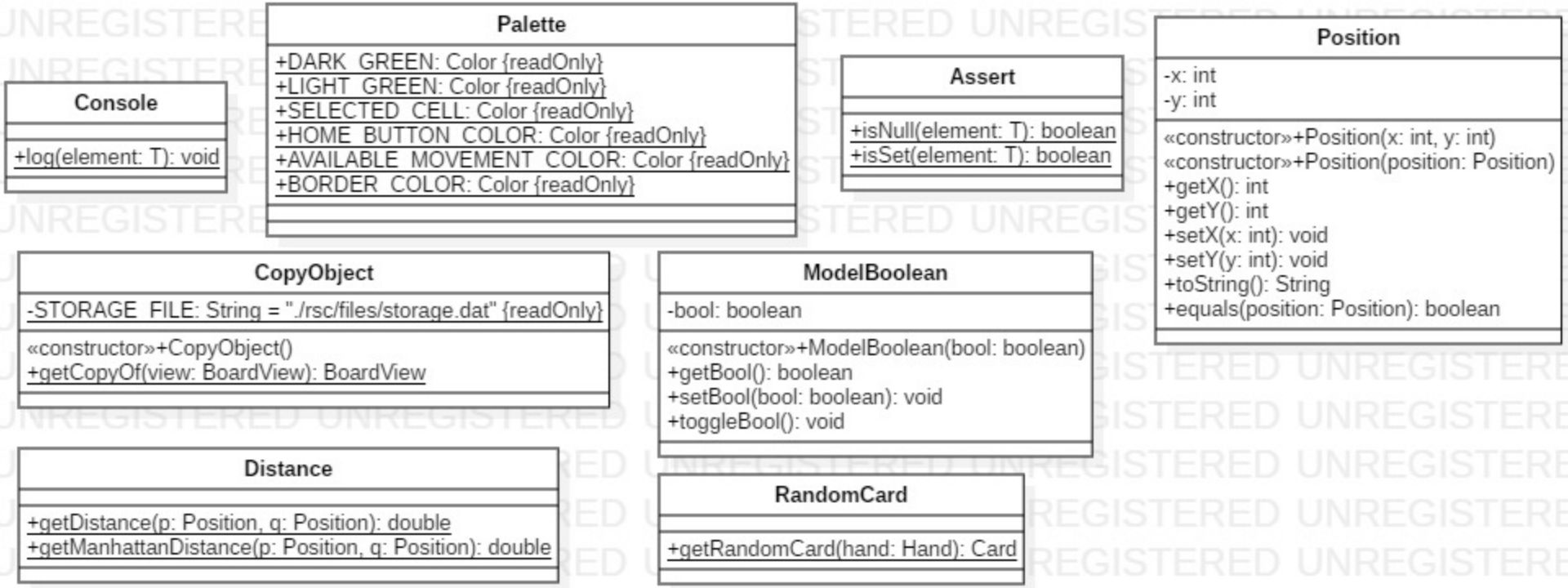


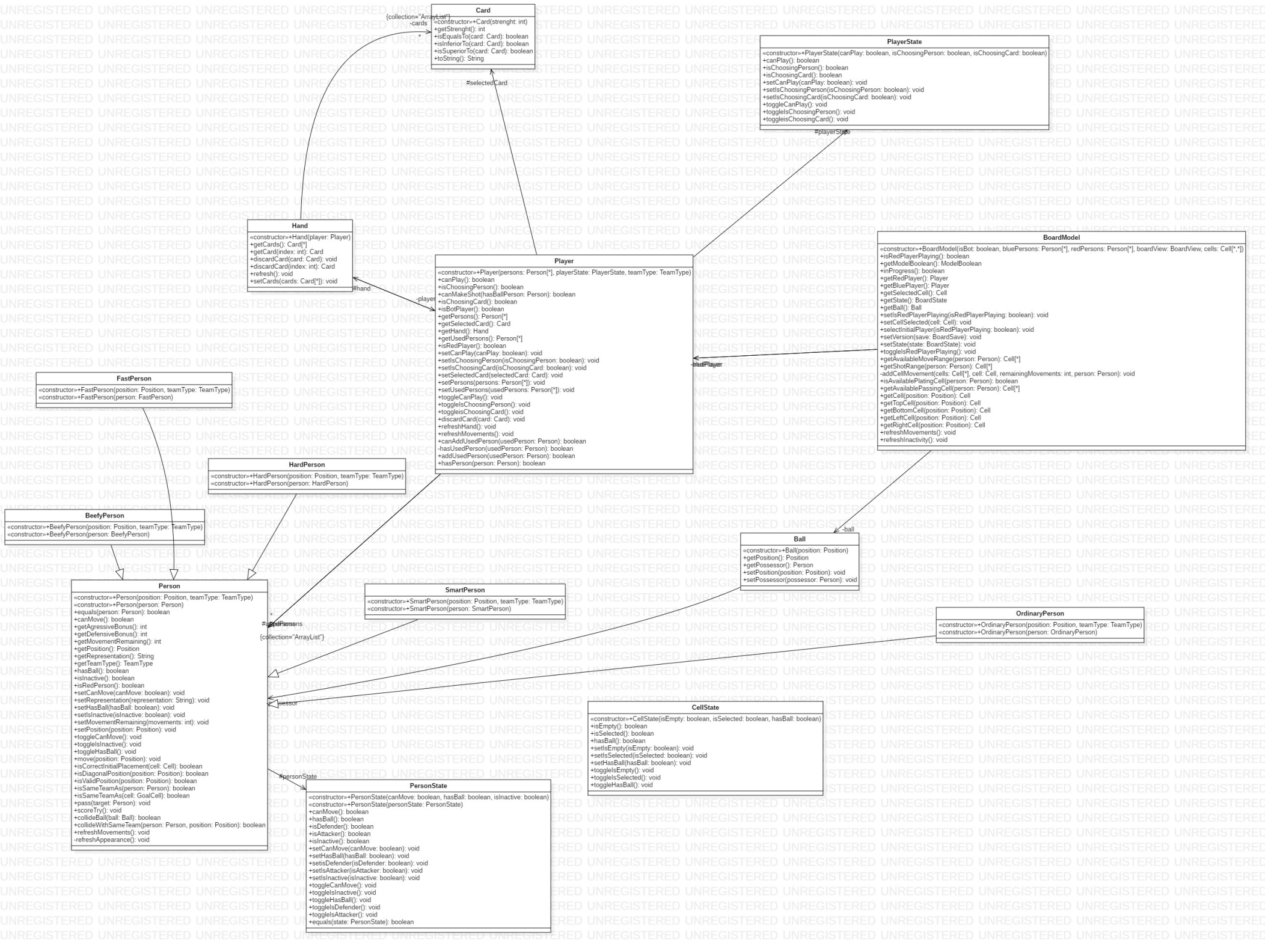

SelectionPlayerController
-BLUE_PLAYER_ACTION_COMMAND: String = "BLUE_PLAYER_ACTION_COMMAND" {readOnly}
-RED_PLAYER_ACTION_COMMAND: String = "RED_PLAYER_ACTION_COMMAND" {readOnly}
-VALIDATION_BUTTON_ACTION_COMMAND: String = "VALIDATION_BUTTON_ACTION_COMMAND" {readOnly}
-model: BoardModel
-bluePlayerRadioButton: JRadioButton
-redPlayerRadioButton: JRadioButton
«constructor»+SelectionPlayerController(bluePlayerRadioButton: JRadioButton, redPlayerRadioButton: JRadioButton, validationButton: JButton, model: BoardModel)
+actionPerformed(event: ActionEvent): void
StartGameController
-START_GAME_ACTION_COMMAND: String = "START_GAME_ACTION_COMMAND" {readOnly}
-view: BoardView
«constructor»+StartGameController(startButton: JButton, view: BoardView)
+actionPerformed(event: ActionEvent): void
ChoiceController
-END_TURN_ACTION_COMMAND: String = "END_TURN_ACION_COMMAND" {readOnly}
-UNDO_ACTION_COMMAND: String = "UNDO_ACTION_COMMAND" {readOnly}
-REDO_ACTION_COMMAND: String = "REDO_ACTION_COMMAND" {readOnly}
-container: ChoiceContainer
-model: BoardModel
«constructor»+ChoiceController(container: ChoiceContainer, endTurn: JButton, undoButton: JButton, redoButton: JButton, model: BoardModel)
+actionPerformed(event: ActionEvent): void

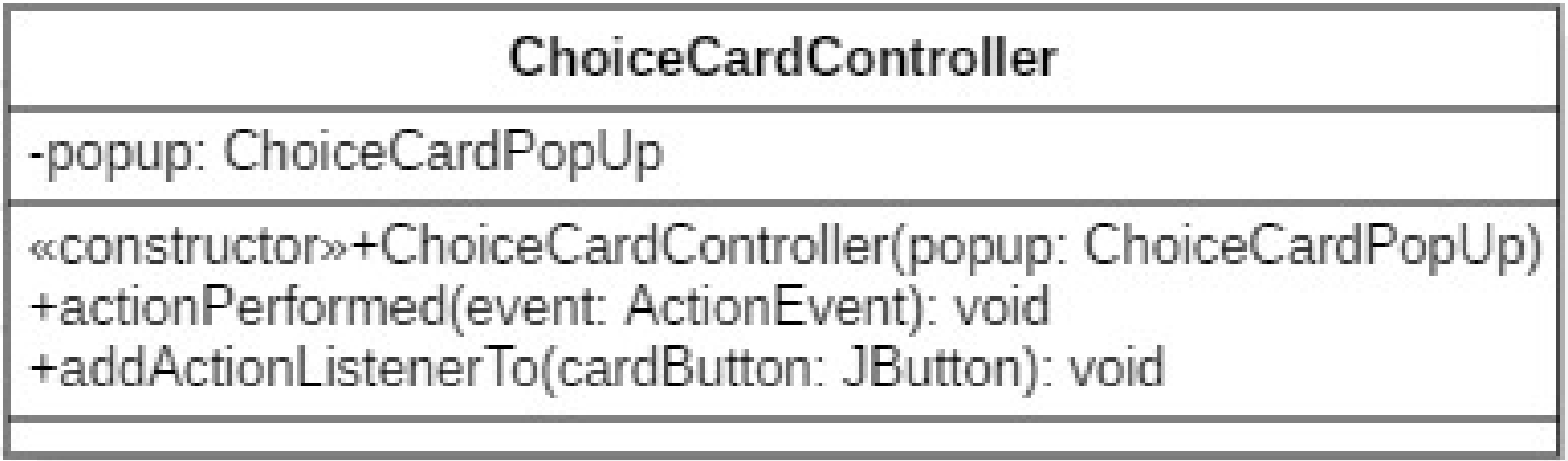

Container
«constructor»+Container()
StartGameContainer
-controller: StartGameController
«constructor»+StartGameContainer(boardview: BoardView)
SelectionPlayerContainer
-bluePlayerRadioButton: JRadioButton
-redPlayerRadioButton: JRadioButton
-controller: SelectionPlayerController
«constructor»+SelectionPlayerContainer(model: BoardModel)
ChoiceContainer
-RED_PLAYER_PLAYING: String = "Le joueur rouge peut jouer" {readOnly}
-BLUE_PLAYER_PLAYING: String = "Le joueur bleu peut jouer" {readOnly}
-whoMustPlay: JLabel
-model: BoardModel
-controller: ChoiceController
«constructor»+ChoiceContainer(model: BoardModel)
+setVersion(save: BoardSave): void
+changeMessage(): void
+updateRound(isRedPlayerPlaying: boolean): void

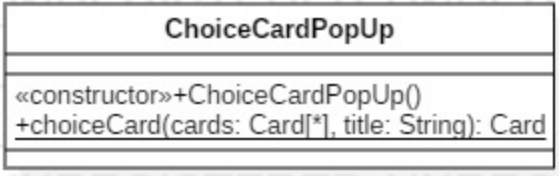


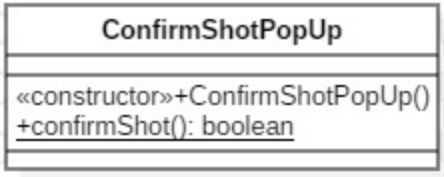





Palette
+DARK_GREEN: Color {readOnly}
+LIGHT_GREEN: Color {readOnly}
+SELECTED_CELL: Color {readOnly}
+HOME_BUTTON_COLOR: Color {readOnly}
+AVAILABLE_MOVEMENT_COLOR: Color {readOnly}
+BORDER_COLOR: Color {readOnly}
Position
-x: int
-y: int
«constructor»+Position(x: int, y: int)
«constructor»+Position(position: Position)
+getX(): int
+getY(): int
+setX(x: int): void
+setY(y: int): void
+toString(): String
+equals(position: Position): boolean
Console
+log(element: T): void
Assert
+isNull(element: T): boolean
+isSet(element: T): boolean
CopyObject
-STORAGE_FILE: String = "./rsc/files/storage.dat" {readOnly}
«constructor»+CopyObject()
+getCopyOf(view: BoardView): BoardView
ModelBoolean
-bool: boolean
«constructor»+ModelBoolean(bool: boolean)
+getBool(): boolean
+setBool(bool: boolean): void
+toggleBool(): void
Distance
+getDistance(p: Position, q: Position): double
+getManhattanDistance(p: Position, q: Position): double
RandomCard
+getRandomCard(hand: Hand): Card

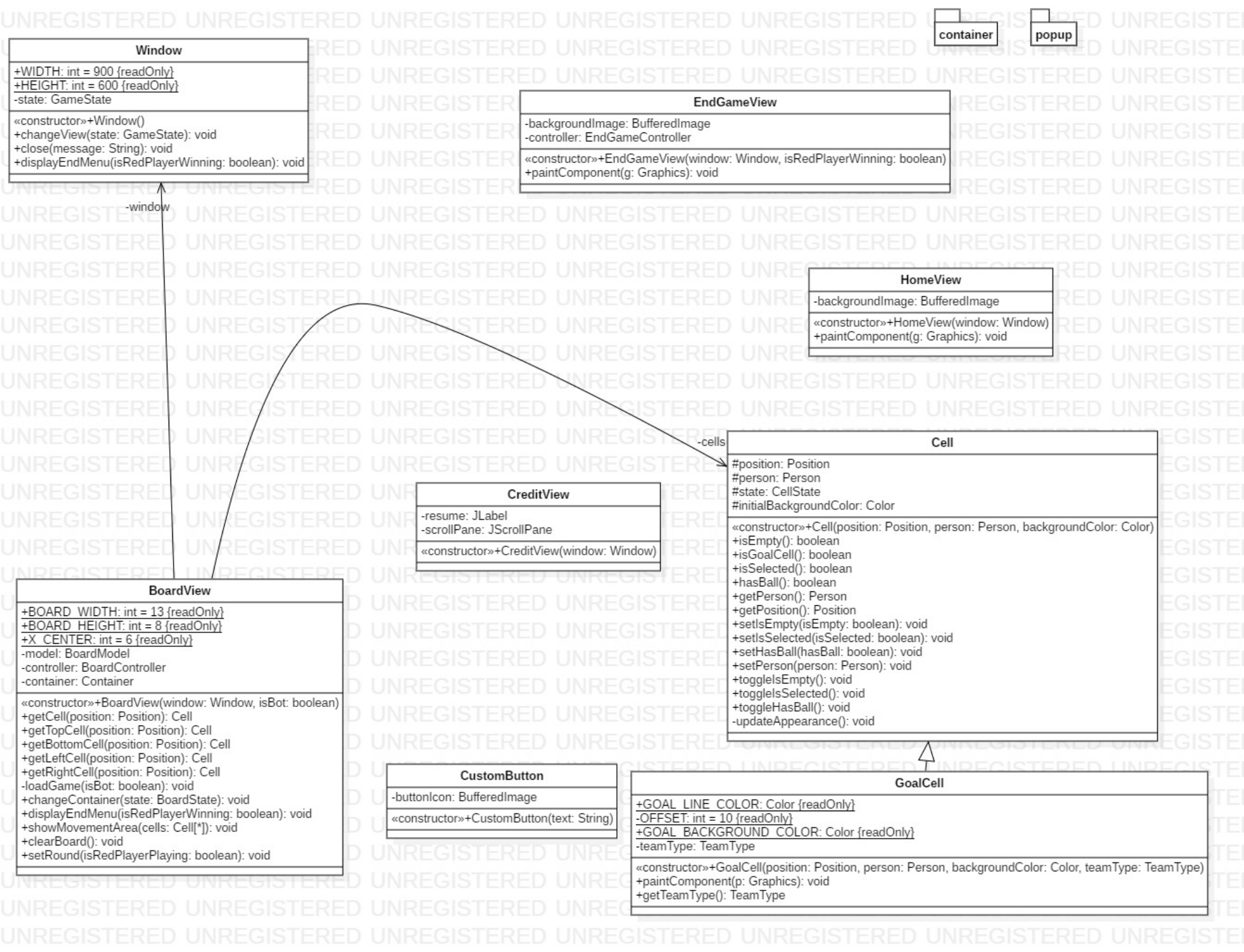

Card
«constructor»+Card(strenght: int)
+getStrenght(): int
+isEqualsTo(card: Card): boolean
+isInferiorTo(card: Card): boolean
+isSuperiorTo(card: Card): boolean
+toString(): String
{collection="ArrayList"}
-cards
*
#selectedCard
PlayerState
«constructor»+PlayerState(canPlay: boolean, isChoosingPerson: boolean, isChoosingCard: boolean)
+canPlay(): boolean
+isChoosingPerson(): boolean
+isChoosingCard(): boolean
+setCanPlay(canPlay: boolean): void
+setIsChoosingPerson(isChoosingPerson: boolean): void
+setIsChoosingCard(isChoosingCard: boolean): void
+toggleCanPlay(): void
+toggleIsChoosingPerson(): void
+toggleisChoosingCard(): void
#playerState
Hand
«constructor»+Hand(player: Player)
+getCards(): Card[*]
+getCard(index: int): Card
+discardCard(card: Card): void
+discardCard(index: int): Card
+refresh(): void
+setCards(cards: Card[*]): void
#hand
-player
Player
«constructor»+Player(persons: Person[*], playerState: PlayerState, teamType: TeamType)
+canPlay(): boolean
+isChoosingPerson(): boolean
+canMakeShot(hasBallPerson: Person): boolean
+isChoosingCard(): boolean
+isBotPlayer(): boolean
+getPersons(): Person[*]
+getSelectedCard(): Card
+getHand(): Hand
+getUsedPersons(): Person[*]
+isRedPlayer(): boolean
+setCanPlay(canPlay: boolean): void
+setIsChoosingPerson(isChoosingPerson: boolean): void
+setIsChoosingCard(isChoosingCard: boolean): void
+setSelectedCard(selectedCard: Card): void
+setPersons(persons: Person[*]): void
+setUsedPersons(usedPersons: Person[*]): void
+toggleCanPlay(): void
+toggleIsChoosingPerson(): void
+toggleisChoosingCard(): void
+discardCard(card: Card): void
+refreshHand(): void
+refreshMovements(): void
+canAddUsedPerson(usedPerson: Person): boolean
-hasUsedPerson(usedPerson: Person): boolean
+addUsedPerson(usedPerson: Person): boolean
+hasPerson(person: Person): boolean
-redPlayer
BoardModel
«constructor»+BoardModel(isBot: boolean, bluePersons: Person[*], redPersons: Person[*], boardView: BoardView, cells: Cell[*,*])
+isRedPlayerPlaying(): boolean
+getModelBoolean(): ModelBoolean
+inProgress(): boolean
+getRedPlayer(): Player
+getBluePlayer(): Player
+getSelectedCell(): Cell
+getState(): BoardState
+getBall(): Ball
+setIsRedPlayerPlaying(isRedPlayerPlaying: boolean): void
+setCellSelected(cell: Cell): void
+selectInitialPlayer(isRedPlayerPlaying: boolean): void
+setVersion(save: BoardSave): void
+setState(state: BoardState): void
+toggleIsRedPlayerPlaying(): void
+getAvailableMoveRange(person: Person): Cell[*]
+getShotRange(person: Person): Cell[*]
-addCellMovement(cells: Cell[*], cell: Cell, remainingMovements: int, person: Person): void
+isAvailablePlatingCell(person: Person): boolean
+getAvailablePassingCell(person: Person): Cell[*]
+getCell(position: Position): Cell
+getTopCell(position: Position): Cell
+getBottomCell(position: Position): Cell
+getLeftCell(position: Position): Cell
+getRightCell(position: Position): Cell
+refreshMovements(): void
+refreshInactivity(): void
FastPerson
«constructor»+FastPerson(position: Position, teamType: TeamType)
«constructor»+FastPerson(person: FastPerson)
HardPerson
«constructor»+HardPerson(position: Position, teamType: TeamType)
«constructor»+HardPerson(person: HardPerson)
BeefyPerson
«constructor»+BeefyPerson(position: Position, teamType: TeamType)
«constructor»+BeefyPerson(person: BeefyPerson)
Ball
«constructor»+Ball(position: Position)
+getPosition(): Position
+getPossessor(): Person
+setPosition(position: Position): void
+setPossessor(possessor: Person): void
-ball
Person
«constructor»+Person(position: Position, teamType: TeamType)
«constructor»+Person(person: Person)
+equals(person: Person): boolean
+canMove(): boolean
+getAgressiveBonus(): int
+getDefensiveBonus(): int
+getMovementRemaining(): int
+getPosition(): Position
+getRepresentation(): String
+getTeamType(): TeamType
+hasBall(): boolean
+isInactive(): boolean
+isRedPerson(): boolean
+setCanMove(canMove: boolean): void
+setRepresentation(representation: String): void
+setHasBall(hasBall: boolean): void
+setIsInactive(isInactive: boolean): void
+setMovementRemaining(movements: int): void
+setPosition(position: Position): void
+toggleCanMove(): void
+toggleIsInactive(): void
+toggleHasBall(): void
+move(position: Position): void
+isCorrectInitialPlacement(cell: Cell): boolean
+isDiagonalPosition(position: Position): boolean
+isValidPosition(position: Position): boolean
+isSameTeamAs(person: Person): boolean
+isSameTeamAs(cell: GoalCell): boolean
+pass(target: Person): void
+scoreTry(): void
+collideBall(ball: Ball): boolean
+collideWithSameTeam(person: Person, position: Position): boolean
+refreshMovements(): void
+refreshAppearance(): void
*
#usedPersons
{collection="ArrayList"}
#possessor
SmartPerson
«constructor»+SmartPerson(position: Position, teamType: TeamType)
«constructor»+SmartPerson(person: SmartPerson)
OrdinaryPerson
«constructor»+OrdinaryPerson(position: Position, teamType: TeamType)
«constructor»+OrdinaryPerson(person: OrdinaryPerson)
CellState
«constructor»+CellState(isEmpty: boolean, isSelected: boolean, hasBall: boolean)
+isEmpty(): boolean
+isSelected(): boolean
+hasBall(): boolean
+setIsEmpty(isEmpty: boolean): void
+setIsSelected(isSelected: boolean): void
+setHasBall(hasBall: boolean): void
+toggleIsEmpty(): void
+toggleIsSelected(): void
+toggleHasBall(): void
#personState
PersonState
«constructor»+PersonState(canMove: boolean, hasBall: boolean, isInactive: boolean)
«constructor»+PersonState(personState: PersonState)
+canMove(): boolean
+hasBall(): boolean
+isDefender(): boolean
+isAttacker(): boolean
+isInactive(): boolean
+setCanMove(canMove: boolean): void
+setHasBall(hasBall: boolean): void
+setisDefender(isDefender: boolean): void
+setIsAttacker(isAttacker: boolean): void
+setIsInactive(isInactive: boolean): void
+toggleCanMove(): void
+toggleIsInactive(): void
+toggleHasBall(): void
+toggleIsDefender(): void
+toggleIsAttacker(): void
+equals(state: PersonState): boolean

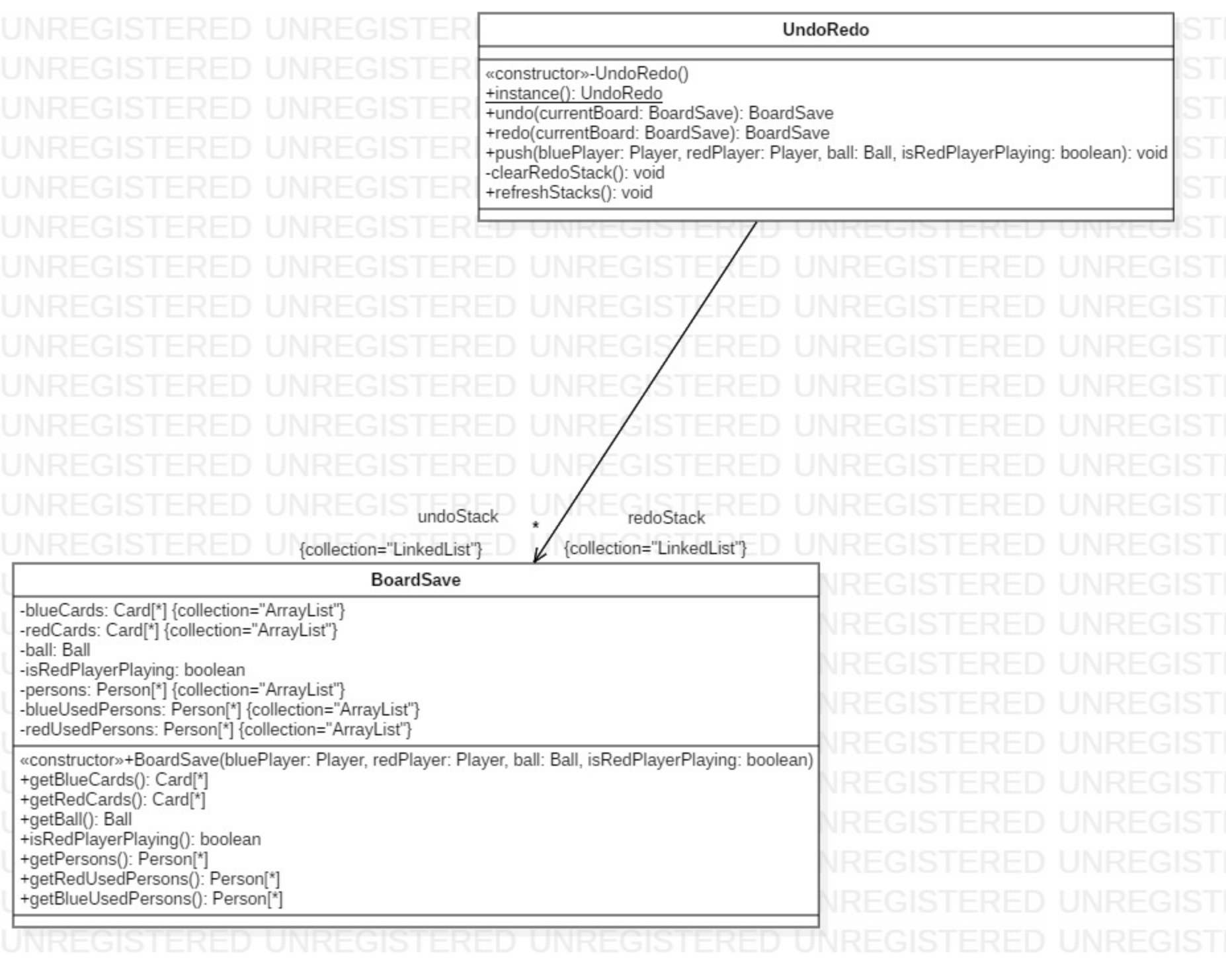

ChoiceCardController
-popup: ChoiceCardPopUp
«constructor»+ChoiceCardController(popup: ChoiceCardPopUp)
+actionPerformed(event: ActionEvent): void
+addActionListenerTo(cardButton: JButton): void




«enumeration»
BotState
DEFENSIVE
AGRESSIVE
NORMAL


container
popup
Window
+WIDTH: int = 900 {readOnly}
+HEIGHT: int = 600 {readOnly}
-state: GameState
«constructor»+Window()
+changeView(state: GameState): void
+close(message: String): void
+displayEndMenu(isRedPlayerWinning: boolean): void
-window
EndGameView
-backgroundImage: BufferedImage
-controller: EndGameController
«constructor»+EndGameView(window: Window, isRedPlayerWinning: boolean)
+paintComponent(g: Graphics): void
HomeView
-backgroundImage: BufferedImage
«constructor»+HomeView(window: Window)
+paintComponent(g: Graphics): void
-cells
Cell
#position: Position
#person: Person
#state: CellState
#initialBackgroundColor: Color
«constructor»+Cell(position: Position, person: Person, backgroundColor: Color)
+isEmpty(): boolean
+isGoalCell(): boolean
+isSelected(): boolean
+hasBall(): boolean
+getPerson(): Person
+getPosition(): Position
+setIsEmpty(isEmpty: boolean): void
+setIsSelected(isSelected: boolean): void
+setHasBall(hasBall: boolean): void
+setPerson(person: Person): void
+toggleIsEmpty(): void
+toggleIsSelected(): void
+toggleHasBall(): void
-updateAppearance(): void
CreditView
-resume: JLabel
-scrollPane: JScrollPane
«constructor»+CreditView(window: Window)
BoardView
+BOARD_WIDTH: int = 13 {readOnly}
+BOARD_HEIGHT: int = 8 {readOnly}
+X_CENTER: int = 6 {readOnly}
-model: BoardModel
-controller: BoardController
-container: Container
«constructor»+BoardView(window: Window, isBot: boolean)
+getCell(position: Position): Cell
+getTopCell(position: Position): Cell
+getBottomCell(position: Position): Cell
+getLeftCell(position: Position): Cell
+getRightCell(position: Position): Cell
-loadGame(isBot: boolean): void
+changeContainer(state: BoardState): void
+displayEndMenu(isRedPlayerWinning: boolean): void
+showMovementArea(cells: Cell[*]): void
+clearBoard(): void
+setRound(isRedPlayerPlaying: boolean): void
CustomButton
-buttonIcon: BufferedImage
«constructor»+CustomButton(text: String)
GoalCell
+GOAL_LINE_COLOR: Color {readOnly}
-OFFSET: int = 10 {readOnly}
+GOAL_BACKGROUND_COLOR: Color {readOnly}
-teamType: TeamType
«constructor»+GoalCell(position: Position, person: Person, backgroundColor: Color, teamType: TeamType)
+paintComponent(p: Graphics): void
+getTeamType(): TeamType


UndoRedo
«constructor»-UndoRedo()
+instance(): UndoRedo
+undo(currentBoard: BoardSave): BoardSave
+redo(currentBoard: BoardSave): BoardSave
+push(bluePlayer: Player, redPlayer: Player, ball: Ball, isRedPlayerPlaying: boolean): void
-clearRedoStack(): void
+refreshStacks(): void
undoStack
{collection="LinkedList"}
*
redoStack
{collection="LinkedList"}
BoardSave
-blueCards: Card[*] {collection="ArrayList"}
-redCards: Card[*] {collection="ArrayList"}
-ball: Ball
-isRedPlayerPlaying: boolean
-persons: Person[*] {collection="ArrayList"}
-blueUsedPersons: Person[*] {collection="ArrayList"}
-redUsedPersons: Person[*] {collection="ArrayList"}
«constructor»+BoardSave(bluePlayer: Player, redPlayer: Player, ball: Ball, isRedPlayerPlaying: boolean)
+getBlueCards(): Card[*]
+getRedCards(): Card[*]
+getBall(): Ball
+isRedPlayerPlaying(): boolean
+getPersons(): Person[*]
+getRedUsedPersons(): Person[*]
+getBlueUsedPersons(): Person[*]