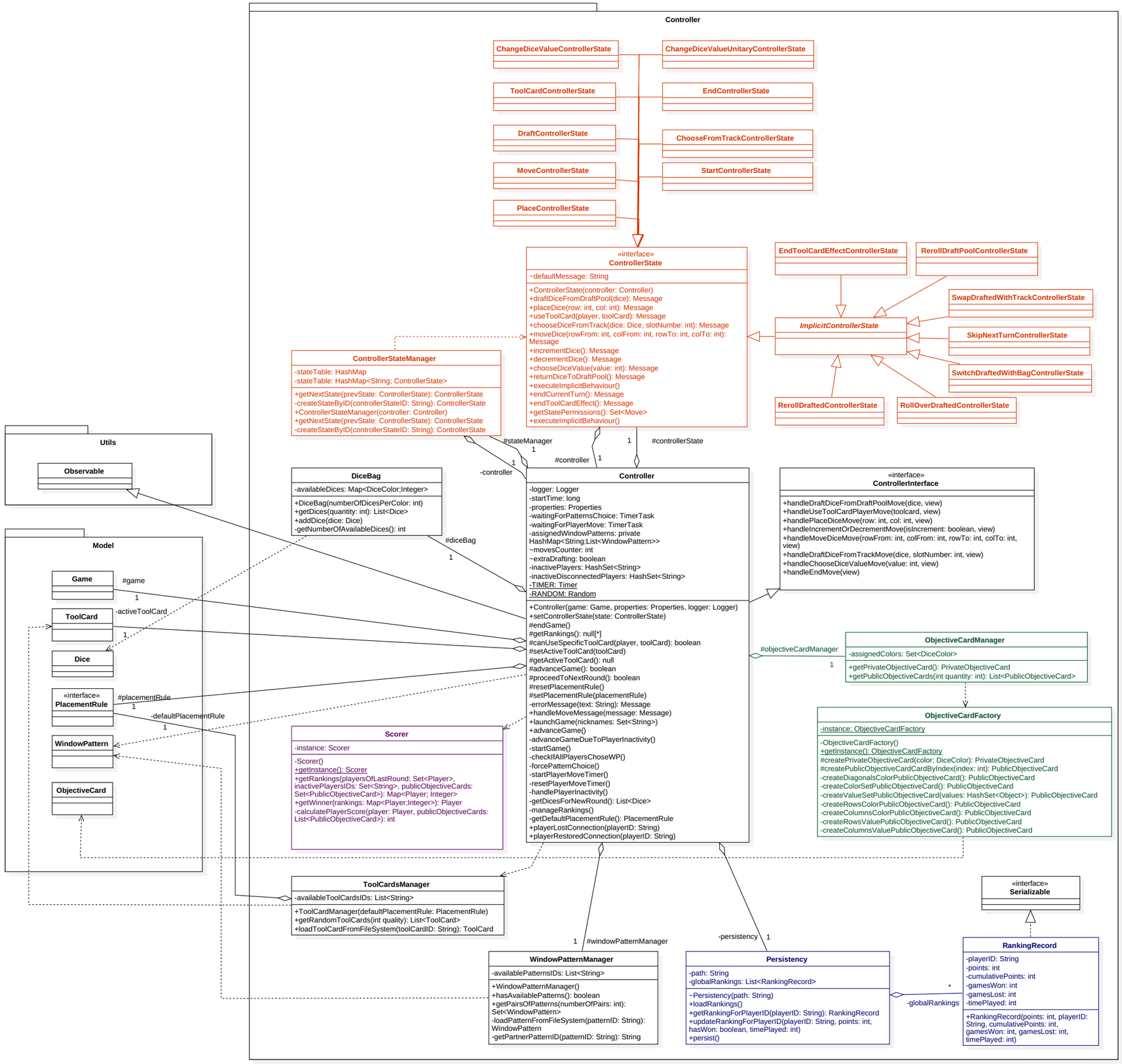
Controller
ChangeDiceValueControllerState
ChangeDiceValueUnitaryControllerState
ToolCardControllerState
EndControllerState
DraftControllerState
ChooseFromTrackControllerState
MoveControllerState
StartControllerState
PlaceControllerState
«interface»
ControllerState
~defaultMessage: String
+ControllerState(controller: Controller)
+draftDiceFromDraftPool(dice): Message
+placeDice(row: int, col: int): Message
+useToolCard(player, toolCard): Message
+chooseDiceFromTrack(dice: Dice, slotNumbe: int): Message
+moveDice(rowFrom: int, colFrom: int, rowTo: int, colTo: int): Message
+incrementDice(): Message
+decrementDice(): Message
+chooseDiceValue(value: int): Message
+returnDiceToDraftPool(): Message
+executeImplicitBehaviour()
+endCurrentTurn(): Message
+endToolCardEffect(): Message
+getStatePermissions(): Set<Move>
+executeImplicitBehaviour()
EndToolCardEffectControllerState
RerollDraftPoolControllerState
SwapDraftedWithTrackControllerState
ImplicitControllerState
SkipNextTurnControllerState
SwitchDraftedWithBagControllerState
RerollDraftedControllerState
RollOverDraftedControllerState
ControllerStateManager
-stateTable: HashMap
-stateTable: HashMap<String; ControllerState>
+getNextState(prevState: ControllerState): ControllerState
-createStateByID(controllerStateID: String): ControllerState
+ControllerStateManager(controller: Controller)
+getNextState(prevState: ControllerState): ControllerState
-createStateByID(controllerStateID: String): ControllerState
#stateManager
1
1
#controller
1
1
#controllerState
-controller
Utils
Observable
DiceBag
-availableDices: Map<DiceColor;Integer>
+DiceBag(numberOfDicesPerColor: int)
+getDices(quantity: int): List<Dice>
+addDice(dice: Dice)
-getNumberOfAvailableDices(): int
Controller
-logger: Logger
-startTime: long
-properties: Properties
-waitingForPatternsChoice: TimerTask
-waitingForPlayerMove: TimerTask
-assignedWindowPatterns: private HashMap<String;List<WindowPattern>>
~movesCounter: int
~extraDrafting: boolean
-inactivePlayers: HashSet<String>
-inactiveDisconnectedPlayers: HashSet<String>
-TIMER: Timer
-RANDOM: Random
+Controller(game: Game, properties: Properties, logger: Logger)
+setControllerState(state: ControllerState)
#endGame()
#getRankings(): null[*]
#canUseSpecificToolCard(player, toolCard): boolean
#setActiveToolCard(toolCard)
#getActiveToolCard(): null
#advanceGame(): boolean
#proceedToNextRound(): boolean
#resetPlacementRule()
#setPlacementRule(placementRule)
-errorMessage(text: String): Message
+handleMoveMessage(message: Message)
+launchGame(nicknames: Set<String>)
+advanceGame()
-advanceGameDueToPlayerInactivity()
-startGame()
-checkIfAllPlayersChoseWP()
-forcePatternChoice()
-startPlayerMoveTimer()
-resetPlayerMoveTimer()
-handlePlayerInactivity()
-getDicesForNewRound(): List<Dice>
-manageRankings()
-getDefaultPlacementRule(): PlacementRule
+playerLostConnection(playerID: String)
+playerRestoredConnection(playerID: String)
«interface»
ControllerInterface
+handleDraftDiceFromDraftPoolMove(dice, view)
+handleUseToolCardPlayerMove(toolcard, view)
+handlePlaceDiceMove(row: int, col: int, view)
+handleIncrementOrDecrementMove(isIncrement: boolean, view)
+handleMoveDiceMove(rowFrom: int, colFrom: int, rowTo: int, colTo: int, view)
+handleDraftDiceFromTrackMove(dice, slotNumber: int, view)
+handleChooseDiceValueMove(value: int, view)
+handleEndMove(view)
#diceBag
1
Model
Game
#game
1
ToolCard
-activeToolCard
1
Dice
«interface»
PlacementRule
#placementRule
1
-defaultPlacementRule
1
WindowPattern
ObjectiveCard
#objectiveCardManager
1
ObjectiveCardManager
-assignedColors: Set<DiceColor>
+getPrivateObjectiveCard(): PrivateObjectiveCard
+getPublicObjectiveCards(int quantity: int): List<PublicObjectiveCard>
ObjectiveCardFactory
-instance: ObjectiveCardFactory
-ObjectiveCardFactory()
+getInstance(): ObjectiveCardFactory
#createPrivateObjectiveCard(color: DiceColor): PrivateObjectiveCard
#createPublicObjectiveCardCardByIndex(index: int): PublicObjectiveCard
-createDiagonalsColorPublicObjectiveCard(): PublicObjectiveCard
-createColorSetPublicObjectiveCard(): PublicObjectiveCard
-createValueSetPublicObjectiveCard(values: HashSet<Object>): PublicObjectiveCard
-createRowsColorPublicObjectiveCard(): PublicObjectiveCard
-createColumnsColorPublicObjectiveCard(): PublicObjectiveCard
-createRowsValuePublicObjectiveCard(): PublicObjectiveCard
-createColumnsValuePublicObjectiveCard(): PublicObjectiveCard
Scorer
-instance: Scorer
-Scorer()
+getInstance(): Scorer
+getRankings(playersOfLastRound: Set<Player>, inactivePlayersIDs: Set<String>, publicObjectiveCards: Set<PublicObjectiveCard>): Map<Player; Integer>
+getWinner(rankings: Map<Player;Integer>): Player
-calculatePlayerScore(player: Player, publicObjectiveCards: List<PublicObjectiveCard>): int
ToolCardsManager
-availableToolCardsIDs: List<String>
+ToolCardManager(defaultPlacementRule: PlacementRule)
+getRandomToolCards(int quality): List<ToolCard>
+loadToolCardFromFileSystem(toolCardID: String): ToolCard
«interface»
Serializable
1
#windowPatternManager
-persistency
1
WindowPatternManager
-availablePatternsIDs: List<String>
+WindowPatternManager()
+hasAvailablePatterns(): boolean
+getPairsOfPatterns(numberOfPairs: int): Set<WindowPattern>
-loadPatternFromFileSystem(patternID: String): WindowPattern
-getPartnerPatternID(patternID: String): String
Persistency
-path: String
-globalRankings: List<RankingRecord>
~Persistency(path: String)
+loadRankings()
+getRankingForPlayerID(playerID: String): RankingRecord
+updateRankingForPlayerID(playerID: String, points: int, hasWon: boolean, timePlayed: int)
+persist()
*
-globalRankings
RankingRecord
-playerID: String
-points: int
-cumulativePoints: int
-gamesWon: int
-gamesLost: int
-timePlayed: int
+RankingRecord(points: int, playerID: String, cumulativePoints: int, gamesWon: int, gamesLost: int, timePlayed: int)