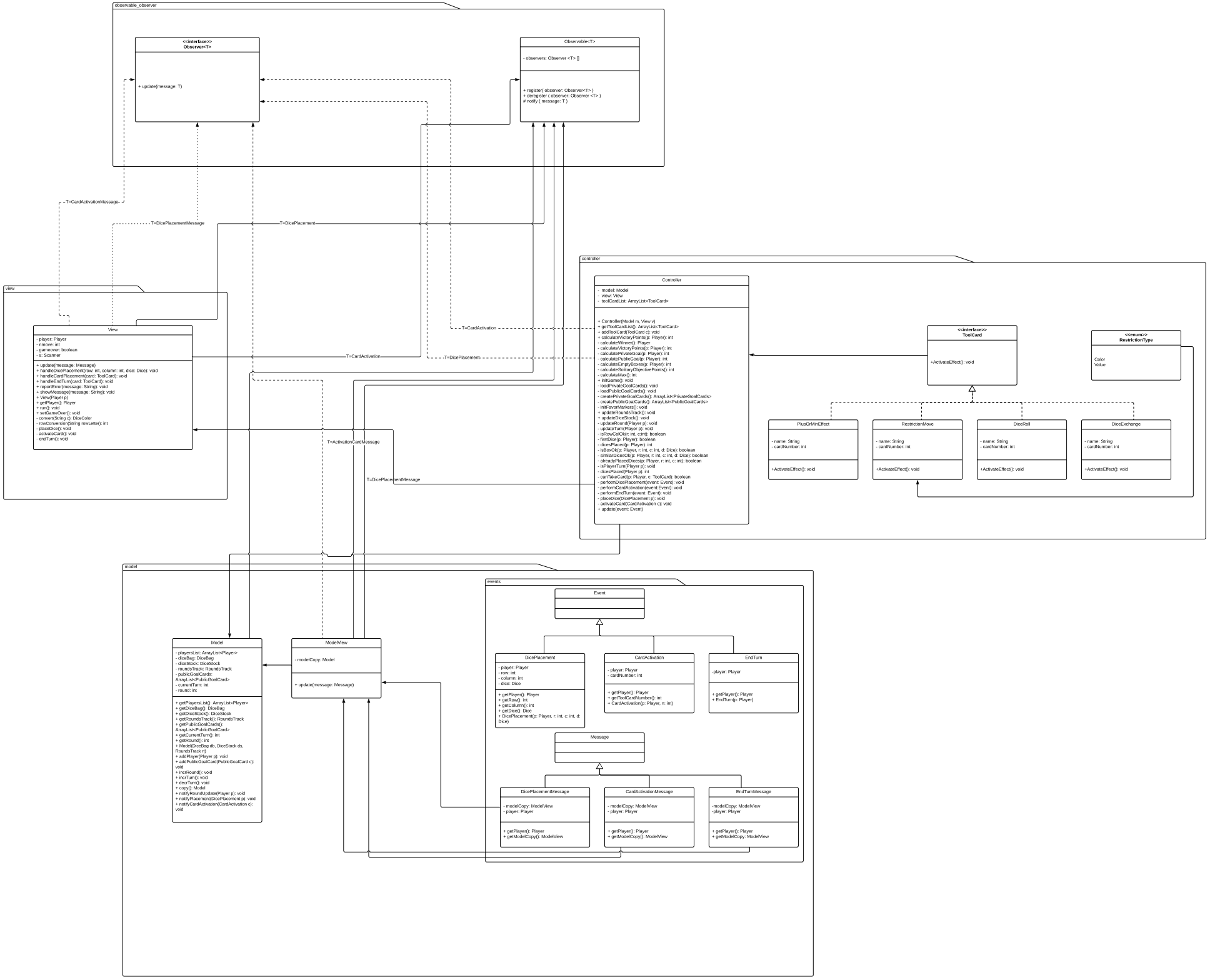
observable_observer
<<interface>>
Observer<T>
+ update(message: T)
Observable<T>
- observers: Observer <T> []
+ register(observer: Observer<T>)
+ deregister (observer: Observer <T>)
# notify (message: T)
T=CardActivationMessage
T=DicePlacementMessage
T=DicePlacement
view
View
- player: Player
- rmove: int
- gameover: boolean
- s: Scanner
+ update(message: Message)
+ handleDicePlacement(row: int, column: int, dice: Dice): void
+ handleCardPlacement(card: ToolCard): void
+ handleEndTurn(card: ToolCard): void
+ reportError(message: String): void
+ showMessage(message: String): void
+ View(Player p)
+ getPlayer(): Player
+ run(): void
+ setGameOver(): void
- convert(String c): DiceColor
- rowConversion(String rowLetter): int
- placeDice(): void
- activateCard(): void
- endTurn(): void
T=CardActivation
T=DicePlacement
T=CardActivation
T=DicePlacement
T=ActivationCardMessage
T=DicePlacementMessage
controller
Controller
- model: Model
- view: View
- toolCardList: ArrayList<ToolCard>
+ Controller(Model m, View v)
+ getToolCardList(): ArrayList<ToolCard>
+ addToolCard(ToolCard c): void
+ calculateVictoryPoints(p: Player): int
- calculateWinner(): Player
- calculateVictoryPoints(p: Player): int
- calculatePrivateGoal(p: Player): int
- calculatePublicGoal(p: Player): int
- calculateEmptyBoxes(p: Player): int
- calculateSolitaryObjectivePoints(): int
- calculateMax(): int
+ initGame(): void
- loadPrivateGoalCards(): void
- loadPublicGoalCards(): void
- createPrivateGoalCards(): ArrayList<PrivateGoalCards>
- createPublicGoalCards(): ArrayList<PublicGoalCards>
- initFavorMarkers(): void
+ updateRoundsTrack(): void
+ updateDiceStock(): void
- updateRound(Player p): void
- updateTurn(Player p): void
- isRowColOk(r: int, c:int): boolean
- firstDice(p: Player): boolean
- dicesPlaced(p: Player): int
- isBoxOk(p: Player, r: int, c: int, d: Dice): boolean
- similarDicesOk(p: Player, r: int, c: int, d: Dice): boolean
- alreadyPlacedDices(p: Player, r: int, c: int): boolean
- isPlayerTurn(Player p): void
- dicesPlaced(Player p): int
- canTakeCard(p: Player, c: ToolCard): boolean
- performDicePlacement(event: Event): void
- performCardActivation(event:Event): void
- performEndTurn(event: Event): void
- placeDice(DicePlacement p): void
- activateCard(CardActivation c): void
+ update(event: Event)
<<interface>>
ToolCard
+ActivateEffect(): void
<<enum>>
RestrictionType
Color
Value
PlusOrMinEffect
- name: String
- cardNumber: int
+ActivateEffect(): void
RestrictionMove
- name: String
- cardNumber: int
+ActivateEffect(): void
DiceRoll
- name: String
- cardNumber: int
+ActivateEffect(): void
DiceExchange
- name: String
- cardNumber: int
+ActivateEffect(): void
model
Model
- playersList: ArrayList<Player>
- diceBag: DiceBag
- diceStock: DiceStock
- roundsTrack: RoundsTrack
- publicGoalCards: ArrayList<PublicGoalCard>
- currentTurn: int
- round: int
+ getPlayersList(): ArrayList<Player>
+ getDiceBag(): DiceBag
+ getDiceStock(): DiceStock
+ getRoundsTrack(): RoundsTrack
+ getPublicGoalCards(): ArrayList<PublicGoalCard>
+ getCurrentTurn(): int
+ getRound(): int
+ Model(DiceBag db, DiceStock ds, RoundsTrack rt)
+ addPlayer(Player p): void
+ addPublicGoalCard(PublicGoalCard c): void
+ incrRound(): void
+ incrTurn(): void
+ decrTurn(): void
+ copy(): Model
+ notifyRoundUpdate(Player p): void
+ notifyPlacement(DicePlacement p): void
+ notifyCardActivation(CardActivation c): void
ModelView
- modelCopy: Model
+ update(message: Message)
events
Event
DicePlacement
- player: Player
- row: int
- column: int
- dice: Dice
+ getPlayer(): Player
+ getRow(): int
+ getColumn(): int
+ getDice(): Dice
+ DicePlacement(p: Player, r: int, c: int, d: Dice)
CardActivation
- player: Player
- cardNumber: int
+ getPlayer(): Player
+ getToolCardNumber(): int
+ CardActivation(p: Player, n: int)
EndTurn
-player: Player
+ getPlayer(): Player
+ EndTurn(p: Player)
Message
DicePlacementMessage
- modelCopy: ModelView
- player: Player
+ getPlayer(): Player
+ getModelCopy(): ModelView
CardActivationMessage
- modelCopy: ModelView
- player: Player
+ getPlayer(): Player
+ getModelCopy(): ModelView
EndTurnMessage
-modelCopy: ModelView
-player: Player
+ getPlayer(): Player
+ getModelCopy: ModelView