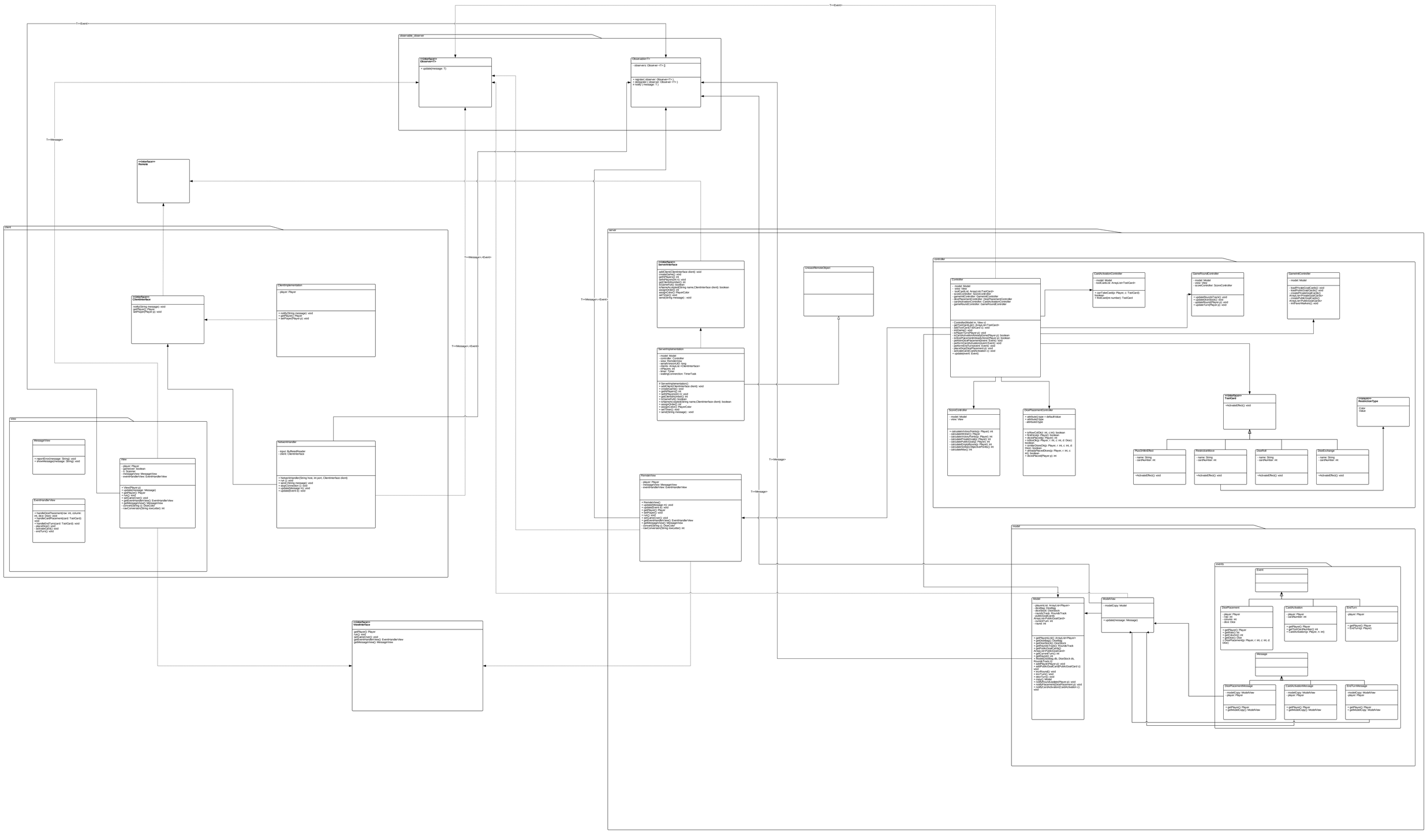
T=Event>
observable_observer
<<InterfaceObserver>>
Observer<T>
+ update(message: T)
Observable<T>
- observers: Observer <T> []
+ registerObserver(observer: Observer<T>)
+ deregisterObserver(observer: Observer <T>)
# notify (message: T)
T=Message>
<<interface>>
Remote
client
<<InterfaceObject>>
ClientInterface
+notify(String message): void
getPlayer() Player
setPlayer(Player p): void
ClientImplementation
- player: Player
+ notify(String message): void
+ getPlayer() Player
+ setPlayer(Player p): void
T=Message>,<Event>
View
MessageView
+ reportError(message: String): void
+ showMessage(message: String): void
View
- player: Player
- gameover: boolean
- model: Model
- messageView: MessageView
- eventHanderView: EvetHanderView
+ ViewPlayer(p)
+ update(message: Message)
+ getPlayer(): Player
+ run(): void
+ setGameOver(): void
+ setEventHanderView(): EventHanderView
+ getMessageView(): MessageView
- convertString (c: DiceColor
- rowConversion(String rowLetter): int
EventHanderView
+ handleDicePlacement(row: int, column: int, dice Dice): void
+ handleCardPlacement(card: TraitCard): void
+ handleEndTurn(card: TraitCard): void
- placeDice(): void
- activateCard(): void
- endTurn(): void
NetworkHandler
- input: BufferedReader
- client: ClientInterface
+ NetworkHandler(String host, int port, ClientInterface client)
+ run(): void
+ send (String message): void
+ stopConnection () void
+ setGameOver(): void
+ update(Event e): void
server
<<InterfaceObject>>
ServerInterface
+addClient(ClientInterface client): void
startGame(): void
getNPlayers() int
setNPlayers(int n): void
getClientsNumber(): int
isGameFull(): boolean
checkNickname(String name,ClientInterface client): boolean
assignOrder() int
assignColor() PlayerColor
setTimer() void
send(String message) : void
ServerImplementation
- model: Model
- controller: Controller
- view: RemoteView
- serialVersionUID: long
- clients: HashMap <ClientInterface>
- nPlayers: int
- nClients: int
- waitingConnection: TimerTask
+ ServerImplementation()
+ addClient(ClientInterface client): void
+ createGame() void
+ startGame(): void
+ setNPlayers(int n): void
+ getClientsNumber(): int
+ isGameFull(): boolean
+ checkNickname(String name,ClientInterface client): boolean
+ assignOrder() int
+ assignColor() PlayerColor
+ setTimer() void
+ send(String message) : void
RemoteView
- player: Player
- messageView: MessageView
- eventHanderView: EvetHanderView
+ RemoteView()
+ setNPlayers(int n): void
+ update(Event e): void
+ getPlayer() Player
+ setPlayer() void
+ run() void
+ setGameOver() void
+ setEventHanderView(): EvetHanderView
+ getMessageView(): MessageView
- convertString (c: DiceColor
- rowConversion(String rowLetter): int
UnicastRemoteObject
controller
Controller
- model: Model
- view: View
- toolCardList: ArrayList<TraitCard>
- publicGoalCards: ArrayList
- gameInitController: GameInitController
- dicePlacementController: DicePlacementController
- cardActivationController: CardActivationController
- gameRoundController: GameRoundController
- controllerModel m, View v)
- getToolCardList() ArrayList<TraitCard>
- activateCard (c: int) void
- endGame() void
- isPlayerTurnPlayer (p) void
- isDicePlacementAllowed(DicePlayer p): boolean
- isCardPlacementAllowed(CardPlayer p): boolean
- performDicePlacement(event: Event): void
- performCardPlacement(event: Event): void
- performEndTurn(event: Event): void
- placeDice(DicePlacement p) void
- activateCard(CardActivation c) void
- update(event: Event)
CardActivationController
- model: Model
- toolCardList: ArrayList<TraitCard>
+ canTakeCard(p: Player, c: TraitCard): boolean
+ findCard(int number): TraitCard
GameRoundController
- model: Model
- view: View
- scoreController: ScoreController
+ updateRoundTrack() void
+ updateDiceStock() void
+ updatePublicGoalCards() void
+ updateTurnPlayer (p): void
GameInitController
- model: Model
- loadPrivateGoalCards() void
- loadPublicGoalCards() void
- createPrivateGoalCards()
ArrayList<PrivateGoalCard>
- createPublicGoalCards()
ArrayList<PublicGoalCard>
- initRoundMarkers() void
ScoreController
- model: Model
- calculateVictoryPoints(p: Player): int
- calculateWinner(): Player
- calculateVictoryPoints(p: Player): int
- calculateVictoryPoints(): void
- calculatePublicGoal(p: Player): int
- calculateEmptyBlocks(p: Player): int
- calculateMinimumDiceOnBoard(p): int
- calculateMax() int
DicePlacementController
- attribute1 type = defaultValue
- attribute2 type
- attribute3 type
+ isRowColOk(r: int, c:int): boolean
+ firstDice(p: Player): boolean
+ diceInPlace(p: Player): int
+ isDiceOk(p: Player, r: int, c: int, d: Dice): boolean
+ similarDicesOnDep(p: Player, r: int, c: int, d: Dice): int
+ alreadyPlaceDiceInDep(p: Player, r: int, c: int): boolean
+ diceInPlacedPlayer (p): int
<<InterfaceObject>>
ToolCard
+ActivateEffect() void
<<Enumeration>>
RestrictionType
Color
Value
PlusOneMoveEffect
- name: String
- cardNumber: int
+ActivateEffect() void
RestrictedMove
- name: String
- cardNumber: int
+ActivateEffect() void
DiceRoll
- name: String
- cardNumber: int
+ActivateEffect() void
DiceExchange
- name: String
- cardNumber: int
+ActivateEffect() void
T=Message>,<Event>
T=Message>
T=Message>
<<InterfaceObject>>
ViewInterface
getPlayer() Player
run() void
setGameOver() void
getEventHanderView() EvetHanderView
getMessageView() MessageView
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
+ setDiceBag(bag: BL DiceStock ds, RoundsTrack r)
+ addPlayer(Player p) void
+ addPublicGoalCard(PublicGoalCard c) void
+ incRound() void
+ setTurn() void
+ decTurn() void
+ copy() Model
+ notifyPlacedDice(Player p): void
+ notifyPlacementDicePlacement (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