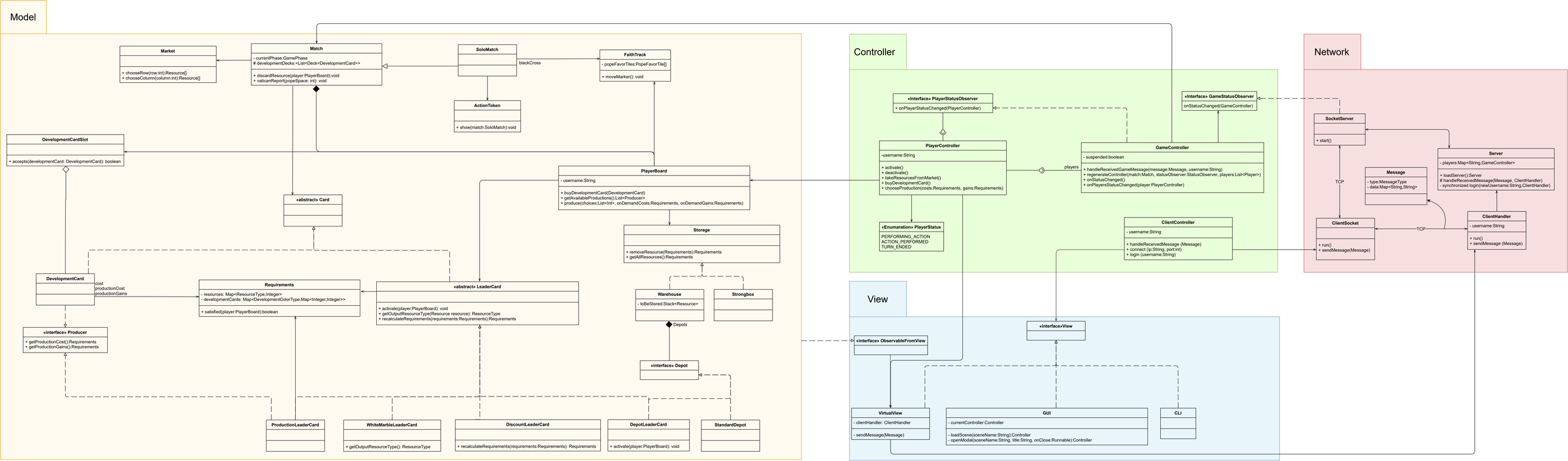
Model
Market
+ chooseRow(row:int):Resource[]
+ chooseColumn(column:int):Resource[]
Match
- currentPhase:GamePhase
developmentDecks:<List<Deck<DevelopmentCard>>
+ discardResource(player:PlayerBoard):void
+ vaticanReport(popeSpace: int): void
SoloMatch
blackCross
FaithTrack
- popeFavorTiles:PopeFavorTile[]
+ moveMarker(): void
ActionToken
+ show(match:SoloMatch):void
DevelopmentCardSlot
+ accepts(developmentCard: DevelopmentCard): boolean
«abstract» Card
PlayerBoard
- username:String
+ buyDevelopmentCard(DevelopmentCard)
+ getAvailableProductions():List<Producer>
+ produce(choices:List<Int>, onDemandCosts:Requirements, onDemandGains:Requirements)
Storage
+ removeResource(Requirements):Requirements
+ getAllResources():Requirements
DevelopmentCard
cost
productionCost
productionGains
Requirements
- resources: Map<ResourceType,Integer>
- developmentCards: Map<DevelopmentColorType,Map<Integer,Integer>>
+ satisfied(player:PlayerBoard):boolean
«abstract» LeaderCard
+ activate(player:PlayerBoard): void
+ getOutputResourceType(Resource resource): ResourceType
+ recalculateRequirements(requirements:Requirements):Requirements
Warehouse
- toBeStored:Stack<Resource>
Strongbox
Depots
«interface» Producer
+ getProductionCost():Requirements
+ getProductionGains():Requirements
«interface» Depot
ProductionLeaderCard
WhiteMarbleLeaderCard
+ getOutputResourceType(): ResourceType
DiscountLeaderCard
+ recalculateRequirements(requirements:Requirements): Requirements
DepotLeaderCard
+ activate(player:PlayerBoard): void
StandardDepot
Controller
«Interface» PlayerStatusObserver
+ onPlayerStatusChanged(PlayerController)
«Interface» GameStatusObserver
onStatusChanged(GameController)
PlayerController
-username:String
+ activate()
+ deactivate()
+ takeResourcesFromMarket()
+ buyDevelopmentCard()
+ chooseProduction(costs:Requirements, gains:Requirements)
GameController
- suspended:boolean
+ handleReceivedGameMessage(message:Message, username:String)
+ regenerateController(match:Match, statusObserver:StatusObserver, players:List<Player>)
+ onStatusChanged()
+ onPlayersStatusChanged(player:PlayerController)
players
«Enumaration» PlayerStatus
PERFORMING_ACTION
ACTION_PERFORMED
TURN_ENDED
ClientController
- username:String
+ handleReceivedMessage (Message)
+ connect (ip:String, port:int)
+ login (username:String)
View
«interface» ObservableFromView
«interface»View
VirtualView
- clientHandler: ClientHandler
- sendMessage(Message)
GUI
- currentController:Controller
- loadScene(sceneName:String):Controller
- openModal(sceneName:String, title:String, onClose:Runnable):Controller
CLI
Network
SocketServer
+ start()
Server
- players:Map<String,GameController>
+ loadServer():Server
# handleReceivedMessage(Message, ClientHandler)
- synchronized login(newUsername:String,ClientHandler)
Message
- type:MessageType
- data:Map<String,String>
TCP
ClientSocket
+ run()
+ sendMessage(Message)
TCP
ClientHandler
- username:String
+ run()
+ sendMessage (Message)