UNREGISTERED
it.polimi.ingsw.Main

it.polimi.ingsw.model

Player
-hasInkwell: boolean {readOnly}
-nickname: String {readOnly}
-isHisTurn: boolean
-leaderCards: LeaderCard[2]
-strategies[3]: Strategy
-game: Game {readOnly}
-personalBoard: PersonalBoard {readOnly}
-victoryPoints: int
-faithPoints: int
+insertMarble(): Resource
+buyDevCard()
+basicProduction()
+extraProduction()
+activateProduction(cards: DevelopmentCard[7])
+useLeader(card: LeaderCard, choice: LeaderAction)
+chooseLeaderCard(leader: LeaderCard[4]): LeaderCard[2]
+addStrategy(newStrategy: Strategy)
+getPlayerFaith(): int
+addPlayerFaith(amount: int)
+getVictoryPoints(): int
+addVictoryPoints(amount: int)
+getNickname(): String
+discardRes(resource: Resource)
+reorderDepot(fromLayer: int, toLayer: int, amount: int)

«enumeration»
LeaderAction
DISCARD
USE_ABILITY

«enumeration»
AbilityChoice
STANDARD
FIRST
SECOND
BOTH

FaithTrack
-popeCards: List<PopeFavorCards>
+checkPopeTile()
+vaticanReport()
+inProximityOfVaticanReport()
+vaticanReportOthers()
+getPopeFavorBySection()

- Discount is optional
- Production is optional
- If two MarblesDecorator are present, player has to choose one

Player's takeResource() and buyDevCard() just call the correct method of the Strategy class

DevelopmentSlot
-cards: Deque<DevelopmentCard>
-isProductionActive: boolean
+getCards(): List<DevelopmentCard>
+addCard(card: DevelopmentCard)
+numberOfElements(): int
+getLevelSlot(): int
+getTopCard(): DevelopmentCard
+isProductionActive(): boolean
+setProductionActive(productionActive: boolean)
+canBeAdded(card: DevelopmentCard): boolean

PersonalBoard
-devSlots: DevelopmentSlot[3]
-depot: Depot
-strongbox: Strongbox
-faithTrack: FaithTrack
+basicProduction(input: ResourceType, output: ResourceType)
+updateFaithTrack(): List<Player>
+getAllCards(): List<DevelopmentCards>

DevCardStrategy
-discounts[2]: Discount
-size: int
+addAbility(ability: Discount)
+buyDevCard(player: Player, level: int, color: CardColor)

ProductionStrategy
-productions[2]: ExtraProduction
-size: int
+addAbility(ability: ExtraProduction)
+extraProduction(player: Player, choice: AbilityChoice)

ResourceStrategy
-resTypes[2]: ChangeWhiteMarbles
-size: int
+addAbility(ability: ChangeWhiteMarbles)
+insertMarble(player: Player, choice: AbilityChoice): Resource

ClientError
-exception: Throwable
+setException(exception: Throwable)
+getError(): String

BasicStrategies
+insertMarble(player: Player, position: int): Resource
+buyDevCards(player: Player, level: int, color: CardColor)
+basicProduction(player: Player, input: ResourceType, output: ResourceType)
+checkAndCompare(depot: Depot, strongbox: Strongbox, expectedRes: Resource)

TempResource
-toHandle: Resource
+setToHandle(toHandle: Resource)
+getToHandle(): Resource
+isEmpty(): boolean

Strongbox
-resource: Resource
+queryAllRes(): Map<Resource Type - Integer>
+addResources(resource: Resource)
+retrieveResources(resource: Resource)
+moveToDepot(resType: ResourceType, depot: Depot, layer: int)

DepotLayer
-resource: ResourceType
-amount: int
+MAX: int {readOnly}
+isEmpty(): boolean
+resetLayer()
+getAmount(): int
+setAmount(amount: int)
+getResource(): ResourceType
+setResAndAmount(res: ResourceType, amount: int)
+canInsert(res: ResourceType, amount: int): boolean

Game
-playersList: List<Player>
-playerAmount: int
-activePlayer: player
-market: Market
+nextTurn()
+computeFinalScore()
+checkEndGame()
+lastTurn()

«interface»
Layer
+isEmpty(): Boolean
+getResource(): ResourceType
+getAmount(): int
+setAmount(amount: int)
+resetLayer()
+setResAndAmount(res: ResourceType, amount: int)
+canInsert(res: ResourceType, amount: int): boolean

Depot
-mapping: Map<Integer - DepotLayer>
+getLayer(layerNumber: int)
+modifyLayer(layerNumber: int)
+moveResources(amount: int, fromLayer: int, toLayer: int)
-moveToLayer(amount: int, fromLayer: DepotLayer, toLayer: DepotLayer)
-swapLayers(fromLayer: DepotLayer, toLayer: DepotLayer)
+queryAllRes(): Resource
+retrieveRes(res: Resource)
+discardRes(layer: DepotLayer)
#getMapCopy(): Map<Integer - DepotLayer>
#addToMap(key: int, depotLayer: DepotLayer)
+canInsert(settings: List<DepotSetting>): boolean

MultiGame
-finalDeckLeader: List<LeaderCard>
+lastTurn: boolean
+nextTurn()
+computeFinalScore()
+checkEndGame()
+lastTurn()
-startGame()
-giveInkwell()
-giveResources()
-giveLeaderCards(player: Player)
+computeRanks(): List<Player>

SoloGame
-soloTokens: List<SoloActionToken>
-blackCrossPosition: int
+nextTurn()
+computeFinalScore()
+checkEndGame()
+lastTurn()
-startGame()
-initSoloTokens()
+shuffleTokens()
+moveBlackCross(space: int)

«enumeration»
CardColor
GREEN
BLUE
YELLOW
PURPLE
+getNumberColumn(): int

Card
-victoryPoints: int
+getVictoryPoint(): int

PopeFavorCard
-faceUp: boolean
-reported
-VaticanReportSection section
+assingSection(): int
+isReported(): boolean
+isFaceUP(): boolean
+setReportedAndFlip()
+getSection()

DevelopmentCard
-cost: Resource
-type: CardColor
-level: int
+canBeBought(playerResource: Resource): boolean
+getLevel(): int
+getType(): CardColor
+getCost(): Resource
+getProdPower(): ProductionPower
+equals(other: Object): boolean

LeaderCard
-specialAbility: LeaderAbility
-isActive: boolean
+getSpecialAbility(): LeaderAbility
+canBeActivated(playerResource: Resource): boolean
+canBeActivated(playerCards: List<DevelopmentCard>): boolean

ExtraLayer
-resource: ResourceType {readOnly}
-amount: int
+MAX: int
+isEmpty(): boolean
+resetLayer()
+getAmount(): int
+setAmount(amount: int)
+getResource(): ResourceType
+setResAndAmount(res: ResourceType, amount: int)
+canInsert(res: ResourceType, amount: int): boolean

Resource
-map: Map<ResourceType - Integer>
+getValue(key: ResourceType): int
+addResource(key: ResourceType, value: int)
+modifyValue(key: ResourceType, value: int)
+getAllRes(): Map<ResourceType - Integer>
+compare(that: Resource): boolean
+sum(that: Resource): Resource
+equals(that: Resource): boolean

ProductionPower
-input: Resource
-output: Resource
+getinput(): Resource
+getOutput(): Resource

ResLeaderCard
-cost: Resource
+getCost(): Resource
+canBeActivated(playerResource: Resource): boolean

DevLeaderCard
-requires: List<LeaderDevCost>
+canBeActivated(playerCards: List<DevelopmentCard>): boolean

LeaderDevCost
-color: CardColor
-level: int
-amount: int
+getColor(): CardColor
+getLevel(): int
+getAmount(): int

LeaderAbility
+activate(player: Player)

Discount
-resource: ResourceType
-amount: int
+getResource(): ResourceType
+getAmount(): int
+activate(player: Player)

ExtraSlot
-resource: ResourceType
+getResource(): ResourceType
+activate(player: Player)

ExtraProduction
-production: ProductionPower
+getProduction(): ProductionPower
+activate(player: Player)

ChangeWhiteMarbles
-resourceType: ResourceType
+getResourceType(): ResourceType
+activate(player: Player)

«enumeration»
ResourceType
STONE
COIN
SHIELD
SERVANT
FAITH
ALL

DepotDecorator
#depot: Depot

SoloActionToken
-game: SoloGame
+reveal()
+getGame(): SoloGame

ConcreteDepot
+getLayer(layerNumber: int)
+modifyLayer(layerNumber: int)
+moveResources(amount: int, fromLayer: int, toLayer: int)
-moveToLayer(amount: int, fromLayer: DepotLayer, toLayer: DepotLayer)
-swapLayers(fromLayer: DepotLayer, toLayer: DepotLayer)
+queryAllRes(): Resource
+retrieveRes(res: Resource)
+discardRes(layer: DepotLayer)
#getMapCopy(): Map<Integer - DepotLayer>
#addToMap(key: int, depotLayer: DepotLayer)

ConcreteDepotDecorator
+getLayer(layerNumber: int)
+modifyLayer(layerNumber: int)
+moveResources(amount: int, fromLayer: int, toLayer: int)
-moveToLayer(amount: int, fromLayer: DepotLayer, toLayer: DepotLayer)
-swapLayers(fromLayer: DepotLayer, toLayer: DepotLayer)
+queryAllRes(): Resource
+retrieveRes(res: Resource)
+discardRes(layer: DepotLayer)
#getMapCopy(): Map<Integer - DepotLayer>
#addToMap(key: int, depotLayer: DepotLayer)

DiscardDevCards
-color: CardColor
+reveal()

MoveBlackCross
+reveal()

MoveAndShuffle
+reveal()

CardDeck
-cards: Deque<DevelopmentCard>
+getLevel(): int
+getType(): CardColor
+addCard(DevelopmentCard)
-shuffle()
+getNumberOfCard(): int
+removeCard(): DevelopmentCard
+getCard(): DevelopmentCard

Market
-cards[4][4]: CardDeck
+buyCards(level: int, color: CardColor): DevelopmentCard
+getMarketTray(): MarketTray
-initCards()
+getAvailableCards(): List<DevelopmentCard>
+getCard(level: int, color: CardColor): DevelopmentCard
+endgame()

Marbles
-marbles: Map<MarblesColor- Integer>
+getValue(key: MarblesColor): int
+add(key: MarblesColor)
+getResource(): Resource
+equals(other: Marbles): boolean
+containFaith(): boolean
+Operation1()
+getAllMarblesList(map: Map<MarbleColor-Integer>): List<>
+getAllMarblesMap(): Map<MarbelColor- Integer>

MarketTray
-marketMarbles[3][4]: MarblesColor
-remainingMarble: MarblesColor
+insertMarble(position: int): Marbles
-updateMarbles(position: int)

«enumeration»
MarblesColor
WHITE
BLUE
GREY
YELLOW
PURPLE
RED
+getResourceType(): ResourceType
+contains(): boolean