Model.Main

Market

-marketArrangement[][]: Marble
-excessMarble: Marble
-row: int
-col: int

+1

+13

Marble

+drawMarble(players[]: Player, playerNumber: int): void

BlueMarble

GreyMarble

PurpleMarble

YellowMarble

RedMarble

LeaderCardDeck

-leaderCardDeck[]: LeaderCard

LeaderCard

-played: boolean
-victoryPoints: int

+isPlayed(): boolean
+setPlayed(play: boolean): void
+getVictoryPoints(): int
+checkRequisites(playerboard: Playerboard): boolean
+activateAbility(player: Player): void

DevelopmentCardDecksGrid

-developmentCardsDecksGrid[][][]: DevelopmentCard
-developmentCardsDecksColours: Map<String;Integer>
-developmentCardsLevels: Collection<Integer>

+getDevelopmentCardsDecks(): DevelopmentCard
+getDevelopmentCardColours(): Map<String;Integer>
+getDevelopmentCardLevels(): Collection<Integer>
+printDevelopmentCardsDecks(): void
+stillCardsInTheDeck(developmentCardDeck[]: DevelopmentCard): boolean
+removeDevelopmentCards(column: int): void

+1
+48

DevelopmentCard

-colour: String
-level: int
-cost: Map<String;Integer>
-input: Map<String;Integer>
-output: Map<String;Integer>
-faithPoints: int
-victoryPoints: int

+getDevelopmentCardColour(): String
+getDevelopmentCardLevel(): int
+getDevelopmentCardCost(): Map<String;Integer>
+getDevelopmentCardInput(): Map<String;Integer>
+getDevelopmentCardOutput(): Map<String;Integer>
+getFaithPoints(): int
+getVictoryPoints(): int
+checkResourcesAvailability(playerBoard: Playerboard, requisites: Map<String;Integer>): boolean
+checkPlayerBoardDevelopmentCardsCompatibility(playerBoard: Playerboard): boolean
+payDevelopmentCard(playerBoard: PlayerBoard): void
+activateProduction(playerBoard: PlayerBoard): void
+printDevelopmentCard(): void