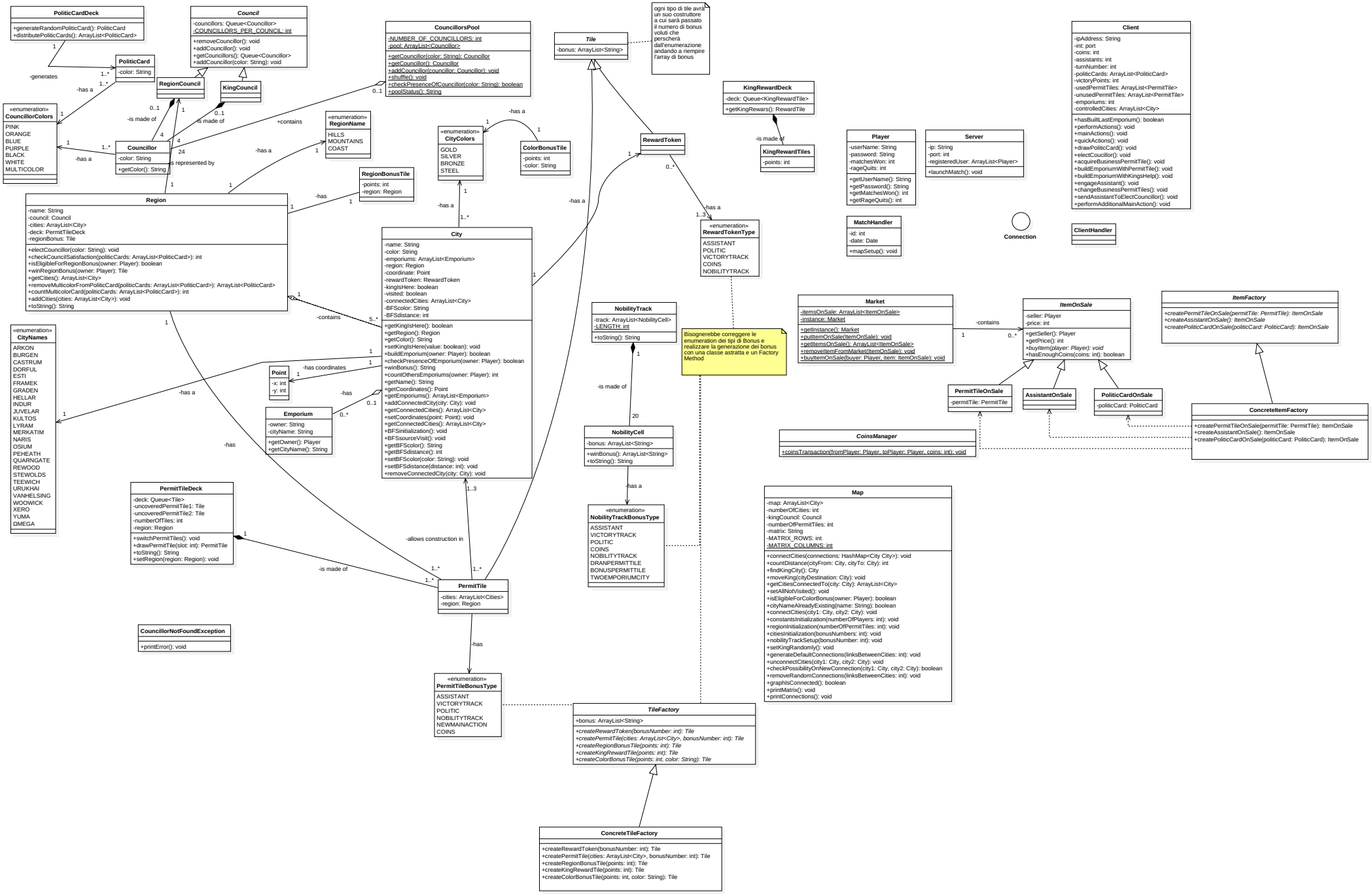
Model::Main


PoliticCardDeck
+generateRandomPoliticCard(): PoliticCard
+distributePoliticCards(): ArrayList<PoliticCard>
PoliticCard
-color: String
Council
-councillors: Queue<Councillor>
-COUNCILLORS_PER_COUNCIL: int
+removeCouncillor(): void
+addCouncillor(): void
+getCouncillors(): Queue<Councillor>
+addCouncillor(color: String): void
RegionCouncil
KingCouncil
CouncillorsPool
-NUMBER_OF_COUNCILLORS: int
-pool: ArrayList<Councillor>
+getCouncillor(color: String): Councillor
+getCouncillor(): Councillor
+addCouncillor(councillor: Councillor): void
+shuffle(): void
+checkPresenceOfCouncillor(color: String): boolean
+poolStatus(): String
Tile
-bonus: ArrayList<String>
ogni tipo di tile avrà un suo costruttore a cui sarà passato il numero di bonus voluti che perscherà dall'enumerazione andando a riempire l'array di bonus
Client
-ipAddress: String
-int: port
-coins: int
-assistants: int
-turnNumber: int
-politicCards: ArrayList<PoliticCard>
-victoryPoints: int
-usedPermitTiles: ArrayList<PermitTile>
-unusedPermitTiles: ArrayList<PermitTile>
-emporiums: int
-controlledCities: ArrayList<City>
+hasBuiltLastEmporium(): boolean
+performActions(): void
+mainActions(): void
+quickActions(): void
+drawPoliticCard(): void
+electCouncillor(): void
+acquireBusinessPermitTile(): void
+buildEmporiumWithPermitTile(): void
+buildEmporiumWithKingsHelp(): void
+engageAssistant(): void
+changeBusinessPermitTiles(): void
+sendAssistantToElectCouncillor(): void
+performAdditionalMainAction(): void
KingRewardDeck
-deck: Queue<KingRewardTile>
+getKingRewars(): RewardTile
KingRewardTiles
-points: int
Player
-userName: String
-password: String
-matchesWon: int
-rageQuits: int
+getUserName(): String
+getPassword(): String
+getMatchesWon(): int
+getRageQuits(): int
Server
-ip: String
-port: int
-registeredUser: ArrayList<Player>
+launchMatch(): void
«enumeration»
CouncillorColors
PINK
ORANGE
BLUE
PURPLE
BLACK
WHITE
MULTICOLOR
Councillor
-color: String
+getColor(): String
«enumeration»
RegionName
HILLS
MOUNTAINS
COAST
«enumeration»
CityColors
GOLD
SILVER
BRONZE
STEEL
ColorBonusTile
-points: int
-color: String
RewardToken
«enumeration»
RewardTokenType
ASSISTANT
POLITIC
VICTORYTRACK
COINS
NOBILITYTRACK
MatchHandler
-id: int
-date: Date
+mapSetup(): void
Connection
ClientHandler
RegionBonusTile
-points: int
-region: Region
Region
-name: String
-council: Council
-cities: ArrayList<City>
-deck: PermitTileDeck
-regionBonus: Tile
+electCouncillor(color: String): void
+checkCouncilSatisfaction(politicCards: ArrayList<PoliticCard>): int
+isEligibleForRegionBonus(owner: Player): boolean
+winRegionBonus(owner: Player): Tile
+getCities(): ArrayList<City>
+removeMulticolorFromPoliticCard(politicCards: ArrayList<PoliticCard>): ArrayList<PoliticCard>
+countMulticolorCard(politicCards: ArrayList<PoliticCard>): int
+addCities(cities: ArrayList<City>): void
+toString(): String
City
-name: String
-color: String
-emporiums: ArrayList<Emporium>
-region: Region
-coordinate: Point
-rewardToken: RewardToken
-kingIsHere: boolean
-visited: boolean
-connectedCities: ArrayList<City>
-BFScolor: String
-BFSdistance: int
+getKingIsHere(): boolean
+getRegion(): Region
+getColor(): String
+setKingIsHere(value: boolean): void
+buildEmporium(owner: Player): boolean
+checkPresenceOfEmporium(owner: Player): boolean
+winBonus(): String
+countOthersEmporiums(owner: Player): int
+getName(): String
+getCoordinates(): Point
+getEmporiums(): ArrayList<Emporium>
+addConnectedCity(city: City): void
+getConnectedCities(): ArrayList<City>
+setCoordinates(point: Point): void
+getConnectedCities(): ArrayList<City>
+BFSinitialization(): void
+BFSsourceVisit(): void
+getBFScolor(): String
+getBFSdistance(): int
+setBFScolor(color: String): void
+setBFSdistance(distance: int): void
+removeConnectedCity(city: City): void
«enumeration»
CityNames
ARKON
BURGEN
CASTRUM
DORFUL
ESTI
FRAMEK
GRADEN
HELLAR
INDUR
JUVELAR
KULTOS
LYRAM
MERKATIM
NARIS
OSIUM
PEHEATH
QUARNGATE
REWOOD
STEWOLDS
TEEWICH
URUKHAI
VANHELSING
WOOWICK
XERO
YUMA
OMEGA
Point
-x: int
-y: int
Emporium
-owner: String
-cityName: String
+getOwner(): Player
+getCityName(): String
NobilityTrack
-track: ArrayList<NobilityCell>
-LENGTH: int
+toString(): String
Bisognerebbe correggere le enumeration dei tipi di Bonus e realizzare la generazione dei bonus con una classe astratta e un Factory Method
Market
-itemsOnSale: ArrayList<ItemOnSale>
-instance: Market
+getInstance(): Market
+putItemOnSale(ItemOnSale): void
+getItemsOnSale(): ArrayList<ItemOnSale>
+removeItemFromMarket(ItemOnSale): void
+buyItemOnSale(buyer: Player, item: ItemOnSale): void
ItemOnSale
-seller: Player
-price: int
+getSeller(): Player
+getPrice(): int
+buyItem(player: Player): void
+hasEnoughCoins(coins: int): boolean
ItemFactory
+createPermitTileOnSale(permitTile: PermitTile): ItemOnSale
+createAssistantOnSale(): ItemOnSale
+createPoliticCardOnSale(politicCard: PoliticCard): ItemOnSale
PermitTileOnSale
-permitTile: PermitTile
AssistantOnSale
PoliticCardOnSale
-politicCard: PoliticCard
ConcreteItemFactory
+createPermitTileOnSale(permitTile: PermitTile): ItemOnSale
+createAssistantOnSale(): ItemOnSale
+createPoliticCardOnSale(politicCard: PoliticCard): ItemOnSale
NobilityCell
-bonus: ArrayList<String>
+winBonus(): ArrayList<String>
+toString(): String
CoinsManager
+coinsTransaction(fromPlayer: Player, toPlayer: Player, coins: int): void
«enumeration»
NobilityTrackBonusType
ASSISTANT
VICTORYTRACK
POLITIC
COINS
NOBILITYTRACK
DRANPERMITTILE
BONUSPERMITTILE
TWOEMPORIUMCITY
Map
-map: ArrayList<City>
-numberOfCities: int
-kingCouncil: Council
-numberOfPermitTiles: int
-matrix: String
-MATRIX_ROWS: int
-MATRIX_COLUMNS: int
+connectCities(connections: HashMap<City City>): void
+countDistance(cityFrom: City, cityTo: City): int
+findKingCity(): City
+moveKing(cityDestination: City): void
+getCitiesConnectedTo(city: City): ArrayList<City>
+setAllNotVisited(): void
+isEligibleForColorBonus(owner: Player): boolean
+cityNameAlreadyExisting(name: String): boolean
+connectCities(city1: City, city2: City): void
+constantsInitialization(numberOfPlayers: int): void
+regionInitialization(numberOfPermitTiles: int): void
+citiesInitialization(bonusNumbers: int): void
+nobilityTrackSetup(bonusNumber: int): void
+setKingRandomly(): void
+generateDefaultConnections(linksBetweenCities: int): void
+unconnectCities(city1: City, city2: City): void
+checkPossibilityOnNewConnection(city1: City, city2: City): boolean
+removeRandomConnections(linksBetweenCities: int): void
+graphIsConnected(): boolean
+printMatrix(): void
+printConnections(): void
PermitTileDeck
-deck: Queue<Tile>
-uncoveredPermitTile1: Tile
-uncoveredPermitTile2: Tile
-numberOfTiles: int
-region: Region
+switchPermitTiles(): void
+drawPermitTile(slot: int): PermitTile
+toString(): String
+setRegion(region: Region): void
PermitTile
-cities: ArrayList<Cities>
-region: Region
CouncillorNotFoundException
+printError(): void
«enumeration»
PermitTileBonusType
ASSISTANT
VICTORYTRACK
POLITIC
NOBILITYTRACK
NEWMAINACTION
COINS
TileFactory
+bonus: ArrayList<String>
+createRewardToken(bonusNumber: int): Tile
+createPermitTile(cities: ArrayList<City>, bonusNumber: int): Tile
+createRegionBonusTile(points: int): Tile
+createKingRewardTile(points: int): Tile
+createColorBonusTile(points: int, color: String): Tile
ConcreteTileFactory
+createRewardToken(bonusNumber: int): Tile
+createPermitTile(cities: ArrayList<City>, bonusNumber: int): Tile
+createRegionBonusTile(points: int): Tile
+createKingRewardTile(points: int): Tile
+createColorBonusTile(points: int, color: String): Tile
-generates
-has a
-is made of
+contains
is represented by
-has
-has coordinates
-contains
-allows construction in
-is made of