Model

Game

- game : Game

- civilizations : Civilization[]

- originalCapitals : City[]

- turn : int

- map : Hex[][]

- Game(users : user[]) : << constructor>>

+ startNewGame(user : user[]) : void

+ getGame() : Game

+ getPlayingCivilization() : Civilization

Civilization

- user: User

- isYourTurn : boolean

- visibilityMap : HexVisibility[][]

- technologies : Hashmap<TechnologyName , Technology>

- technologyInProgress : Technology

- openedUnits : ArrayList<Unit>

- openedResources : ArrayList<Resource>

- capital : City

- units : ArrayList<Unit>

- cities : ArrayList<City>

- gold : int

- sciencePerTurn : int

- scienceStorage : int

- happiness : int

+ Civilization (user : User) : <<constructor>>

+ getmap (int x , int y) : String

+ adjustVisibility(x : int , y : int) : void

User

- listOfUsers : ArrayList<User>

- username: String

- password : String

- nickname : String

- score : int

+ User(username : String, password : String, nickname : String): <<constructor>>

+ addUser (username : String, password : String, nickname : String) : void

Technology

- technologyTree : ArrayList<Technology>

- futureTechnologies : Hashmap<TechnologyName , Technology>

- prerequisiteTechnologies : Hashmap<TechnologyName , Technology

- openingUnits : ArrayList<Unit>

- openingResources : ArrayList<Resource>

- cost : int

- name : TechnologyName

+ Technology(name : TechnologyName, cost : int): <<constructor>>

City

- name : String

- owner : Civilization

- neededProduction : int

- unitInProgress: Unit

- numberOfCitizen : int

- coordinatesOfCenter : Hashmap<char, int>

- foodStorage : int

- foodPerTurn : int

- productionPerTurn : int

- goldPerTurn : int

- sciencePerTurn : int

- cityHexes : ArrayList<Hex>

- cityUnit : RangedMilitary

+ City(name : String, x : int , y : int): <<constructor>>

Model

Hex

- owner : Civilization
- terrain : Terrain
- feature : Feature
-resource: Resource
- imrovement : Improvement
- hasDestroyedImprovement : boolean
-percentOfBuldingImprovement : int
-isAnyCitizenWorking: boolean
-movementPrice: int
- hasRiver : boolean
- hasRoad : boolean
- hasRailRoad : boolean
- coordinates : Hashmap<char, int>
-militaryUnit: MilitaryUnit
-civilUnit : CivilUnit

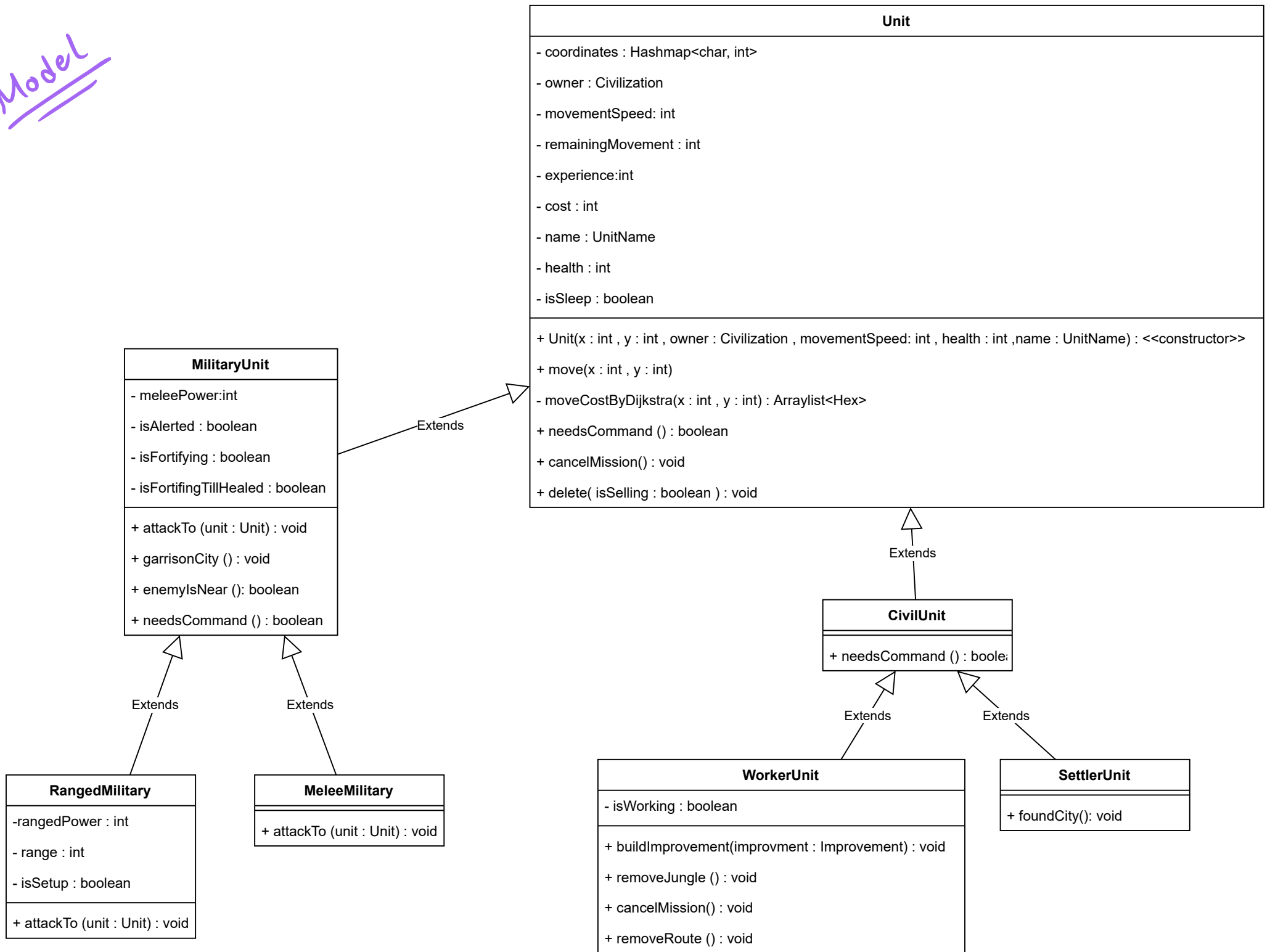+ Hex() : <<constructor>>
+ getUnitInside() : Unit

Feature

+ features : Hashmap<FeatureName, Feature>
 - gold : int
- production : int
- combatEffect : int
-movementPrice: int
- food : int
- name : FeatureName

+ Feature(gold : int , production : int , combatEffect : int , mp : int , food : int , name : FeatureName) : << constructor >>

Improvement

+ improvements : Hashmap< improvementName, Improvement>
- food : int
- gold : int
- production : int
- name : ImprovementName

+ Improvement(gold : int , production : int, food : int , name : ImprovementName) : << constructor >>

Resource

+ resources : Hashmap<ResourceName, resource>
- food : int
- gold : int
- production : int
- name : ResourceName

+ Resource(gold : int , production : int, food : int , name : ResourceName) : << constructor >>
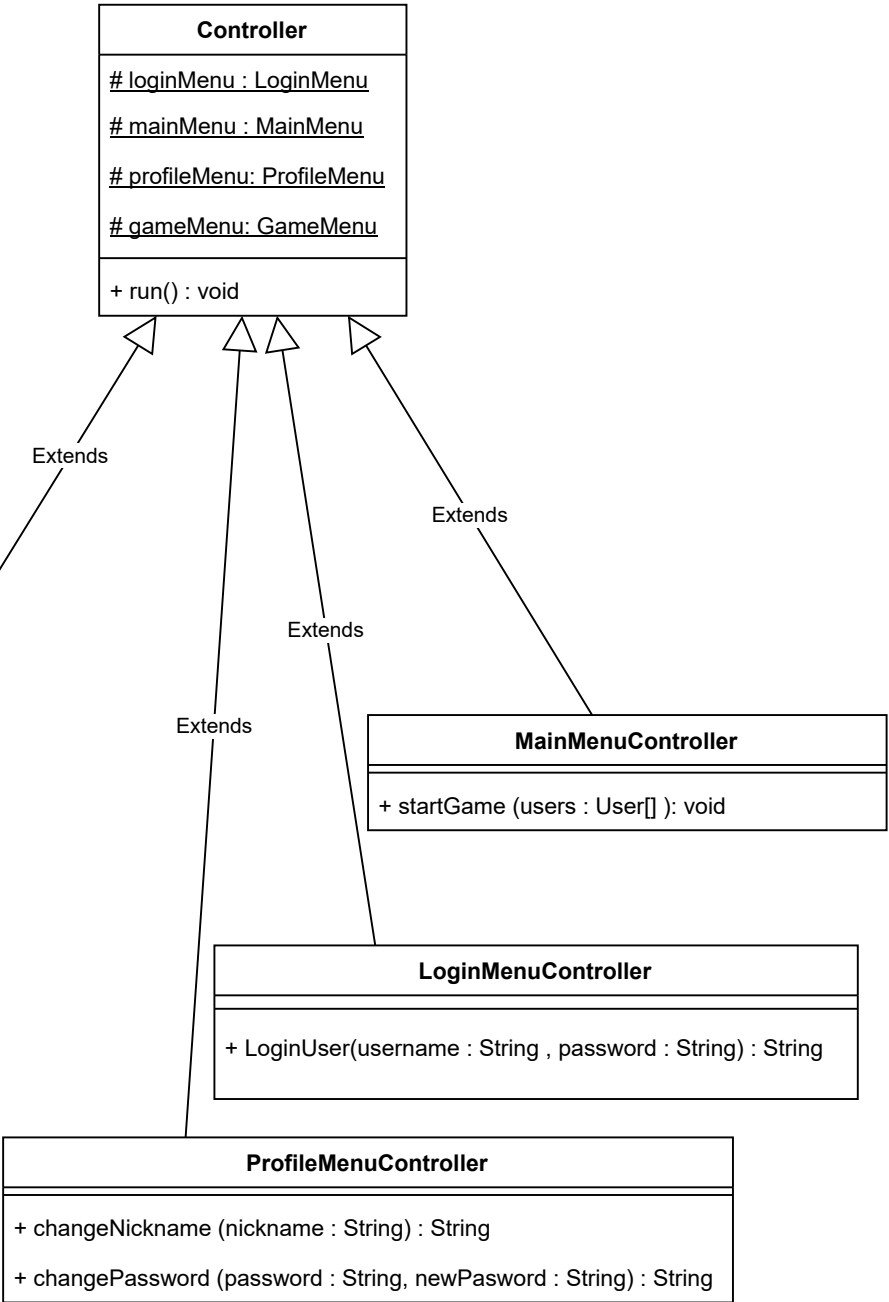
Terrain

+ terrains : Hashmap<TerrainName, Terrain>
- food : int
-movementPrice: int
-combatEffect : int
- gold : int
- production : int
- name : TerrainName

+ Terrain(gold : int , production : int , combatEffect : int , mp : int ,food : int , name : TerrainName) : << constructor >>

Model

Unit

- coordinates : Hashmap<char, int>
- owner : Civilization
- movementSpeed: int
- remainingMovement : int
- experience:int
- cost : int
- name : UnitName
- health : int
- isSleep : boolean

+ Unit(x : int , y : int , owner : Civilization , movementSpeed: int , health : int ,name : UnitName) : <<constructor>>
+ move(x : int , y : int)
- moveCostByDijkstra(x : int , y : int) : Arraylist<Hex>
+ needsCommand () : boolean
+ cancelMission() : void
+ delete(isSelling : boolean) : void

MilitaryUnit

- meleePower:int
- isAlerted : boolean
- isFortifying : boolean
- isFortifingTillHealed : boolean

+ attackTo (unit : Unit) : void
+ garrisonCity () : void
+ enemyIsNear (): boolean
+ needsCommand () : boolean

Extends

CivilUnit

+ needsCommand () : boolea

Extends

RangedMilitary

-rangedPower : int
- range : int
- isSetup : boolean

+ attackTo (unit : Unit) : void

Extends

MeleeMilitary

+ attackTo (unit : Unit) : void

Extends

WorkerUnit

- isWorking : boolean

+ buildImprovement(improvment : Improvement) : void
+ removeJungle () : void
+ cancelMission() : void
+ removeRoute () : void

Extends

SettlerUnit

+ foundCity(): void

Controller

Controller

loginMenu : LoginMenu

mainMenu : MainMenu

profileMenu: ProfileMenu
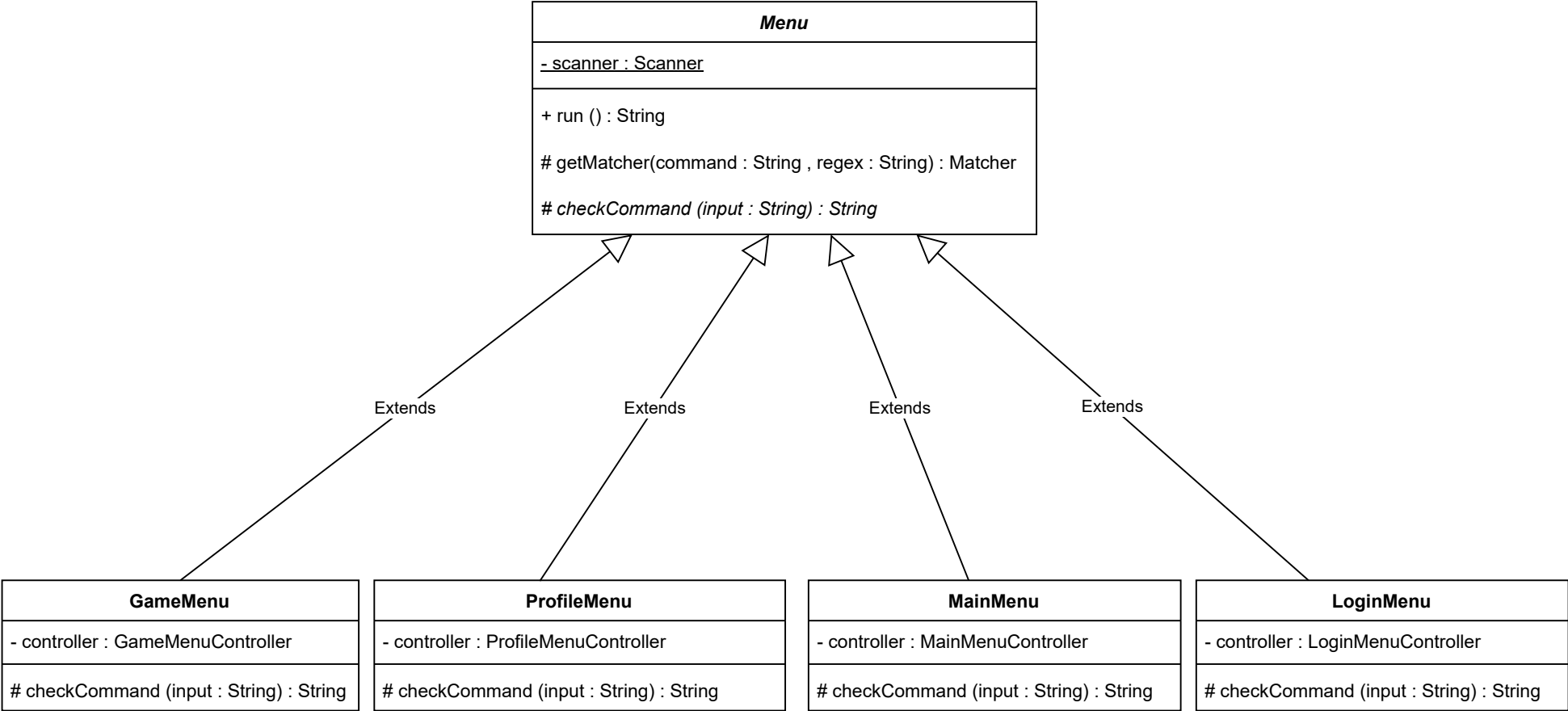
gameMenu: GameMenu

+ run() : void

GameMenuController

- selectedUnit : Unit

- selectedCity : City

- notificationsHistory : ArrayList<String>

+ changeTurn() : String

+ showTechnologyInfo() : String

+ showUnitsPanel() : String

+ showCitiesPanel() : String

+ showDiplomacyPanel() : String

+ showVictoryPanel() : String

+ showDemographicsPanel() : String

+ showNotificationHistory() : String

+ showMilitaryPanel() : String

+ showEconomyPanel() : String

+ showDiplomaticSituation() : String

+ showDealsPanel() : String

+ selectMilitaryUnit(x : int, y : int) : String

+ selectCivilUnit(x : int, y : int) : String

+ selectCity(name : String) : String

+ selectCity(x : int , y : int) : String

+ moveTo(x : int , y : int) : String

+ sleepUnit() : String

+ alertUnit() : String

+ fortifyUnit() : String

+ fortifyUnitUntilHeal() : String

+ garrisonUnitInCity() : String

+ setupRangedUnit() : String

+ attackTo(x : int , y : int) : String

+ foundCityHere() : String

+ wakeUpUnit() : String

+ deleteUnit() : String

+ cancelMission() : String

+ build(improvement : ImprovementName) : String

+ removeJungle() : String

+ removeRoute() : String

+ repair() : String

+ showMap(x : int , y : int) : String

+ showMap(cityName : String) : String

+ moveMap(direction : Direction , amount : int) : String

Extends

Extends

Extends

Extends

MainMenuController

+ startGame (users : User[]): void

LoginMenuController

+ LoginUser(username : String , password : String) : String

ProfileMenuController

+ changeNickname (nickname : String) : String

+ changePassword (password : String, newPasword : String) : String

View

Menu

- scanner : Scanner

+ run () : String

getMatcher(command : String , regex : String) : Matcher

checkCommand (input : String) : String

Extends | Extends | Extends | Extends

GameMenu

- controller : GameMenuController

checkCommand (input : String) : String

ProfileMenu

- controller : ProfileMenuController

checkCommand (input : String) : String

MainMenu

- controller : MainMenuController

checkCommand (input : String) : String

LoginMenu

- controller : LoginMenuController

checkCommand (input : String) : String

enums

<<enumeration>> UnitName

- ARCHER
- CHARIOT_ARCHER
- SCOUT
- SETTLER
- SPEARMAN
- WARRIOR
- WORKER
- CATAPULT
- HORSEMAN
- SWORDSMAN
- CROSSBOWMAN
- KNIGHT
- LONGSWORDSMAN
- PIKEMAN
- TREBUCHET
- CANON
- CAVALRY
- LANCER
- MUSKETMAN
- RIFLEMAN
- ANTI_TANK_GUN
- ARTILLERY
- INFANTRY
- PANZER
- TANK

<<enumeration>> ResourceName

- BANANA
- COW
- GAZELLE
- SHEEP
- WHEAT
- COAL
- HORSE
- IRON
- COTTON
- COLOR
- FUR
- GEM
- GOLD
- FUMIGATION
- TUSK
- MARBLE
- SILK
- SILVER
- SUGAR

<<enumeration>> TechnologyName

- AGRICULTURE
- ANIMAL_HUSBANDRY
- ARCHERY
- BRONZE_WORKING
- CALENDAR
- MASONRY
- MINING
- POTTERY
- WHEEL
- TRAPPING
- WRITING
- CONSTRUCTION
- HORSEBACK_RIDING
- IRON_WORKING
- MATHEMATICS
- PHILOSOPHY
- CHIVALRY
- CIVIL_SERVICE
- CURRENCY
- EDUCATION
- ENGINEERING
- MACHINERY
- METAL_CASTING
- PHYSICS
- STEEL
- THEOLOGY
- ACOUSTICS
- ARCHAEOLOGY
- BANKING
- CHEMISTRY
- ECONOMICS
- FERTILIZER
- GUNPOWDER
- METALLURGY
- MILITARY_SCIENCE
- PRINTING_PRESS
- RIFLING
- SCIENTIFIC_THEORY
- BIOLOGY
- COMBUSTION
- DYNAMITE
- ELECTRICITY
- RADIO
- RAILROAD
- REPLACEABLE
- PARTS
- STEAM_POWER
- TELEGRAPH

<<enumeration>> TerrainName

- DESERT
- GRASSLAND
- HILL
- MOUNTAIN
- OCEAN
- PLAIN
- SNOWLAND
- TUNDRA

<<enumeration>> FeatureName

- FLAT
- JUNGLE
- ICE
- DENSE_FOREST
- SWAMP
- OASIS
- RIVER

<<enumeration>> ImprovementName

- CAMP
- FARM
- LUMBER_MILL
- MINE
- PASTURE
- PLANTATION
- QUARRY
- TRADING_POST
- FACTORY
- ROAD
- RAILROAD

<<enumeration>> HexVisibility

- TRANSPARENT
- DETERMINED
- FOG_OF_WAR

<<enumeration>> Direction

- RIGHT
- LEFT
- UP
- DOWN