Main

- scanner: Scanner

- controller: ChangeMenuController

+ Main(args: String[]): void

Model

User

- users: ArrayList<User>
- username: String
- password: String
- nickname: String
- email: String
- slogan: String
- highscore: int
- securityQuestion: String
- securityAnswer: String

+ User(username: String, nickname: String, email: String, slogan: String, question: String, answer: String)<<constructor>>
+ isUsernameUniqe(username: String): boolean
+ isEmailUniqe(email: String): boolean
+ addUser(user: User): void
+ getUserByUsername(username: String): User
+ checkPassword(password: String): boolean
+ setUsername(username: String): void
+ setNickname(nickname: String): void
+ setPassword(password: String): void

OtherBuildings

- isWorking: boolean

+ OtherBuildings (building: Building, isWorking: boolean)<<constructor>>

Extends

ShopBuildings

Extends

+ ShopBuilding (building: Building)<<constructor>>
+ hasEnoughProduct(product: Product, count: int): boolean

Building

- allBuildings: ArrayList<Building>
- type: String
- category: String
- buildingCosts: ArrayList<Source>
- workerCounter: int
- hitPoint: int
- location: Cell
- king: User

+ Building(type: String, category: String, buildingCosts: ArrayList<Source>, workerCouner, hitpoint: int)<<constructor>>
+ Building (building: Building, location: Cell)<<constructor>>
+ getBuildingByType(type: String): Building

FightingBuilding

- military: ArrayList<MilitaryPerson>
- fireRange: int
- defendRange: int

+ FightingBuilding (building: Building, rate: int)<<constructor>>
+ calculateRepairCost(): int

Extends

StorageBuilding

- storedProducts: ArrayList<Product>
- capacity: int
- storageCategory: String

+ StorageBuilding (building: Building, capacity: int, storageCategory: String)<<constructor>>
+ countFreeSpace(): int

Extends

Kingdom

- kingBuildings: ArrayList<Building>
- kingProducts: ArrayList<Product>
- kingPeople: ArrayList<Person>
- kingPopularityFactors: ArrayList<PopularityFactor>
- kingAttackEquipments: ArrayList<AttackEquipment>
- king: User
- inventory: int
- joblessCounter: int

+ Kingdom(king: User)<<constructor>>
+ getProductByName(name: String): Product
+ addToKingBuildings(building: Building): void
+ changeKingProductCount(product: Product, count: int): void
+ addToKingPopularityFactors(popularityFactor: PopularityFactor): void
+ getKingFoods(): ArrayList<Product>
- getBuildingByLocation(location: Cell): Building

ProductionBuilding

- rate: int

+ ProductionBuilding (building: Building, rate: int)<<constructor>>
+ hasMatterial(): boolean

Extends

Extends

TrainingBuilding

- outputMilitaryPeople: ArrayList<MilitaryPerson>

+ TrainingBuilding (building: Building, - outputMilitaryPeople: ArrayList<MilitaryPerson>)<<constructor>>
+ canTrainWantedUnit(personName: String): boolean

PopularityFactor

- rate: int
- popularityAmount: int

+ PopularityFactor(rate: int)<<constructor>>

Product

- allProducts: ArrayList<Product>
- name: String
- cost: int
- price: int
- count: int
- catogory: String
- usedMatterials: ArrayList<Product>

+ Product(name: String, price: int, cost: int, category: String, usedMatterials: ArrayList<Product>)<<constructor>>
+ changeProductCount(count: int): void

Cell

- material: String
- building: Building
- people : ArrayList<Person>
- naturalBlocks : ArrayList<NaturalBlock>
- hasTunnel: boolean

+ setMaterial(material: String): void
+ setBuilding(building: Building): void
+ addPerson(person: Person): void

MilitaryPerson

- allMilitaryPeople: ArrayList<MilitaryPerson>
- firePower: int
- defendPower: int
- speed: int
- neededProducts: ArrayList<Product>
- trainingCost: int
- mode: String

+ MilitaryPerson(king: User, type: String, location: Cell, militaryPerson: MilitaryPerson)<<constructor>>
+ MilitaryPerson(king: User, type: String, location: Cell, firePower: int, defendPower: int, speed: int, neededProducts: ArrayList<Product>, trainingCost: int)<<constructor>>
+ getMilitaryPersonByType(type: String): MilitartPerson

Person

- king: User
- type: String
- location: Cell

+ Person(type: String, king: User, location: Cell)<<constructor>>

Extends

WorkerPerson

- workPlace: Building

+ WorkerPerson(name: String, workPlace: Building)<<constructor>>

Extends

AttackEquipment

- name: String
- usedMatterials: ArrayList<Product>
- king: User
- engineers: ArrayList<MilitaryPerson>

+ AttackEquipment(name: String, usedMaterials: ArrayList<Product>, king: User, engineers: ArrayList<MilitaryPerson>)<<constructor>>

Map

- templateMaps: Map[3]
- cells: Cell[][]

+ Map(height: int, width int)<<constructor>>
+ getCellByLocation(x: int, y: int): Cell

Turn

- TurnCounter: int
- currentKing: User

+ Turn(currentKing: user)<<constructor>>

Turn

- destinationOfMovingMilitaryPerson: HashMap<MilitaryPerson,Cell>

+ Turn()<<constructor>>
+ addMovingMilitaryPerson(militaryPerson : MilitaryPerson, cell: Cell): void

NaturalBlock

- name: String
- category: String

+ NaturalBlock(name: String, category: String)<<constructor>>

Trade

- allTrades: ArrayList<Trade>
- id: int
- tradeType: String
- resourceType: String
- resourceAmount: String
- price: int
- message: String

+ Trade(tradeType: String, resourceType: String, resourceAmount: String, price: int, message: String)<<constructor>>

Game

- kingdoms: ArrayList<Kingdom>
- map: Map
- mapTemplateNumber: int

+ Game(map: Map)<<constructor>>
+ addKingdom(kingdom: Kingdom): void

RegisterLoginController

- currentUser: User

+ RegisterLoginController(scanner: Sccaner)<<constructor>>
+ setCurrentUser(user: User): void
+ getCurrentUser(): User
+ register(matcher: Matcher): String
+ isUsernameValid(username: String): boolean
+ isPasswordWeak(password: String): boolean
+ isEmailValid(email: String): boolean
+ generateRandomPassword(): String
+ generateRandomSlogan(): String
+ generateCaptcha(): String
+ setSequrityQuestion(scanner: Scanner): String
+ login(matcher: Matcher): String
+ resetPassword(): String
+ changeInfo(field: String, content: String): String
+ changePassword(oldPassword: String, newPassword: String): String
+ getRank(): int
+ displayProfile(matcher: Matcher): String

ProfileController

+ ProfileController(scanner: Sccaner)<<constructor>>
+ changeInfo(field: String, content: String): String
+ changePassword(oldPassword: String, newPassword: String): String
+ removeSlogan(): void
+ displayProfile(matcher: Matcher): String
+ getRank(): int

InitializeController

+ InitializeController(scanner: Sccaner)<<constructor>>
+ readFromFile(path: String): ArrayList<String>
+ writeInFile(content: String, path: Stirng): void
+ intializBulidingsFromFile(path: String): void
+ intializeProductsFromFile(path: String): void
+ intializeMilitaryPersonFromFile(path: String): void
+ initializeMapTemplate1(): void
+ initializeMapTemplate2(): void
+ initializeMapTemplate3(): void

GameController

- currentX: int
- currentY: int
- selectedBuilding: Building
- selectedUnit: MilitaryPerson

+ GameController(scanner: Sccaner)<<constructor>>
+ setCellMatterial(matcher: Matcher): void
+ clearCell(): void
+ dropRock(mathcer: Matcher): String
+ dropTree(mathcer: Matcher): String
+ showMap(mathcer: Matcher): String
+ showMapWithNewCenter(mathcer: Matcher): String
+ showDetails(mathcer: Matcher): void
+ rateFood(mathcer: Matcher): void
+ rateTax(mathcer: Matcher): void
+ showPopularity(): String
+ showPopularityFactor(): String
+ showPopularity(): String
+ showFoodList(): String
+ buyProduct(name: String, count: int): String
+ showPopularityFactorRate(matcher: Matcher): String
+ hasFreeSpaceForBuilding(x: int, y: int): boolean
+ dropBuilding(matcher: Matcher): String
+ selectBuilding(matcher: Matcher): String
+ countTotalFreeSpaceOfStorage(storageCategory: String): int
+ createUnit(matcher: Matcher): String
+ repairBuilding(): String
+ increasePopulation(): void
+ selectUnit(matcher: Matcher): String
+ moveUnit(matcher: Matcher): String
+ patrolUnit(matcher: Matcher): String
+ setMode(matcher: Matcher): String
+ attackEnemy(matcher: Matcher): String
+ attackByShooting(matcher: Matcher): String
+ pourOil(matcher: Matcher): String
+ digTunnel(matcher: Matcher): String
+ buildEquipment(matcher: Matcher): String
+ disbandUnit(): String
+ digMoat(mathcer: Matcher): String
+ fillEnemyMoat(matcher: Matcher): String
+ burnOil(matcher: Matcher): String
+ showProductsPriceList(): String
+ buyProduct(matcher: Matcher): String
+ sellProduct(matcher: Matcher): String
+ nextTurn(): String

Controller

MainController

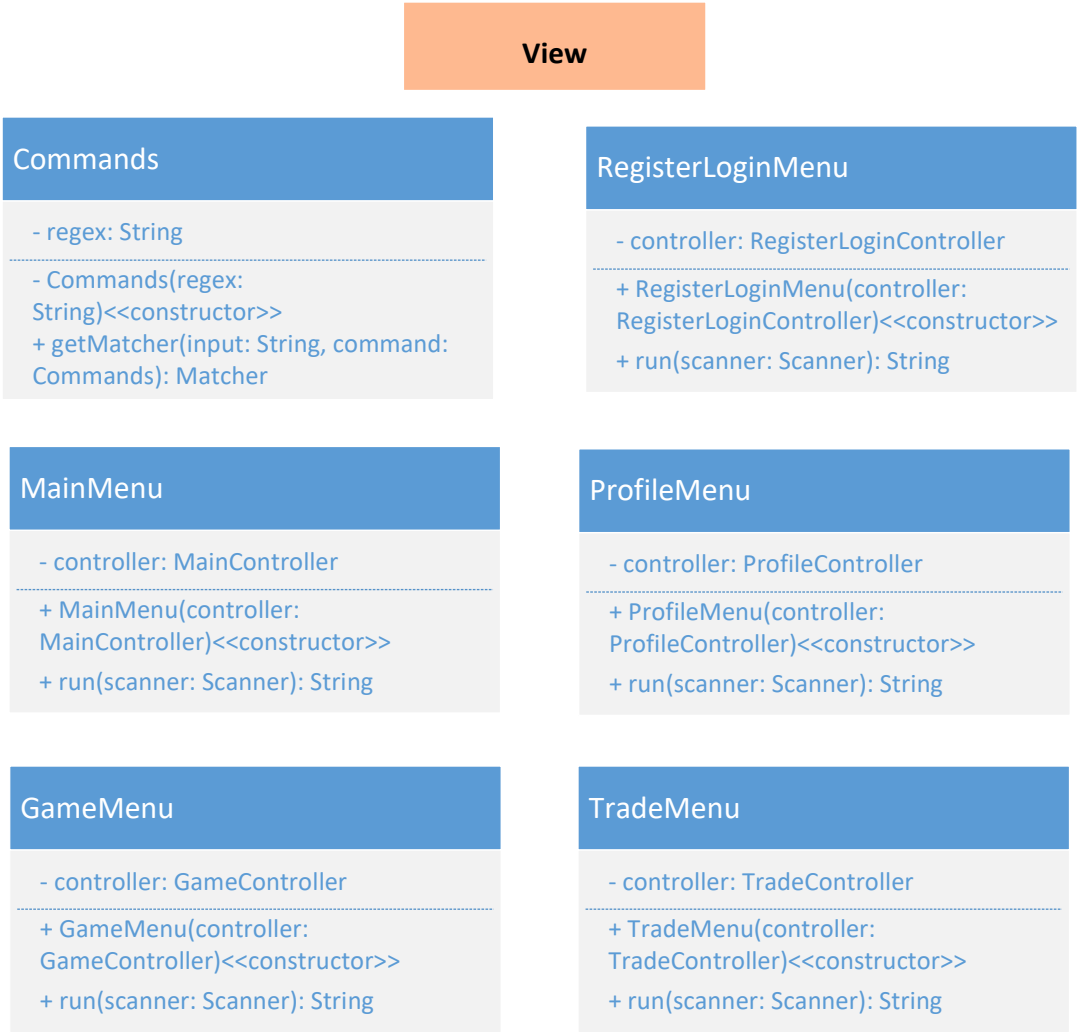
+ MainController(scanner: Sccaner)<<constructor>>
+ showTemplate(templateNumber: int): void
+ createMap(templateNumber: int): Map

ChangeMenuController

+ ChangeMenuController(scanner: Sccaner)<<constructor>>
+ getLoggedInUsername(): String
+ initializeDataFromFile(): void
+ run(scanner: Scanner): void

TradeController

+ TradeController(scanner: Sccaner)<<constructor>>
+ sendRequest(matcher: Matcher): String
+ acceptRequest(matcher: Matcher): String
+ showRequests(): String
+ showHistory(): String


View
Commands
- regex: String
- Commands(regex: String)<<constructor>>
+ getMatcher(input: String, command: Commands): Matcher
RegisterLoginMenu
- controller: RegisterLoginController
+ RegisterLoginMenu(controller: RegisterLoginController)<<constructor>>
+ run(scanner: Scanner): String
MainMenu
- controller: MainController
+ MainMenu(controller: MainController)<<constructor>>
+ run(scanner: Scanner): String
ProfileMenu
- controller: ProfileController
+ ProfileMenu(controller: ProfileController)<<constructor>>
+ run(scanner: Scanner): String
GameMenu
- controller: GameController
+ GameMenu(controller: GameController)<<constructor>>
+ run(scanner: Scanner): String
TradeMenu
- controller: TradeController
+ TradeMenu(controller: TradeController)<<constructor>>
+ run(scanner: Scanner): String