COSC 3P91 – Assignment 1 – 7698368, 8330508

COLIN GILBERT, LUKE SZABO, Brock University, Canada

1 PROJECT OVERVIEW

Compilation Code:
cd 3P91Code
javac Main.java

This Report is to showcase our design and implementation for a strategy-based simulation game focused on village development, resource management, and combat interactions between attacking and defending forces.
Note: Some methods like getter or setter methods have not been commented because their functionality is obvious.

The project is divided into four primary packages:

- **UI**: Handles user interaction and presentation logic.
- **UtilThings**: Provides shared utility classes, enumerations, and entity statistics.
- **Game**: Implements the core gameplay logic and simulation control.
- **GameComponents**: Defines the entities, units, buildings, and interfaces used throughout the game.

2 PACKAGE: GAME

The `Game` package contains the main game logic and orchestration mechanisms. Manages players, villages, combat simulations, and time-based events.

2.1 Classes in Game Package

2.1.1 *Class: Army.* Variables

- units : List<ArmyUnit>
- attackScore : int

Methods

- addUnit(unit : ArmyUnit) : void
- removeUnit(unit : ArmyUnit) : void
- getUnits() : List<ArmyUnit>
- getAttackScore() : int

2.1.2 *Class: Defences.* Variables

- buildings : List<DefenceBuilding>
- defenseScore : int

Methods

- addBuilding(building : DefenceBuilding) : void
- removeBuilding(building : DefenceBuilding) : void
- getDefenceBuildings() : List<DefenceBuilding>

Author's address: Colin Gilbert, Luke Szabo, Brock University, 1812 Sir Isaac Brock Way, St. Catharines, ON, L2S 3A1, Canada.

Update date: 8 February 2026

- getDefenceScore() : int

2.1.3 *Class: GameEngine.* **Variables**

- villages : List<Village>
- time : Time

Methods

- runGame() : void
- generateVillage(v : Village) : Village
- simulateAttack(a : Army, d : Defences) : SimulationResult
- setGuardTime(v : Village)

2.1.4 *Class: Player.* **Variables**

- playerID : int
- village : Village

Methods

- getPlayerID() : int
- setVillage(v : Village) : void
- getVillage() : Village
- exploreAttack() : Village

2.1.5 *Class: SimulationResult.* **Variables**

- attackerWin : boolean
- loot : Resource

Methods

- isAttackerWin() : boolean
- getLoot() : Resource

2.1.6 *Class: Time.* **Variables**

- time : int

Methods

- getTime() : int
- eventFinished(time : int) : boolean

2.1.7 *Class: Village.* **Variables**

- army : Army
- defences : Defences
- MAX_NUM_BUILDINGS : int
- villageHall : VillageHall
- buildings : List<Building>
- inhabitants : List<Inhabitant>
- resources : Resource
- guardTimeDuration : int

Methods

- addBuilding(b : Building) : void
- removeBuilding(b : Building) : void
- addInhabitant(i : Inhabitant) : void
- removeInhabitant(i : Inhabitant) : void

- getVillageHall() : VillageHall
- getBuildings() : List<Building>
- getInhabitants() : List<Inhabitant>
- getResource() : Resource
- getDefences() : Defences
- collectResources() : void
- isUnderProtection(t : Time) : boolean
- build(b : Building) : void
- train(i : Inhabitant) : void
- upgradeBuilding(b : Building) : void
- upgradeInhabitant(i : Inhabitant) : void

2.2 Game Package Design Decisions

For most of the class, the design decisions are very basic. Classes outline the main game components like army, defences, game engine, player, village all being very basic game components, with simulation result allowing for simulation. Within the village class we have a variable for the max number of buildings to limit how strong a village can be.

3 PACKAGE: GAMECOMPONENTS

The `GameComponents` package defines all core entities, inhabitants, combat units, buildings, workers, and interfaces used throughout the game. These components represent the physical and logical objects that exist within a village and participate in combat, production, and upgrades.

3.1 Interfaces

3.1.1 IAttacker.

- attack(target : IAttackable) : void

3.1.2 IAttackable.

- takeDamage(amount : int) : void
- isDestroyed() : boolean
- getHP() : int

3.1.3 IUpgradeable.

- getEntityType() : EntityType
- getStats() : EntityStats
- setStats(newStats : EntityStats) : void
- upgrade(nextLevel : EntityStats) : void

3.2 Core Classes

3.2.1 Class: Entity (implements IUpgradeable). **Variables**

- nextId : int
- id : int
- stats : EntityStats

Methods

- getId() : int
- getEntityType() : EntityType

- upgrade(nextLevelStats : EntityStats) : void
- getStats() : EntityStats
- setStats(newStats : EntityStats) : void

3.2.2 Class: Inhabitant (extends Entity).

3.2.3 Class: ArmyUnit (extends Inhabitant, implements IAttacker, IAttackable). **Variables**

- hp : int
- damage : int
- range : int

Methods

- attack(target : IAttackable) : int
- takeDamage(damage : int) : void
- isDestroyed() : boolean
- getDamage() : int
- getRange() : int
- getHP() : int

3.2.4 Class: Archer (extends ArmyUnit). **Methods**

- Archer()

3.2.5 Class: Soldier (extends ArmyUnit). **Methods**

- Soldier()

3.2.6 Class: Knight (extends ArmyUnit). **Methods**

- Knight()

3.2.7 Class: Catapult (extends ArmyUnit). **Methods**

- Catapult()

3.2.8 Class: Building (extends Entity, implements IAttackable). **Variables**

- currentHP : int
- stats : EntityStats
- underConstruction : boolean

Methods

- takeDamage(damage : int) : void
- isDestroyed() : boolean
- getHP() : int

3.2.9 Class: DefenceBuilding (extends Building, implements IAttacker). **Variables**

- damage : int
- range : int

Methods

- attack(target : IAttackable) : int

3.2.10 Class: ResourceBuilding (extends Building). **Variables**

- workers : List<ResourceWorker>
- workerCapacity : int

3.2.11 *Class: Farm (extends Building).* **Variables**

- workers : List<Worker>

3.2.12 *Class: GoldMine (extends ResourceBuilding).*

3.2.13 *Class: IronMine (extends ResourceBuilding).*

3.2.14 *Class: LumberMill (extends ResourceBuilding).*

3.2.15 *Class: VillageHall (extends Building).*

3.2.16 *Class: Worker (extends Inhabitant).* **Variables**

- isIdle : boolean
- stats : EntityStats

Methods

- getEntityType() : EntityType
- getIdleStatus() : boolean

3.2.17 *Class: ResourceWorker (extends Inhabitant).* **Variables**

- productionRate : int

Methods

- getEntityType() : EntityType

3.3 GameComponents Discussion

Within the Game Components package we start to define objects and how they can be used. Our Interfaces, IAttacker, IAttackable, and IUpgradeable Determine whether an Entity can Attack, Be Attacked, Or be Upgraded respectively . Entity is used as a base outline for anything that can be upgraded and stores it stats and id, inhabitant is all entities that live in a village, made for any type of person. The other classes within this section are largely part to use the game. For example ArmyUnit and Classes for each type of unit. Resource Buildings and different types of resource buildings.

For the resources specifically, we are currently doing a hard coded approach for convenience sake, but if later on we find it easier to implement using something like an enum type then we will switch then.

4 PACKAGE: UTILTHINGS

The `UtilThings` package provides shared data structures and enumerations.

4.0.1 *Class: EntityStats.* **Variables : int**

- id, level, hp, damage, range
- productionRate, goldCost, ironCost, lumberCost
- timeToCompletion, villageHallReq

Methods : int

- getDamage(), getRange(), getProductionRate()
- getGoldCost(), getIronCost(), getLumberCost()
- getTimeToCompletion(), getVillageHallReq()
- printStats(e : EntityStats)

4.0.2 *Enumeration: EntityType.*

- SOLDIER, ARCHER, KNIGHT, CATAPULT
- RESOURCE_WORKER, WORKER
- GOLD_MINE, IRON_MINE, LUMBER_MILL
- FARM, VILLAGE_HALL
- ARCHER_TOWER, CANNON

4.0.3 *Enumeration: ResourceType.*

- GOLD, IRON, WOOD

4.1 UtilThings Discussion

Within this section we define what the Entities stats are, and store a class for all information about their class. have getter methods to get each stat, and a way to print out a Entities stats. We also have two enumeration types to make defining types of resources and entity types easier.

5 PACKAGE: UI

The `UI` package is responsible for user interaction and presentation.

5.0.1 *Class: UserInterface.* **Methods**

- (To be defined)

5.1 UI Package Discussion

There is nothing here currently. but this section will be used to store all UI Related Things.