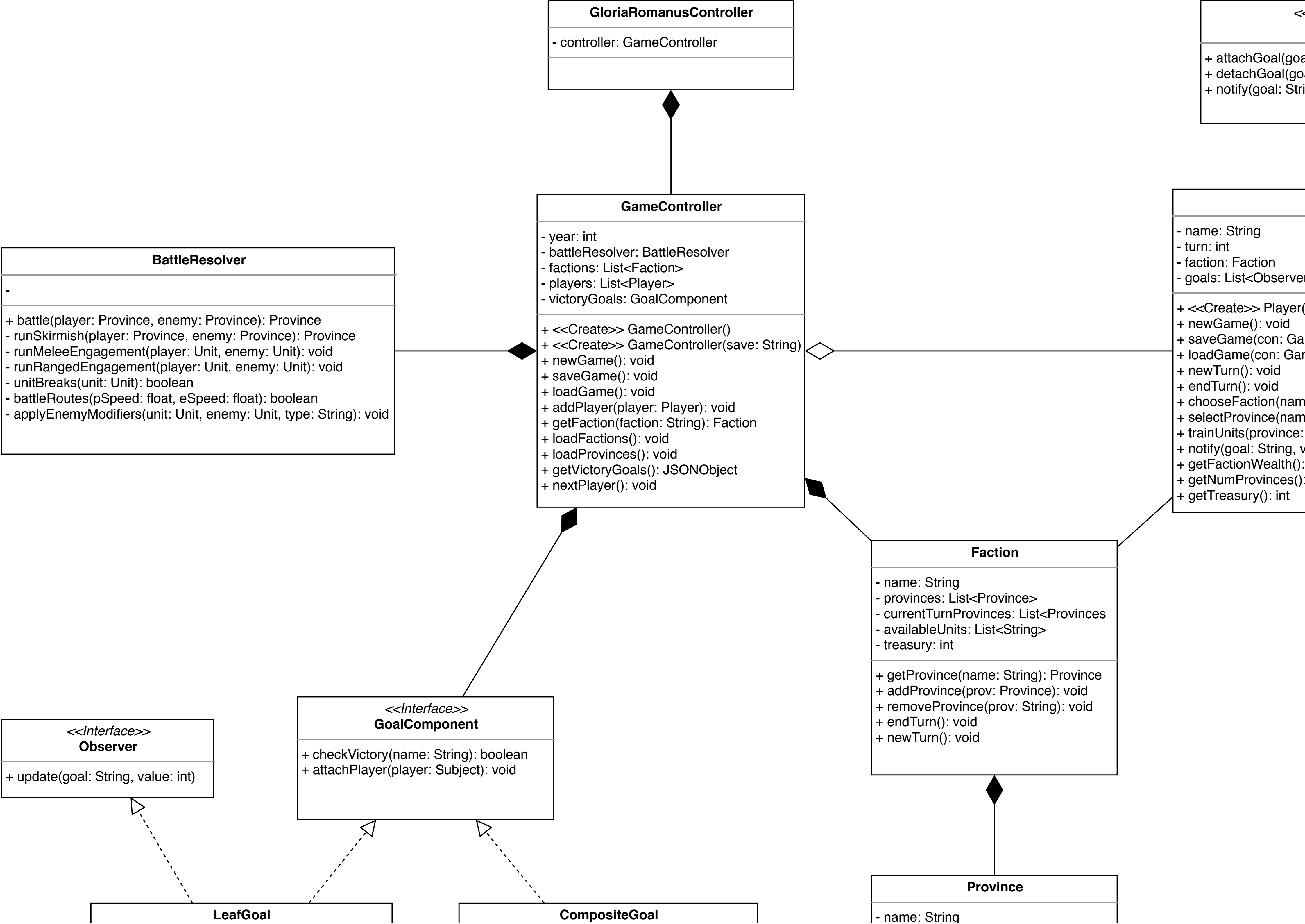


GloriaRomanusController

- controller: GameController

<<

+ attachGoal(goa
+ detachGoal(goa
+ notify(goal: Stri

GameController

- year: int
- battleResolver: BattleResolver
- factions: List<Faction>
- players: List<Player>
- victoryGoals: GoalComponent

+ <<Create>> GameController()
+ <<Create>> GameController(save: String)
+ newGame(): void
+ saveGame(): void
+ loadGame(): void
+ addPlayer(player: Player): void
+ getFaction(faction: String): Faction
+ loadFactions(): void
+ loadProvinces(): void
+ getVictoryGoals(): JSONObject
+ nextPlayer(): void

BattleResolver

-

+ battle(player: Province, enemy: Province): Province
- runSkirmish(player: Province, enemy: Province): Province
- runMeleeEngagement(player: Unit, enemy: Unit): void
- runRangedEngagement(player: Unit, enemy: Unit): void
- unitBreaks(unit: Unit): boolean
- battleRoutes(pSpeed: float, eSpeed: float): boolean
- applyEnemyModifiers(unit: Unit, enemy: Unit, type: String): void

- name: String
- turn: int
- faction: Faction
- goals: List<Observer

+ <<Create>> Player(
+ newGame(): void
+ saveGame(con: Ga
+ loadGame(con: Gar
+ newTurn(): void
+ endTurn(): void
+ chooseFaction(nam
+ selectProvince(nam
+ trainUnits(province:
+ notify(goal: String, v
+ getFactionWealth():
+ getNumProvinces():
+ getTreasury(): int

Faction

- name: String
- provinces: List<Province>
- currentTurnProvinces: List<Provinces
- availableUnits: List<String>
- treasury: int

+ getProvince(name: String): Province
+ addProvince(prov: Province): void
+ removeProvince(prov: String): void
+ endTurn(): void
+ newTurn(): void

<<Interface>>
Observer

+ update(goal: String, value: int)

<<Interface>>
GoalComponent

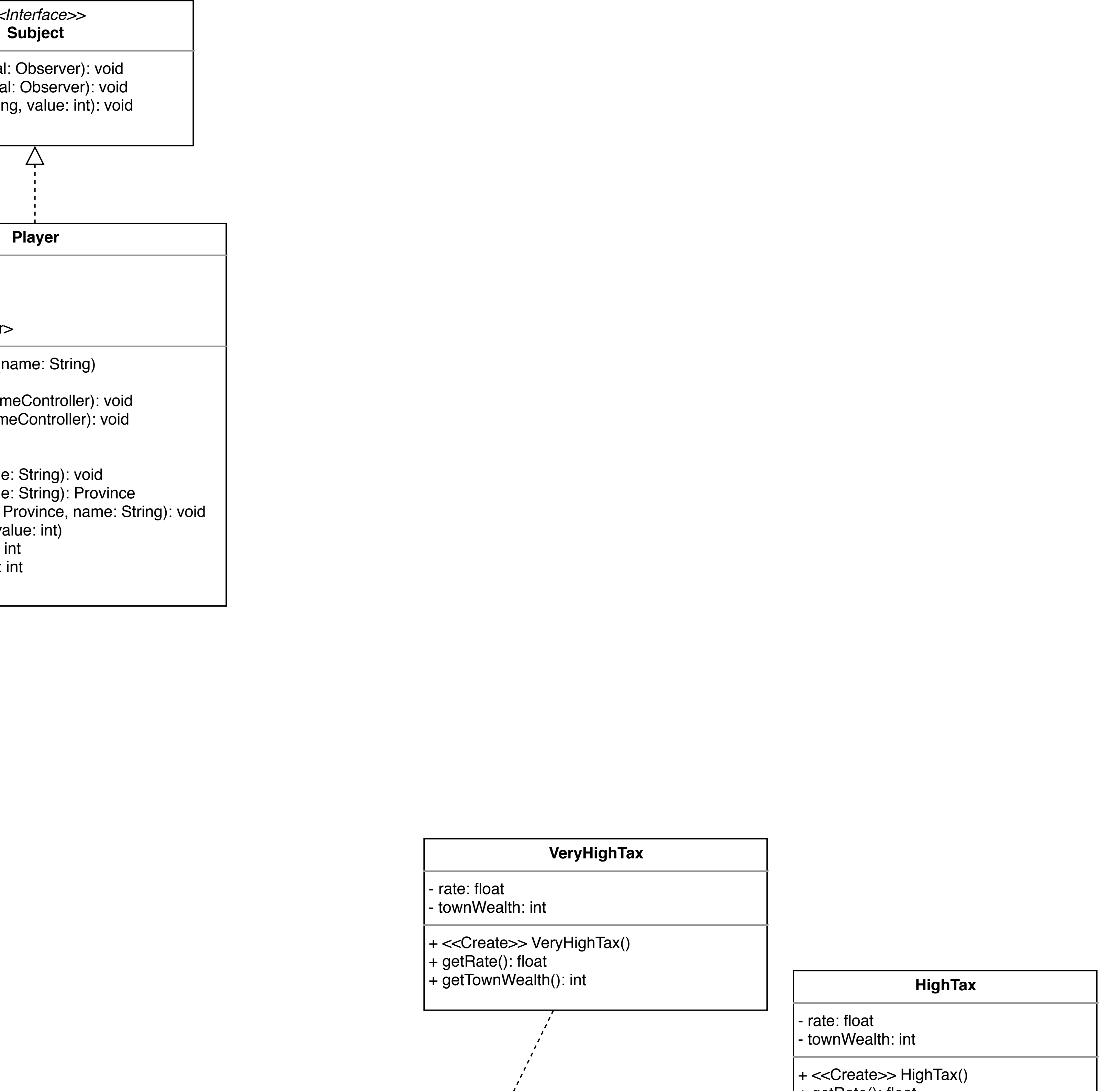
+ checkVictory(name: String): boolean
+ attachPlayer(player: Subject): void

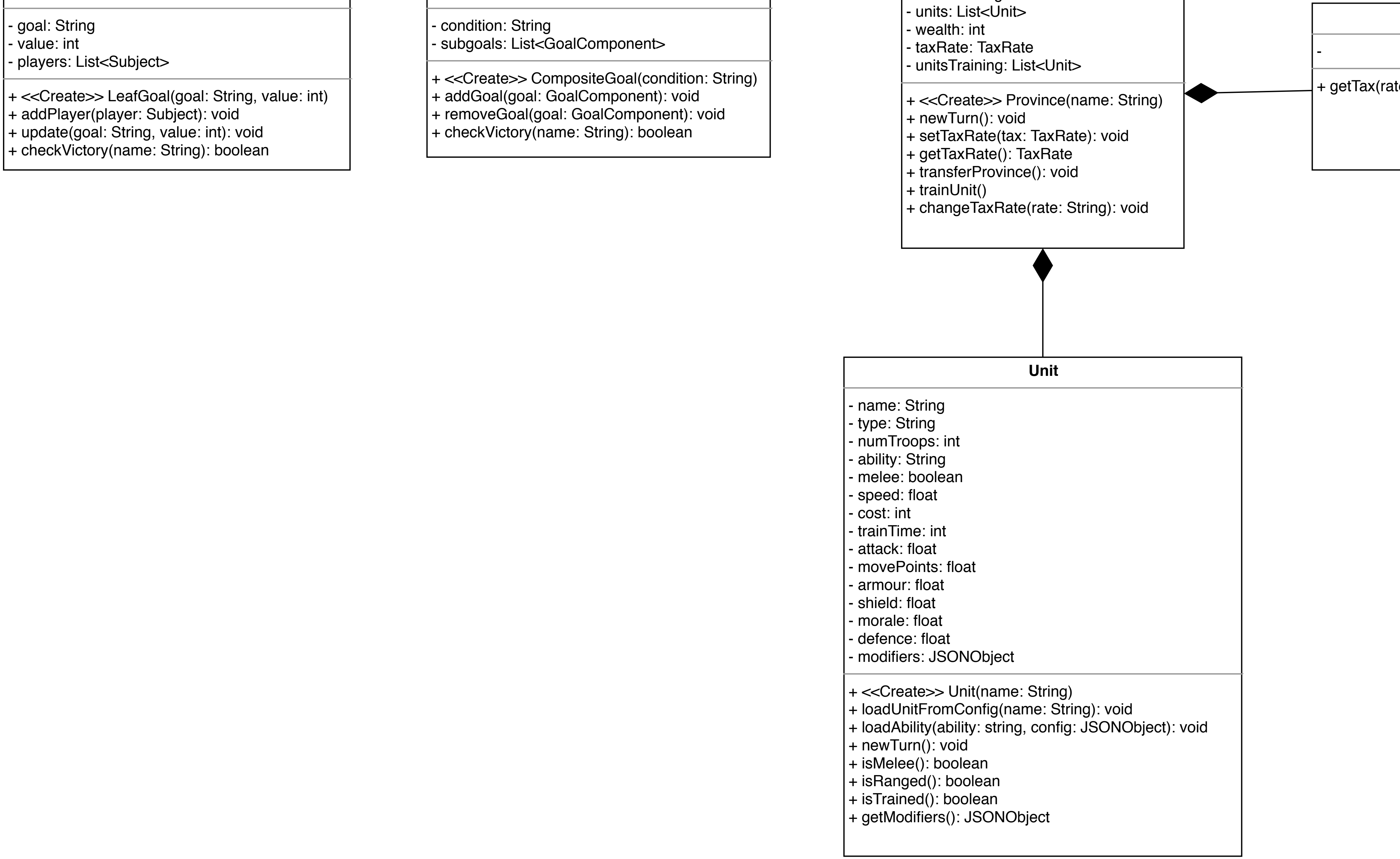
LeafGoal

CompositeGoal

Province

- name: String


<Interface>>
Subject
al: Observer): void
al: Observer): void
ng, value: int): void
Player
r>
name: String)
meController): void
meController): void
e: String): void
e: String): Province
Province, name: String): void
value: int)
int
int
VeryHighTax
- rate: float
- townWealth: int
+ <<Create>> VeryHighTax()
+ getRate(): float
+ getTownWealth(): int
HighTax
- rate: float
- townWealth: int
+ <<Create>> HighTax()



- goal: String
- value: int
- players: List<Subject>

+ <<Create>> LeafGoal(goal: String, value: int)
+ addPlayer(player: Subject): void
+ update(goal: String, value: int): void
+ checkVictory(name: String): boolean

- condition: String
- subgoals: List<GoalComponent>

+ <<Create>> CompositeGoal(condition: String)
+ addGoal(goal: GoalComponent): void
+ removeGoal(goal: GoalComponent): void
+ checkVictory(name: String): boolean

- units: List<Unit>
- wealth: int
- taxRate: TaxRate
- unitsTraining: List<Unit>

+ <<Create>> Province(name: String)
+ newTurn(): void
+ setTaxRate(tax: TaxRate): void
+ getTaxRate(): TaxRate
+ transferProvince(): void
+ trainUnit()
+ changeTaxRate(rate: String): void

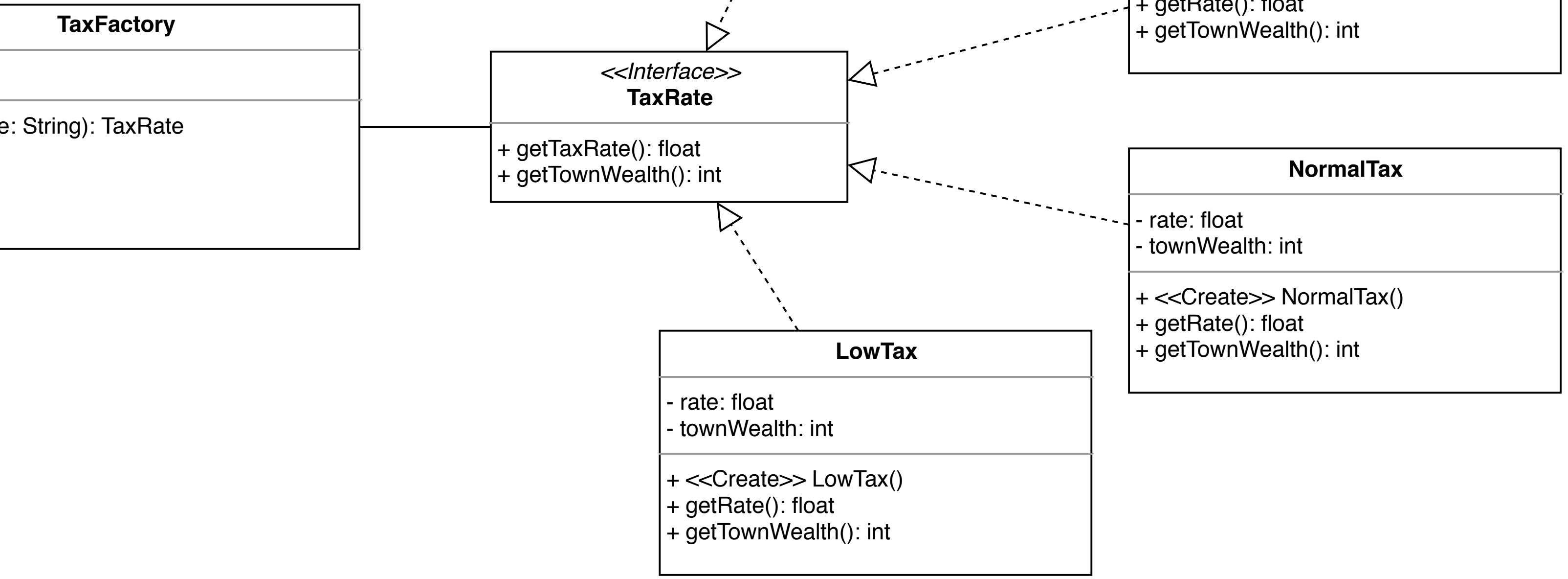
-

+ getTax(rat

Unit

- name: String
- type: String
- numTroops: int
- ability: String
- melee: boolean
- speed: float
- cost: int
- trainTime: int
- attack: float
- movePoints: float
- armour: float
- shield: float
- morale: float
- defence: float
- modifiers: JSONObject

+ <<Create>> Unit(name: String)
+ loadUnitFromConfig(name: String): void
+ loadAbility(ability: string, config: JSONObject): void
+ newTurn(): void
+ isMelee(): boolean
+ isRanged(): boolean
+ isTrained(): boolean
+ getModifiers(): JSONObject


TaxFactory
e: String): TaxRate
<<Interface>>
TaxRate
+ getTaxRate(): float
+ getTownWealth(): int
+ getRate(): float
+ getTownWealth(): int
NormalTax
- rate: float
- townWealth: int
+ <<Create>> NormalTax()
+ getRate(): float
+ getTownWealth(): int
LowTax
- rate: float
- townWealth: int
+ <<Create>> LowTax()
+ getRate(): float
+ getTownWealth(): int