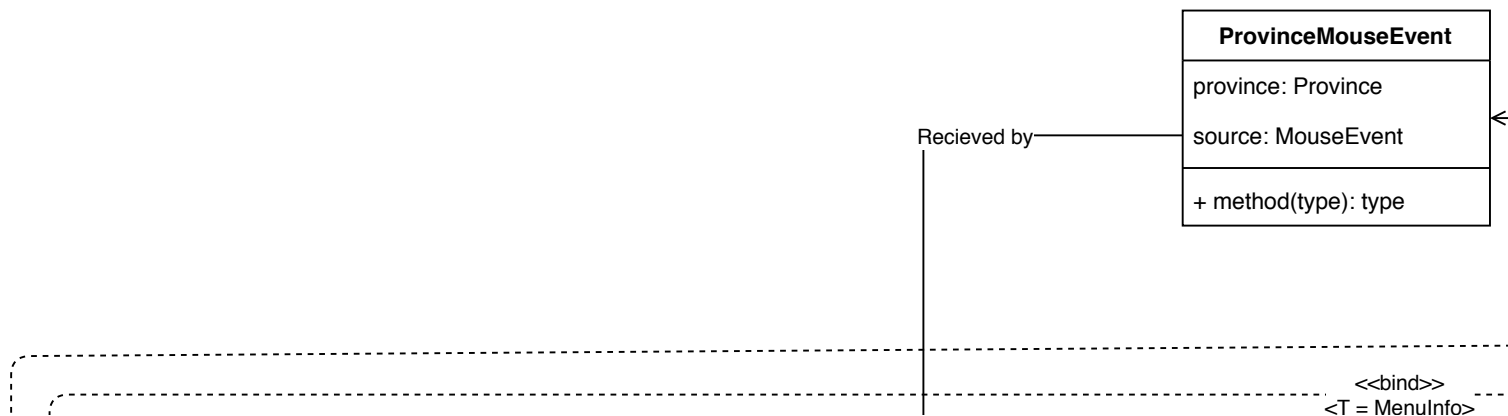
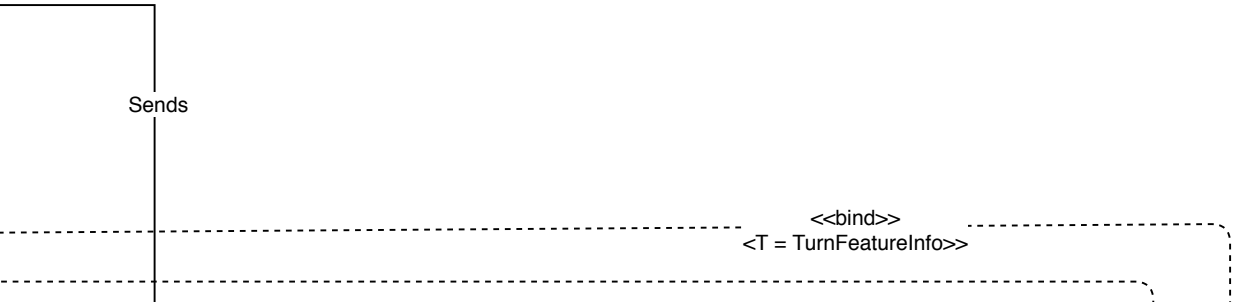
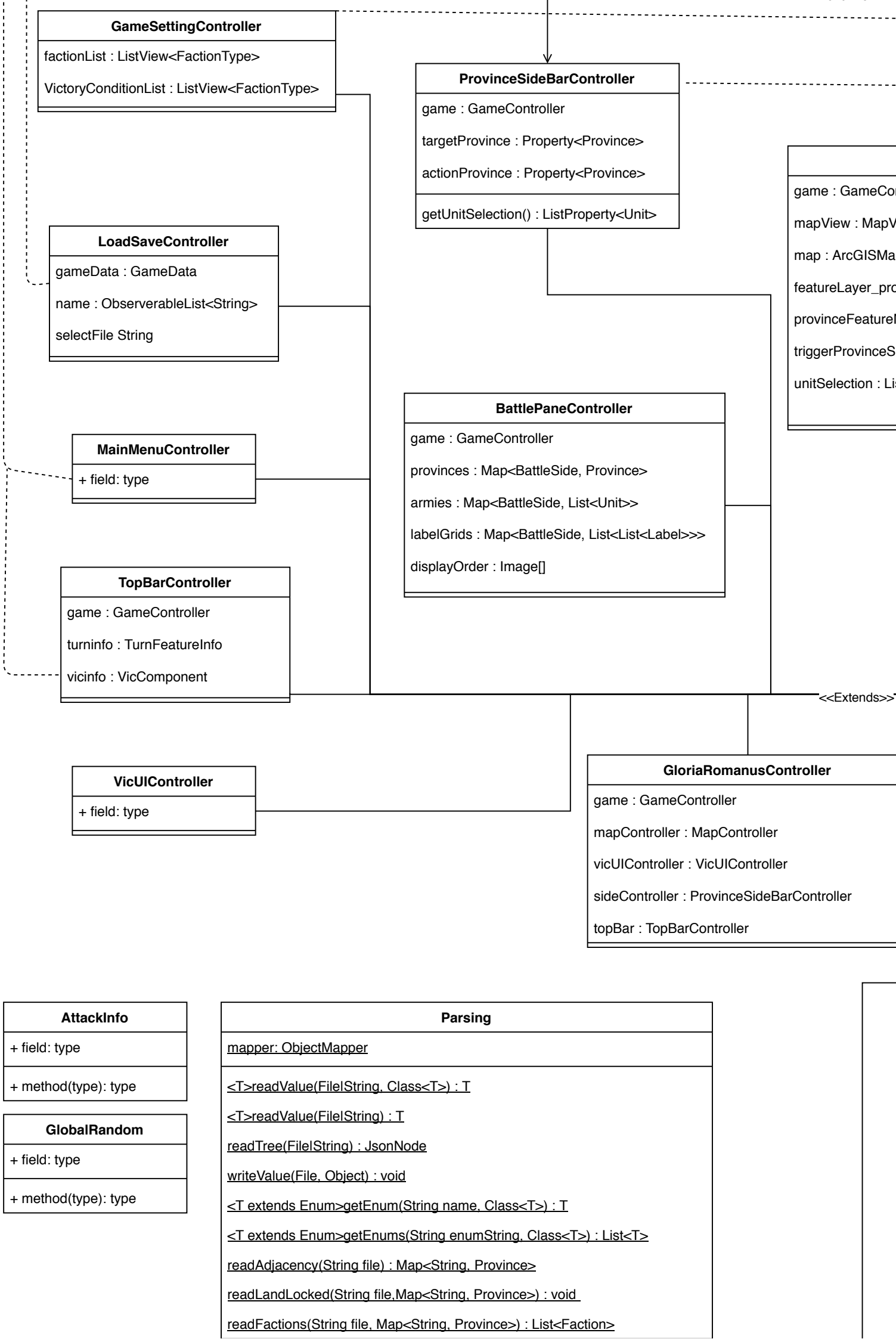
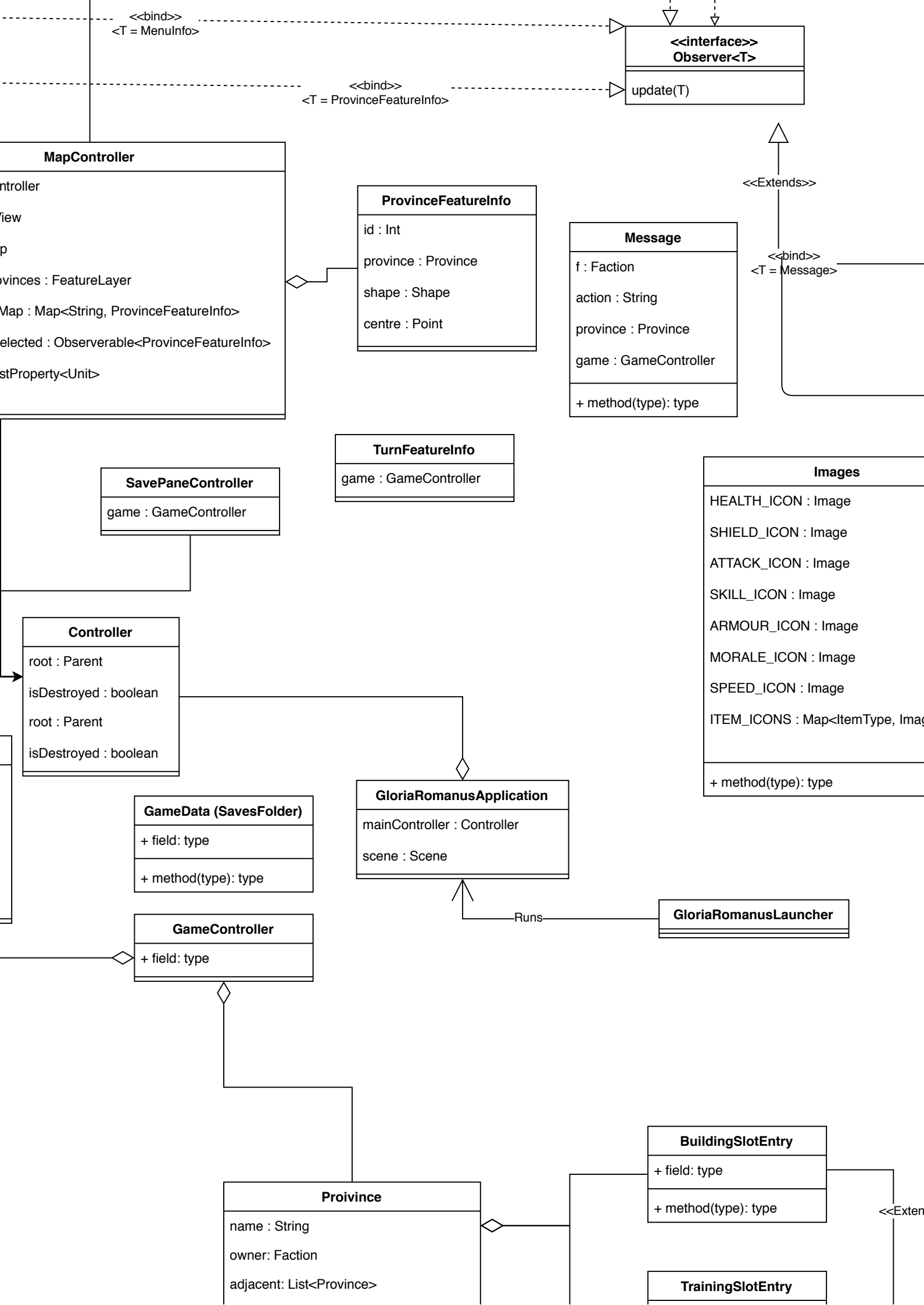
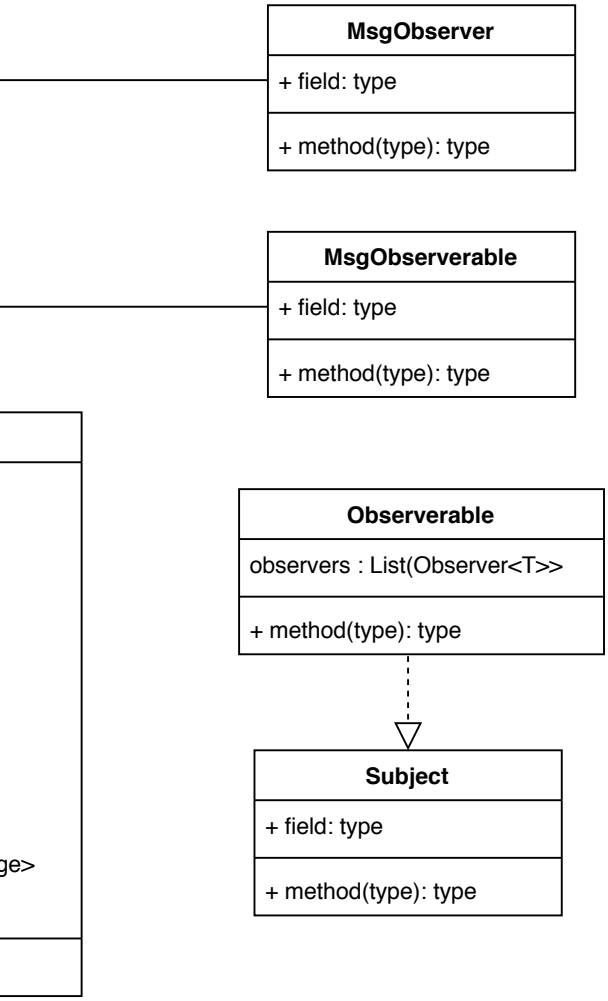

ProvinceMouseEvent
province: Province
source: MouseEvent
+ method(type): type
Recieved by
<<bind>>
<T = MenuInfo>

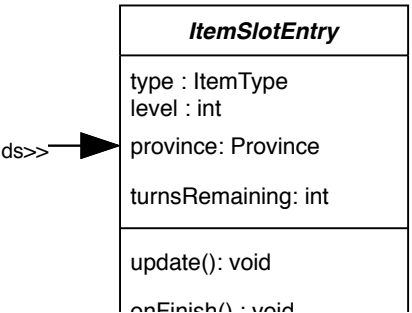

Sends
<<bind>>
<T = TurnFeatureInfo>>

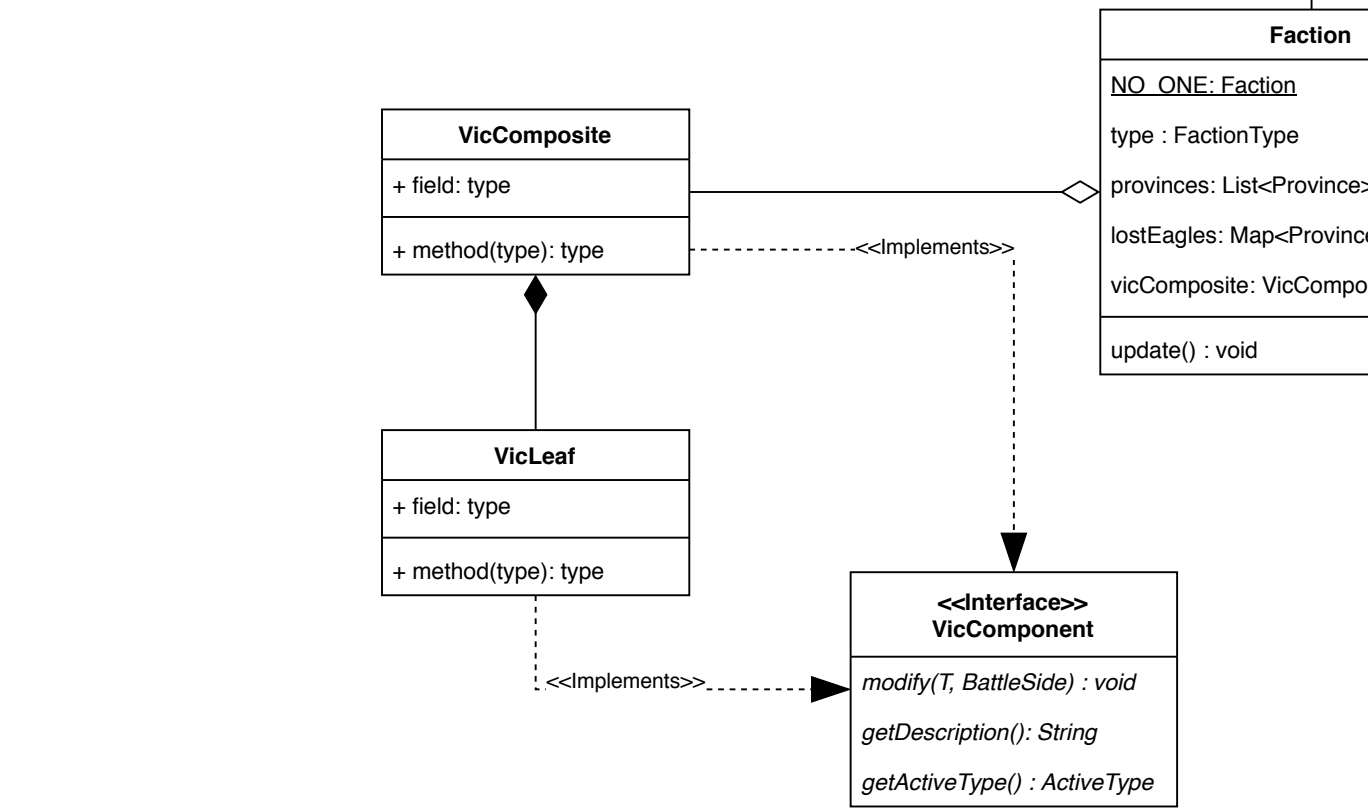

GameSettingController
factionList : ListView<FactionType>
VictoryConditionList : ListView<FactionType>
ProvinceSideBarController
game : GameController
targetProvince : Property<Province>
actionProvince : Property<Province>
getUnitSelection() : ListProperty<Unit>
LoadSaveController
gameData : GameData
name : ObserverableList<String>
selectFile String
BattlePaneController
game : GameController
provinces : Map<BattleSide, Province>
armies : Map<BattleSide, List<Unit>>
labelGrids : Map<BattleSide, List<List<Label>>>
displayOrder : Image[]
MainMenuController
+ field: type
TopBarController
game : GameController
turninfo : TurnFeatureInfo
vicinfo : VicComponent
<<Extends>>
VicUIController
+ field: type
GloriaRomanusController
game : GameController
mapController : MapController
vicUIController : VicUIController
sideController : ProvinceSideBarController
topBar : TopBarController
AttackInfo
+ field: type
+ method(type): type
GlobalRandom
+ field: type
+ method(type): type
Parsing
mapper: ObjectMapper
<T>readValue(FileIString, Class<T>) : T
<T>readValue(FileIString) : T
readTree(FileIString) : JsonNode
writeValue(File, Object) : void
<T extends Enum>getEnum(String name, Class<T>) : T
<T extends Enum>getEnums(String enumString, Class<T>) : List<T>
readAdjacency(String file) : Map<String, Province>
readLandLocked(String file,Map<String, Province>) : void
readFactions(String file, Map<String, Province>) : List<Faction>

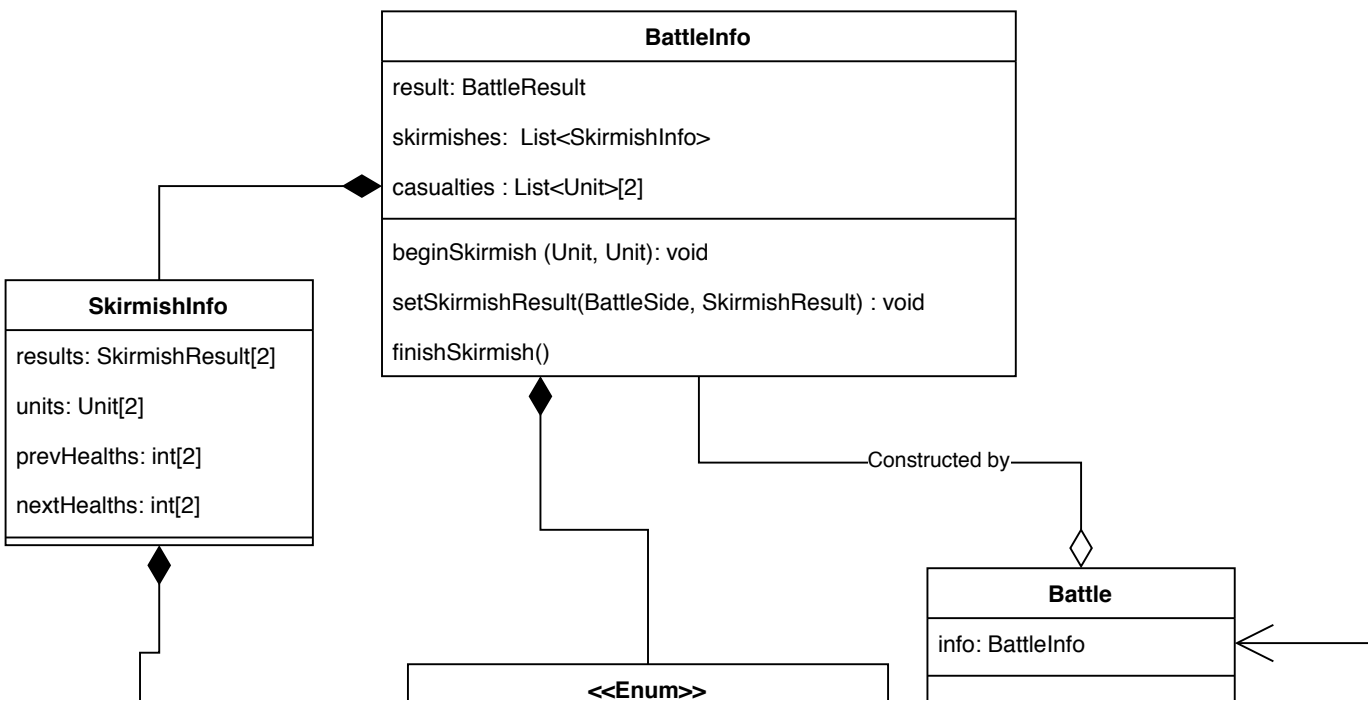

<<bind>>
<T = MenuInfo>
<<interface>>
Observer<T>
update(T)
<<bind>>
<T = ProvinceFeatureInfo>
MapController
ntroller
iew
p
ovinces : FeatureLayer
Map : Map<String, ProvinceFeatureInfo>
elected : Observerable<ProvinceFeatureInfo>
stProperty<Unit>
ProvinceFeatureInfo
id : Int
province : Province
shape : Shape
centre : Point
<<Extends>>
Message
f : Faction
action : String
province : Province
game : GameController
+ method(type): type
<<bind>>
<T = Message>
TurnFeatureInfo
game : GameController
SavePaneController
game : GameController
Images
HEALTH_ICON : Image
SHIELD_ICON : Image
ATTACK_ICON : Image
SKILL_ICON : Image
ARMOUR_ICON : Image
MORALE_ICON : Image
SPEED_ICON : Image
ITEM_ICONS : Map<ItemType, Imag
+ method(type): type
Controller
root : Parent
isDestroyed : boolean
root : Parent
isDestroyed : boolean
GameData (SavesFolder)
+ field: type
+ method(type): type
GloriaRomanusApplication
mainController : Controller
scene : Scene
Runs
GloriaRomanusLauncher
GameController
+ field: type
BuildingSlotEntry
+ field: type
+ method(type): type
<<Exten
Proivince
name : String
owner: Faction
adjacent: List<Province>
TrainingSlotEntry

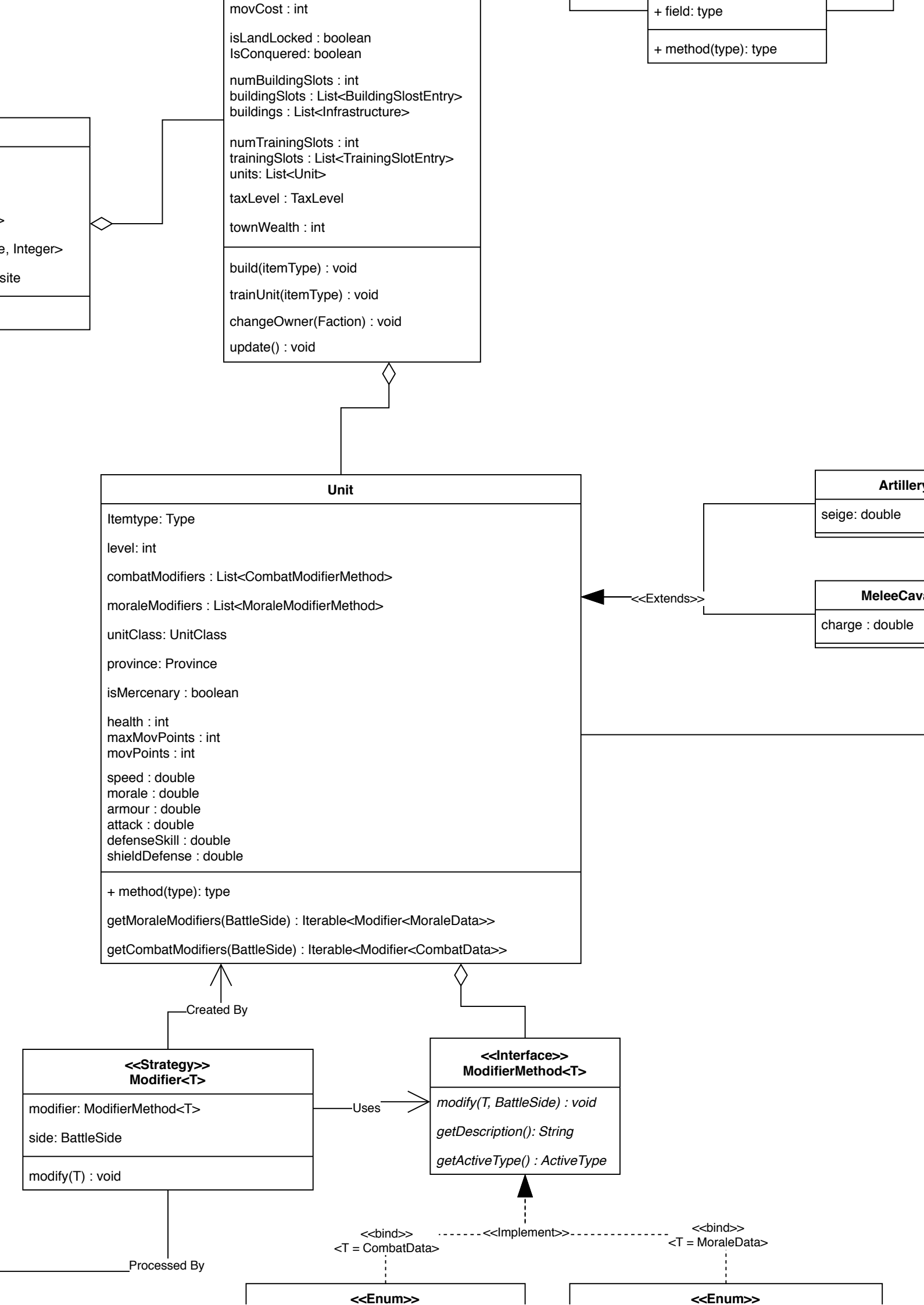
- All controllers aggregates
GameController






Faction
NO_ONE: Faction
type : FactionType
provinces: List<Province>
lostEagles: Map<Provinc
vicComposite: VicCompo
update() : void
VicComposite
+ field: type
+ method(type): type
<<Implements>>
VicLeaf
+ field: type
+ method(type): type
<<Implements>>
<<Interface>>
VicComponent
modify(T, BattleSide) : void
getDescription(): String
getActiveType() : ActiveType

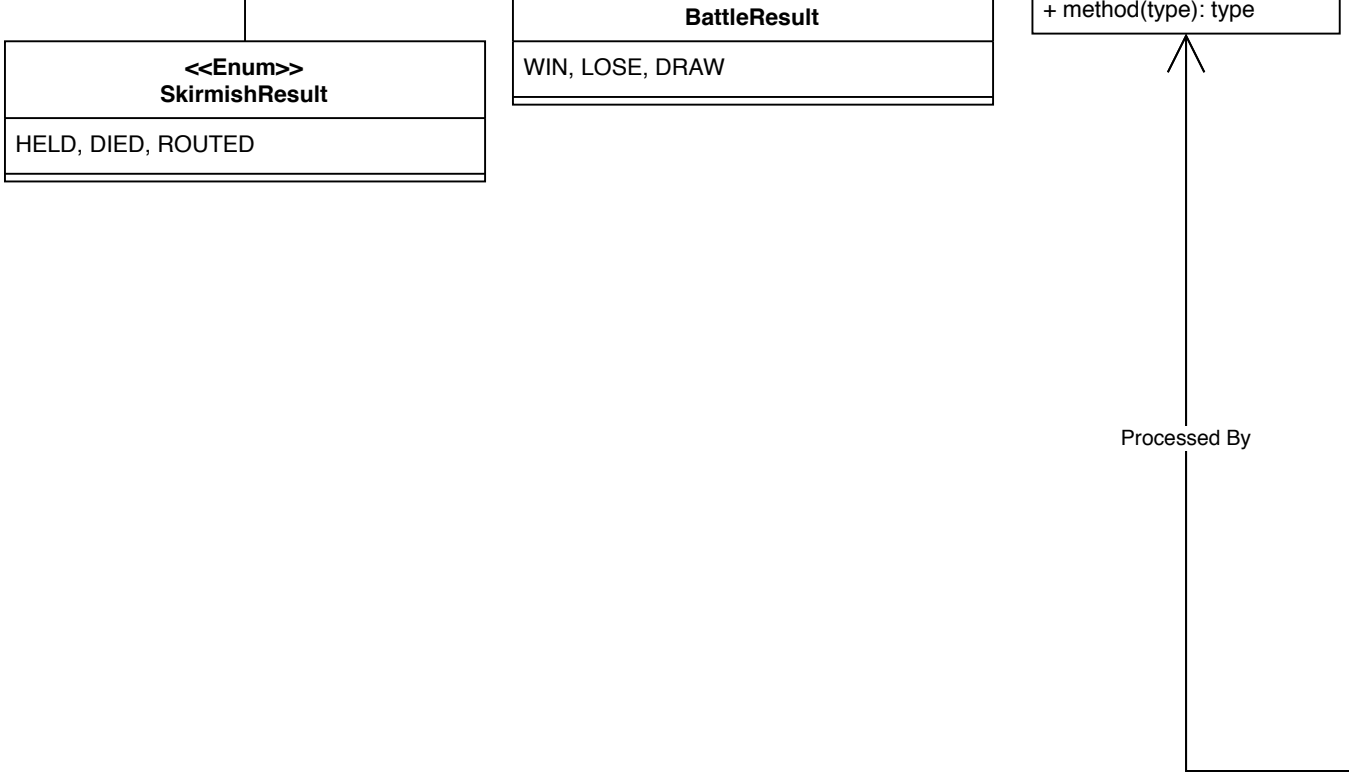

BattleInfo
result: BattleResult
skirmishes: List<SkirmishInfo>
casualties : List<Unit>[2]
beginSkirmish (Unit, Unit): void
setSkirmishResult(BattleSide, SkirmishResult) : void
finishSkirmish()
SkirmishInfo
results: SkirmishResult[2]
units: Unit[2]
prevHealths: int[2]
nextHealths: int[2]
Constructed by
Battle
info: BattleInfo
<<Enum>>

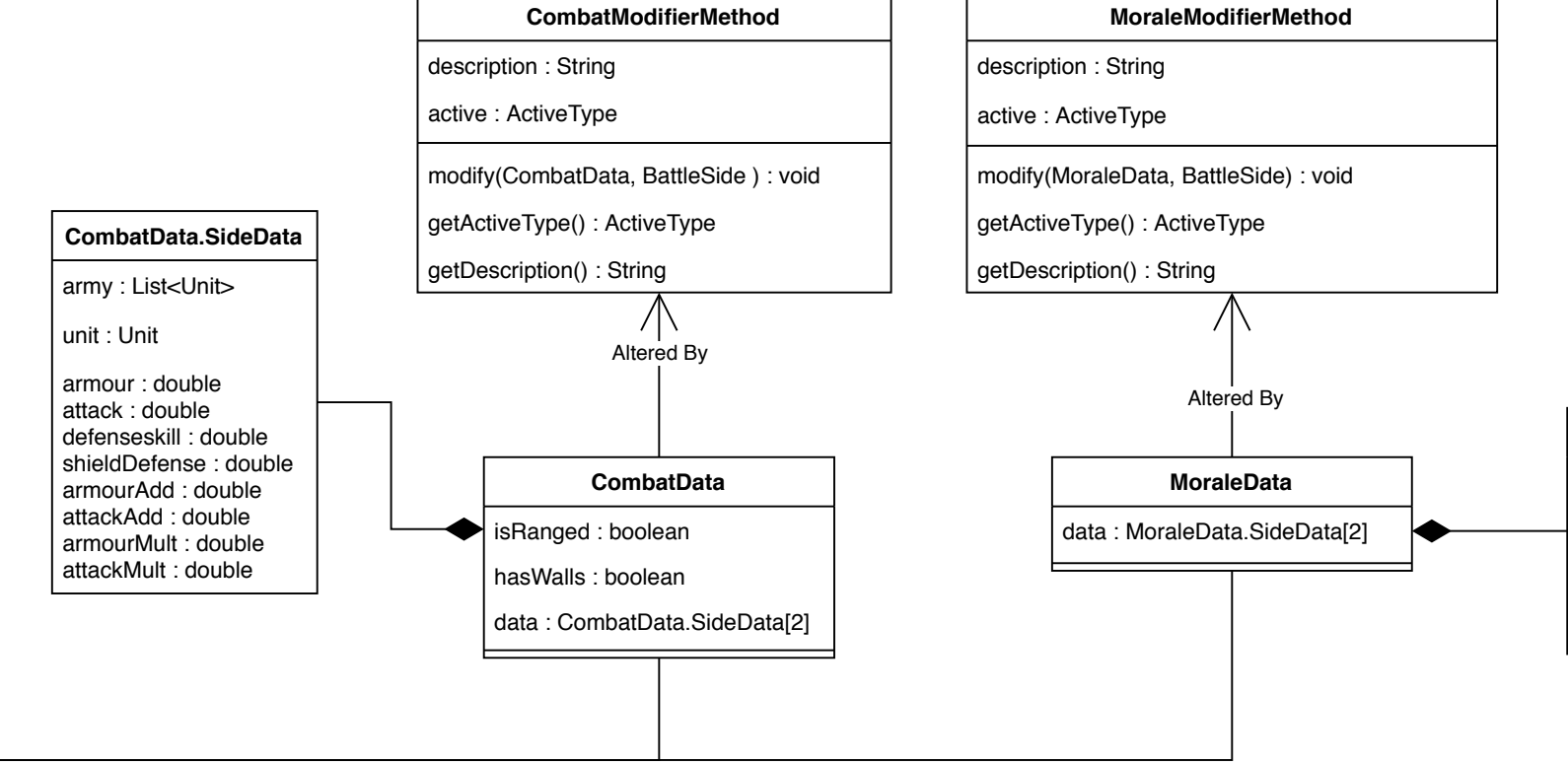

movCost : int
isLandLocked : boolean
IsConquered: boolean
numBuildingSlots : int
buildingSlots : List<BuildingSlostEntry>
buildings : List<Infrastructure>
numTrainingSlots : int
trainingSlots : List<TrainingSlotEntry>
units: List<Unit>
taxLevel : TaxLevel
townWealth : int
build(itemType) : void
trainUnit(itemType) : void
changeOwner(Faction) : void
update() : void
+ field: type
+ method(type): type
Unit
Itemtype: Type
level: int
combatModifiers : List<CombatModifierMethod>
moraleModifiers : List<MoraleModifierMethod>
unitClass: UnitClass
province: Province
isMercenary : boolean
health : int
maxMovPoints : int
movPoints : int
speed : double
morale : double
armour : double
attack : double
defenseSkill : double
shieldDefense : double
+ method(type): type
getMoraleModifiers(BattleSide) : Iterable<Modifier<MoraleData>>
getCombatModifiers(BattleSide) : Iterable<Modifier<CombatData>>
<<Extends>>
seige: double
charge : double
Created By
<<Strategy>>
Modifier<T>
modifier: ModifierMethod<T>
side: BattleSide
modify(T) : void
Uses
<<Interface>>
ModifierMethod<T>
modify(T, BattleSide) : void
getDescription(): String
getActiveType() : ActiveType
<<Implement>>
<<bind>>
<T = CombatData>
<<bind>>
<T = MoraleData>
Processed By

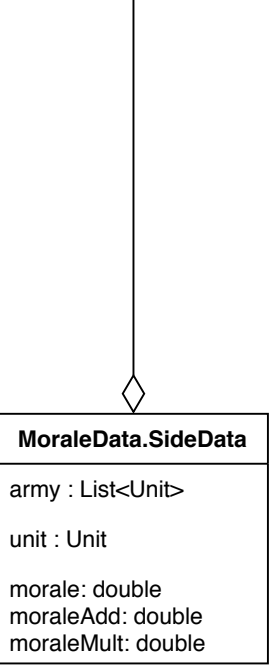
onFinish() : void

y

alry


<<Enum>>
SkirmishResult
HELD, DIED, ROUTED
BattleResult
WIN, LOSE, DRAW
+ method(type): type
Processed By


CombatModifierMethod
description : String
active : ActiveType
modify(CombatData, BattleSide) : void
getActiveType() : ActiveType
getDescription() : String
MoraleModifierMethod
description : String
active : ActiveType
modify(MoraleData, BattleSide) : void
getActiveType() : ActiveType
getDescription() : String
CombatData.SideData
army : List<Unit>
unit : Unit
armour : double
attack : double
defenseskill : double
shieldDefense : double
armourAdd : double
attackAdd : double
armourMult : double
attackMult : double
Altered By
Altered By
CombatData
isRanged : boolean
hasWalls : boolean
data : CombatData.SideData[2]
MoraleData
data : MoraleData.SideData[2]


MoraleData.SideData
army : List<Unit>
unit : Unit
morale: double
moraleAdd: double
moraleMult: double