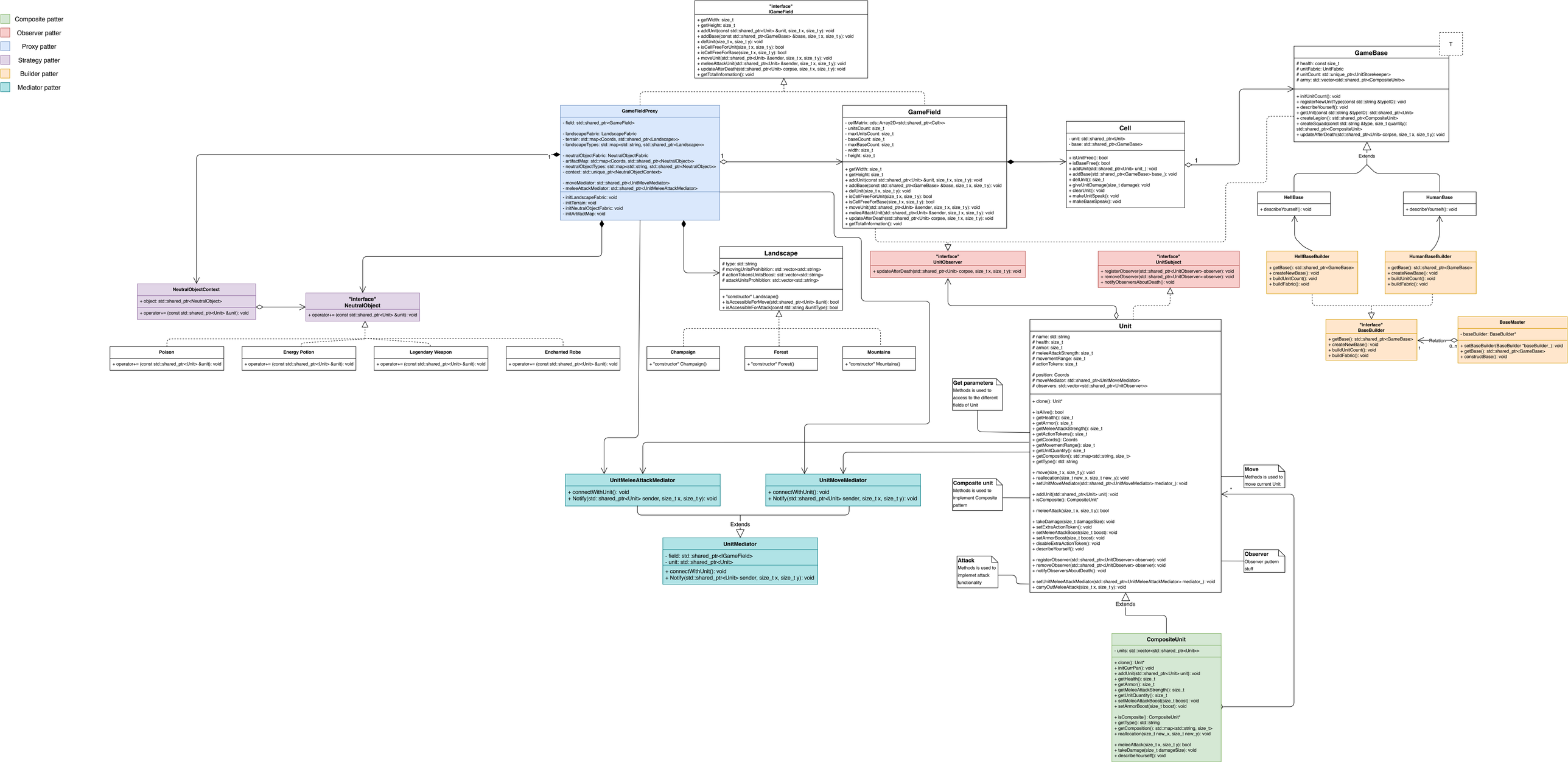
Composite patter
Observer patter
Proxy patter
Strategy patter
Builder patter
Mediator patter
"interface"
IGameField
+ getWidth: size_t
+ getHeight: size_t
+ addUnit(const std::shared_ptr<Unit> &unit, size_t x, size_t y): void
+ addBase(const std::shared_ptr<GameBase> &base, size_t x, size_t y): void
+ delUnit(size_t x, size_t y): void
+ isCellFreeForUnit(size_t x, size_t y): bool
+ isCellFreeForBase(size_t x, size_t y): bool
+ moveUnit(std::shared_ptr<Unit> &sender, size_t x, size_t y): void
+ meleeAttackUnit(std::shared_ptr<Unit> &sender, size_t x, size_t y): void
+ updateAfterDeath(std::shared_ptr<Unit> corpse, size_t x, size_t y): void
+ getTotalInformation(): void
GameFieldProxy
- field: std::shared_ptr<GameField>
- landscapeFabric: LandscapeFabric
- terrain: std::map<Coords, std::shared_ptr<Landscape>>
- landscapeTypes: std::map<std::string, std::shared_ptr<Landscape>>
- neutralObjectFabric: NeutralObjectFabric
- artifactMap: std::map<Coords, std::shared_ptr<NeutralObject>>
- neutralObjectTypes: std::map<std::string, std::shared_ptr<NeutralObject>>
- context: std::unique_ptr<NeutralObjectContext>
- moveMediator: std::shared_ptr<UnitMoveMediator>
- meleeAttackMediator: std::shared_ptr<UnitMeleeAttackMediator>
- initLandscapeFabric: void
- initTerrain: void
- initNeutralObjectFabric: void
- initArtifactMap: void
GameField
- cellMatrix: cds::Array2D<std::shared_ptr<Cell>>
- unitsCount: size_t
- maxUnitsCount: size_t
- baseCount: size_t
- maxBaseCount: size_t
- width: size_t
- height: size_t
+ getWidth: size_t
+ getHeight: size_t
+ addUnit(const std::shared_ptr<Unit> &unit, size_t x, size_t y): void
+ addBase(const std::shared_ptr<GameBase> &base, size_t x, size_t y): void
+ delUnit(size_t x, size_t y): void
+ isCellFreeForUnit(size_t x, size_t y): bool
+ isCellFreeForBase(size_t x, size_t y): bool
+ moveUnit(std::shared_ptr<Unit> &sender, size_t x, size_t y): void
+ meleeAttackUnit(std::shared_ptr<Unit> &sender, size_t x, size_t y): void
+ updateAfterDeath(std::shared_ptr<Unit> corpse, size_t x, size_t y): void
+ getTotalInformation(): void
Cell
- unit: std::shared_ptr<Unit>
- base: std::shared_ptr<GameBase>
+ isUnitFree(): bool
+ isBaseFree(): bool
+ addUnit(std::shared_ptr<Unit> unit_): void
+ addBase(std::shared_ptr<GameBase> base_): void
+ delUnit(): size_t
+ giveUnitDamage(size_t damage): void
+ clearUnit(): void
+ makeUnitSpeak(): void
+ makeBaseSpeak(): void
T
GameBase
health: const size_t
unitFabric: UnitFabric
unitCount: std::unique_ptr<UnitStorekeeper>
army: std::vector<std::shared_ptr<CompositeUnit>>
+ initUnitCount(): void
+ registerNewUnitType(const std::string &typeID): void
+ describeYourself(): void
+ getUnit(const std::string &typeID): std::shared_ptr<Unit>
+ createLegion(): std::shared_ptr<CompositeUnit>
+ createSquad(const std::string &type, size_t quantity): std::shared_ptr<CompositeUnit>
+ updateAfterDeath(std::shared_ptr<Unit> corpse, size_t x, size_t y): void
Extends
HellBase
+ describeYourself(): void
HumanBase
+ describeYourself(): void
"interface"
UnitObserver
+ updateAfterDeath(std::shared_ptr<Unit> corpse, size_t x, size_t y): void
"interface"
UnitSubject
+ registerObserver(std::shared_ptr<UnitObserver> observer): void
+ removeObserver(std::shared_ptr<UnitObserver> observer): void
+ notifyObserversAboutDeath(): void
HellBaseBuilder
+ getBase(): std::shared_ptr<GameBase>
+ createNewBase(): void
+ buildUnitCount(): void
+ buildFabric(): void
HumanBaseBuilder
+ getBase(): std::shared_ptr<GameBase>
+ createNewBase(): void
+ buildUnitCount(): void
+ buildFabric(): void
"interface"
BaseBuilder
+ getBase(): std::shared_ptr<GameBase>
+ createNewBase(): void
+ buildUnitCount(): void
+ buildFabric(): void
Relation
0..n
1
BaseMaster
- baseBuilder: BaseBuilder*
+ setBaseBuilder(BaseBuilder *baseBuilder_): void
+ getBase(): std::shared_ptr<GameBase>
+ constructBase(): void
NeutralObjectContext
+ object: std::shared_ptr<NeutralObject>
+ operator+= (const std::shared_ptr<Unit> &unit): void
"interface"
NeutralObject
+ operator+= (const std::shared_ptr<Unit> &unit): void
Poison
+ operator+= (const std::shared_ptr<Unit> &unit): void
Energy Potion
+ operator+= (const std::shared_ptr<Unit> &unit): void
Legendary Weapon
+ operator+= (const std::shared_ptr<Unit> &unit): void
Enchanted Robe
+ operator+= (const std::shared_ptr<Unit> &unit): void
Landscape
type: std::string
movingUnitsProhibition: std::vector<std::string>
actionTokensUnitsBoost: std::vector<std::string>
attackUnitsProhibition: std::vector<std::string>
+ "constructor" Landscape()
+ isAccessibleForMove(std::shared_ptr<Unit> &unit): bool
+ isAccessibleForAttack(const std::string &unitType): bool
Champaign
+ "constructor" Champaign()
Forest
+ "constructor" Forest()
Mountains
+ "constructor" Mountains()
Unit
name: std::string
health: size_t
armor: size_t
meleeAttackStrength: size_t
movementRange: size_t
actionTokens: size_t
position: Coords
moveMediator: std::shared_ptr<UnitMoveMediator>
observers: std::vector<std::shared_ptr<UnitObserver>>
+ clone(): Unit*
+ isAlive(): bool
+ getHealth(): size_t
+ getArmor(): size_t
+ getMeleeAttackStrength(): size_t
+ getActionTokens(): size_t
+ getCoords(): Coords
+ getMovementRange(): size_t
+ getUnitQuantity(): size_t
+ getComposition(): std::map<std::string, size_t>
+ getType(): std::string
+ move(size_t x, size_t y): void
+ reallocation(size_t new_x, size_t new_y): void
+ setUnitMoveMediator(std::shared_ptr<UnitMoveMediator> mediator_): void
+ addUnit(std::shared_ptr<Unit> unit): void
+ isComposite(): CompositeUnit*
+ meleeAttack(size_t x, size_t y): bool
+ takeDamage(size_t damageSize): void
+ setExtraActionToken(): void
+ setMeleeAttackBoost(size_t boost): void
+ setArmorBoost(size_t boost): void
+ disableExtraActionToken(): void
+ describeYourself(): void
+ registerObserver(std::shared_ptr<UnitObserver> observer): void
+ removeObserver(std::shared_ptr<UnitObserver> observer): void
+ notifyObserversAboutDeath(): void
+ setUnitMeleeAttackMediator(std::shared_ptr<UnitMeleeAttackMediator> mediator_): void
+ carryOutMeleeAttack(size_t x, size_t y): void
Get parameters
Methods is used to access to the different fields of Unit
Composite unit
Methods is used to implement Composite pattern
Attack
Methods is used to implemet attack functionality
Move
Methods is used to move current Unit
Observer
Observer puttern stuff
Extends
CompositeUnit
- units: std::vector<std::shared_ptr<Unit>>
+ clone(): Unit*
+ initCurrPar(): void
+ addUnit(std::shared_ptr<Unit> unit): void
+ getHealth(): size_t
+ getArmor(): size_t
+ getMeleeAttackStrength(): size_t
+ getUnitQuantity(): size_t
+ setMeleeAttackBoost(size_t boost): void
+ setArmorBoost(size_t boost): void
+ isComposite(): CompositeUnit*
+ getType(): std::string
+ getComposition(): std::map<std::string, size_t>
+ reallocation(size_t new_x, size_t new_y): void
+ meleeAttack(size_t x, size_t y): bool
+ takeDamage(size_t damageSize): void
+ describeYourself(): void
UnitMeleeAttackMediator
+ connectWithUnit(): void
+ Notify(std::shared_ptr<Unit> sender, size_t x, size_t y): void
UnitMoveMediator
+ connectWithUnit(): void
+ Notify(std::shared_ptr<Unit> sender, size_t x, size_t y): void
Extends
UnitMediator
- field: std::shared_ptr<IGameField>
- unit: std::shared_ptr<Unit>
+ connectWithUnit(): void
+ Notify(std::shared_ptr<Unit> sender, size_t x, size_t y): void