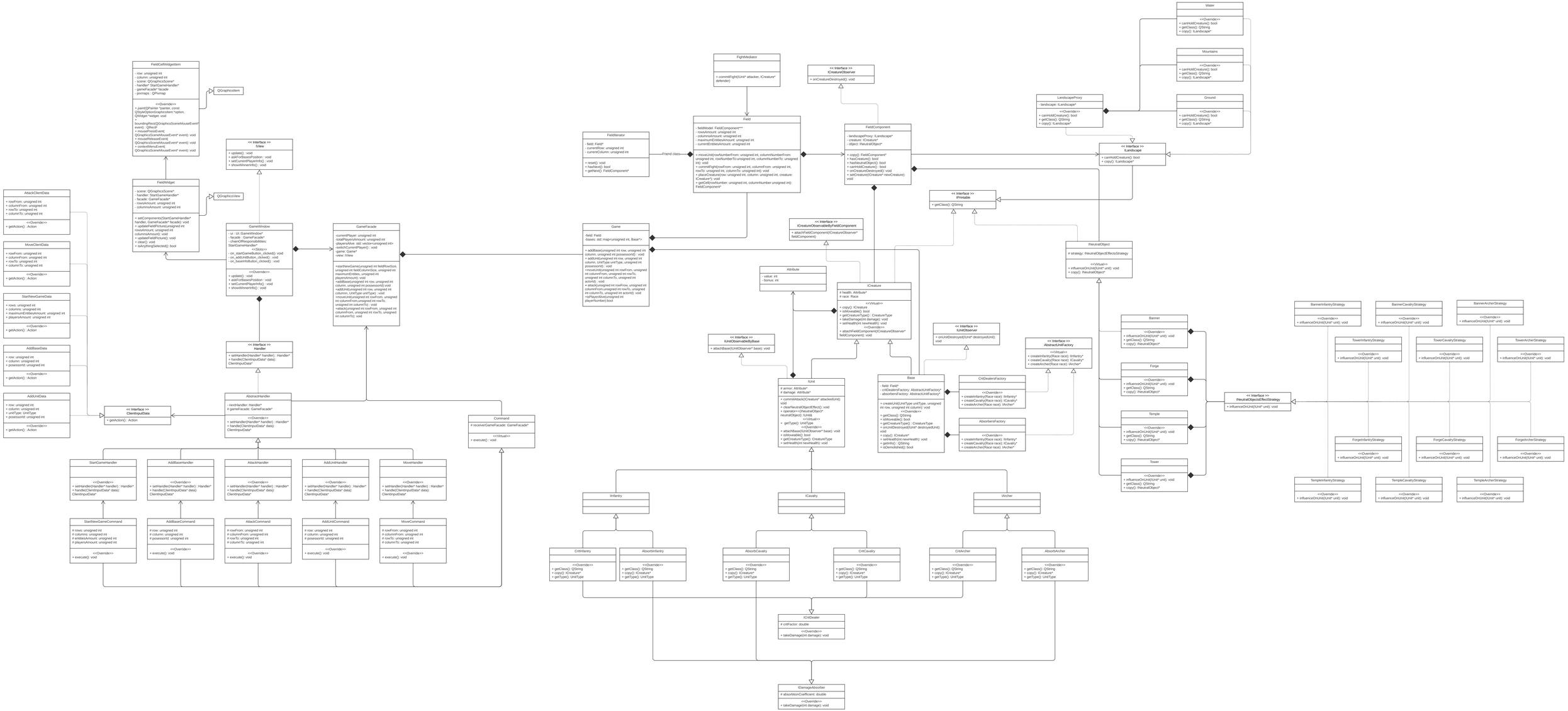
FieldCellWidgetItem
- row: unsigned int
- column: unsigned int
- scene: QGraphicsScene*
- handler* StartGameHandler*
- gameFacade* facade
- pixmaps : QPixmap
<<Override>>
+ paint(QPainter *painter, const QStyleOptionGraphicsItem *option, QWidget *widget: void
+ boundingRect(GraphicsSceneMouseEvent* event) : QRectF
+ mousePressEvent(QGraphicsSceneMouseEvent* event): void
+ mouseReleaseEvent(QGraphicsSceneMouseEvent* event): void
+ contextMenuEvent(QGraphicsSceneMouseEvent* event): void
QGraphicsItem
FieldWidget
- scene: QGraphicsScene*
- handler: StartGameHandler*
- facade: GameFacade*
- rowsAmount: unsigned int
- columnsAmount: unsigned int
+ setComponents(StartGameHandler* handler, GameFacade* facade): void
+ updateFieldPicture(unsigned int rowsAmount, unsigned int columnsAmount): void
+ updateFieldPicture(): void
+ clear(): void
+ isAnything(Selected): bool
QGraphicsView
<< Interface >>
IView
+ update() : void
+ askForBasesPosition : void
+ setCurrentPlayerInfo() : void
+ showWinnerInfo() : void
AttackClientData
+ rowFrom: unsigned int
+ columnFrom: unsigned int
+ rowTo: unsigned int
+ columnTo: unsigned int
<<Override>>
+ getAction() : Action
MoveClientData
+ rowFrom: unsigned int
+ columnFrom: unsigned int
+ rowTo: unsigned int
+ columnTo: unsigned int
<<Override>>
+ getAction() : Action
StartNewGameData
+ rows: unsigned int
+ columns: unsigned int
+ maximumEntitiesAmount: unsigned int
+ playersAmount: unsigned int
<<Override>>
+ getAction() : Action
AddBaseData
+ row: unsigned int
+ column: unsigned int
+ possessorId: unsigned int
<<Override>>
+ getAction() : Action
AddUnitData
+ row: unsigned int
+ column: unsigned int
+ unitType: UnitType
+ possessorId: unsigned int
<<Override>>
+ getAction() : Action
GameWindow
- ui : Ui::GameWindow*
- facade : GameFacade*
- chainOfResponsibilities: StartGameHandler*
<<Slots>>
- on_startGameButton_clicked(): void
- on_addUnitButton_clicked() : void
- on_baseInfoButton_clicked() : void
<<Override>>
+ update() : void
+ askForBasesPosition : void
+ setCurrentPlayerInfo() : void
+ showWinnerInfo() : void
GameFacade
-currentPlayer: unsigned int
-totalPlayersAmount: unsigned int
-playersAlive: std::vector<unsigned int>
-switchCurrentPlayer() : void
-game: Game*
-view: IView
+startNewGame(unsigned int fieldRowSize, unsigned int fieldColumnSize, unsigned int maximumEntities, unsigned int playersAmount): void
+addBase(unsigned int row, unsigned int column, unsigned int possessorId): void
+addUnit(unsigned int row, unsigned int column, UnitType unitType) : void
+moveUnit(unsigned int rowFrom, unsigned int columnFrom,unsigned int rowTo, unsigned int columnTo) : void
+attack(unsigned int rowFrom, unsigned int columnFrom,unsigned int rowTo, unsigned int columnTo, unsigned int actionId): void
+isPlayerAlive(unsigned int playerNumber):bool
<< Interface >>
Handler
+ setHandler(Handler* handler) : Handler*
+ handle(ClientInputData* data): ClientInputData*
AbstractHandler
-next Handler: Handler*
# gameFacade: GameFacade*
<<Override>>
+ setHandler(Handler* handler) : Handler*
+ handle(ClientInputData* data): ClientInputData*
<< Interface >>
ClientInputData
+ getAction() : Action
Command
# receiverGameFacade: GameFacade*
<<Virtual>>
+ execute() : void
StartGameHandler
<<Override>>
+ setHandler(Handler* handler) : Handler*
+ handle(ClientInputData* data): ClientInputData*
AddBaseHandler
<<Override>>
+ setHandler(Handler* handler) : Handler*
+ handle(ClientInputData* data): ClientInputData*
AttackHandler
<<Override>>
+ setHandler(Handler* handler) : Handler*
+ handle(ClientInputData* data): ClientInputData*
AddUnitHandler
<<Override>>
+ setHandler(Handler* handler) : Handler*
+ handle(ClientInputData* data): ClientInputData*
MoveHandler
<<Override>>
+ setHandler(Handler* handler) : Handler*
+ handle(ClientInputData* data): ClientInputData*
StartNewGameCommand
# rows: unsigned int
# columns: unsigned int
# entitiesAmount: unsigned int
# playersAmount: unsigned int
<<Override>>
+ execute() : void
AddBaseCommand
# row: unsigned int
# column: unsigned int
# possessorId: unsigned int
<<Override>>
+ execute() : void
AttackCommand
# rowFrom: unsigned int
# columnFrom: unsigned int
# rowTo: unsigned int
# columnTo: unsigned int
<<Override>>
+ execute() : void
AddUnitCommand
# row: unsigned int
# column: unsigned int
# possessorId: unsigned int
<<Override>>
+ execute() : void
MoveCommand
# rowFrom: unsigned int
# columnFrom: unsigned int
# rowTo: unsigned int
# columnTo: unsigned int
<<Override>>
+ execute() : void
FightMediator
+ commitFight(IUnit* attacker, ICreature* defender)
Field
- fieldModel: FieldComponent***
- rowsAmount: unsigned int
- columnsAmount: unsigned int
- maximumEntitiesAmount: unsigned int
- currentEntitiesAmount: unsigned int
+ moveUnit(rowNumberFrom: unsigned int, columnNumberFrom: unsigned int, rowNumberTo:unsigned int, columnNumberTo: unsigned int) : void
+ commitFight(rowFrom: unsigned int, columnFrom: unsigned int, rowTo: unsigned int, columnTo: unsigned int): void
+ placeCreature(row: unsigned int; column: unsigned int, creature: ICreature*) : void
+ getCell(rowNumber: unsigned int, columnNumber:unsigned int): FieldComponent*
FieldIterator
- field: Field*
- currentRow: unsigned int
- currentColumn: unsigned int
+ next() : void
+ hasNext() : bool
+ getNext() : FieldComponent*
Friend class
<< Interface >>
ICreatureObserver
+ onCreatureDestroyed() : void
FieldComponent
- landscapeProxy: ILandscape*
- creature: ICreature*
- object: INeutralObject*
+ copy() : FieldComponent*
+ hasCreature() : bool
+ hasNeutralObject() : bool
+ canHoldCreature() : bool
+ onCreatureDestroyed() : void
+ setCreature(ICreature* newCreature): void
LandscapeProxy
- landscape: ILandscape*
<<Override>>
+ canHoldCreature() : bool
+ getClass() : QString
+ copy() : ILandscape*
Water
<<Override>>
+ canHoldCreature() : bool
+ getClass() : QString
+ copy() : ILandscape*
Mountains
<<Override>>
+ canHoldCreature() : bool
+ getClass() : QString
+ copy() : ILandscape*
Ground
<<Override>>
+ canHoldCreature() : bool
+ getClass() : QString
+ copy() : ILandscape*
<< Interface >>
ILandscape
+ canHoldCreature() : bool
+ copy() : ILandscape*
<< Interface >>
IPrintable
+ getClass() : QString
Game
-field: Field
-bases: std::map<unsigned int, Base*>
+ addBase(unsigned int row, unsigned int column, unsigned int possessorId) : void
+ addUnit(unsigned int row, unsigned int column, UnitType unitType, unsigned int possessorId) : void
+moveUnit(unsigned int rowFrom, unsigned int columnFrom, unsigned int rowTo, unsigned int columnTo, unsigned int actionId) : void
+ attack(unsigned int rowFrom, unsigned int columnFrom,unsigned int rowTo, unsigned int columnTo, unsigned int actionId): void
+isPlayerAlive(unsigned int playerNumber):bool
<< Interface >>
ICreatureObservableByFieldComponent
+ attachFieldComponent(ICreatureObserver* fieldComponent)
INeutralObject
# strategy: INeutralObjectEffectsStrategy
<<Virtual>>
+ influenceOnUnit(IUnit* unit): void
+ copy() : INeutralObject*
Attribute
- value: int
- bonus: int
ICreature
# health: Attribute*
# race: Race
<<Virtual>>
+ copy() : ICreature
+ isMoveable() : bool
+ getCreatureType() : CreatureType
+ takeDamage(int damage) : void
+ setHealth(int newHealth): void
<<Override>>
+ attachFieldComponent(ICreatureObserver* fieldComponent): void
<< Interface >>
IUnitObservableByBase
+ attachBase(IUnitObserver* base): void
<< Interface >>
IUnitObserver
+ onUnitDestroyed(IUnit* destroyedUnit): void
<< Interface >>
AbstractUnitFactory
<<Virtual>>
+ createInfantry(Race race): IInfantry*
+ createCavalry(Race race): ICavalry*
+ createArcher(Race race): IArcher*
Banner
<<Override>>
+ influenceOnUnit(IUnit* unit): void
+ getClass() : QString
+ copy() : INeutralObject*
Forge
<<Override>>
+ influenceOnUnit(IUnit* unit): void
+ getClass() : QString
+ copy() : INeutralObject*
Temple
<<Override>>
+ influenceOnUnit(IUnit* unit): void
+ getClass() : QString
+ copy() : INeutralObject*
Tower
<<Override>>
+ influenceOnUnit(IUnit* unit): void
+ getClass() : QString
+ copy() : INeutralObject*
<< Interface >>
INeutralObjectEffectsStrategy
+ influenceOnUnit(IUnit* unit): void
BannerInfantryStrategy
<<Override>>
+ influenceOnUnit(IUnit* unit): void
BannerCavalryStrategy
<<Override>>
+ influenceOnUnit(IUnit* unit): void
BannerArcherStrategy
<<Override>>
+ influenceOnUnit(IUnit* unit): void
TowerInfantryStrategy
<<Override>>
+ influenceOnUnit(IUnit* unit): void
TowerCavalryStrategy
<<Override>>
+ influenceOnUnit(IUnit* unit): void
TowerArcherStrategy
<<Override>>
+ influenceOnUnit(IUnit* unit): void
ForgeInfantryStrategy
<<Override>>
+ influenceOnUnit(IUnit* unit): void
ForgeCavalryStrategy
<<Override>>
+ influenceOnUnit(IUnit* unit): void
ForgeArcherStrategy
<<Override>>
+ influenceOnUnit(IUnit* unit): void
TempleInfantryStrategy
<<Override>>
+ influenceOnUnit(IUnit* unit): void
TempleCavalryStrategy
<<Override>>
+ influenceOnUnit(IUnit* unit): void
TempleArcherStrategy
<<Override>>
+ influenceOnUnit(IUnit* unit): void
IUnit
# armor: Attribute*
# damage: Attribute*
+ commitAttack(ICreature* attackedUnit): void
+ clearNeutralObjectEffect(): void
+ operator+=(INeutralObject* neutralObject): IUnit&
<<Virtual>>
+ getType(): UnitType
<<Override>>
+ attachBase(IUnitObserver* base): void
+ isMoveable() : bool
+ getCreatureType() : CreatureType
+ setHealth(int newHealth): void
Base
- field: Field*
- crit Dealers Factory: AbstractUnitFactory*
- absorbersFactory: AbstractUnitFactory*
+ createUnit(UnitType unitType, unsigned int row, unsigned int column): void
<<Override>>
+ getClass() : QString
+ isMoveable() : bool
+ getCreatureType() : CreatureType
+ onUnitDestroyed(IUnit* destroyedUnit): void
+ copy() : ICreature*
+ setHealth(int newHealth): void
+ getInfo() : QString
+ isDemolished() : bool
CritDealersFactory
<<Override>>
+ createInfantry(Race race): IInfantry*
+ createCavalry(Race race): ICavalry*
+ createArcher(Race race): IArcher*
AbsorbersFactory
<<Override>>
+ createInfantry(Race race): IInfantry*
+ createCavalry(Race race): ICavalry*
+ createArcher(Race race): IArcher*
IInfantry
ICavalry
IArcher
CritInfantry
<<Override>>
+ getClass() : QString
+ copy() : ICreature*
+ getType() : UnitType
AbsorbInfantry
<<Override>>
+ getClass() : QString
+ copy() : ICreature*
+ getType() : UnitType
AbsorbCavalry
<<Override>>
+ getClass() : QString
+ copy() : ICreature*
+ getType() : UnitType
CritCavalry
<<Override>>
+ getClass() : QString
+ copy() : ICreature*
+ getType() : UnitType
CritArcher
<<Override>>
+ getClass() : QString
+ copy() : ICreature*
+ getType() : UnitType
AbsorbArcher
<<Override>>
+ getClass() : QString
+ copy() : ICreature*
+ getType() : UnitType
ICritDealer
# critFactor: double
<<Override>>
+ takeDamage(int damage): void
IDamageAbsorber
# absorbtionCoefficient: double
<<Override>>
+ takeDamage(int damage): void