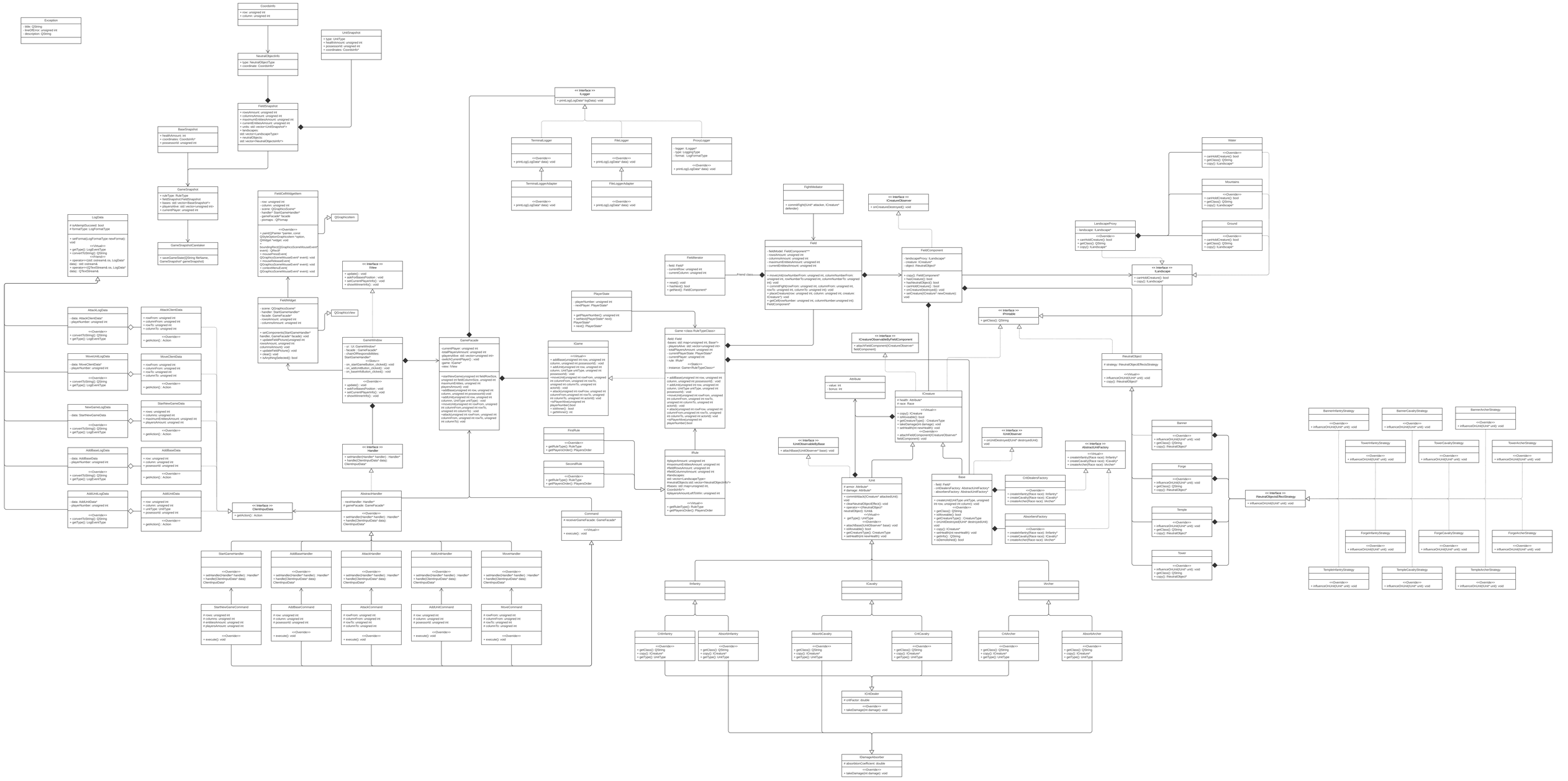
Exception
- title: QString
- lineOfError: unsigned int
- description: QString
CoordsInfo
+ row: unsigned int
+ column: unsigned int
NeutralObjectInfo
+ type: NeutralObjectType
+ coordinate: CoordsInfo*
UnitSnapshot
+ type: UnitType
+ healthAmount: unsigned int
+ possessorId: unsigned int
+ coordinates: CoordsInfo*
FieldSnapshot
+ rowsAmount: unsigned int
+ columnsAmount: unsigned int
+ maximumEntitiesAmount: unsigned int
+ currentEntitiesAmount: unsigned int
+ units: std::vector<UnitSnapshot*>
+ landscapes: std::vector<LandscapeType>
+ neutralObjects: std::vector<NeutralObjectInfo*>
BaseSnapshot
+ healthAmount: int
+ coordinates: CoordsInfo*
+ possessorId: unsigned int
GameSnapshot
+ rulesType: RuleType
+ fieldSnapshot:FieldSnapshot
+ bases: std::vector<BaseSnapshot*>
+ playersAlive: std::vector<unsigned int>
+ currentPlayer: unsigned int
GameSnapshotCaretaker
+ saveGameState(QString fileName, GameSnapshot* gameSnapshot)
<< Interface >>
ILogger
+ printLog(LogData* logData): void
TerminalLogger
<<Override>>
+ printLog(LogData* data): void
FileLogger
<<Override>>
+ printLog(LogData* data): void
ProxyLogger
- logger: ILogger*
- type: LoggingType
- format: LogFormatType
<<Override>>
+ printLog(LogData* data): void
TerminalLoggerAdapter
<<Override>>
+ printLog(LogData* data): void
FileLoggerAdapter
<<Override>>
+ printLog(LogData* data): void
LogData
isAttempt:Succeed: bool
formatType: LogFormatType
+ setFormat(LogFormatType newFormat): void
<<Virtual>>
+ getType(): LogEventType
+ convertToString(): QString
<<Override>>
+ operator<<(std::ostream& os, LogData* data) : std::ostream&
+ operator<<(QTextStream& os, LogData* data) : QTextStream&
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(QGraphicsSceneMouseEvent 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
+ addForBasesPosition : void
+ setCurrentPlayerInfo() : void
+ showWinnerInfo() : void
AttackLogData
- data: AttackClientData*
- playerNumber: unsigned int
<<Override>>
+ convertToString(): QString
+ getType(): LogEventType
AttackClientData
+ rowFrom: unsigned int
+ columnFrom: unsigned int
+ rowTo: unsigned int
+ columnTo: unsigned int
<<Override>>
+ getAction() : Action
MoveLogLogData
- data: MoveClientData*
- playerNumber: unsigned int
<<Override>>
+ convertToString(): QString
+ getType(): LogEventType
MoveClientData
+ rowFrom: unsigned int
+ columnFrom: unsigned int
+ rowTo: unsigned int
+ columnTo: unsigned int
<<Override>>
+ getAction() : Action
NewGameLogData
- data: StartNewGameData
<<Override>>
+ convertToString(): QString
+ getType(): LogEventType
StartNewGameData
+ rows: unsigned int
+ columns: unsigned int
+ maximumEntitiesAmount: unsigned int
+ playersAmount: unsigned int
<<Override>>
+ getAction() : Action
AddBaseLogData
- data: AddBaseData
- playerNumber: unsigned int
<<Override>>
+ convertToString(): QString
+ getType(): LogEventType
AddBaseData
+ row: unsigned int
+ column: unsigned int
+ possessorId: unsigned int
<<Override>>
+ getAction() : Action
AddUnitLogData
- data: AddUnitData*
- playerNumber: unsigned int
<<Override>>
+ convertToString(): QString
+ getType(): LogEventType
AddUnitData
+ row: unsigned int
+ column: unsigned int
+ unitType: UnitTypes
+ possessorId: unsigned int
<<Override>>
+ getAction() : Action
<< Interface >>
ClientInputData
+ 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
+ addForBasesPosition : void
+ setCurrentPlayerInfo() : void
+ showWinnerInfo() : void
GameFacade
-currentPlayer: unsigned int
-numOfPlayers: unsigned int
-playersAlive: std::vector<unsigned int>
-switchCurrentPlayer() : void
-game: IGame*
-view: IView
+startNewGame(unsigned int fieldRowSize, unsigned int fieldColumnSize, unsigned int maximumEntities, unsigned int playersAmount)
+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) : void
+attackUnsigned int rowFrom, unsigned int columnFrom, unsigned int rowTo, unsigned int columnTo): void
<< Interface >>
Handler
+ setHandler(Handler* handler) : Handler*
+ handle(ClientInputData* data): ClientInputData*
AbstractHandler
-nextHandler: Handler*
gameFacade: GameFacade*
<<Override>>
+ setHandler(Handler* handler) : Handler*
+ handle(ClientInputData* data): ClientInputData*
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
Command
receiverGameFacade: GameFacade*
<<Virtual>>
+ execute() : void
IGame
<<Virtual>>
+ addBase(unsigned int row, unsigned int column, unsigned int possessorId) : void
+ addUnit(unsigned int row, unsigned int column, UnitTypes 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
+ isWinner() : bool
+ getWinner() : int
FirstRule
<<Override>>
+ getRuleType(): RuleType
+ getPlayersOrder(): PlayersOrder
SecondRule
<<Override>>
+ getRuleType(): RuleType
+ getPlayersOrder(): PlayersOrder
FightMediator
+ commitFight(IUnit* attacker, ICreature* defender):
<< Interface >>
ICreatureObserver
+ onCreatureDestroyed(): void
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
+ getCellByNumber: unsigned int, columnNumber:unsigned int): FieldComponent*
Friend class
FieldIterator
- field: Field*
- currentRow: unsigned int
- currentColumn: unsigned int
+ reset(): void
+ hasNext() :bool
+ getNext(): FieldComponent*
FieldComponent
- landscapeProxy : ILandscape*
- creature: ICreature*
- object: IneutralObject*
+ copy() : FieldComponent*
+ isAvailable() :bool
+ hasOwnerId() :bool
+ canHoldCreature() : bool
+ onCreatureDestroyed(): void
+ setCreature(ICreature* newCreature): void
PlayerState
- playerNumber: unsigned int
- nextPlayer: PlayerState*
+ getPlayerNumber(): unsigned int
+ setNextPlayerState* next): PlayerState*
+ next(): PlayerState*
Game <class RuleTypeClass>
-field: Field*
-bases: std::map<unsigned int, Base*>
-playersAlive: std::vector<unsigned int>
-totalPlayersAmount: unsigned int
-currentPlayerState: PlayerState*
-currentPlayer: unsigned int
-rule: IRule*
<<Static>>
- instance: Game<RuleTypeClass>*
+ 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
+ isWinner() : bool
+ getWinner() : int
IRule
#playersAmount: unsigned int
#maximumEntitiesAmount: unsigned int
#fieldRowsAmount: unsigned int
#fieldColumnsAmount: unsigned int
#landscapes: std::vector<LandscapeType>
#neutralObjects: std::vector<NeutralObjectInfo*>
#bases: std::map<unsigned int, CoordsInfo*>
#playersAmountLeftToWin: unsigned int
<<Virtual>>
+ getRuleType(): RuleType
+ getPlayersOrder(): PlayersOrder
<< Interface >>
ICreatureObserverBeforeFieldComponent
+ attachFieldComponent(ICreatureObserver* fieldComponent)
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 >>
IUnitObserverBeforeBase
+ attachBase(IUnitObserver* base): void
<< Interface >>
IUnitObserver
+ onUnitDestroyed(IUnit* destroyedUnit): void
<< Interface >>
AbstractUnitFactory
+ createInfantry(Race race): Infantry*
+ createCavalry(Race race): ICavalry*
+ createArcher(Race race): IArcher*
IUnit
armor: Attribute*
damage: Attribute*
+ commitAttack(ICreature* attackedUnit): void
+ clearNeutralObjectEffect() : void
+ operator+=(INeutralObject neutralObject): IUnit&
<<Virtual>>
~ getType(): UnitTypes
<<Override>>
+ attachBase(IUnitObserver* base): void
+ isMoveable() :bool
+ getCreatureType(): CreatureType
+ setHealth(int newHealth): void
Base
- field: Field*
- crtDealersFactory: AbstractUnitFactory*
- absorbersFactory : AbstractUnitFactory*
+ createUnit(UnitTypeUnitType, unsigned int row, unsigned int column): void
<<Override>>
+ getClass(): QString
+ isMoveable() :bool
+ getCreatureType(): CreatureType
+ setUnitDestroyed(IUnit* destroyedUnit): void
+ copy(): ICreature*
+ setHealth(int newHealth): void
+ getInfo() : QString
+ isDemolished() :bool
CrtDealersFactory
<<Override>>
+ createInfantry(Race race): Infantry*
+ createCavalry(Race race): ICavalry*
+ createArcher(Race race): IArcher*
AbsorbersFactory
<<Override>>
+ createInfantry(Race race): Infantry*
+ createCavalry(Race race): ICavalry*
+ createArcher(Race race): IArcher*
Infantry
ICavalry
IArcher
CrtInfantry
<<Override>>
+ getClass(): QString
+ copy(): ICreature*
+ getType(): UnitType
AbsorbInfantry
<<Override>>
+ getClass(): QString
+ copy(): ICreature*
+ getType(): UnitType
AbsorbCavalry
<<Override>>
+ getClass(): QString
+ copy(): ICreature*
+ getType(): UnitType
CrtCavalry
<<Override>>
+ getClass(): QString
+ copy(): ICreature*
+ getType(): UnitType
CrtArcher
<<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
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
INeutralObject
strategy: INeutralObjectEffectsStrategy
<<Virtual>>
+ influenceOnUnit(IUnit* unit): void
+ copy(): INeutralObject*
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 >>
INeutralObjectEffectStrategy
+ 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
TemplelnfantryStrategy
<<Override>>
+ influenceOnUnit(IUnit* unit): void
TemplecavalryStrategy
<<Override>>
+ influenceOnUnit(IUnit* unit): void
TemplearcherStrategy
<<Override>>
+ influenceOnUnit(IUnit* unit): void