


Software Engineering project 2019


Implementation of Adrenaline by Filip Neduk.


Marco Bagatella, Riccardo Bassani, Davide Aldé.

model

 Package board

 Package cards

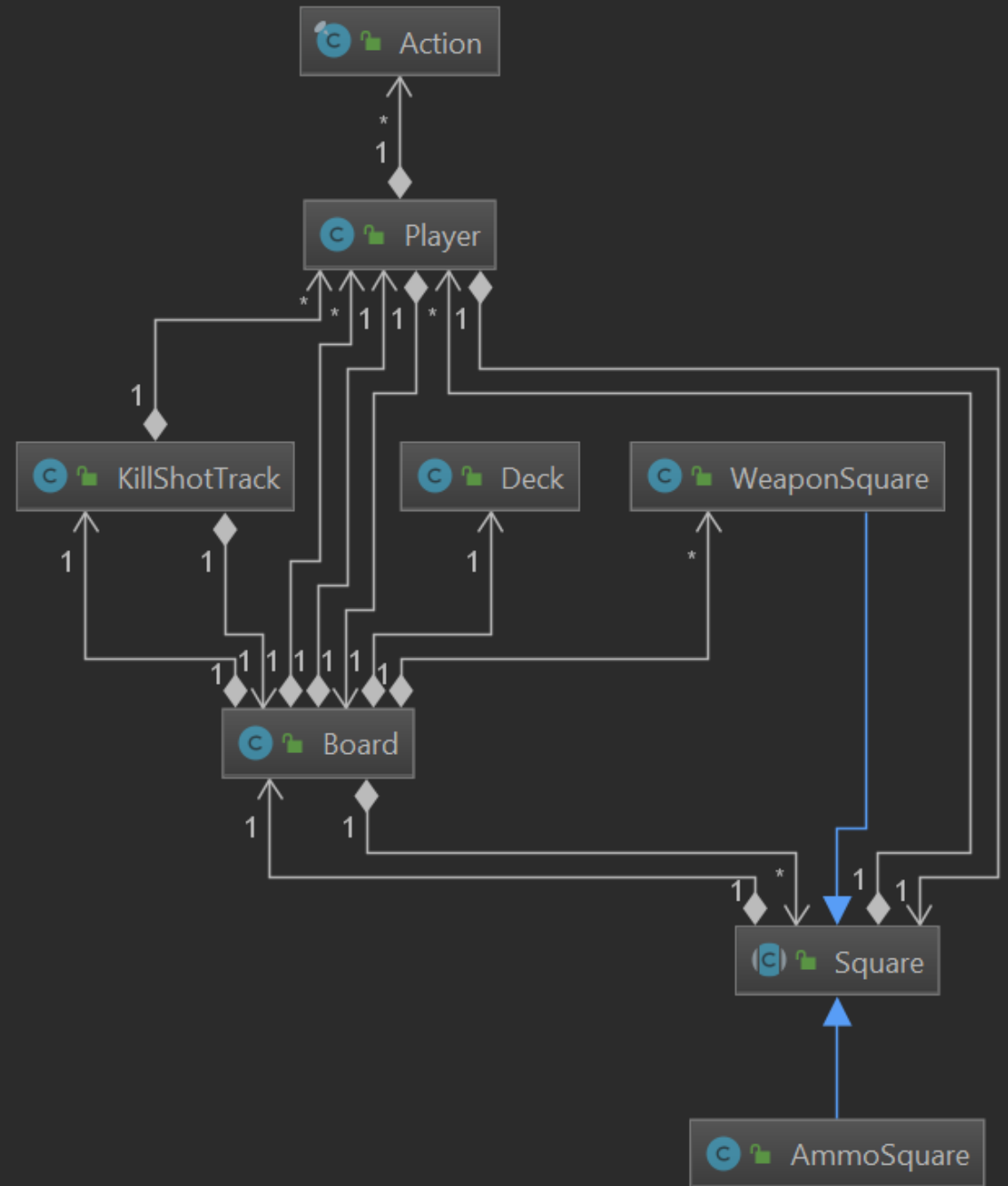
 Updater

 Package exceptions

Powered by yFiles

model.board

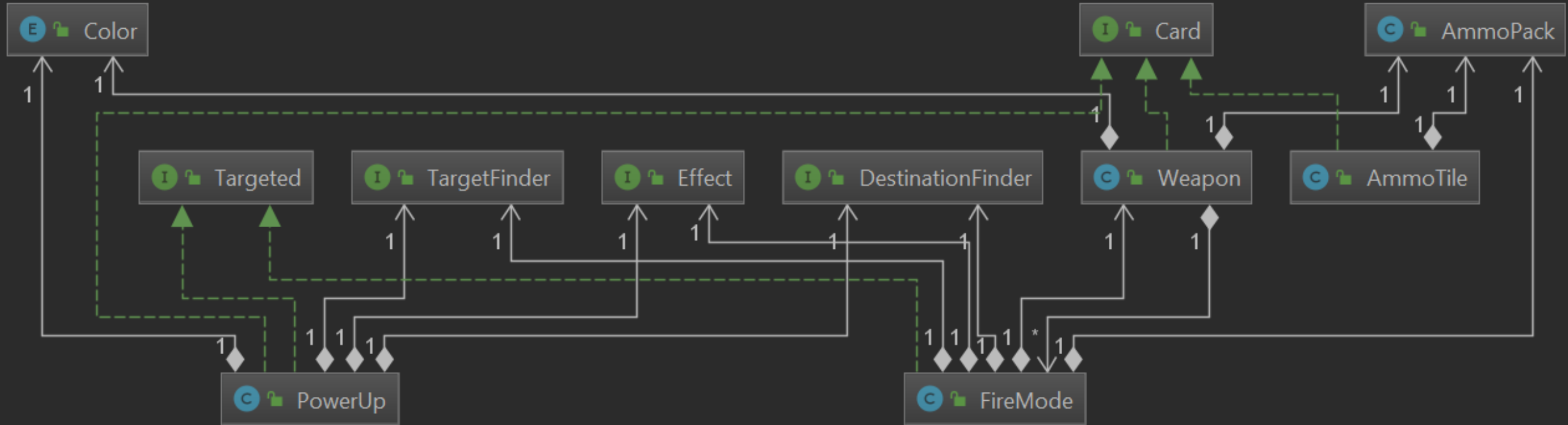
- Contains a class representing the Board and other classes representing the elements of the Board.
- A Board is made of Squares, that can be AmmoSquares or WeaponSquares.
- A Board contains a KillShotTrack, some Decks and the Players of the game.
- Every Player can do some Actions.



model.cards

- Weapon, PowerUp and AmmoTile implement the interface Card.
- An AmmoTile contains an Ammopack. Weapons, PowerUps and FireModes have a cost in AmmoPack.
- FireMode and PowerUp implement the interface Targeted.
- TargetFinder, Effect and DestinationFinder are functional interfaces used by FireMode, PowerUp.
- Cards, as well as other components of the game, can have a Color. Namely, Weapons and PowerUp must have a Color while AmmoTile must not.

model.cards

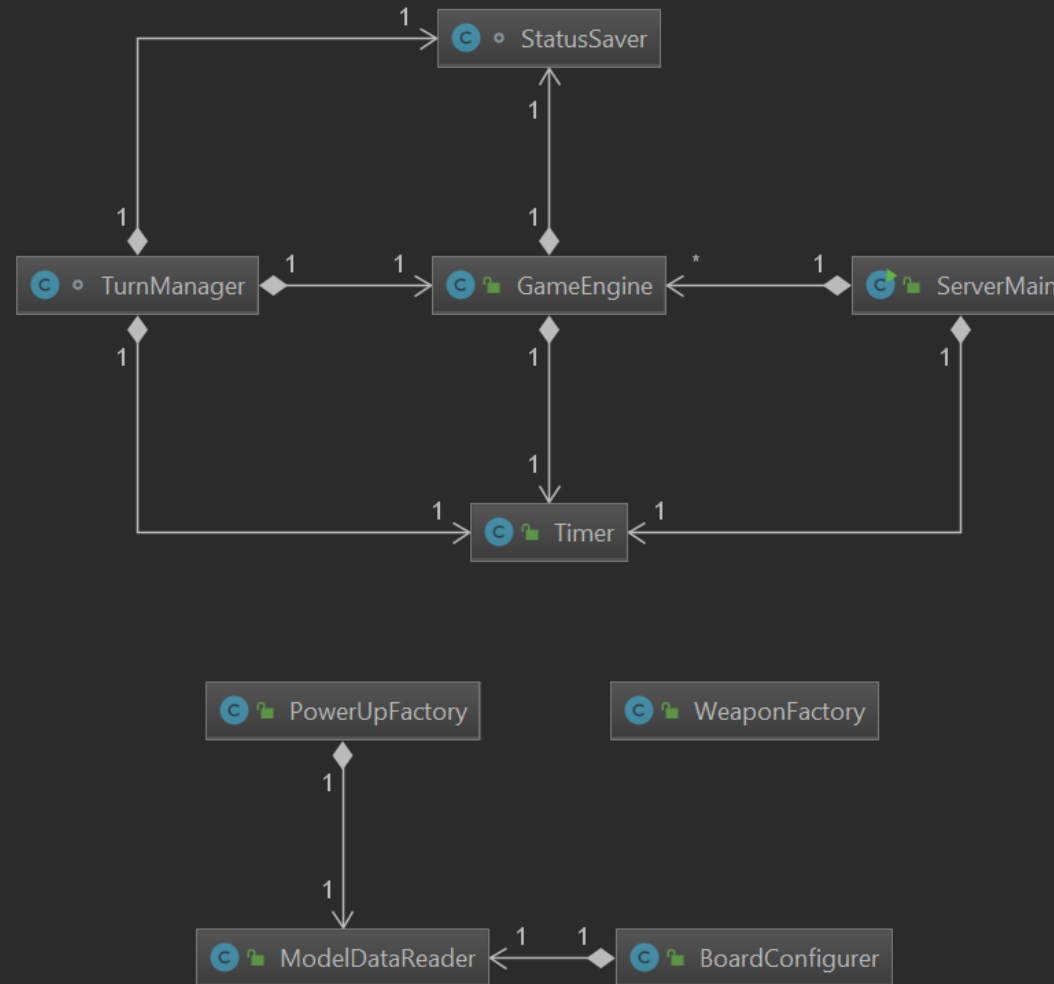


model.Updater

- The class responsible for updating the ClientModel by copying the necessary values from the ServerModel.
- The ClientModel class will be presented later.

Updater		
get(String, Weapon, boolean)		JsonObject
get(String, int)		JsonObject
get(String, int, Player, boolean)		JsonObject
get(String, Player, PowerUp)		JsonObject
get(String, Player, Weapon)		JsonObject
get(String, Player, AmmoPack)		JsonObject
get(String, Player, Square)		JsonObject
get(String, Player)		JsonObject
get(String, Player, int)		JsonObject
get(String, Player, List<Player>)		JsonObject
get(String, Square, Weapon)		JsonObject
get(String, Player, boolean)		JsonObject
get(String, Square)		JsonObject
getModel(Board, Player)		JsonObject
createSimplePlayer(Player, Board)		SimplePlayer
getFreshUpdate(String)		JsonObject
getRenderMessage()		JsonObject
toSimpleWeapon(Weapon)		SimpleWeapon
toSimpleSquare(WeaponSquare)		SimpleSquare
toSimpleSquare(AmmoSquare)		SimpleSquare

controller



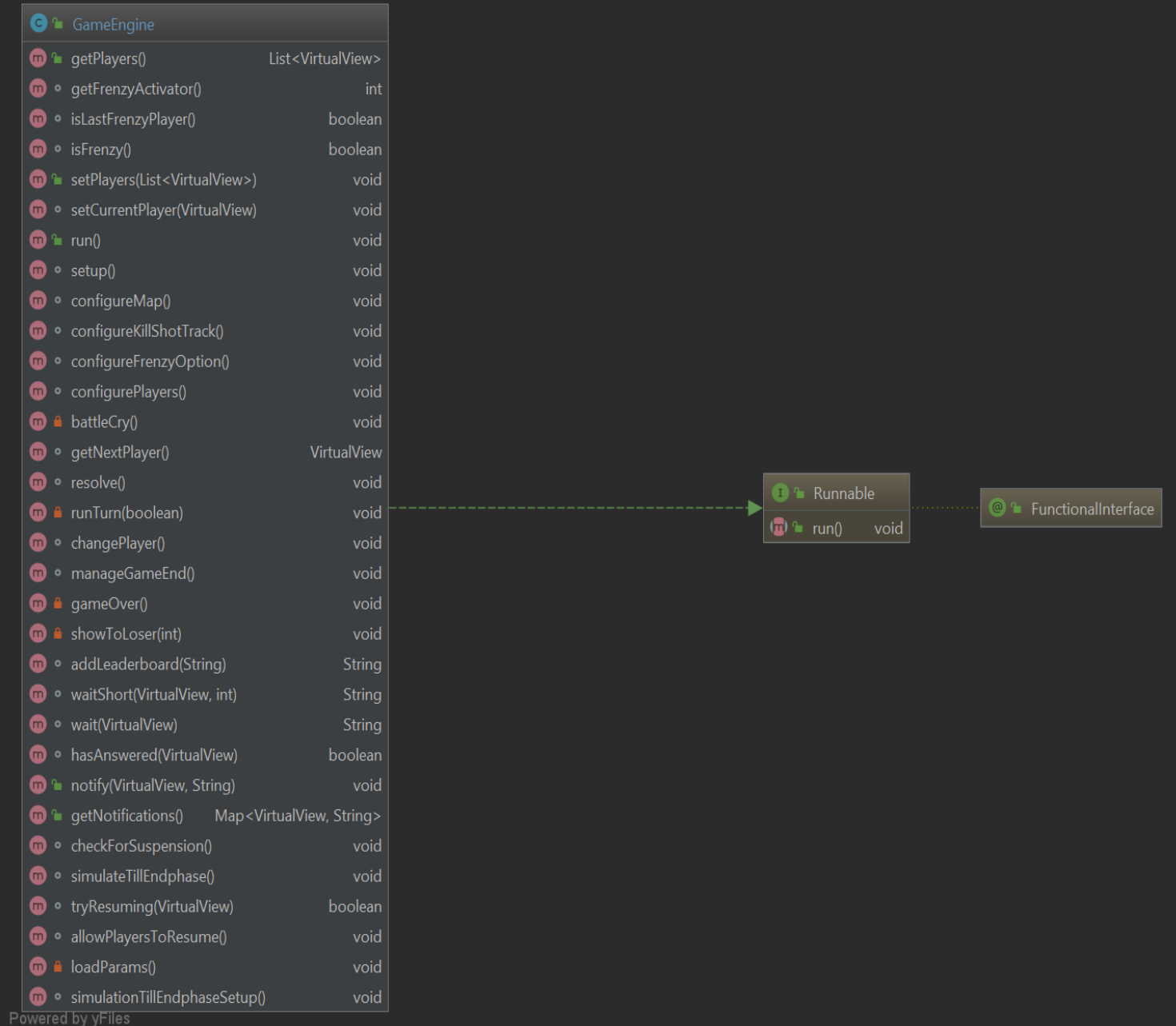
controller.ServerMain

- The class containing the main() method of the Server.
- Manages connections and matchmaking.

ServerMain		
getInstance()		ServerMain
main(String[])		void
◦ setup()		void
◦ untrackGame(GameEngine)		void
◦ addPlayer(VirtualView)		void
login(VirtualView)		boolean
canResume(String)		boolean
resume(VirtualView)		boolean
◦ removeSuspendedPlayers()		void
getPlayers()		List<VirtualView>
◦ initializeLogger()		void
◦ loadConfig()		Properties
manageInput()		void
refreshConnections()		void
◦ matchmaking()		void
getAlreadyConnected()		String

controller.GameEngine

- The class responsible for running a game.
- Is instantiated and run by ServerMain.



controller.TurnManager





- The class responsible for running a turn.
- Handles the exchange of messages with the user.

TurnManager		
runTurn()		void
joinBoard(Player, int, boolean)		void
executeAction()		boolean
executeActualAction(Action)		void
handleUsingPowerUp()		boolean
usePowerUp()		void
handleMoving(Action)		void
handleCollecting()		void
handleShooting()		void
applyFireMode(FireMode)		void
reload(int)		boolean
reloadMandatory()		void
handleTargetingScope(Player, List<Player>)		boolean
askTargetsForGrenade()		void
handleTagbackGrenade(Player)		boolean
handleDeaths()		void
replaceWeapons()		void
replaceAmmoTiles()		void
getDamagesList()		List<Integer>
updateDead()		void
askConfirmation(String)		boolean
askConfirmation(String, Player)		boolean
resetJoinBoard(Player, boolean)		void
resetPowerUp()		void
resetAction()		void
toStringList(List)		List<String>
toUserStringList(List<List<Player>>)		List<String>
updateAndSendModel()		void
updateAndNotifyAll()		void
restoreAndNotify()		void
getVirtualView(Player)		VirtualView
getDead()		List<Integer>
handlePayment(AmmoPack)		void
mandatoryConversion(Color)		void
setDead(List<Integer>)		void

controller.StatusSaver

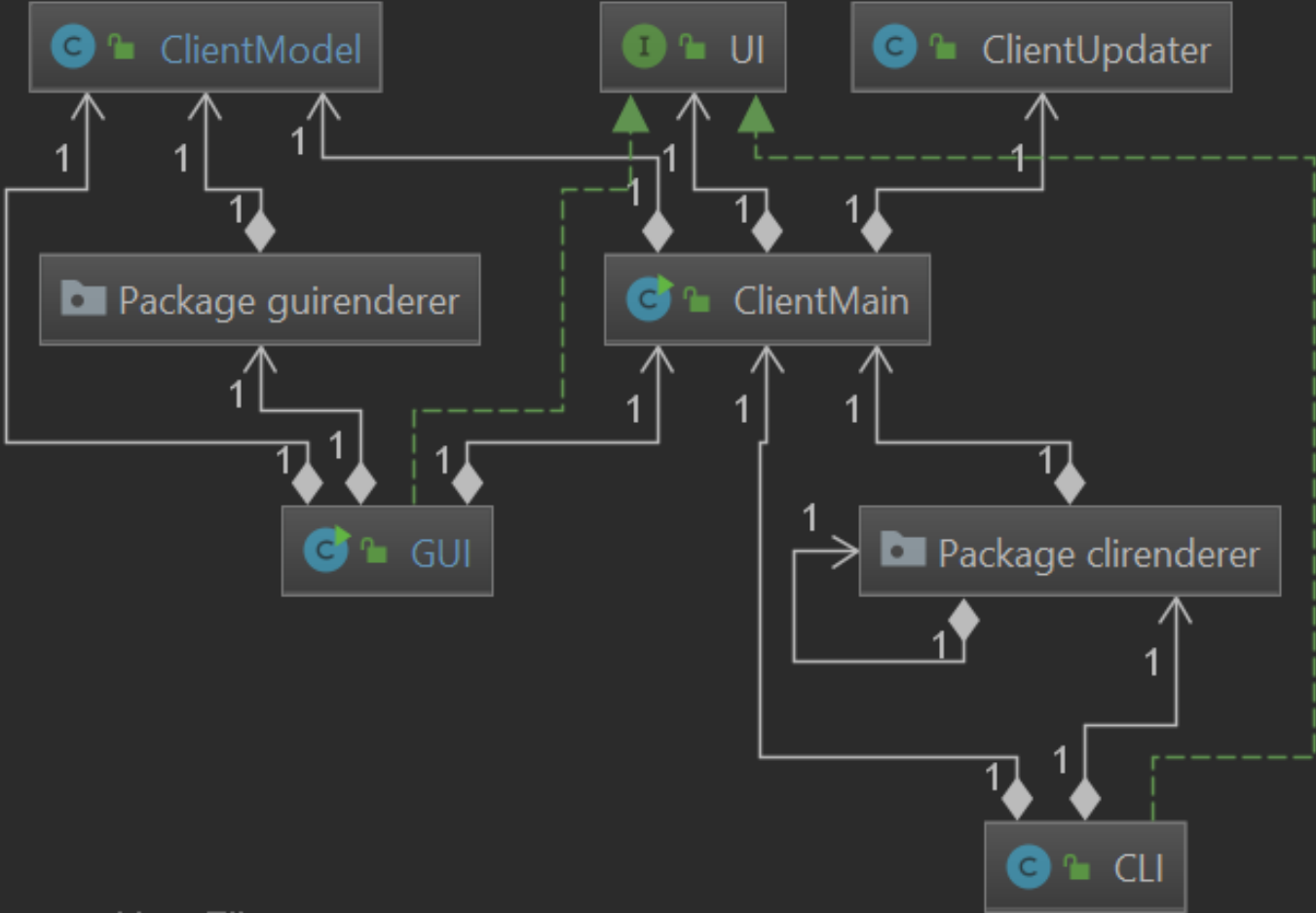
- Supports the TurnManager allowing it to save checkpoints in order to make possible for the player to reset his actions.

StatusSaver

	◦ updateCheckpoint()	void
	◦ updatePowerups()	void
	◦ restoreCheckpoint()	void
	◦ restorePowerUps()	void



























Powered by yFiles

view



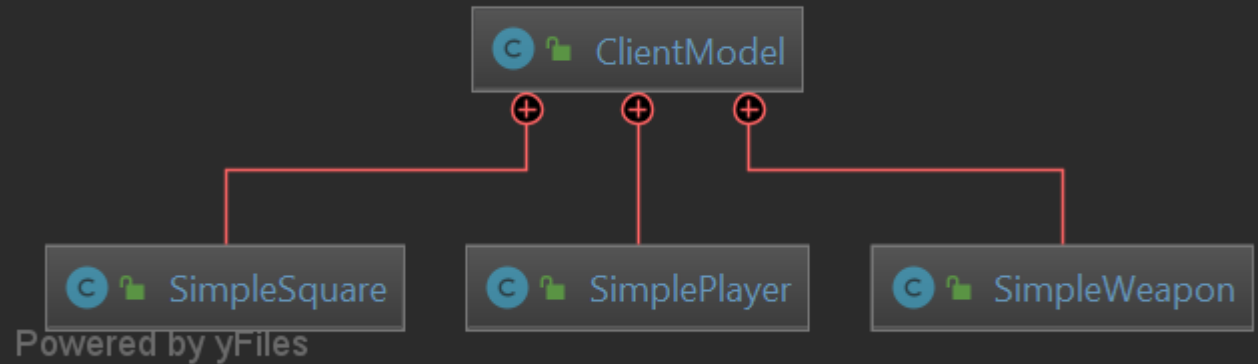
view.ClientMain

- The class containing the main() method of the Server.

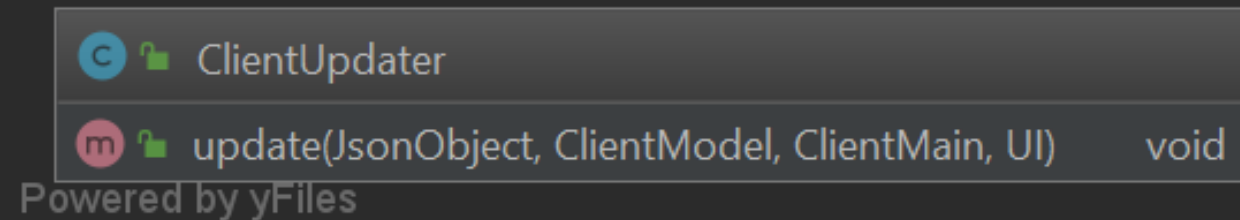
ClientMain		
	 main(String[])	void
	 initializeLogger()	void
	 loadConfig(String[])	Properties
	 setup(String[])	void
	 choose(String, String, List<String>)	int
	 display(String)	void
	 getInput(String, int)	String
	 update(JsonObject)	void
	 getClientModel()	ClientModel
	 setClientModel(ClientModel)	void
	 showSuspension()	void
	 showDisconnection()	void
	 showEnd(String)	void

view.ClientModel & view.ClientUpdater

Contains a simplified version
of the model.
Contains three inner classes.

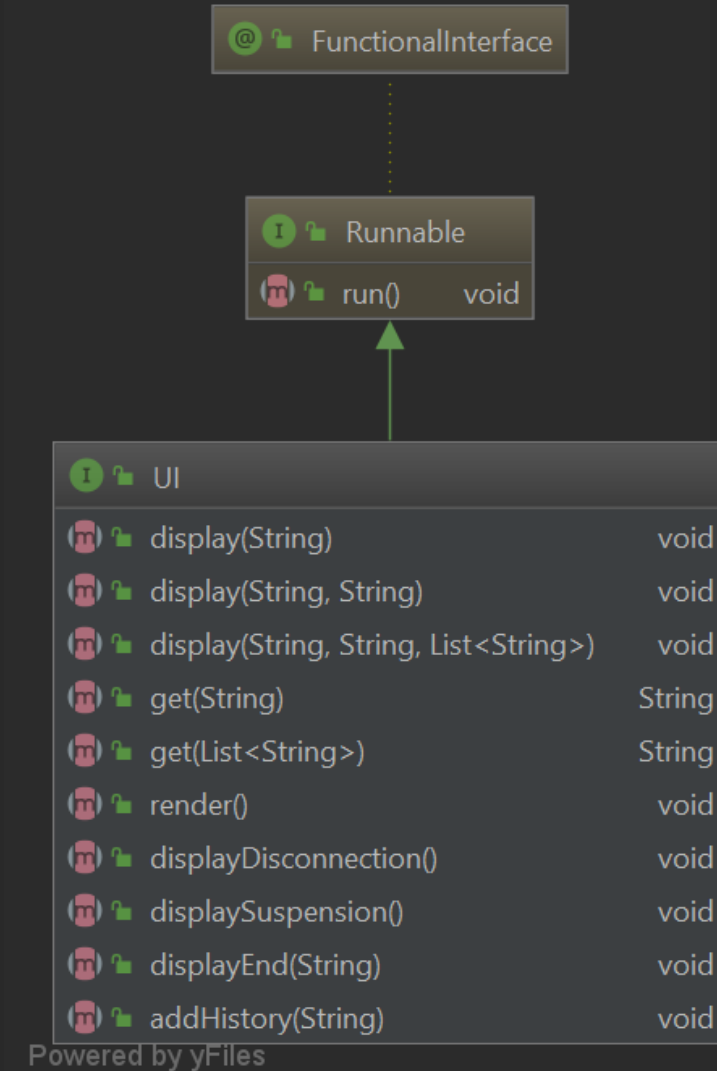


Updates the ClientModel,
depending on the type of
JsonObject it receives.



view.UI

- Interface for user interface and for managing client's input/output.
- Is implemented by CLI, GUI.



network



Package client

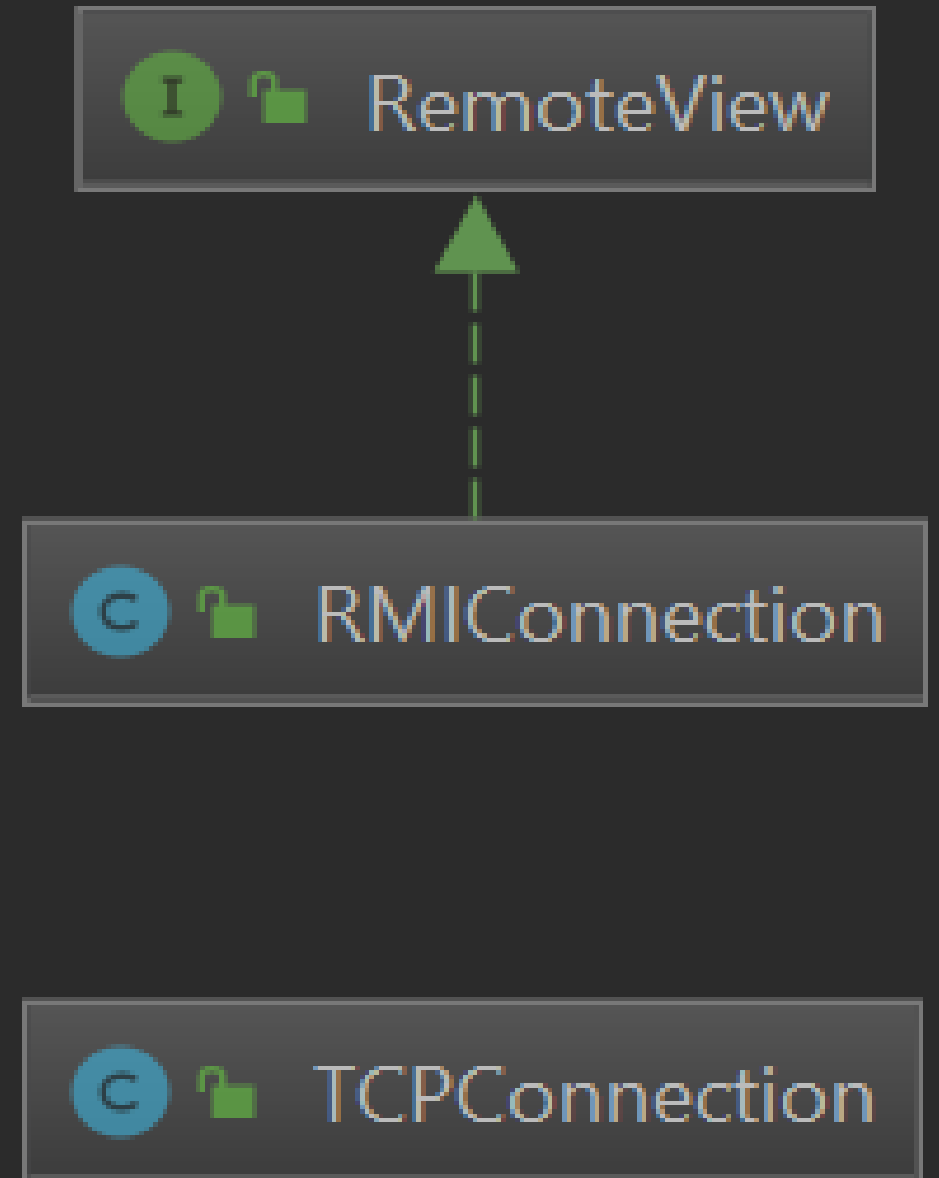


Package server

Powered by yFiles

network.client

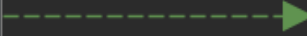
- Two classes for two different types of connection: RMI and TCP.
- RMIConnection implements the interface RemoteView.



network.client

C RMIConnection		
m	run()	void
m	choose(String, String, List<String>)	int
m	display(String)	void
m	getInput(String, int)	String
m	ping()	void
m	update(String)	void
m	shutdown()	void
m	showSuspension()	void
m	showEnd(String)	void

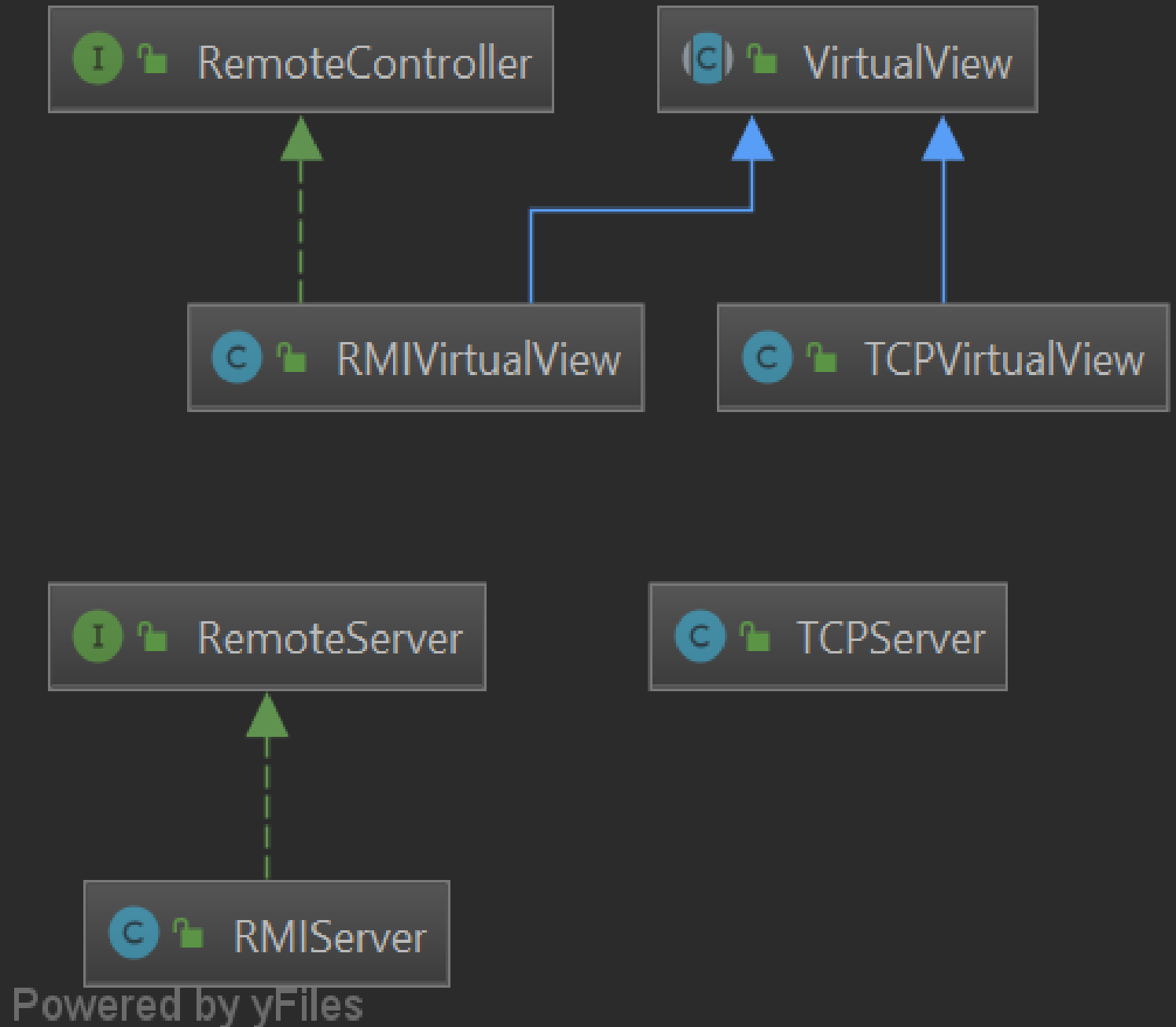
I RemoteView		
m	choose(String, String, List<String>)	int
m	display(String)	void
m	getInput(String, int)	String
m	ping()	void
m	update(String)	void
m	showSuspension()	void
m	showEnd(String)	void



C TCPConnection		
m	run()	void
m	handleRequest(JsonObject)	void
m	send(String)	void
m	receive()	String
m	shutdown()	void

network.server

- A VirtualView allows the Server to communicate with the Client.
- Depending on the user choice, for every connection the Server can implement a TCPVirtualView or an RMIVirtualView, taking advantage of a TCPServer class or of an RMIServer class.



network.server

TCPVirtualView		
run()		void
refresh()		void
shutdown()		void
showSuspension()		void
showEnd(String)		void
choose(String, String, List<?>)		void
choose(String, String, List<?>, int)		void
chooseNow(String, String, List<?>)		int
display(String)		void
getInputNow(String, int)		String
update(JsonObject)		void
receive()		String
send(JsonObject)		void

TCPServer		
run()		void
getPort()		int
shutdown()		void

VirtualView		
run()		void
getName()		String
getBattlecry()		String
getGame()		GameEngine
setGame(GameEngine)		void
getModel()		Player
isSuspended()		boolean
setPlayer(Player)		void
isJustSuspended()		boolean
setJustSuspended(boolean)		void
setSuspended(boolean)		void
setName(String)		void
refresh()		void
shutdown()		void
showSuspension()		void
showEnd(String)		void
suspend()		void
choose(String, String, List<?>)		void
choose(String, String, List<?>, int)		void
chooseNow(String, String, List<?>)		int
display(String)		void
getInputNow(String, int)		String
update(JsonObject)		void
notifyObservers(String)		void
toString()		String

RMIVirtualView		
refresh()		void
shutdown()		void
ping()		void
showSuspension()		void
showEnd(String)		void
choose(String, String, List<?>)		void
choose(String, String, List<?>, int)		void
chooseNow(String, String, List<?>)		int
display(String)		void
getInputNow(String, int)		String
update(JsonObject)		void

RemoteController		
ping()		void

RemoteServer		
getVirtualView(RemoteView)		String

RMIServer		
setup()		void
getVirtualView(RemoteView)		String
shutdown()		void

COMPREHENSIVE DIAGRAM

Methods summary: model.board

Player	
sufferDamage(int, Player)	void
sufferDamageNoMarksExtra(int, Player)	void
addDamages(int, Player)	void
addMarks(int, Player)	void
addWeapon(Weapon)	void
addAmmoPack(AmmoPack)	void
collect(Card)	boolean
drawPowerUp()	void
discardWeapon(Card)	void
discardPowerUp(Card)	void
hasUsableTeleporterOrNewton()	boolean
hasUsableTagbackGrenade()	boolean
hasUsableTargetingScope()	boolean
canPay(AmmoPack)	boolean
useAmmo(AmmoPack)	void
getReloadableWeapons()	List<Weapon>
getLoadedWeapons()	List<Weapon>
getAvailableWeapons()	List<Weapon>
getPowerUps(PowerUpName)	List<PowerUp>
getPowerUps(Color)	List<PowerUp>
addMainTarget(Player)	void
addMainTargets(List<Player>)	void
addOptionalTarget(Player)	void
addOptionalTargets(List<Player>)	void
updateAwards()	void
rewardKillers()	void
addPoints(int)	void
refreshActionList()	void
getShootingSquares(int, List<Weapon>)	List<Square>
getAvailableActions()	List<Action>
removeShootingAction(List<Action>)	List<Action>
removeCollectingAction(List<Action>)	List<Action>
getCollectibleWeapons(WeaponSquare)	List<Weapon>
getColor()	String
toString()	String
userToString()	String
equals(Object)	boolean
hashCode()	int

Square	
getId()	int
getRoomId()	int
getRow()	int
getColumn()	int
getColor()	Color
getPlayers()	List<Player>
getBoard()	Board
containsPlayer(Player)	boolean
addPlayer(Player)	void
removePlayer(Player)	void
removeCard(Card)	Card
addAllCards()	void
isEmpty()	boolean
equals(Object)	boolean
hashCode()	int
toString()	String

AmmoSquare	
isEmpty()	boolean
getAmmoTile()	AmmoTile
removeCard(Card)	Card
addAllCards()	void
hasAmmoTile()	boolean
equals(Object)	boolean
hashCode()	int

WeaponSquare	
setWeapons(List<Weapon>)	void
isEmpty()	boolean
getWeapons()	List<Weapon>
removeCard(Card)	Card
addAllCards()	void
addCard()	void
addCard(Weapon)	void
equals(Object)	boolean
hashCode()	int

Deck	
getDrawable()	List<Card>
getDiscarded()	List<Card>
addCard(Card)	void
addDiscardedCard(Card)	void
drawCard()	Card
shuffleDeck()	void
regenerate()	void

Action	
getSteps()	int
isCollect()	boolean
isShoot()	boolean
isReload()	boolean
equals(Object)	boolean
hashCode()	int
toString()	String

Board	
getPlayersInside(Square)	List<Player>
getAdjacent(Square)	List<Square>
getReachable(Square, int)	List<Square>
getVisible(Square)	List<Square>
getSquaresInRoom(int)	List<Square>
getSquaresInLine(Square, Direction)	List<Square>
getSquaresInLineIgnoringWalls(Square, Direction)	List<Square>
getDistance(Square, Square)	int
inLineTop(int, int, int, int)	boolean
inLineDown(int, int, int, int)	boolean
inLineLeft(int, int, int, int)	boolean
inLineRight(int, int, int, int)	boolean
sort(List<Player>, List<Player>)	void
getAmmoSquares()	List<AmmoSquare>
registerObserver(VirtualView)	void
removeObserver(VirtualView)	void
notifyObservers()	void
notifyObserver(VirtualView)	void
addToUpdateQueue(JsonObject)	void
addToUpdateQueue(JsonObject, VirtualView)	void
revertUpdates(VirtualView)	void

KillShotTrack	
getSkullsLeft()	int
getKillers()	List<Player>
removeSkulls(int)	void
registerKill(Player, Player, boolean)	void
rewardKillers()	void

Attributes summary: model.board

Player	
id	int
name	HeroName
username	String
status	Status
points	int
dead	boolean
flipped	boolean
damages	List<Player>
marks	List<Player>
position	Square
previousPosition	Square
board	Board
weaponList	List<Weapon>
powerUpList	List<PowerUp>
ammoPack	AmmoPack
actionList	List<Action>
mainTargets	List<Player>
optionalTargets	List<Player>
deaths	int
pointsToGive	int
justDamaged	boolean
overkilled	boolean
inGame	boolean
j	ModelDataReader

Action	
steps	int
collect	boolean
shoot	boolean
reload	boolean
MOVE_UP_TO	String
SQUARES	String
COLLECT_TAG	String
RELOAD_TAG	String
SHOOT_TAG	String

Square	
board	Board
id	int
roomId	int
row	int
column	int
color	Color
players	List<Player>
MIN_SQUARE_ID	int
MAX_SQUARE_ID	int
MAX_ROOM_ID	int
MIN_SQUARE_ROW_INDEX	int
MIN_SQUARE_COLUMN_INDEX	int

AmmoSquare	
ammoTile	AmmoTile
LOGGER	Logger

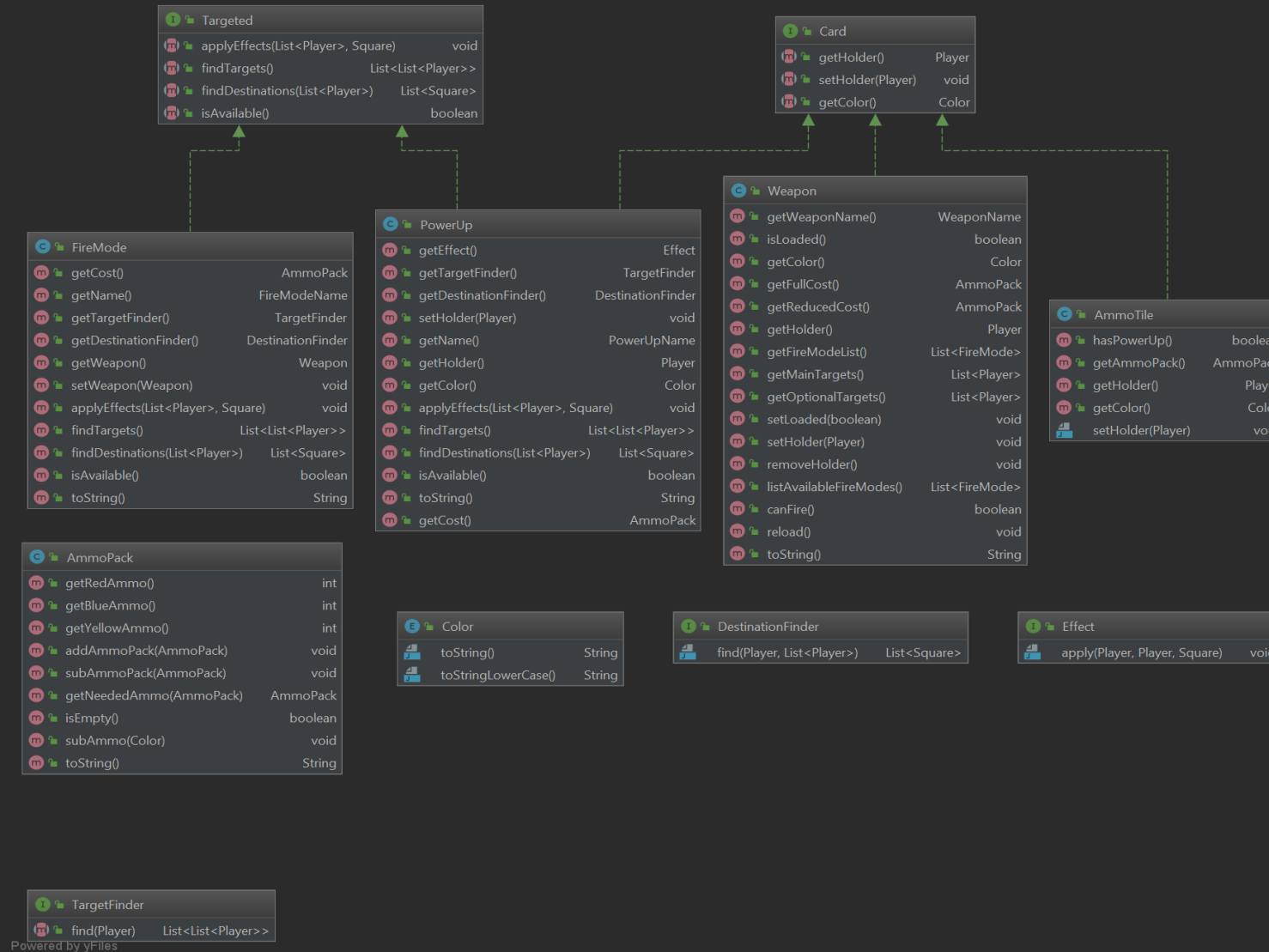
WeaponSquare	
weapons	List<Weapon>
MAX_WEAPONS_ON_SQUARE	int

Deck	
drawable	List<Card>
discarded	List<Card>
DRAWN_CARD_INDEX	int

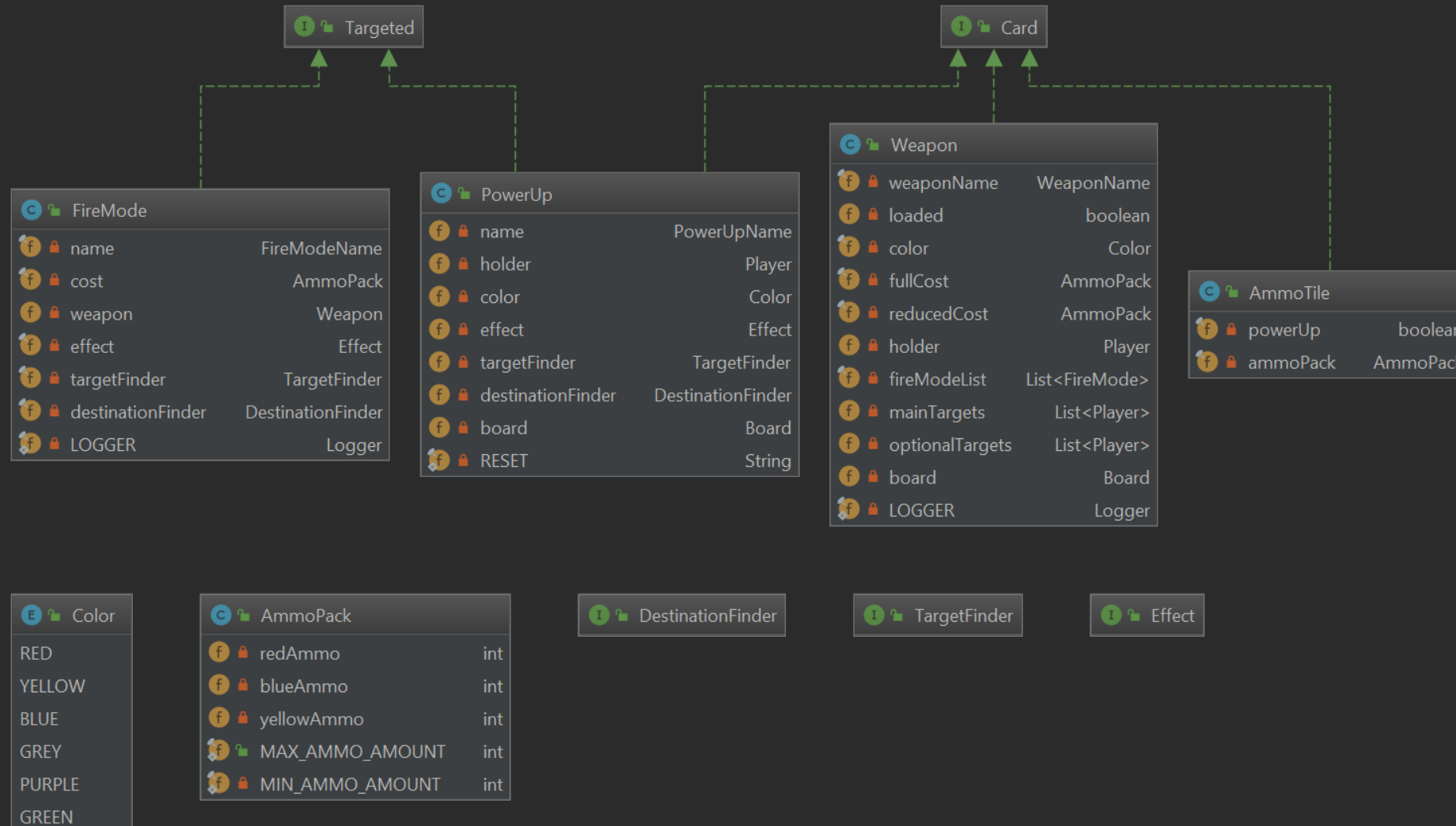
Board	
id	int
map	List<Square>
leftWalls	boolean[]
topWalls	boolean[]
spawnPoints	List<WeaponSquare>
players	List<Player>
currentPlayer	Player
weaponDeck	Deck
powerUpDeck	Deck
ammoDeck	Deck
killShotTrack	KillShotTrack
observers	List<VirtualView>
updates	Map<VirtualView, List<JsonObject>>
reset	boolean

KillShotTrack	
skullsLeft	int
killers	List<Player>
board	Board
LOGGER	Logger
MAX_POINTS_TO_GIVE_KILLSHOT_TRACK	int
POINTS_TO_GIVE_STANDARD_DECREASE	int
POINTS_TO_GIVE_LAST_DECREASE	int
THRESHOLD_FOR_LAST_DECREASE	int

Methods summary: model.cards



Attributes summary: model.cards



Methods summary: controller

TurnManager	
runTurn()	void
joinBoard(Player, int, boolean)	void
executeAction()	boolean
executeActualAction(Action)	void
handleUsingPowerUp()	boolean
usePowerUp()	void
handleMoving(Action)	void
handleCollecting()	void
handleShooting()	void
applyFireMode(FireMode)	void
reload(int)	boolean
reloadMandatory()	void
handleTargetingScope(Player, List<Player>)	boolean
askTargetsForGrenade()	void
handleTagbackGrenade(Player)	boolean
handleDeaths()	void
replaceWeapons()	void
replaceAmmoTiles()	void
getDamagesList()	List<Integer>
updateDead()	void
askConfirmation(String)	boolean
askConfirmation(String, Player)	boolean
resetJoinBoard(Player, boolean)	void
resetPowerUp()	void
resetAction()	void
toStringList(List)	List<String>
toUserStringList(List<List<Player>>)	List<String>
updateAndSendModel()	void
updateAndNotifyAll()	void
restoreAndNotify()	void
getVirtualView(Player)	VirtualView
getDead()	List<Integer>
handlePayment(AmmoPack)	void
mandatoryConversion(Color)	void
setDead(List<Integer>)	void

GameEngine	
getPlayers()	List<VirtualView>
getFrenzyActivator()	int
isLastFrenzyPlayer()	boolean
isFrenzy()	boolean
setPlayers(List<VirtualView>)	void
setCurrentPlayer(VirtualView)	void
run()	void
setup()	void
configureMap()	void
configureKillShotTrack()	void
configureFrenzyOption()	void
configurePlayers()	void
battleCry()	void
getNextPlayer()	VirtualView
resolve()	void
runTurn(boolean)	void
changePlayer()	void
manageGameEnd()	void
gameOver()	void
showToLoser(int)	void
addLeaderboard(String)	String
waitShort(VirtualView, int)	String
wait(VirtualView)	String
hasAnswered(VirtualView)	boolean
notify(VirtualView, String)	void
getNotifications()	Map<VirtualView, String>
checkForSuspension()	void
simulateTillEndphase()	void
tryResuming(VirtualView)	boolean
allowPlayersToResume()	void
loadParams()	void
simulationTillEndphaseSetup()	void

PowerUpFactory	
createPowerUp(PowerUpName, Color)	PowerUp

ServerMain	
getInstance()	ServerMain
main(String[])	void
setup()	void
untrackGame(GameEngine)	void
addPlayer(VirtualView)	void
login(VirtualView)	boolean
canResume(String)	boolean
resume(VirtualView)	boolean
removeSuspendedPlayers()	void
getPlayers()	List<VirtualView>
initializeLogger()	void
loadConfig()	Properties
manageInput()	void
refreshConnections()	void
matchmaking()	void
getAlreadyConnected()	String

StatusSaver	
updateCheckpoint()	void
updatePowerups()	void
restoreCheckpoint()	void
restorePowerUps()	void

WeaponFactory	
createWeapon(WeaponName)	Weapon
getWeaponTree(WeaponName)	JsonObject
getColor(JsonObject)	Color
getFullCost(JsonObject)	AmmoPack
getReducedCost(AmmoPack, Color)	AmmoPack
getFireModeName(String)	FireModeName
getFireModeCost(JsonObject)	AmmoPack
getTargetFinder(JsonObject)	TargetFinder
getDestinationFinder(JsonObject)	DestinationFinder
getEffect(JsonObject)	Effect
createEffect(int, int)	Effect
cartesian(List<List<Player>>, List<List<Player>>)	List<List<Player>>

ModelDataReader	
analyzer(String)	JsonObject
analyzer(String, String, int)	JsonObject
getIntBC(String)	int
getIntBC(String, String, int)	int
getBooleanBC(String, String, int)	boolean
getInt(String)	int
getInt(String, String, int)	int
getBoolean(String, String, int)	boolean
getColorBC(String, String, int)	Color
getColorBC(String)	Color
getInt(JsonObject, String)	int
getBoolean(JsonObject, String)	boolean
getColor(JsonObject, String)	Color

BoardConfigurer	
configureMap(int)	Board
configurePlayerOptions(int, Board)	void
configureDecks(Board)	void
setAmmoTilesAndWeapons(Board)	void
configureKillShotTrack(int, Board)	void
simulateScenario()	Board

Timer	
start()	void
stop()	void
reset()	void
update()	void
isRunning()	boolean
isOver()	boolean
pause()	void
resume()	void
getTimeLeft()	long

Attributes summary: controller

GameEngine	
players	List<VirtualView>
leaderboard	List<VirtualView>
currentPlayer	VirtualView
frenzyActivator	int
board	Board
killShotTrack	KillShotTrack
timer	Timer
statusSaver	StatusSaver
gameOver	boolean
frenzy	boolean
lastFrenzyPlayer	boolean
exitGame	boolean
notifications	Map<VirtualView, String>
resuming	List<VirtualView>
endphaseSimulation	boolean
turnDuration	int

Timer	
over	boolean
start	long
duration	long
pausedAt	long
running	boolean

ServerMain	
instance	ServerMain
players	List<VirtualView>
waitingPlayers	List<VirtualView>
currentGames	List<GameEngine>
tcpServer	TCPServer
rmiServer	RMIServer
timer	Timer
executor	ExecutorService
in	BufferedReader
running	boolean
oldMessage	String
LOGGER	Logger
SLEEP_TIMEOUT	int
SERVER_LOG_FILENAME	String

StatusSaver	
board	Board
playersPositions	List<Square>
playersDamages	List<List<Player>>
playersMarks	List<List<Player>>
playersPowerups	List<List<PowerUp>>
playersAmmoPacks	List<AmmoPack>
currentPlayerWeapons	List<Weapon>
currentPlayerLoadedWeapons	List<Boolean>
squareWeapons	List<List<Weapon>>
LOGGER	Logger

TurnManager	
board	Board
statusSaver	StatusSaver
playerConnections	List<VirtualView>
currentPlayerConnection	VirtualView
currentPlayer	Player
dead	List<Integer>
killShotTrack	KillShotTrack
gameEngine	GameEngine
timer	Timer
frenzy	boolean
actionsLeft	int

ModelDataReader	
LOGGER	Logger
parser	JsonParser
boardConfFile	String
miscellaneous	String
DATA_NOT_FOUND	String

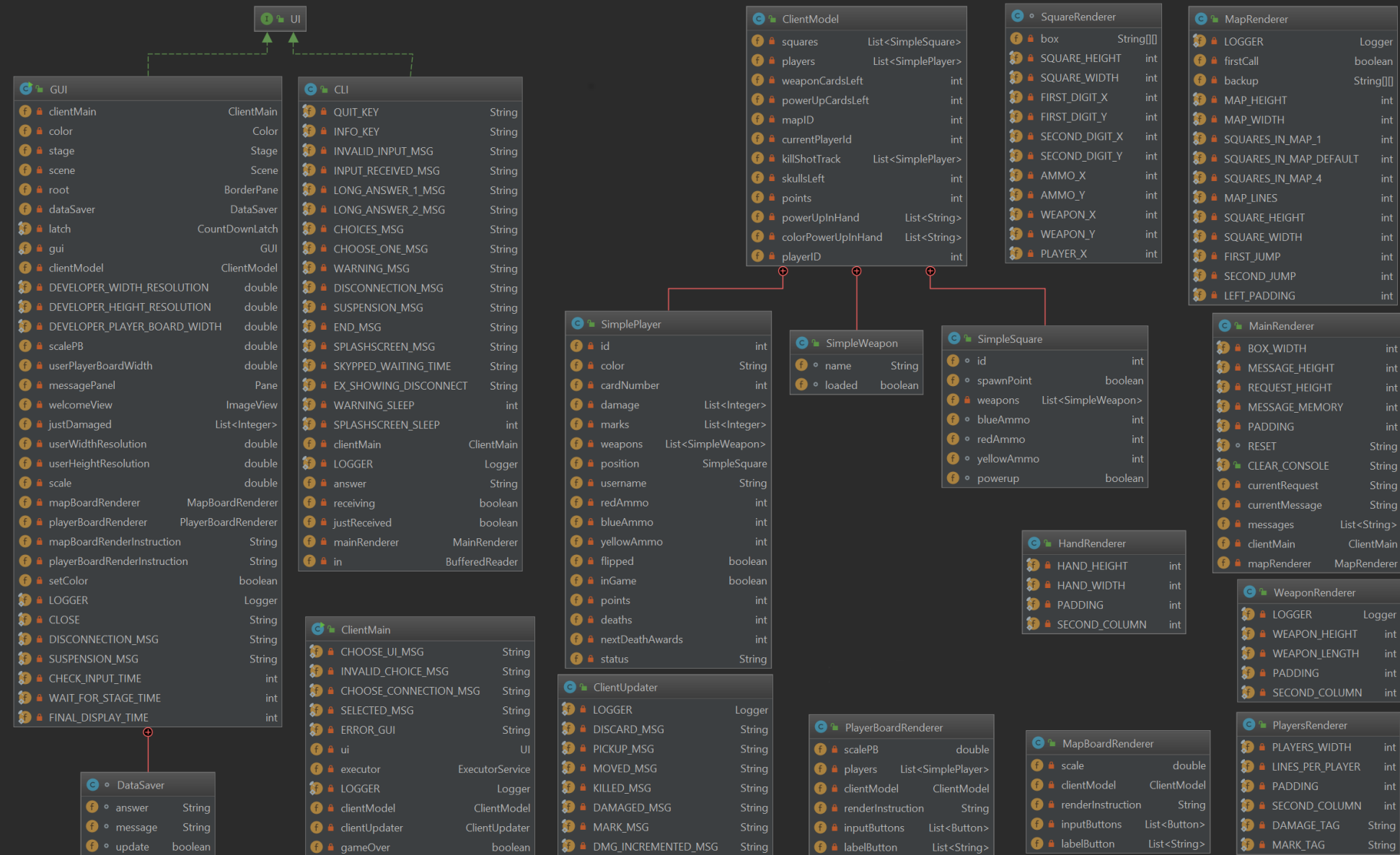
BoardConfigurer	
j	ModelDataReader
BOARDS	String
AMMO_TILES	String
AMMO_SQUARE	String
WEAPON_SQUARE	String
ID	String

PowerUpFactory	
board	Board
j	ModelDataReader
TARGETING_SCOPE_DMG	String
NEWTON_MAX_DISTANCE	String
TAGBACK_GRENADE_MARKS	String

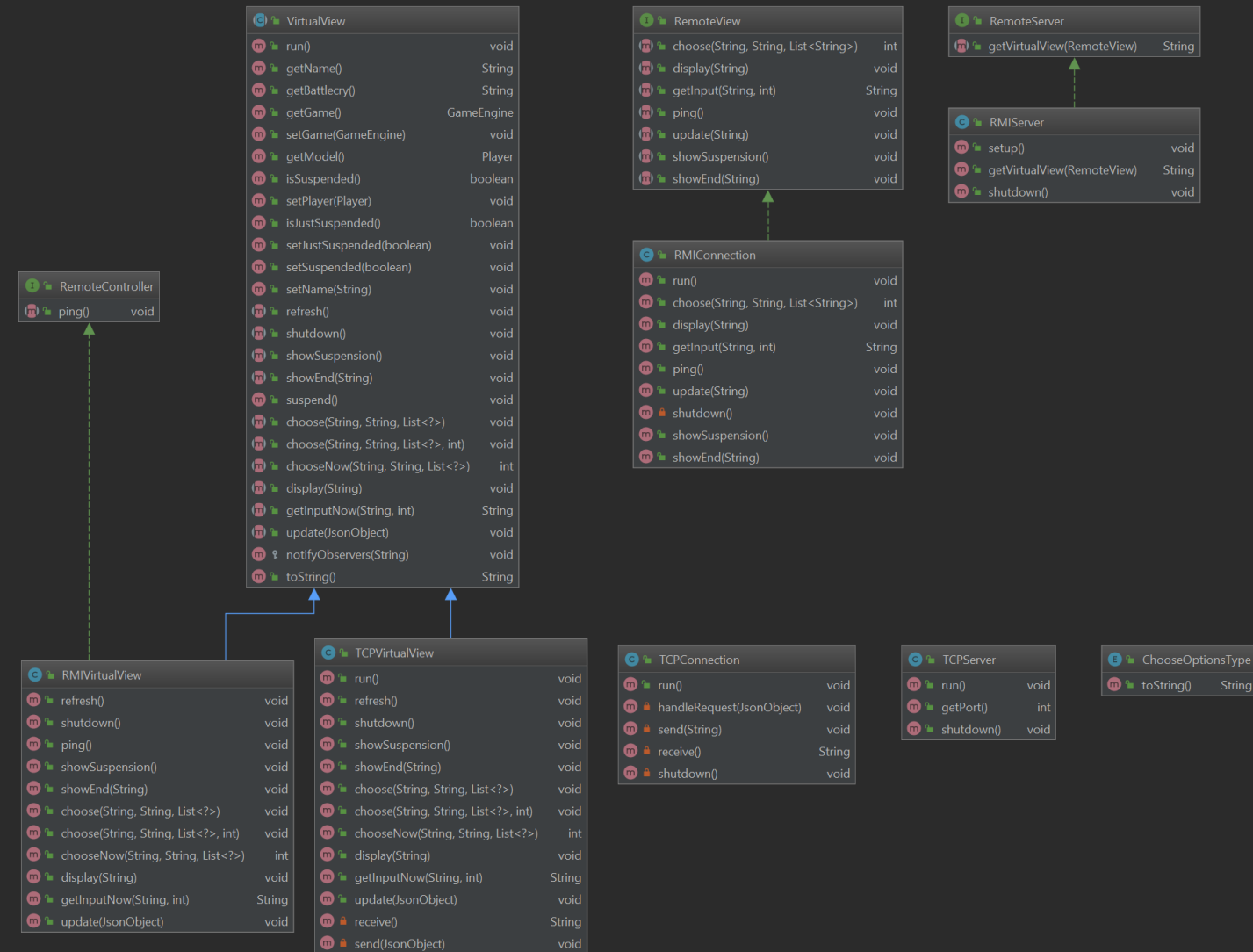
WeaponFactory	
board	Board

Methods summary: view

Attributes summary: view



Methods summary: network



Attributes summary: network

