DefaultRules

- defaultRules : DefaultRules

- DefaultRules()
+ getDefaultRules() : DefaultRules
# dealTool(toolPerGame : int) : ActionCommand
# dealPublicObjective(pubObjPerGame : int) : ActionCommand
# giveTokens() : ActionCommand
# dealGlassWindow(perPlayer : int) : ActionCommand
# dealPrivateObjective(objPerPlayer : int) : ActionCommand

Rules holds the info
on the single actions.

Since the single player mode
is different only in the actions
and not in the structure,
to implement the mode is
only required to make another
rules extension(once the
Server and Client are modified
to accept multiple game
modes)

«Rules»

getSetupGameActions() : List<ActionCommand>
getSetupRoundAction() : ActionCommand
getTurnAction(turnPlayer : Player, secondTurn : boolean) : TurnActionComman
getEndRoundAction() : ActionCommand
getEndGameAction() : ActionCommand
getScore(Player player, Game actionReceiver) : Pair<Player, Integer>
getDraftAction(marker : String, dieColor : String, dieNumber : String) : ActionCommand
getPlaceAction(marker : String, adjacencyRestriction : boolean,
colorRestriction : boolean, numberRestriction : boolean, forced : boolean) : ActionCommand

---

TurnCommand

- actionReceiver : Game

TurnActionCommand(player : Player, secondTurn : boolean)
- startTimer(timer : Timer) : void
- doActionChosen(actionChosen : Identifiable) : void
- doToolAction() : void
- availableTools() : Collection<Tool>
+ getPlayer() : Player
+ reset() : void
- backUp() : void

Invokes
commands
from
set
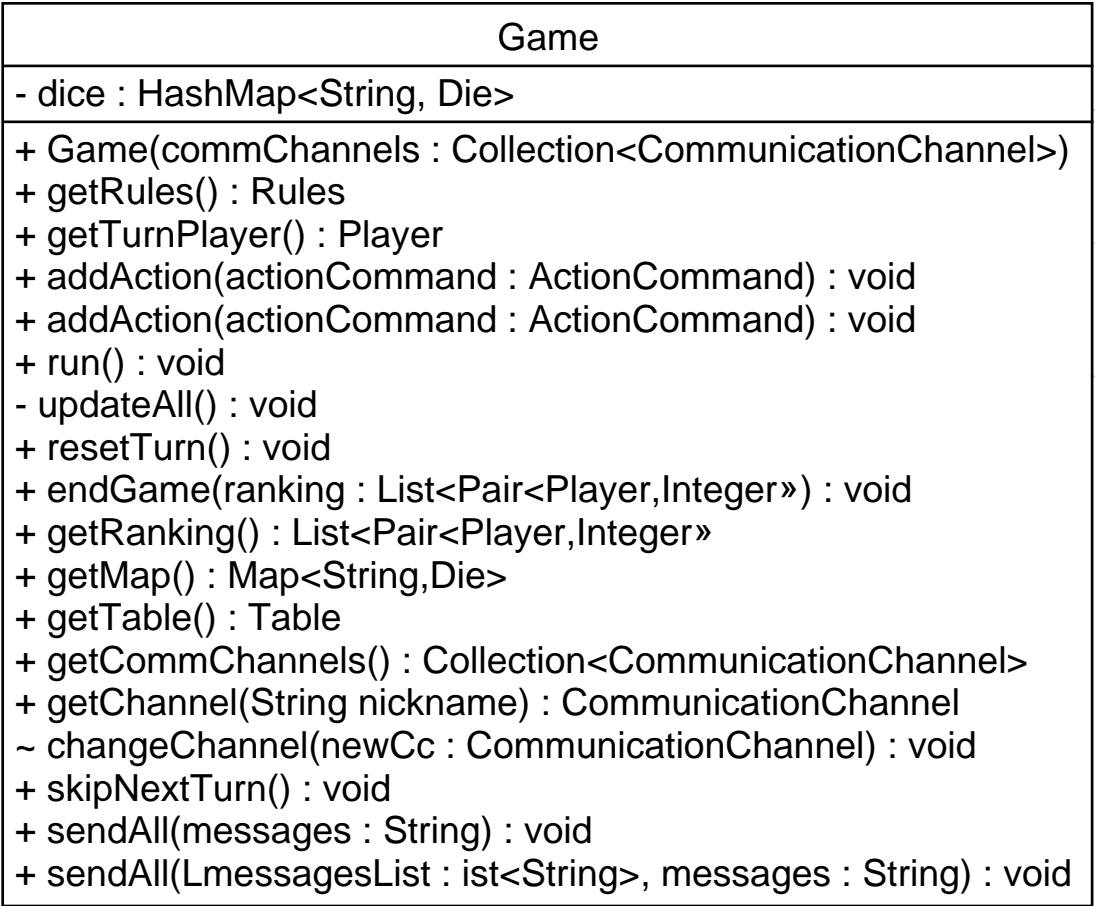
1

0..n

0..n   contains

0..n

Game holds the info
on the main structure of the
game and of the single
rounds.
Upon creations it gets the
single actions from the
Rules it contains and builds
a list of them.
When the game is running
it executes all the actions on
the list and, if only one player
remains online, it stops and
immidiately ends the game.

Game

- dice : HashMap<String, Die>

+ Game(commChannels : Collection<CommunicationChannel>)
+ getRules() : Rules
+ getTurnPlayer() : Player
+ addAction(actionCommand : ActionCommand) : void
+ addAction(actionCommand : ActionCommand) : void
+ run() : void
- updateAll() : void
+ resetTurn() : void
+ endGame(ranking : List<Pair<Player,Integer») : void
+ getRanking() : List<Pair<Player,Integer»
+ getMap() : Map<String,Die>
+ getTable() : Table
+ getCommChannels() : Collection<CommunicationChannel>
+ getChannel(String nickname) : CommunicationChannel
~ changeChannel(newCc : CommunicationChannel) : void
+ skipNextTurn() : void
+ sendAll(messages : String) : void
+ sendAll(LmessagesList : ist<String>, messages : String) : void

---

«ActionCommand»

execute(actionReceiver : Game) : void

0..n   1..n   1   0..n   1   0..n