Package Server GamesHandler waitingRoom: List<GameFlowHandler> - playingUsers: List<GameFlowHandler> ServerConfigFile timer: Timer static final CONFIG_FILE_NAME: String = "server_config" static final secondsTimerSchema: int - GamesHandler() throws NoMorePlayersException - static final secondsTimerStartGame: int + login(name: String, password: String, connection: ClientInterface) throws LoginFailedException, RemoteException static final secondsTimerTurn: int - findGameFlow(nickname: String): Optional<GameFlowHandler> - reconnection(name: String, password: String, connection: ClientInterface): GameFlowHandler # logout(nickname: String) ServerConfigFile() - waitingRoomNewPlayer() # static initConfigParameters() # waitingRoomDisconnection(gameFlow: GameFlowHandler) + static getSecondsTimerSchema(): int startGame() + static getSecondsTimerStartGame(): int # getWaitingPlayers(): List<String> + static getSecondsTimerTurn(): int # getPlayersNick(): List<String> # class TimerExpired{ # goToWaitingRoom(gameFlow: GameFlowHandler) CliListener Server Logger server: Server socketPort: int input: BufferedReader + static print(o: Object) executor: ExecutorService gamesHandler: GamesHandler serverSocket: ServerSocket CliListener(server: Server) listening: boolean + run(): int Server(port: int) - startServer() throws IOException Login setKiller() gamesHandler: GamesHandler # stop() + static main(args: String[]) + Login(gamesHandler: GamesHandler) throws RemoteException + login(name: String, password: String, connection: ClientInterface): FlowHandlerInterface throws LoginFailedException, RemoteException SocketHandler socket: Socket Package RmiInterfaces input: BufferedReader «interface» - output: PrintWriter nickname: String LoginInterface connected: boolear gameFlowHandler: GameFlowHandler gamesHandler: GamesHandler gson: Gson - parser: JsonParser GameFlowHandler - static final NICKNAME_STRING: String = "nickname" - player: Player # SocketHandler(socket: Socket, gamesHandler: GamesHandler) -gameRoom: GameRoom - gamsHandler: GamesHandler getJsonStringValue(message: JsonObject, key: String): String initialSchemas: List<Schema> getJsonPositiveIntValue(message: JsonObject, key: String): int - activeToolcard: ToolCard - connection: ClientInterface placeDice(message: JsonObject): int - toolCardUsed: boolean pass() chooseSchema(message: JsonObject) timer: Timer logout() ,----useToolCard(message: JsonObject) + GameFlowHandler(gamesHandler: GamesHandler, connection: ClientInterface, continueToolCard() player: Player) throws RemoteException notifyLogin(nickname: String) - getPlaver(): Plaver notifyLogin(nicknames: List<String>) getConnection(): ClientInterface notifyLogout(nickname: String) - setGame(game: GameRoom) rotifySchema(schemas: List<Schema>) - checkGameReadv() Package RmiInterfaces - notifyGameInfo(toolCards: Map<String, Boolean>. + disconnected() «interface» publicGoal: List<String>, privateGoal: String) + getInitialSchemas: List<Schema> notifyReconInfo(windows: Map<String, Window>, favorToken: Map<String, Integer>, + reconnection(connection: ClientInterface) FlowHandlerInterface roundTrack: List<Dice>, cardName: String # class TimerExpired{ - notifyToolCardUse(player: String, toolCard: String, window: Window, + run() } draftPool: List<Dice>, roundTrack: List<Dice> startTimer() notifyOtherSchema(playersSchema: Map<String, Schema>) - chooseSchema(schemaNumber: Integer) throws GameNotStartedException, - notifyRound(currentPlayer: String, draftPool: List<Dice>, newRound: boolean, GameOverException, WindowAlreadySetException roundTrack: List<Dice> placeDice(row: int, column: int, dice: Dice) throws GameNotStartedException, notifyDicePlaced(nickname: String, row: int, column: int, dice: Dice) GameOverException, ToolCardInUseException, NotYourTurnException, + notifyEndGame(scores: List<Score>) NoAdjacentDiceException, DiceAlreadyExtractedException, BadAdjacentDiceException, + notifySuspension(nickname: String) FirstDiceMisplacedException, ConstraintViolatedException, DiceNotInDraftPoolException. + askDiceWindow(prompt: String, rollback: boolean): Coordinates DiceAlreadyHereException throws RollbackException, DisconnectionException pass() throws GameOverException, GameNotStartedException, NotYourTurnException askDiceDraftPool(prompt: String, rollback: boolean): Dice + logout() throws RollbackException, DisconnectionException newGame() - askDiceRoundTrack(prompt: String, rollback: boolean): int - useToolCard(cardName: String) throws GameNotStartedException, GameOverException, throws RollbackException, DisconnectionException ToolCardInUseException, NoSuchToolCardException, ToolcardAlreadyUsedException, - asklfPlus(prompt: String, rollback: boolean): boolean NotYourSecondTurnException, AlreadyDraftedException, NoDiceInRoundTrackException, throws RollbackException, DisconnectionException InvalidFavorTokenNumberException. NotEnoughFavorTokenException. + askDiceValue(prompt: String, rollback: boolean): int NoDiceInWindowException, NotYourTurnException, NotDraftedYetException, throws RollbackException, DisconnectionException NotYourFirstTurnException, NoSameColorDicesException, NothingCanBeMovedException, + askMoveNumber(prompt: String, rollback: boolean): int NotEnoughDiceToMoveException, PlayerSuspendedException throws RollbackException, DisconnectionException continueToolCard() throws GameNotStartedException, GameOverException, showDice(dice: Dice) NoSuchToolCardException, ToolcardAlreadyUsedException, + alertDiceInDraftPool(dice: Dice) $Invalid Favor Token Number Exception,\ Not Enough Favor Token Exception,\ Note that the property of the prop$ - login (): boolean NotYourTurnException, PlayerSuspendedException socketSendMessage(json: JsonObject) socketReadLine(): String socketClose() static createErrorMessage(description: String): JasonObject - static createMessage(message: String): JsonObject GameRoom connections: List<ClientInterface> - notifyEndGame: boolean - timer: Timer # GameRoom(playerList: List<Player>, connections: List<ClientConnection>) throws NoMorePlayersException # notifyAllToolCardUsed(nickname: String, toolCard: String, window: Window) # notifyAllDicePlaced(nickname: String, row: int, column: int, dice: Dice) # notifyRound(newRound: boolean) # goOn() - checkGameFinished(): boolean # gameReady() # replaceConnection(oldConnection: ClientConnection, newConnection: ClientConnection) # logout(nickname: String, connection: ClientInterface) # getPlayersNick(): List<String> # class TimerExpired{ + run() } - startTimer() + suspendPlayer(nickname: String) + suspendCurrentPlayer()