ulm university universität
uulm

Fakultät für Ingenieurwissenschaften, Informatik und Psychologie
Institut für Softwaretechnik und Programmiersprachen

Softwaregrundprojekt Meilenstein 6

Softwaregrundprojekt an der Universität Ulm

Vorgelegt von:
Gruppe 10

Dozent:
Florian Ege

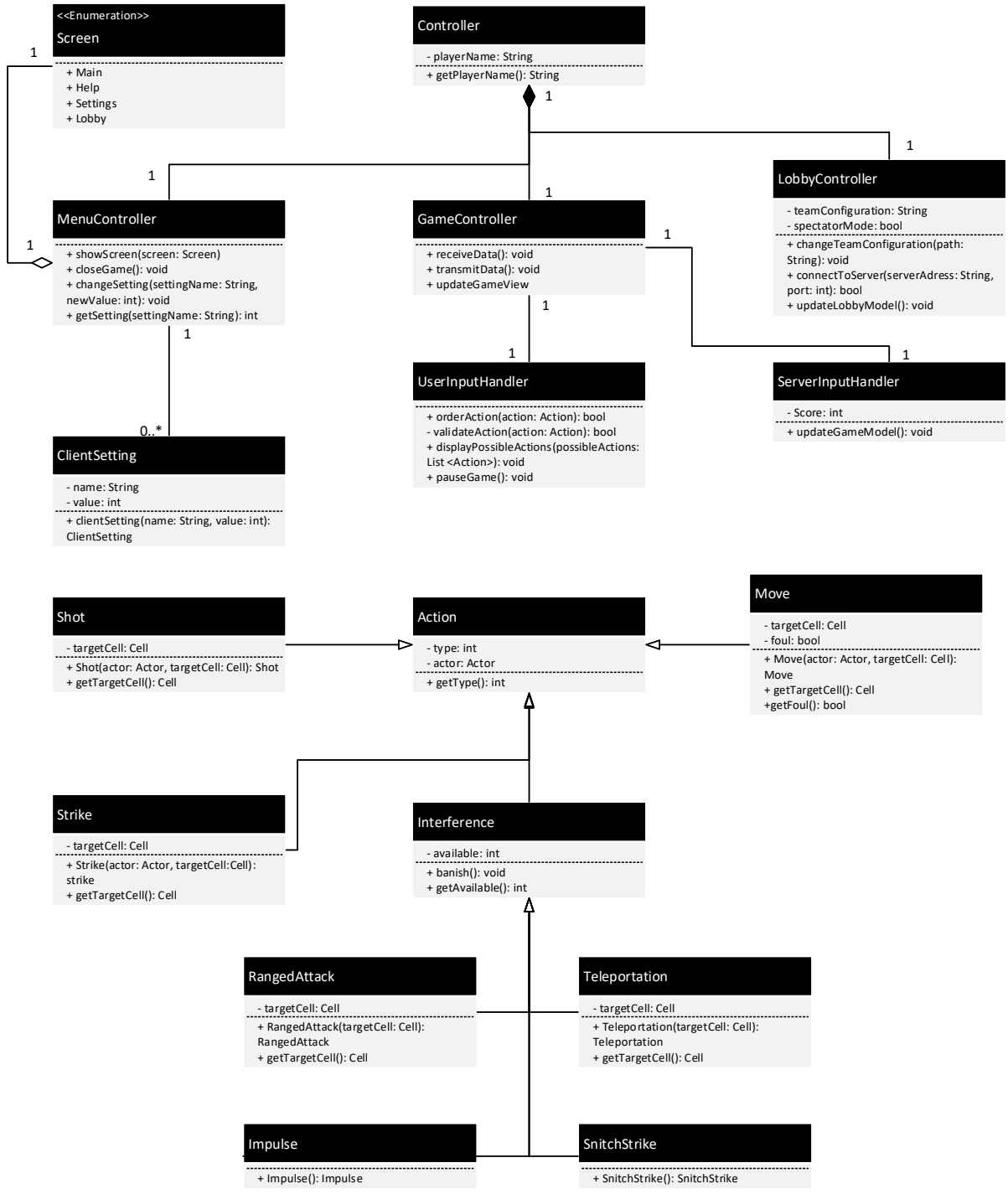
Betreuer:
Stefanos Mytilineos

2019

Inhaltsverzeichnis

1 Controller

1.1 Klassendiagramm


<<Enumeration>>
Screen
+ Main
+ Help
+ Settings
+ Lobby
Controller
- playerName: String
+ getPlayerName(): String
LobbyController
- teamConfiguration: String
- spectatorMode: bool
+ changeTeamConfiguration(path: String): void
+ connectToServer(serverAdress: String, port: int): bool
+ updateLobbyModel(): void
MenuController
+ showScreen(screen: Screen)
+ closeGame(): void
+ changeSetting(settingName: String, newValue: int): void
+ getSetting(settingName: String): int
GameController
+ receiveData(): void
+ transmitData(): void
+ updateGameView
UserInputHandler
+ orderAction(action: Action): bool
- validateAction(action: Action): bool
+ displayPossibleActions(possibleActions: List <Action>): void
+ pauseGame(): void
ServerInputHandler
- Score: int
+ updateGameModel(): void
ClientSetting
- name: String
- value: int
+ clientSetting(name: String, value: int): ClientSetting
Shot
- targetCell: Cell
+ Shot(actor: Actor, targetCell: Cell): Shot
+ getTargetCell(): Cell
Action
- type: int
- actor: Actor
+ getType(): int
Move
- targetCell: Cell
- foul: bool
+ Move(actor: Actor, targetCell: Cell): Move
+ getTargetCell(): Cell
+getFoul(): bool
Strike
- targetCell: Cell
+ Strike(actor: Actor, targetCell:Cell): strike
+ getTargetCell(): Cell
Interference
- available: int
+ banish(): void
+ getAvailable(): int
RangedAttack
- targetCell: Cell
+ RangedAttack(targetCell: Cell): RangedAttack
+ getTargetCell(): Cell
Teleportation
- targetCell: Cell
+ Teleportation(targetCell: Cell): Teleportation
+ getTargetCell(): Cell
Impulse
+ Impulse(): Impulse
SnitchStrike
+ SnitchStrike(): SnitchStrike
1
0..*

1.2 Beschreibung

Im Folgenden werden die eingetragenen Methoden erklärt.

Name	Vorbedingungen	Nachbedingungen	Erklärung
MenuController			
closeGame	-	Positive Antwort in einem Popup-Fenster	Beendet die Anwendung
changeSetting	ClientSetting mit dem angegebenen settingName existiert	Der Wert von newValue wird akzeptiert	Ändert das Attribut value des ClientSetting mit dem Attribut name, der dem Parameter settingName entspricht, auf den Wert des Parameters newValue.
getSetting	ClientSetting mit dem angegebenen settingName existiert	-	Liefert den derzeitigen Wert des Attributs value des ClientSetting, dessen Attribut name mit dem Parameter settingName übereinstimmt.
GameController			
receiveData	-	-	Empfängt Daten vom JSON-Parser und übergibt sie dem ServerInputHandler
transmitData	-	-	Empfängt Daten vom UserInputHandler und sendet sie über den JSON-Parser und den Kommunikator zum Server
LobbyController			
changeTeamConfiguration	-	Parameter path ist ein gültiger Pfad zu einer Team-Konfigurations-Datei	Ändert den Wert von teamConfiguration, in dem der Pfad zu der zu verwendenden Team-Konfigurations-Datei gespeichert ist.

connectToServer	Parameter server-Adress und port sind ungleich Null	-	Veranlasst über den JSON-Parser den Kommunikator dazu, eine Verbindung mit dem in den Parametern spezifiziertem Server aufzubauen. Gibt true zurück, wenn der Verbindungsaufbau erfolgreich war, ansonsten false.
UserInputHandler			
validateAction	-	-	Fordert vom Partie-Model Daten an und entscheidet danach, ob die geforderte Action durchgeführt werden kann. Gibt true zurück wenn die als Parameter übergebene Action gültig ist, ansonsten false.
orderAction	validateAction gibt für die als Parameter übergebene Action true zurück.	-	Übergibt die auszuführende Aktion an den GameController. Ruft die Methode validateAction auf und gibt deren Rückgabewert zurück.
displayPossibleActions	-	-	Fordert vom Partie-Model Daten an und berechnet alle möglichen Aktionen und gibt sie an die Partie-View weiter, um sie dem Spieler anzuzeigen.
Interference			
banish	-	-	Verringert das Attribut available um eins.

1.3 Zuordnung der Funktionalen Anforderungen

Die Funktionalen Anforderungen werden den Methoden folgendermaßen zugeteilt:

Funktionale Anforderungen	**Methoden**
FA7	Shot::shot Shot::getTargetCell
FA21	Shot::shot
FA26	Move::move
FA27	Shot::shot
FA28	Strike::strike
FA30	Move::move
FA31	RangedAttack::rangedAttack Teleportation::teleportation Impulse::impulse SnitchStrike::snitchStrike Inteference::banish
FA32	Teleportation::teleportation
FA33	RangedAttack::rangedAttack
FA34	Impulse::impulse
FA35	SnitchStrike::snitchStrike
FA36	Inteference::banish Actor::setBanished
FA37	Actor::setBanished Move::move
FA38 - FA42	Move::move
FA48	Actor::setBanished
FA54	LobbyController::changeTeamConfiguration LobbyController::updateLobbyModel
FA55	GameController::receiveData GameController::transmitData LobbyController::connectToServer
FA56	GameController::receiveData GameController::transmitData
FA60-FA61	MenuController::showScreen

FA67	UserInputHandler::orderActions UserInputHandler::validateActions MenuController::showScreen MenuController::changeSetting LobbyController::changeTeamConfiguration LobbyController::connectToServer
FA69	UserInputHandler::pauseGame