Implementation
Project Black Jack

by Andrea Kaminski, Dajana Berthold, Michael Freiwald, Andreas Mayer, Matthias Müller-Brockhausen, Daniel Sikeler

Implementation Overview

As discussed in the System Architecture Document we mostly just implemented what was described in there.
We had to adjust the Blackjack Framework a little for Mutliplayer and we created a lot of Agents so that we can let them challenge each other and maybe gain statistical Data.
We also implemented a Base Agent to facilitate further Implementations by eliminating repeated Code.

Tools / Frameworks / Dev Environment

As our Implementation Language we chose Java because everyone in our Team was equally familiar with it. The next step was to find a Blackjack Framework in that Langauge that is open-source.

For that we found our building Blocks for our AI Implementations in the Blackjack Framework called Monte-Jack (http://garrettsmith.net/montejack/).

As Developement Environment we used Eclipse because it makes building and organizing Java Projects very easy and everyone in our Team was already familiar with it.

To distribute the Code evenly and allow multiple People to work on it simultaneously we choose to use Git. Fortunately one of our Team Members is a Github Educational User so we were able to obtain a private repository.

Obstacles / Issues

Unfortunately the Montejack-Blackjack Framework only supports one player vs. one Dealer so we had to make crucial changes to the core of the Framework to enable multiplayer. This was one of the most tedious Tasks (of course apart from the Agent Implementation)

For one of the Agents the “Wall Hack Agent” who knows all the cards we had to add an additional Game Event Listener since usually Agents should not be able to see more than they are supposed to so this specific Agent had to get an own API to “cheat”.

Changes in Design and Implementation

In our initial Design we did not have to change much since we already thought it through pretty well.
(only the gameeventlistener for the Wall Hack Agent which is an exception)
Design-Wise the following Things are now completed:
- Multiple Agents can compete against each other
- A Base Agent Class that makes implementing the specific Agents a lot easier
- Wall-Hack Agent (Omniscient Agent)
- Reflex Agent
- Learning Agent
- AlwaysStand Agent (for debugging purposes)