Software-Challenge 2021

STARTUP BRIDGE

Inhalt

Software-Challenge Endbenutzer-Dokumentation

Ziel dieser Dokumentation ist es, alle Informationen über die Software-Challenge an einer Stelle zu bündeln. Für die teilnehmenden Teams ist hier alles dokumentiert, was für den Ablauf des Wettbewerbes benötigt wird. Die Software-Challenge wird 2021 wieder mit dem Programmierwettbewerb aus Kiel kombiniert. Die Infrastruktur und die Aufgabenstellung wurden von dem Kieler Software-Challenge Team bereitgestellt. Die Aufgabe besteht darin, einen Spieler für das Brettspiel Blokus zu programmieren.

Teilnahmebedingungen und Registrierung

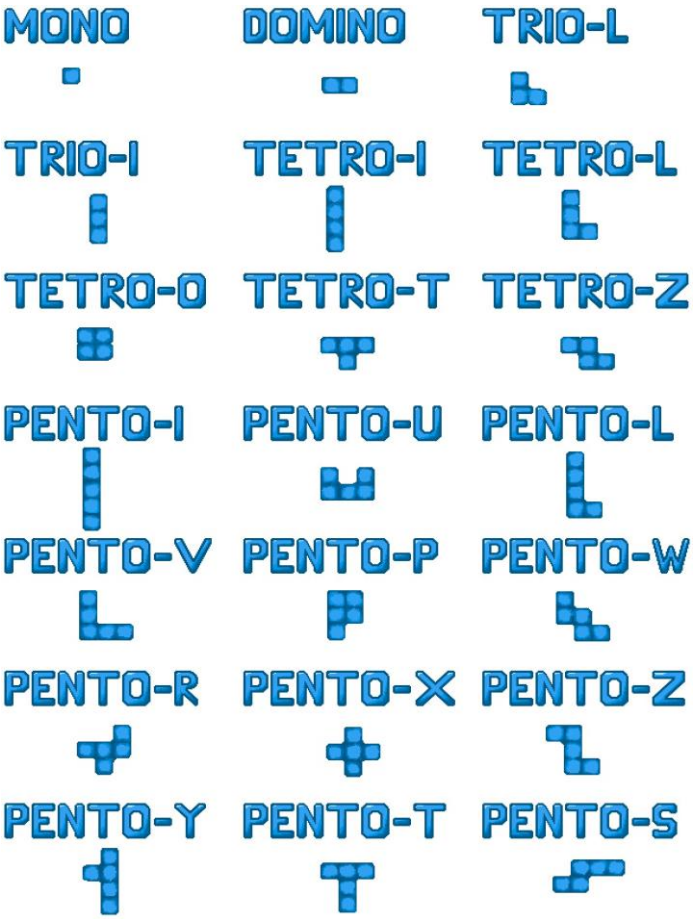
Teilnahmeberechtigt sind alle Studierende und Mitarbeiter der FH Wedel. Externe sind leider nicht erlaubt. Die Anmeldung erfolgt über die folgende Webseite: **Zur Anmeldung**
Eine kurze Anleitung zur Anmeldung befindet sich unter den eben genannten Link.

Das Brettspiel Blokus

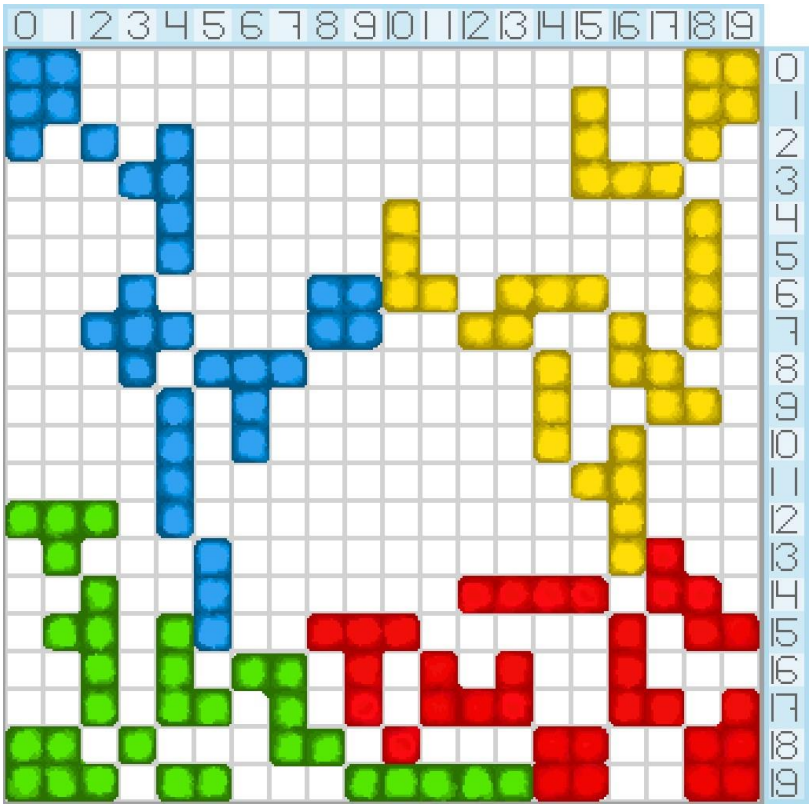
Blokus ist ein abstraktes Strategie-Spiel mit Tetris-artigen, farbigen Spielsteinen, die ihr auf das Board setzt. Ihr müsst dabei beachten, dass ein Spielstein, den ihr neu legen wollt, nicht eure bereits gelegten Spielsteine ganz berühren darf, sondern nur mit einer oder mehreren Ecken. Jeder gelegte Stein zählt so viele Punkte, wie er Quadrate hat. Darüber hinaus können Spieler Bonuspunkte bekommen. Der Spieler mit den meisten Punkten gewinnt das Spiel.

Das Spielmaterial

Jeder Spieler verfügt über zwei Sätze von 21 Spielsteinen (Polyominos), die sich aus kleinen Quadraten zusammensetzen. Der eine Spieler hat blaue und rote Steine, der andere gelbe und grüne. Dabei kommt jede Form, die aus 1–5 Quadraten besteht, in jeder Farbe genau einmal vor: also 1 Monomino, 1 Domino, 2 Triominos, 5 Tetrominos und 12 Pentominos.



Das Brett besteht aus 20 × 20 quadratischen Feldern, welche dieselbe Größe wie die Quadrate der Spielsteine haben.
Die Koordinaten der Felder des Bretts beginnen in der linken oberen Ecke bei x=0, y=0 und folgen den Regeln des kartesischen Koordinatensystems, wobei die positive x-Achse nach rechts und die positive y-Achse nach unten verläuft.



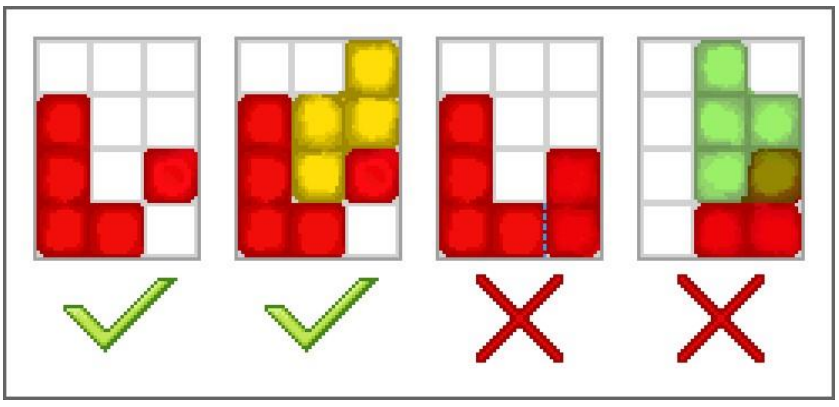
Der Spielablauf

Die Spielsteine werden abwechselnd gelegt, wobei in den Ecken begonnen wird. Die Reihenfolge ist: Blau – Gelb – Rot – Grün. Der erste Stein des zweiten Spielers darf auch diagonal gegenüber dem ersten Stein des ersten Spielers gelegt werden.

Für den ersten Stein jeder Farbe gelten besondere Regeln: Er muss so gesetzt werden, dass ein Eckfeld des Spielbretts (also (0,0), (19,0), (0,19) oder (19,19)) besetzt wird. Die Form des ersten Steins wird vom Server vorgegeben. Sie wird zufällig aus allen verfügbaren Formen ausgewählt, die aus 5 Quadraten bestehen. In einem Spiel ist die Form des ersten Steins für alle Farben gleich.

Für die Platzierung aller weiteren Steine gilt: Zwei Steine derselben Farbe dürfen sich nur an den Ecken berühren. Wird ein neuer Stein gelegt, muss dieser mindestens einen Stein derselben Farbe an mindestens einer Ecke berühren. Dabei ist es egal, wie Steine anderer Farben berührt werden, solange diese nicht überdeckt werden.

Beispielbilder mit gültigen und ungültigen Berührungen



Die Steine dürfen umgedreht werden (die Oberseite nach unten) und sind frei rotierbar (in 90° Schritten). Die Steine müssen vollständig auf dem Feld liegen und dürfen sich nicht mit anderen Steinen überlappen. Gelegte Steine dürfen nicht mehr bewegt werden.

Ziel ist es, möglichst viele Felder auf dem Brett mit den eigenen Farben zu belegen.

Es wird so lange gezogen, bis keine Steine mehr gesetzt werden können. Wenn Steine einer Farbe nicht mehr gelegt werden können, wird diese Farbe im weiteren Verlauf des Spiels übersprungen (die gelegten Steine zählen aber mit in die Wertung). Kann kein Stein irgendeiner Farbe mehr gelegt werden, endet das Spiel.

Der Computerspieler hat für das Legen eines Spielsteines zwei Sekunden Zeit.

Der Spielleiter ruft den Computerspieler nur dann zu einem Zug einer Farbe auf, wenn es auch noch mindestens einen möglichen Zug gibt. Der Computerspieler hat dann jedoch auch die Möglichkeit, mit einem "Passen"-Zug zu antworten. Dadurch wird die aktuelle Farbe in der aktuellen Runde ausgelassen. "Passen" ist erst nach dem ersten Zug erlaubt.

Eine Runde besteht aus vier Zügen. Pro Zug kann eine Farbe einen Stein setzen oder passen. Auch wenn eine Farbe übersprungen wird, zählt dies hierbei als Zug. Das Spiel endet, sobald alle Steine einer Farbe auf das Spielfeld gelegt wurden, spätestens aber nach 25 Runden. In jedem Fall wird die aktuelle Runde noch zu Ende gespielt.

Die Wertung

Jeder gelegte Stein zählt so viele Punkte, wie er Quadrate hat (ein gelegtes Pentomino zählt z.B. fünf Punkte). Wurden alle Steine einer Farbe gelegt, gibt es 15 zusätzliche Punkte. Wurden alle Steine einer Farbe gelegt und war zusätzlich der letzte gelegte Stein der Monomino, gibt es nochmal fünf zusätzliche Punkte (insgesamt also 20). Die Punkte der jeweiligen zwei Farben, die ein Spieler kontrolliert, werden zusammengezählt und sind die Gesamtpunkte des Spielers. Der Spieler mit den meisten Punkten gewinnt das Spiel.

In der Meisterschaftsphase ist die durchschnittliche Punktzahl nachrangiges Kriterium. D.h. haben zwei Teams gleich viele Siege, bekommt das Team mit der höheren durchschnittlichen Punktzahl den besseren Tabellenplatz.

Der Spielleiter (Server)

Die beiden Computerspieler kommunizieren nicht direkt miteinander, sondern übertragen ihre Nachrichten über einen Mittelmann: den Spielleiter. Dadurch ist zum einen sichergestellt, dass man seinen Gegner nicht mit illegalen Nachrichten belästigen kann, zum anderen sorgt der Spielleiter dafür, dass sich die Kontrahenten an die Spielregeln halten.

Der Spielleiter ist direkt im Wettkampfsystem integriert, so dass alle Turnierspiele regelkonform gespielt werden müssen. Zum Testen des eigenen Computerspielers gibt es eine spezielle Version des Spielleiters, die im Downloadbereich (**https://software-challenge.de**) heruntergeladen werden kann. Diese Download-Version, die in diesem Abschnitt beschrieben wird, besitzt eine grafische Oberfläche, durch die man das Spiel gut verfolgen und sogar als Mensch mitspielen kann.

System vorbereiten und Spielleiter starten

Die einzige Voraussetzung ist, dass auf dem Rechner mindestens die Laufzeitumgebung für Java 8 installiert ist. Siehe „Installation von Java“.

Nach der erfolgreichen Installation kann man den Server durch einen Doppelklick auf die Datei software-challenge-gui starten.

Die Programmoberfläche

Die Programmoberfläche besteht aus dem Menü oben sowie der Spielfläche darunter.

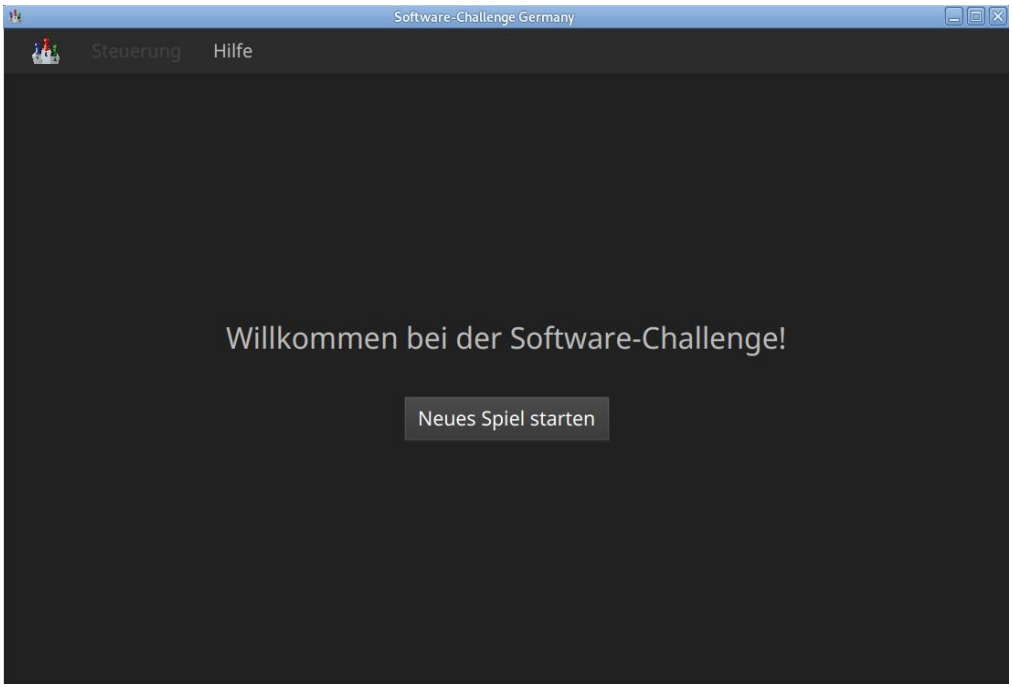


Figure 1. Die Spielleiteroberfläche direkt nach dem Programmstart

Unter dem ersten Menüpunkt (Symbol der Software-Challenge) findet man alle möglichen Aktionen.

Zu Beginn ist die Spielfläche leer.

Ein neues Spiel erstellen

Um ein Spiel zu spielen, muss zunächst "Neues Spiel starten" angeklickt werden.

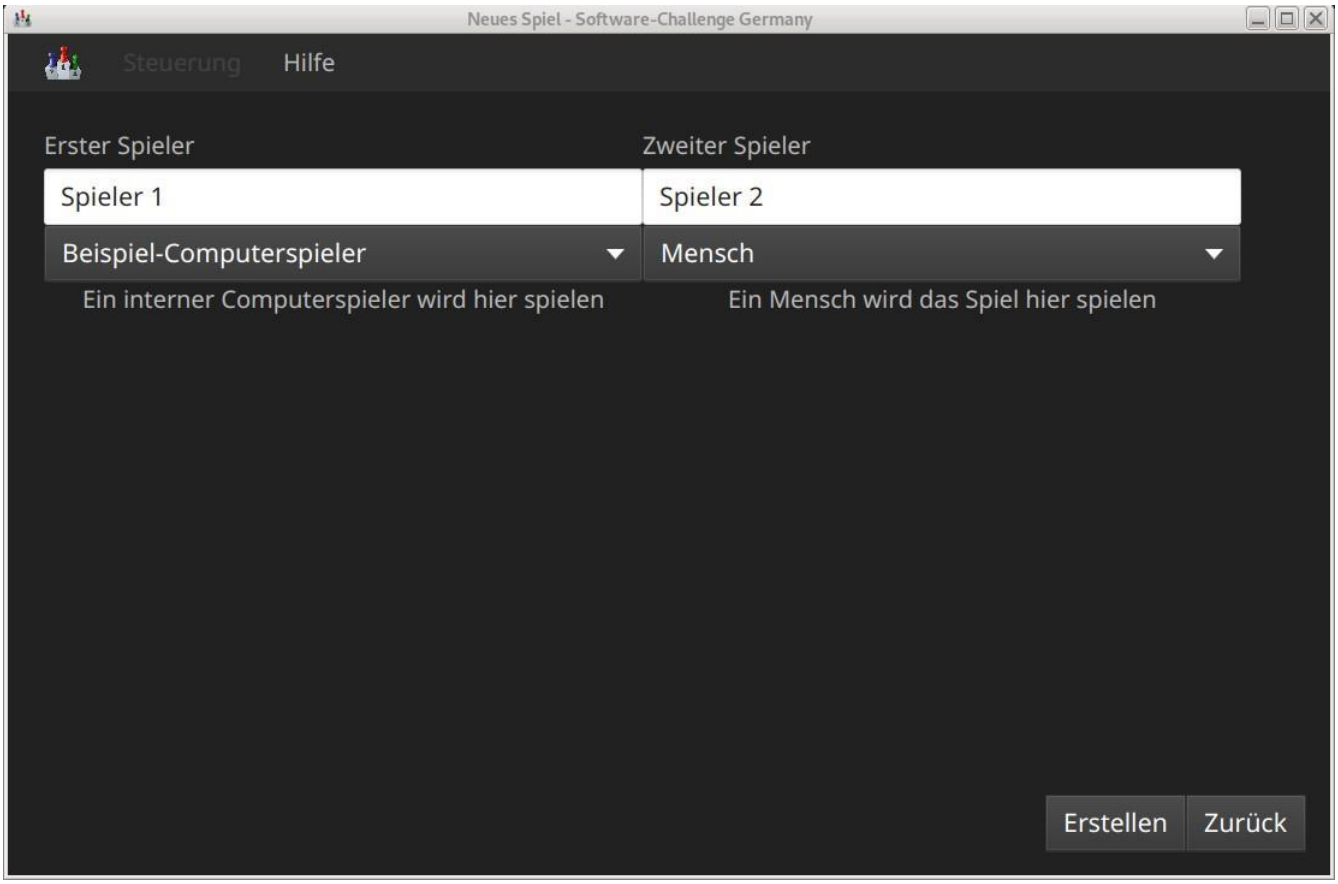


Figure 2. Dialog für ein neues Spiel

In diesem Fenster werden die Spieler ausgewählt, die an dem Spiel teilnehmen sollen. Für jeden Spieler gibt es folgende Optionen:

- **Text-Eingabefeld:** Hier kann für jeden Spieler ein Name eingegeben werden, der dann im Spiel angezeigt wird.
- **Spielertyp:** Es kann zwischen 4 verschiedenen Spielertypen gewählt werden:
  1. Mensch: Ein menschlicher Spieler, der über die Programmoberfläche spielt.
  2. Beispiel-Computerspieler: Ein Computerspieler, der im Server integriert ist.
  3. Computerspieler, von GUI gestartet: Ein Computerspieler in Form eines separaten Programms, das beim Starten des Spiels automatisch vom Server gestartet wird.
  4. Manuell gestarteter Client: Ein Computerspieler in Form eines separaten Programms, das manuell durch den Benutzer gestartet werden muss.

Nach Eingabe der erforderlichen Werte kann das Spiel mithilfe des unteren Knopfs "Erstellen" erstellt werden.

Die Spielfeldoberfläche

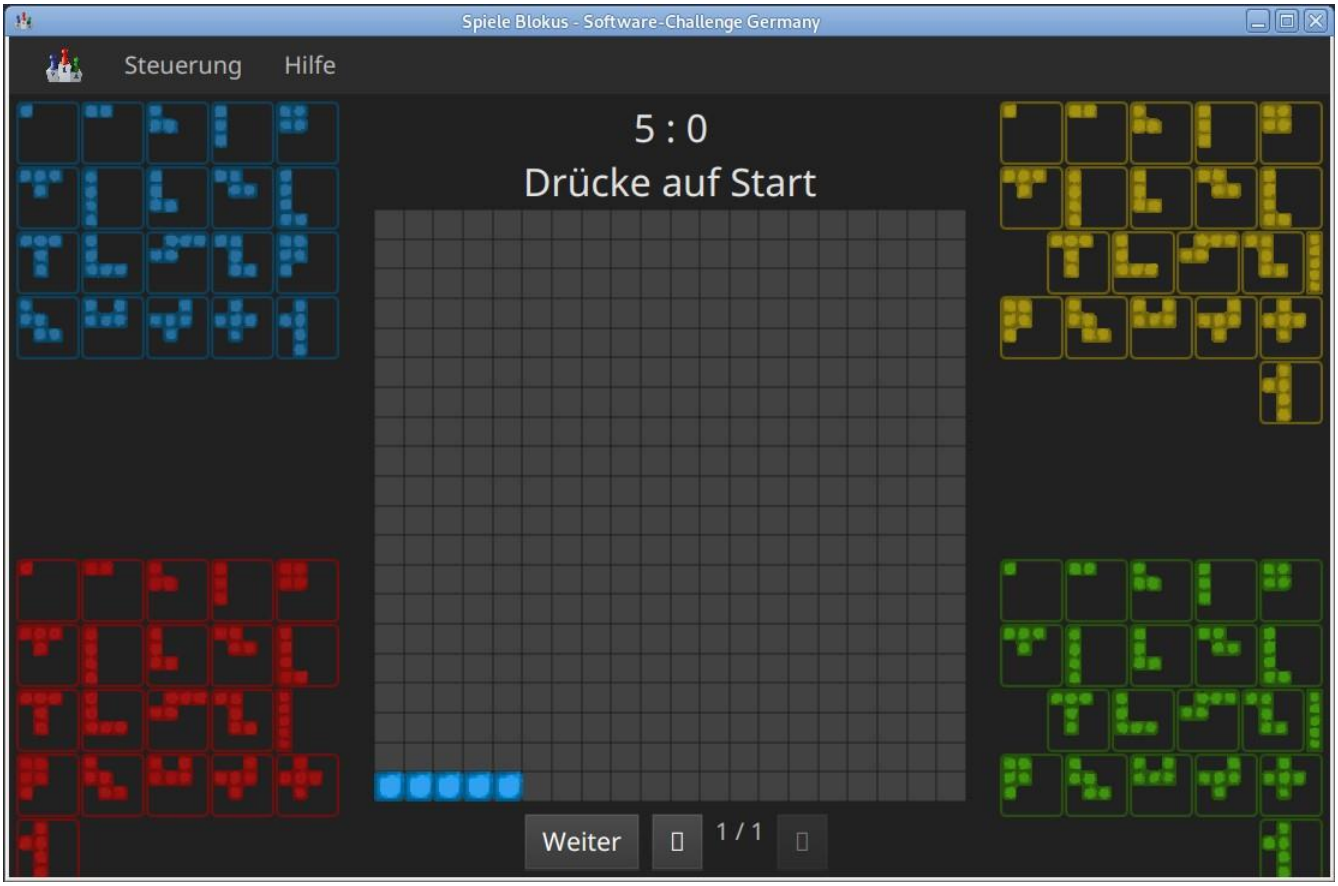


Figure 3. Die Spielfeldoberfläche (hier mit dem Spiel "Blokus")

Auf dem Spielbrett werden das eigentliche Spiel (hier z.B. "Blokus"), die Züge und weitere für das Spiel wichtige Informationen dargestellt. Hier setzt der menschliche Spieler auch seine Züge.

Die Steuerelemente unterscheiden sich je nach Spiel und Spielsituation. Unten gibt es immer die Schaltflächen "Anhalten/Weiter", Für das Spiel Blokus:

- Der aktuell gewählte Spielstein wird unten angezeigt, über das Menü Steuerung kann man diesen rotieren (auch über Mausrad) und spiegeln (auch über rechte Maustaste)
- Aus der Menge noch verfügbarer Spielsteine kann man mit Mausklick einen anderen Spielstein auswählen.
- Mithilfe der Maus wird der gewählte Spielstein auf dem Feld abgelegt.

Spielwiederholungen

Spielwiederholungen oder Replay-Dateien sind aufgezeichnete frühere Spiele, die man sich beliebig oft wieder ansehen kann, um beispielsweise einen Fehler des eigenen Spielers zu analysieren oder eine Strategie zu verbessern.

Um das aktuelle Spiel als Spielwiederholung zu speichern, klickt man auf das Icon ganz rechts unten im Spielbereich. Dann kann man einen Dateinamen und Speicherort festlegen.

Um eine gespeicherte Spielwiederholung zu laden, verwendet man den Eintrag "Replay laden" in der linken Leiste. Nachdem man eine Datei ausgewählt hat, kann man das gespeicherte Spiel abspielen oder Schritt für Schritt durchgehen.

Testdurchläufe

Wenn man einen grundsätzlich funktionierenden Computerspieler programmiert hat, ist es sinnvoll, diesen mit vielen verschiedenen Spielsituationen zu konfrontieren. Dadurch lassen sich Fehler entdecken und die Spielstärke des Computerspielers beurteilen. Für solche Testdurchläufe wird ein Testserver und TestClient zur Verfügung gestellt. Wie man diese verwendet, ist weiter unten unter der Überschrift „Automatische Spiele: Der Testserver“ und „Massentests“ beschrieben.

Spielsituation nachstellen

Wenn ein Fehlerverhalten des Computerspielers nur in einer bestimmten Situation in einem Spiel auftritt, kann es oft wünschenswert sein, diese Situation erneut nachzuspielen um den Computerspieler gezielt zu verbessern.

Dies ist zurzeit nur auf etwas kompliziertem Wege möglich. Es folgt eine Schritt-für-Schritt Anleitung:

1. Laden Sie das betreffende Replay aus dem Wettkampfsystem herunter (.xml.gz Datei).
2. Entpacken Sie das Replay, sodass sie eine .XML-Datei erhalten.
3. Starten Sie den Server und erstellen Sie ein neues Spiel. Wählen Sie den Computerspieler, der für diese Spielsituation getestet werden soll. Dieser Spieler muss als Spieler 1 gestartet werden und ist dann direkt als erstes dran. Der Gegenspieler kann dann ein beliebiger Computerspieler oder auch ein Mensch sein.
4. Setzen Sie einen Haken bei "Spiel aus Datei laden". Wählen Sie über "Datei wählen" das entsprechende Replay aus und spezifizieren sie den Zug in dem gestartet werden soll. Starten Sie dann das Spiel. Das Spiel sollte sich nun in genau der Situation befinden, in der das Fehlerverhalten aufgetreten ist. Dabei ist der Spieler, der nun dran ist immer der rote Spieler. Falls der blaue Spieler eigentlich dran war, werden die Farben der Spieler getauscht.
5. Nun kann der nächste Zug beim Spieler angefordert werden und dabei durch Debugging kontrolliert werden, wo sich der Spieler falsch verhalten hat. Achtung: Wenn weitere Züge angefordert werden, kann das Verhalten vom normalen Spielverlauf abweichen, da evtl. nicht alle Daten für das Spiel in der XML vorhanden sind.

Replay mit Server ohne graphische Oberfläche speichern

Wenn der Server ohne die graphische Oberfläche gestartet wird, kann das --saveReplay Attribut gesetzt werden, damit bei Ende jedes Spiels das Replay des Spiels unter *./replays* gespeichert wird.

```
java -Dfile.encoding=UTF-8 -Dlogback.configurationFile=logback.xml -jar softwarechallenge-server.jar --port 13051 --saveReplay true
```

Automatische Spiele: Der Testserver

Wenn Sie automatisiert Spiele mit Ihrem Computerspieler spielen wollen, um bestimmte Verhaltensweisen bei der Weiterentwicklung regelmäßig zu testen, können Sie dafür einen speziellen Server ohne grafische Oberfläche verwenden, den sogenannten Testserver. Hier ist zu beachten, dass der Testserver auf dem Port 13051 gestartet wird und nicht, wie im normalen Spiel auf Port 13050.

Gehen Sie dazu wie folgt vor:

1. Laden Sie den Testserver von der Download-Seite herunter.
2. Entpacken Sie das heruntergeladene Archiv.
3. Wechseln Sie in einer Kommandozeilenumgebung (Windows: cmd.exe oder Powershell, Linux: beliebige Shell oder Terminal) in das Verzeichnis des entpackten Archives.
4. Starten Sie den Testserver auf dem Port 13051 mit folgendem Befehl:

   ```
   java -Dfile.encoding=UTF-8 -Dlogback.configurationFile=logback.xml -jar softwarechallenge-server.jar --port 13051
   ```

5. Starten Sie Ihren Computerspieler und einen zweiten Computerspieler manuell auf dem Port 13051 (im SimpleClient geht dies mit der Option --port 13051) in weiteren Kommandozeilenumgebungen. Die Computerspieler verbinden sich automatisch zum Testserver und es wird ein Spiel gespielt. Danach sollten sich die Computerspieler automatisch beenden.
6. Wenn Sie weitere Testspiele starten wollen, können Sie die Computerspieler erneut starten. Der Testserver muss nicht neu gestartet werden.

Beachten Sie, dass der Testserver keine Spielaufzeichnungen anlegt, wie es der Server mit grafischer Oberfläche tut. Die Auswertung der Spiele muss in einem der teilnehmenden Computerspieler geschehen (z.B. durch Log-Ausgaben).

Es ist ebenfalls möglich, statt eines zufällig generierten vollständigen Spielplanes eine Spielsituation zu laden und zu testen. Die Spielsituation muss vorher wie unter „Spielsituation nachstellen“ erzeugt werden. Dann kann die Datei mit dem Argument --loadGameFile geladen werden und optional mit --turn ein Zug spezifiziert werden.

```
java -Dfile.encoding=UTF-8 -Dlogback.configurationFile=logback.xml -jar softwarechallenge-server.jar --port 13051 --loadGameFile ./replay.xml --turn 10
```

Unerwartete Zugzeitüberschreitungen (Soft-Timeout)

Wenn Sie den Testserver einige Zeit laufen lassen, um eine größere Anzahl von Testspielen durchzuführen, kann es dazu kommen, dass Computerspieler wegen Zugzeitüberschreitungen vom Server disqualifiziert werden (Soft-Timeout). Dies passiert, obwohl sie ihren Zug innerhalb der erlaubten Zugzeit (abhängig vom Spiel, bisher aber immer zwei Sekunden) an den Server geschickt haben. Der Garbage Collector der Java Virtual Machine löst dieses Verhalten aus. Er pausiert die Anwendung, um nicht mehr genutzten Speicher freizugeben. Wenn der Server dadurch zu einem ungünstigen Zeitpunkt angehalten wird, bemerkt er den Eingang des Zuges vom Computerspieler nicht rechtzeitig und disqualifiziert ihn daraufhin. Damit dieses Problem möglichst selten auftritt, haben sich die folgenden Parameter beim Starten des Servers bewährt:

Unter Linux:

```
java -Dfile.encoding=UTF-8 \
    -Dlogback.configurationFile=logback.xml \
    -server \
    -XX:MaxGCPauseMillis=100 \
    -XX:GCPauseIntervalMillis=2050 \
    -XX:+UseConcMarkSweepGC -XX:+CMSParallelRemarkEnabled \
    -XX:+UseCMSInitiatingOccupancyOnly -XX:CMSInitiatingOccupancyFraction=70 \
    -XX:+ScavengeBeforeFullGC -XX:+CMSScavengeBeforeRemark \
    -jar softwarechallenge-server.jar --port 13051
```

Unter Windows (unterscheidet sich nur durch die Art, den langen Befehl auf mehrere Zeilen zu verteilen):

```
java -Dfile.encoding=UTF-8 ^
    -Dlogback.configurationFile=logback.xml ^
    -server ^
    -XX:MaxGCPauseMillis=100 ^
    -XX:GCPauseIntervalMillis=2050 ^
    -XX:+UseConcMarkSweepGC -XX:+CMSParallelRemarkEnabled ^
    -XX:+UseCMSInitiatingOccupancyOnly -XX:CMSInitiatingOccupancyFraction=70 ^
    -XX:+ScavengeBeforeFullGC -XX:+CMSScavengeBeforeRemark ^
    -jar softwarechallenge-server.jar --port 13051
```

Um das Verhalten des Garbage Collectors noch weiter zu verbessern, kann man auch noch mittels der Optionen

```
-XX:+PrintGCDateStamps -XX:+PrintGC -XX:+PrintGCDetails -Xloggc:"pfad_zum_gc.log"
```

eine Logdatei über die Aktivitäten des Garbage Collectors anlegen. Darin sieht man genau, wann er wie lange lief. Man kann dann die Einstellungen verändern und testen, ob sich das Verhalten verbessert.

Die Konfiguration des Garbage Collectors ist kein Allheilmittel und kann zu neuen Problemen führen, auf die man gefasst sein sollte. Dazu gehören erhöhter Ressourcenverbrauch und Instabilität der Anwendung.

Massentests mit Server ohne graphische Oberfläche

Massentests mit dem eigenen Computerspieler können sehr nützlich sein, beispielsweise um die Stärke gegenüber einer früheren Version zu Testen. Dabei gibt es zwei Varianten:

Variante mit TestClient

Der TestClient muss vom Terminal mit den entsprechenden Argumenten aufgerufen werden. Diese werden unter den Beispielen näher erläutert.

Unter Linux:

```
java -jar -Dlogback.configurationFile=logback-tests.xml test-client.jar\
     --tests 4 \
    --name1 "displayName1" \
     --player1 "./player1.jar" \
    --name2 "displayName2" \
     --player2 "./player2.jar" \
     --start-server\
     --port 13051
```

Unter Windows (unterscheidet sich nur durch die Art, den langen Befehl auf mehrere Zeilen zu verteilen):

```
java -jar -Dlogback.configurationFile=logback-tests.xml test-client.jar ^
    --tests 4 ^
    --name1 "displayName1" ^
    --player1 "./player1.jar" ^
    --name2 "displayName2" ^
    --player2 "./player2.jar" ^
    --start-server ^
    --port 13051
```

Der TestClient kann sich auch mit einem bereits laufenden Server verbinden, wie bei der Variante ohne TestClient, bei Angabe des Arguments --start-server startet er jedoch einfach selbst einen. Wichtig ist, dass nicht versucht wird, zwei Server auf demselben Port zu starten.

Argumente des TestClients

Attribut	Standardwert (Typ)	Beschreibung
--tests	100 (int)	Anzahl der Tests, die gespielt werden sollen
--player1	"./defaultplayer.jar" (Dateipfad)	Erster Computerspieler
--player2	"./defaultplayer.jar" (Dateipfad)	Zweiter Computerspieler
Attribut	**Standardwert (Typ)**	**Beschreibung**
--name1	"player1" (String)	Name des ersten Spielers
--name2	"player2" (String)	Name des zweiten Spielers
--no-timeout	false (bool)	Deaktiviere ausscheiden durch Timeouts. Kann durch --no-timeout1 bzw. --no-timeout2 für beide Spieler unabhängig gesetzt werden.
--start-server	false (bool)	Starte einen Server auf dem angegebenen Port vor dem Starten der Clients.
--server	'server.jar aus dem Classpath' (Dateipfad)	Gib einen bestimmten server an, der für die tests gestartet werden soll.
--port	13051 (int)	Der Port, auf dem der Server läuft.
--host	localhost (IP)	Die Adresse, auf dem der Server läuft.
--loglevel	INFO - ensprechend der logback-tests.xml (Level)	Setzt das Loglevel, um ausführliche oder besonders kompakte Ausgaben zu erhalten.

> Boolesche Parameter werden als true gewertet, sobald sie angegeben werden. Ein Wert hinter dem Parameter hat keine Wirkung.

Bei Argumenten, die nicht angegeben wurden, werden die Standardwerte aus der Tabelle verwendet. Die Ausgabe der Daten erfolgt nach jedem Spiel anhand von gerundeten Werten. Der TestClient beendet sich selbst, nachdem alle Spiele gespielt wurden.

Die Ergebnisse der Spiele werden für den jeweiligen Spielernamen vom Server zusammengezählt, auch über mehrere Starts des TestClients. Die Ergebnisse werden erst zurückgesetzt, wenn der Server neu gestartet wird. Achte also nach einer Veränderung der Spieler darauf, den Server neuzustarten oder andere Spielernamen zu verwenden.

Variante ohne TestClient

Starte den Server aus dem Terminal:

```
java -Dfile.encoding=UTF-8 -Dlogback.configurationFile=logback.xml -jar softwarechallenge-server.jar --port 13051
```

Starte ein Spiel mit Reservierungscode (siehe Spielverlauf in der XML-Dokumentation). Aktiviere mit dem erstellten Administratorclient den Testmodus:

```
<testMode testModus="true"/>
```

Darauf antwortet der Server:

```
<testing testModus="true"/>
```

Mit false als entsprechenden Parameter kann dieser wieder deaktiviert werden. Nun können jederzeit die Testdaten der Spieler anhand ihres Anzeigenamens erfragt werden:

```
<scoreForPlayer displayName="player1" />
```

Der Server antwortet mit:

```
<playerScore>
  <score displayName="player1" numberOfTests="4">
    <values>
      <fragment name="Gewinner">
        <aggregation>SUM</aggregation>
        <relevantForRanking>true</relevantForRanking>
      </fragment>
      <value>4</value>
    </values>
    <values>
      <fragment name="Ø Feldnummer">
        <aggregation>AVERAGE</aggregation>
        <relevantForRanking>true</relevantForRanking>
      </fragment>
      <value>5.0000013</value>
    </values>
    <values>
      <fragment name="Ø Karotten">
        <aggregation>AVERAGE</aggregation>
        <relevantForRanking>true</relevantForRanking>
      </fragment>
      <value>40.500011</value>
    </values>
  </score>
</playerScore>
```

Bei dieser Variante muss sich selbst um das Starten der Clients gekümmert werden.

Der Computerspieler (Client)

Der Computerspieler ist ein Programm, dass sich mit dem Spielleiter (siehe „Der Spielleiter (Server)“) verbindet und die gestellte Aufgabe selbständig lösen kann. Die Aufgabe der Studierenden ist es, sich eine Strategie zu überlegen und zu implementieren, mit der sie gegen die Clients der anderen Studierenden gewinnen können.

Der Computerspieler kann in einer beliebigen Programmiersprache geschrieben sein, jedoch gibt es Muster-Computerspieler nur in Java und Ruby.

Die Muster-Computerspieler können im Downloadbereich der Software Challenge Website (**https://software-challenge.de/**) heruntergeladen werden.

Hinweis: Das Spielleiter-Programm (siehe „Der Spielleiter (Server)“) benötigt Java. Deshalb muss auf den ausführenden Rechnern auch das Java SDK installiert sein.

Der SimpleClient

Der SimpleClient ist ein Computerspieler, den das Institut für Informatik ins Rennen schickt. Er stellt zwar eine korrekte Lösung der gestellten Aufgabe dar, ist aber nicht besonders intelligent. Neben dem eigentlichen Programm ist auch der Quellcode des SimpleClients verfügbar. Auf diese Weise können sich die Schüler anschauen und lernen, wie man die gestellte Aufgabe lösen kann.
Außerdem darf der Code um die eigene Strategie erweitert werden. Auf diese Weise müssen die Schüler nicht den ganzen Computerspieler selbst entwickeln, sondern können sich auf den Entwurf und die Implementierung ihrer eigenen Strategie konzentrieren.

Der NotSoSimpleClient

Wenn die aktuelle Saison der Software-Challenge etwas weiter fortgeschritten ist, stellt das Institut einen stärkeren Computerspieler zur Verfügung: den NotSoSimpleClient. Das ist ein Spieler, der eine effizientere Strategie zur Lösung der Aufgabe als der SimpleClient verfolgt und dadurch nicht mehr so leicht zu schlagen ist. Dieser Spieler wird ohne den Quellcode veröffentlicht, so dass die Schüler den NotSoSimpleClient zwar als Gegenspieler für Testspiele nehmen, jedoch nicht den Quellcode für den eigenen Spieler weiterverwenden können.

Den NotSoSimpleClient findet ihr hier:
https://software-challenge.de/dokumentation-und-material

Der AdminClient

Der AdminClient dient dazu, um Spiele anzulegen und wenn nötig bestimmte Einstellungen zu ändern:

Verbinden als AdminClient:

```
<protocol><authenticate password="examplepassword"/>
```

Sobald der Client sich authentifiziert hat ist folgendes möglich:

Ein AdminClient erhält eine Nachricht, sobald sich ein Client über eine JoinRoomRequest verbindet:

```
<joinedGameRoom roomId=ROOMID existing="false" />
```

existing gibt hier an, ob der GameRoom bereits existierte bevor der Spieler beigetreten ist. Die ROOMID ist ein String, der vom Server eindeutig generiert wird und ist wichtig für die genaue Zuordnung zu dem entsprechenden Raum.

Ein AdminClient kann sich als Observer mit einem GameRoom verbinden, dadurch erhält er alle Nachrichten, die das Spiel betreffen:

```
<observe roomId=ROOMID />
```

Ein AdminClient kann ein Spiel pausieren oder fortsetzen:

```
<pause roomId=ROOMID pause="true/false" />
```

Ein AdminClient kann einen Zug in einem pausierten Spiel anfordern:

```
<step roomId=ROOMID />
```

Ein AdminClient kann ein Spiel abbrechen:

```
<cancel roomId=ROOMID />
```

Ein AdminClient kann ein Spiel vorbereiten:

```
<prepare gameType="pluginUUID">
  <slot displayName="p1" canTimeout="true" shouldBePaused="false"/>  <slot
displayName="p2" canTimeout="true" shouldBePaused="false"/>
</prepare>
```

Ein AdminClient kann den Testmodus aktivieren/deaktivieren:

```
<testMode testMode="true/false" />
```

Der Server antwortet mit:

```
<testing testMode="true/false" />
```

Ein AdminClient kann die Punkte und Gewinnkriterien, falls der Testmodus aktiviert wurde, anhand des Anzeigenamens der Spieler abfragen:

```
<scoreForPlayer displayName="p1"/>
```

Der Server antwortet darauf mit:

```
<playerScore>
  <score displayName="p1" numberOfTests="1">
    <values>
      <fragment name="Gewinner">
        <aggregation>SUM</aggregation>
        <relevantForRanking>true</relevantForRanking>
      </fragment>
      <value>2</value>
    </values>
    <values>
      <fragment name="∅ Feldnummer">
        <aggregation>AVERAGE</aggregation>
        <relevantForRanking>true</relevantForRanking>
      </fragment>
      <value>10.000000</value>
    </values>
    <values>
      <fragment name="∅ Karotten">
        <aggregation>AVERAGE</aggregation>
        <relevantForRanking>true</relevantForRanking>
      </fragment>
      <value>13.000000</value>
    </values>
  </score>
</playerScore>
```

Servereinstellungen

Die Servereinstellungen liegen in der server.properties Datei, die zusammen mit dem Server heruntergeladen wird. In ihr können folgende Werte konfiguriert werden:

Das lokale Administratorpasswort kann geändert werden. Dazu muss das password Attribut neu gesetzt werden (standardmäßig auf examplepassword). Es kann eingestellt werden, ob ein durch einen JoinRequest gestartetes Spiel anfangs pausiert sein soll oder nicht (standardmäßig pausiert), indem der entsprechende Wert auf true oder false gesetzt wird.

Die richtige Programmiersprache

Am einfachsten ist es natürlich den Java/Ruby Simple-Client als Basis zu nutzen, allerdings könnt ihr auch eigene Clients in anderen Sprachen schreiben. Das ist mit mehr Arbeit verbunden, aber wenn ihr eine Sprache kennt, die ihr gerne nutzt, oder die andere Vorteile besitzt, dann kann es sich lohnen!

Wenn ihr also genug Erfahrung habt und euch entscheidet den schwereren Weg zu gehen, dann solltet ihr euch die XML-Dokumentation genau anschauen, da ihr die ganze Kommunikation, sowie das Parsen der XML-Nachrichten, implementieren müsst. Außerdem solltet ihr euch als Beispiel den (inoffiziellen) Swift Client ansehen. Das kann auch helfen, wenn man Swift nicht kann, da die meisten prozeduralen Programmiersprachen viele Ähnlichkeiten haben. Somit sollte es nicht allzu schwer sein den Swift code in eure Sprache zu übersetzen. Auf ähnliche Weise können natürlich auch **der Java Client source code** und **der C# Client source code** helfen.

Installation von Java

Die meisten Programme, die vom Institut für Informatik zur Verfügung gestellt werden, sind in der Programmiersprache Java geschrieben. Diese Anleitung soll die Beschaffung und Installation von Java erleichtern.

Grundsätzliches

Java gibt es in zwei verschiedenen Paketen: Das *Java Runtime Environment (JRE)* und das *Java Development Kit (JDK)*. Möchte man lediglich Java-Programme starten, also nicht selber entwickeln, dann reicht das JRE vollkommen aus. Möchte man auch eigene Programme schreiben, muss das JDK auf jeden Fall installiert sein. Da im JDK auch das JRE integriert ist, kann man aber immer ohne Bedenken gleich zum JDK greifen.

Installation

Das JDK gibt es auf den Seiten von Oracle **http://www.oracle.com/technetwork/java/javase/downloads/index-jsp-138363.html#javasejdk** Dort das aktuelle "JDK" herunterladen. Es gibt auch Installationsanleitungen auf der Seite.

Installation über Paketquellen (Linux)

Meistens ist das Java JDK in den Paketquellen der Linux-Distributionen enthalten, so dass man es einfach über den Paketmanager installieren kann. Sofern möglich, wird diese Art der Installation empfohlen, da es oft noch Paketabhängigkeiten gibt, die dann automatisch mitinstalliert werden.

Weiterführende Informationen

- **Die Java-Seiten von Oracle**
- **Installation von Java auf Ubuntu Linux** (Für andere Distributionen gibt es meist auch Wikis oder Foren mit den entsprechenden Anleitungen)

Einrichtung der (Java)Entwicklungsumgebung

Die Aufgabe einer Entwicklungsumgebung (IDE) ist es, den Programmierer bei seiner Arbeit zu unterstützen. Dazu bietet sie neben dem Editor auch viele Tools, die das Entwickeln eigener Programme stark erleichtern. Zwei große Vertreter an Entwicklungsumgebungen sind Eclipse und IntelliJ.

Hinweis: Bevor man sich um die Einrichtung der Entwicklungsumgebung kümmert, muss unbedingt Java installiert sein.

SimpleClient beschaffen

Der SimpleClient ist schon ein fertiger Computerspieler. Denn Quellcode kann man verwenden, um seinen eigenen Spieler zu programmieren. Den SimpleClient bekommt man im Downloadbereich der Software-Challenge (**https://software-challenge.de**). Man braucht die Version "als Quellcode".

Einrichtung von Eclipse

Beschaffung und Installation der Software

Am einfachsten ist die Installation von Eclipse mittels des Eclipse Installer. Dies ist auf folgender Seite erklärt: **https://www.eclipse.org/downloads/packages/installer**

SimpleClient in Eclipse einbinden

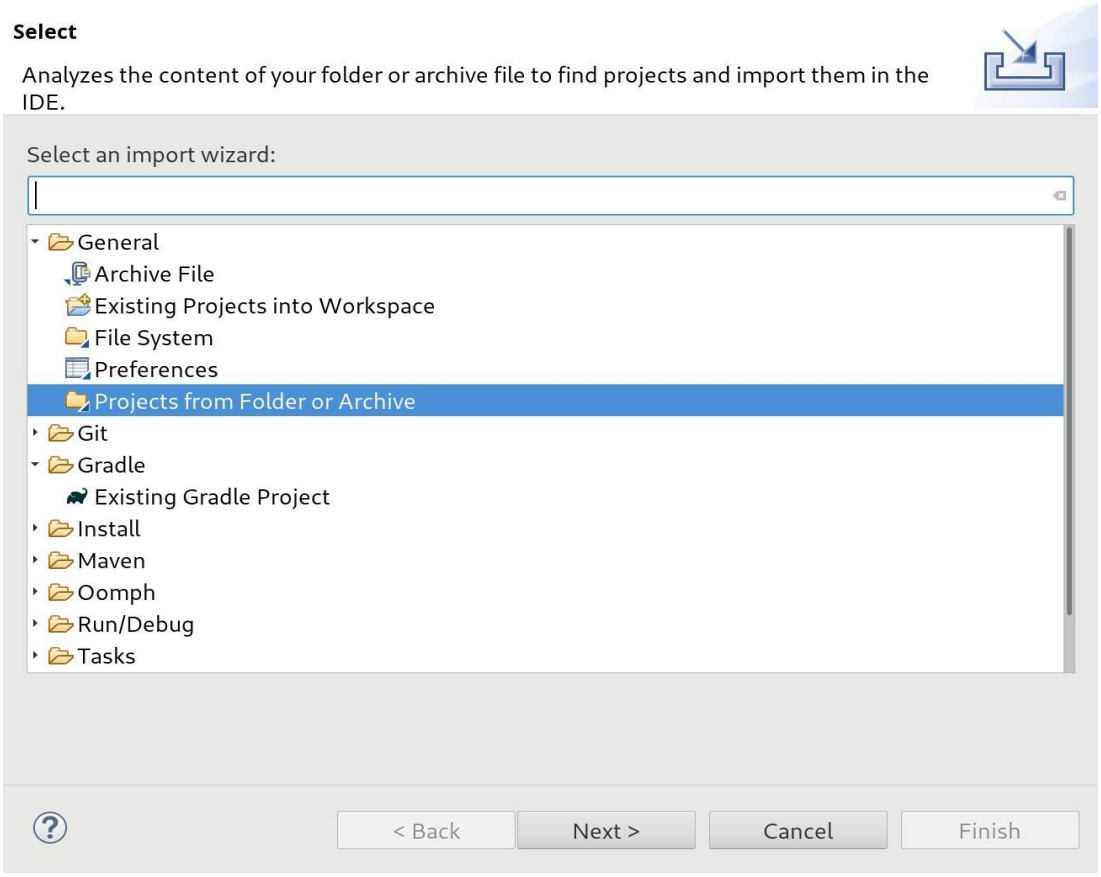


Figure 4. SimpleClient in Eclipse importieren

1. Im Menü auf "File" → "Import…" gehen
2. Im Dialogfenster unter "General" "Projects from Folder or Archive" wählen, dann auf den "Next" Button klicken
3. Oben rechts auf "Archive…" klicken und die heruntergeladene ZIP-Datei mit dem SimpleClient auswählen. Dann auf "Finish" klicken.

Nun muss noch das SDK und Spiel-Plugin eingebunden werden, damit Funktionen wie Autovervollständigung und Anzeige der Dokumentation richtig arbeiten:

1. Im Package Explorer einen Rechtsklick auf den Eintrag sdk.jar unter "Referenced Libraries" machen und Properties wählen
2. Links "Java Source Attachment" auswählen
3. Rechts "Workspace location" aktivieren und den Pfad zu "sdk-sources.jar" (im Ordner "lib" des SimpleClient Quellcode Paketes) einstellen
4. Den Dialog mit "Apply and Close" schließen
5. Im Package Explorer einen Rechtsklick auf den Eintrag für das Spiel-Plugin unter "Referenced Libraries" machen (Spielname und Jahreszahl, also z.B. "piranhas_2019.jar") und Properties wählen
6. Links "Java Source Attachment" auswählen
7. Rechts "Workspace location" aktivieren und den Pfad zur Source-Jar (im Ordner "lib" des SimpleClient Quellcode Paketes) einstellen (heißt genau wie das Spiel-Plugin mit einem "sources" angehängt, also z.B. "piranhas_2019-sources.jar")
8. Den Dialog mit "Apply and Close" schließen

SimpleClient aus Eclipse starten

Den SimpleClient kann man starten, indem man im Project-Explorer einen Rechtsklick auf die Datei Starter.java macht und dann "Run As" → "Java Application" auswählt.

Hinweis: Damit der SimpleClient erfolgreich startet, muss der Spielleiter laufen und auf eine Verbindung warten.

Weiterführende Links

- **http://www.eclipse.org** Homepage der Eclipse-IDE
- **http://www.netbeans.org** Homepage des NetBeans-Projektes

Bedienung von Eclipse

Wenn man bisher noch nicht mit einer Entwicklungsumgebung gearbeitet hat, mag der Anblick erschreckend unübersichtlich sein. Sobald man sich jedoch etwas intensiver damit beschäftigt hat, möchte man den Bedienkomfort eines solchen Entwicklertools gar nicht mehr missen. Dieser Artikel stellt die wichtigsten Komponenten der Entwicklungsumgebung Eclipse vor.

Die Oberfläche

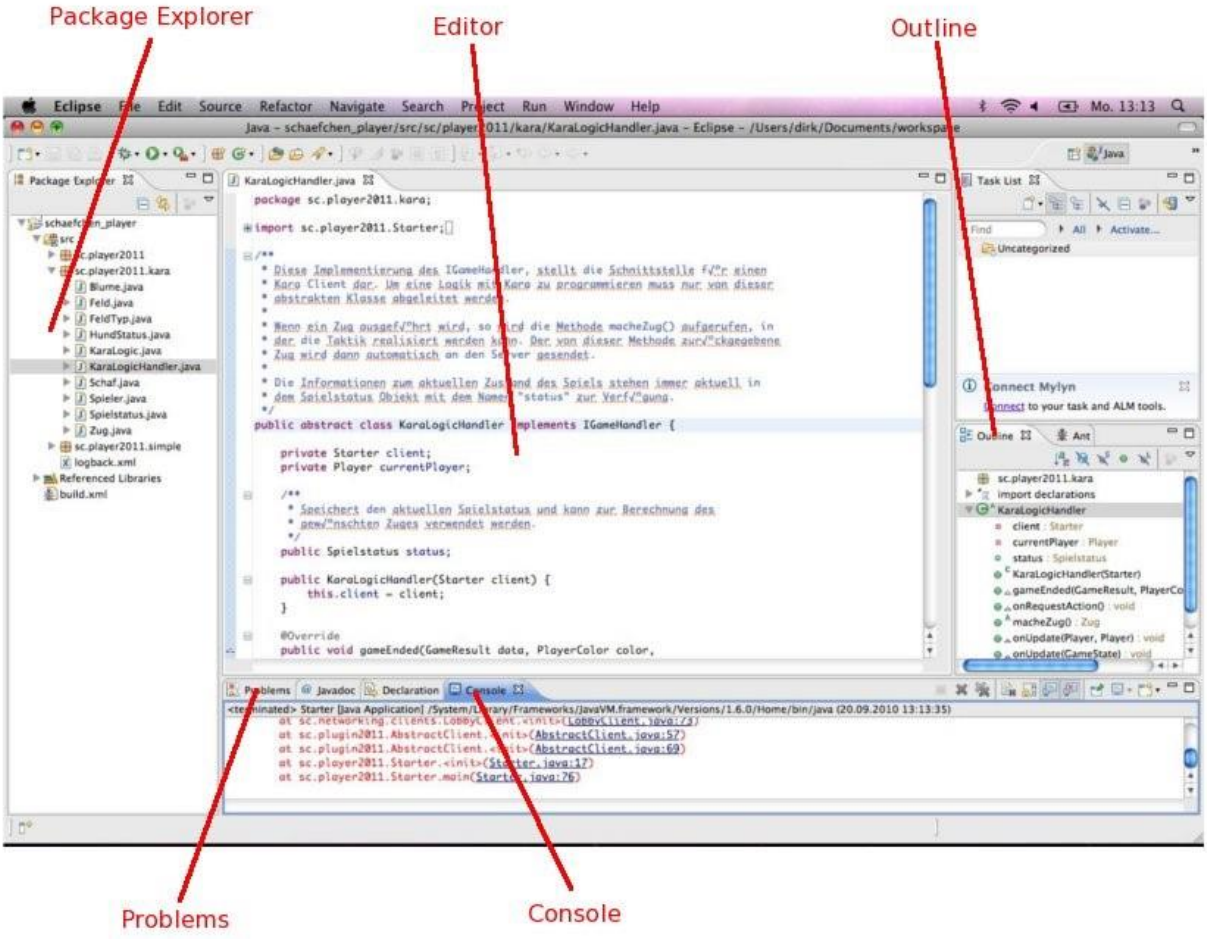


Figure 5. Überblick über die wichtigsten Fenster in Eclipse

Package Explorer

Der Package Explorer befindet sich am linken Rand. Er verwaltet alle importierten Projekte. Wenn man im Package Explorer einen Doppelklick auf eine Datei macht, wird diese im Editor angezeigt. Mit einem Rechtsklick auf eine Datei oder ein Verzeichnis bekommt man viele Optionen, mit denen sich das ausgewählte Objekt bearbeiten lässt.

Editor

Der Editor ist die große Fläche in der Mitte des Eclipse-Fensters. Am oberen Rand befindet sich die Tab-Leiste, die alle geöffneten Dateien beinhaltet.

Outline

Am rechten Bildschirmrand befindet sich die Outline. Sie zeigt alle Variablen und Methoden der Klasse an, die gerade im Editor geöffnet ist. Mit einem Doppelklick auf einen Eintrag springt der Cursor im Editor an die entsprechende Stelle im Code.

Problems

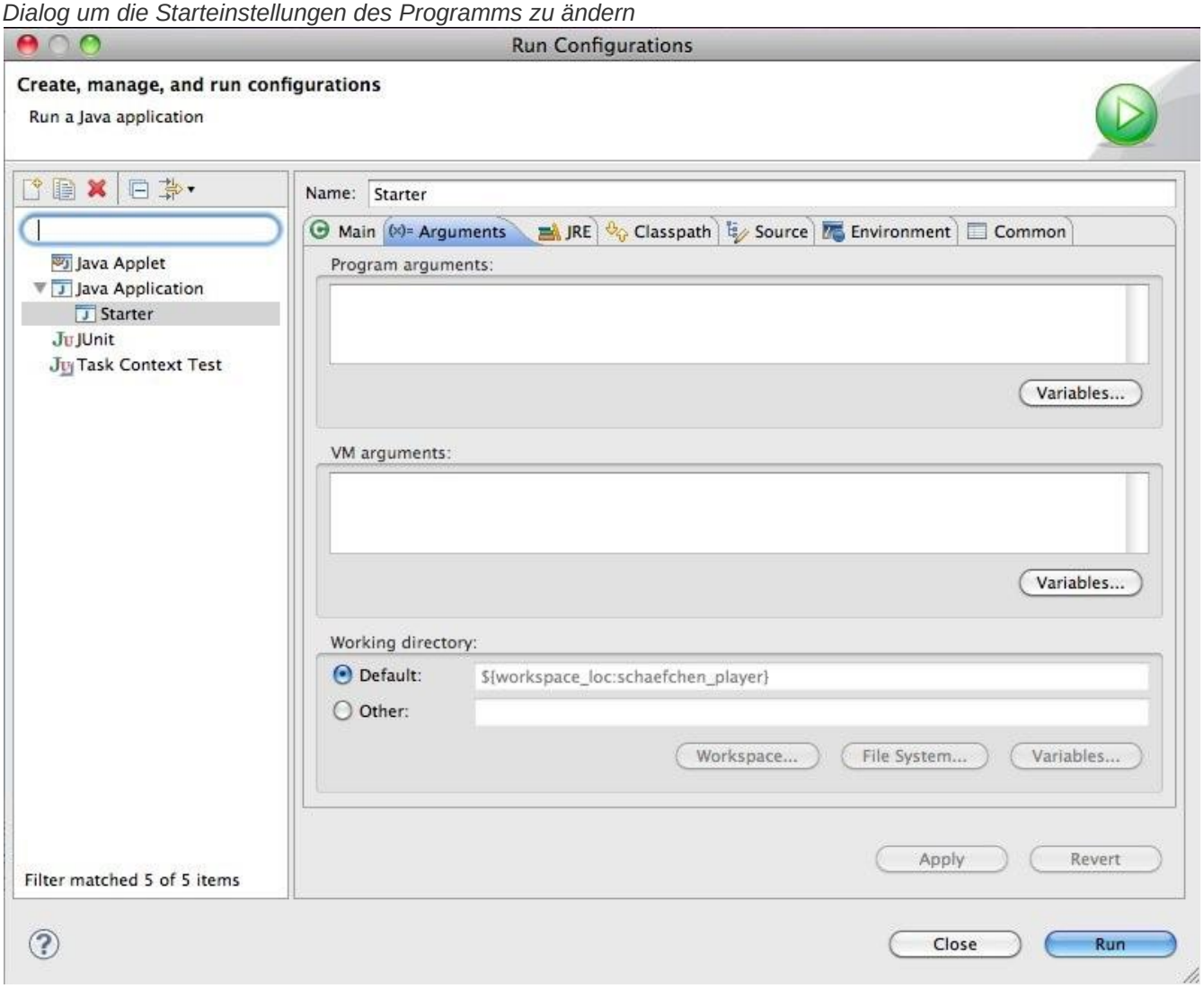
Der Tab Problems befindet sich im Fenster, das am unteren Bildschirmrand zu sehen ist. Hier werden sowohl Programmierfehler als auch Warnungen angezeigt. Mit einem Doppelklick auf einen Eintrag springt der Cursor im Editor an die entsprechende Codezeile.

Console

Die Console ist nicht sofort sichtbar, sondern erscheint erst, nachdem das erste Programm ausgeführt worden ist. In der Console werden alle Systemausgaben angezeigt. Falls ein Fehler (Exception) geworfen wird, kann man durch einen Klick darauf an die entsprechende Zeile im Programmcode gelangen.

Programme starten

Dialog um die Starteinstellungen des Programms zu ändern



Ein Programm lässt sich starten, indem man im Package Explorer einen Rechtsklick auf die Datei mit der Main-Methode macht und dann "Run As" → "Java Application" ausführt.

Im Menü kann man unter "Run" → "Run Configurations" im Tab "Arguments" noch Optionen angeben.

Tastaturkürzel

Eclipse kennt viele Tastenkombinationen, mit Hilfe derer einige Eclipse-Funktionen schneller aufgerufen werden können. Die wichtigsten Shortcuts kann man der folgenden Tabelle entnehmen:

Aktion	**Effekt**
Strg+Shift+F11	Führt die zuletzt ausgeführte Java-Datei erneut aus
Cursor auf Variablen-, Klassen- oder Methodennamen, dann Alt+Strg+R	Bennennt alle Vorkommen des Namens im ganzen Projekt um
Strg+F1	Wenn man diese Tasenkombination über einen Fehler oder eine Warnung eingibt, kriegt man von Eclipse Verbesserungs-, bzw. Reparaturvorschläge
Strg+I	Rückt den markierten Text sauber ein
Strg+F7	Kommentiert die markierten Zeilen ein, bzw. aus
Cursor auf Variablen-, Methoden- oder Klassennamen, dann F3	Der Cursor springt zur der Stelle, wo die Variable oder Klasse definiert wurde
Variablen- oder Klassenname teilweise eingegeben, dann Strg+Space	Eclipse liefert Vorschläge zur Vervollständigung
Eingabe von syso, dann Strg+Space	Erzeugt System.out.println
Eingabe von for, dann Strg+Space	Eclipse liefert eine Auswahl an beliebigen for -Schleifen
Eingabe von if, dann Strg+Space	Liefert eine Auswahl an if-Dialogen

Hinweis: Bei Mac OS X wird statt der Strg-Taste meistens die Apple-Taste benutzt.

Den SimpleClient erweitern

In der Version des Java SimpleClients von der Software Challenge Homepage ist bereits eine Strategie implementiert, die RandomLogic. Man kann jedoch auch noch beliebig viele eigene Strategien hinzufügen.

Erstellen einer neuen Strategie

Die einfachste Möglichkeit ist, die Klasse Logic des SimpleClient zu kopieren und umzubenennen (alle Vorkommen von Logic durch den neuen Klassennamen ersetzen). Der Vollständigkeit halber hier noch das Vorgehen bei einer komplett neuen Klasse:

- Erstellt eine neue Klasse (z.B. MyLogic), die das Interface IGameHandler implementiert:

```
public class MyLogic implements IGameHandler {
    private Starter client;
    private GameState gameState;
    private Player currentPlayer;
```

- Erstellt einen Konstruktor, der eine Instanz des Starters erhält. Diese wird später noch gebraucht

```
public MyLogic(Starter client) {
    this.client = client;
}
```

- Implementiert die 5 Interface-Methoden

```
@Override
public void gameEnded(GameResult result, PlayerColor color, String errorMessage)  {
    // Hier muss nichts getan werden
}

@Override
public void onUpdate(Player player, Player otherPlayer) {
    // Der Spieler wurde aktualisiert
    this.player = player;
}

@Override
public void onUpdate(GameState gameState) {
    // Ein neuer Spielstatus, d.h. etwas ist geschehen. Deshalb
    // alles aktualisieren.
    this.gameState = gameState;
    this.player = gameState.getCurrentPlayer();
}

@Override
public void sendAction(Move move) {
    // Einen Zug an den Server senden
     starter.sendMove(move);
}

@Override
public void onRequestAction() {
    // Ich soll einen Zug machen
    Move move;
    // ... Hier muss die Logik rein, die einen Zug findet.
    sendAction(move);
}
```

Nun kann die Strategie in der Methode onRequestAction (oder in eigenen Klassen, die dort verwendet werden) implementiert werden.

Computerspieler abgabefertig machen

Damit das Wettkampfsystem mit dem Computerspieler arbeiten kann, muss er zu einem ausführbaren Programm gemacht und in ein ZIP-Archiv gepackt werden.

Je nach Programmiersprache, in der der Computerspieler entwickelt wurde, sind unterschiedliche Schritte notwendig.

Wie man den abgabefertigen Computerspieler dann im Wettkampfsystem einsendet, ist unter Wettkampfsystem, Computerspieler beschrieben.

Java

Diese Anleitung beschreibt, wie man für den Java-SimpleClient vorgehen muss. Hierzu gibt es zwei Möglichkeiten: Die Jar-Datei selbst erstellen und die Verwendung eines AntScripts.

Erste Möglichkeit - JAR erstellen

Eclipse

1. In Eclipse im Menü auf "File" → "Export". Dann unter "Java" → "Runnable JAR file" wählen
2. Im nächsten Fenster wird die "Run Configuration" ausgewählt (dazu muss der SimpleClient mindestens einmal mit Eclipse gestartet worden sein). Darunter wird mit "Browse" die Zieldatei (z.B. […]/my_player.jar) ausgewählt. Bei "Library handling" am besten die erste Option nehmen. So wird eine einzige JAR Datei erzeugt, in der alles nötige drin ist. Mit einem Klick auf Finish wird die JAR-Datei erzeugt. Eventuell erhält man einen Hinweis "This operation repacks referenced libraries", den man mit "OK" bestätigen muss.

Wenn alles geklappt hat, wurde der Computerspieler in ein auführbares Programm überführt. Damit der Wettkampfserver den Client verarbeiten kann, muss er noch in ein ZIP-Archiv gepackt werden (auch wenn ein JAR technisch gesehen bereits ein ZIP-Archiv ist).

NetBeans

Nach einem Rechtsklick auf das Projekt in der Projektansicht kann man "Properties" auswählen. Auf der linken Seite unter "Categories" die Kategorie "Run" auswählen und dann rechts unter "Main Class" die Klasse "Starter" eintragen.

Nach einem Rechtsklick auf das Projekt in der Projektansicht kann man "Clean and Build" auswählen. Danach gibt es den Ordner "Pfad/zur/Project/Loacation//dist". In diesem befindet sich eine JAR-Datei und eine Kopie des "lib"-Ordners. Beides zusammen muss jetzt mit einem beliebigen Archivierungsprogramm in eine ZIP-Datei zusammengepackt werden. Dieses Archiv kann dann hochgeladen werden.

Zweite Möglichkeit - Das ANT Script

Man kann auch das dem SimpleClient beiliegende Ant Buildscript benutzen. Dieses kompiliert den SimpleClient und erzeugt automatisch eine JAR Datei sowie ein ZIP-Archiv, das man direkt im Websystem hochladen kann.

Direkt ausführen

Wenn Ant installiert ist, kann man über die Kommandozeile in das Verzeichnis des SimpleClients wechseln und mit dem Aufruf "Ant" den Build ausführen. Am Ende sollte die Meldung "BUILD SUCCESSFUL" erscheinen. Im SimpleClient Ordner findet man dann im Unterordner "build" die JAR Datei im Ordner "jar" sowie die fertig gepackte ZIP-Datei im Ordner "zipped" die direkt im Wettkampfsystem hochgeladen werden kann.

Eclipse

Eclipse kann von Haus aus auch mit Ant-Scripten umgehen.

1. Im Menü "Run" → "External Tools" → "External Tools Configuration" wählen
2. Links in dem neuen Fenster mit einem Rechtsklick auf "Ant Build" → "New" erstellt man eine neue Konfiguration und wählt diese aus
3. Auf der rechten Seite muss man nun das Buildfile auswählen. Das geht entweder mit "Browse Workspace" oder "Browse File System". Das Buildfile heißt "build.xml" und liegt direkt in dem SimpleClient Ordner, den man auf der Software Challenge Homepage heruntergeladen hat. Anschließend mit "Apply" bestätigen und das Fenster schließen
4. Um den Buildprozess zu starten, muss im Menü "Run" → "External Tools" die gerade erstellte Konfiguration ausgewählt werdem. Der Build wird dann durchgeführt (dauert i.d.R. wenige Sekunden).

Am Ende erhält man die Meldung "BUILD SUCCESSFUL".

Im SimpleClient-Ordner findet man dann im Unterordner "build" die JAR Datei im Ordner "jar" sowie die fertig gepackte ZIP-Datei im Ordner "zipped", die direkt ins Wettkampfsystem hochgeladen werden kann.

Ruby

Da Ruby eine interpretierte Sprache ist, muss der Ruby-Quellcode direkt in ein ZIP-Archiv gepackt und auf das Wettkampfsystem hochgeladen werden. Auf dem Wettkampfsystem ist ein Ruby Interpreter sowie das aktuelle software_challenge_client Gem installiert. Alle weiteren Bibliotheken müssen im ZIP-Archiv vorhanden sein. Nach dem Hochladen des ZIP-Archiv muss die auszuführende Hauptdatei in Wettkampfsystem ausgewählt werden. Diese wird dann zum Start des Computerspielers gestartet. Damit dies richtig funktioniert, ist es entscheidend, dass die Hauptdatei mit einer sogenannten "Shebang"-Zeile beginnt:

```
#!/usr/bin/env ruby
```

Weiterhin ist es ratsam, den Magic-Comment zum Encoding direkt unter die Shebang-Zeile zu schreiben:

```
# encoding: UTF-8
```

Ein vollständiges Beispiel für einen abgabefertigen Ruby-Computerspieler gibt es im **example Verzeichnis des Client-Gems bei Github**. Packt man die beiden Dateien client.rb und main.rb in ein ZIP-Archiv, hat man einen abgabefertigen Computerspieler. Beim Hochladen wählt man main.rb als Hauptdatei.

C#

Stellt sicher, dass euer Projekt nur Libraries nutzt, die auch in Mono 5.4 enthalten sind. Kompiliert bzw. erstellt euer Projekt und fügt zu dem build eine start.sh hinzu, wie im nächsten Kapitel beschrieben, sofern ihr noch keine habt, da der Wettkampfserver eure exe sonst nicht ausführen kann. Sie sollte folgenden Inhalt haben, um eure exe mit mono zu starten:

```
#!/bin/sh
chmod +x csharp_exe_dateiname
mono csharp_exe_dateiname "$@"
```

Andere Programmiersprachen

Bei Computerspielern in jeder anderen Programmiersprache muss ein Shell-Script dem ZIP-Archiv beigelegt werden, was die genauen Instruktionen zum Start des Computerspielers enthält. Dieses muss nach dem Hochladen im Wettkampfsystem als Hauptdatei ausgewählt werden.

Beachten Sie hierbei, dass diesem Script vom Wettkampfsystem Parameter übergeben werden, die an Ihr Programm weitergegeben werden müssen. Diese Parameter sind mindestens Host und Port des Spielservers sowie die Reservierungsnummer des Spiels, dem der Computerspieler beitreten soll. Ein Aufruf sieht also in etwa wie folgt aus (falls start.sh als Hauptdatei eingestellt ist):

```
start.sh -h gameserver -p 13050 -r 590e5e6f-cf93-488e-a12d-5c194ecf95c2
```

Die Parameter folgen dabei den GNU Getopt Konventionen. Das heißt, die drei Parameter können in beliebiger Reihenfolge und als kurze oder lange Version übergeben werden. Folgende Variante muss also auch von Ihrem Programm korrekt verarbeitet werden können:

```
start.sh --reservation 590e5e6f-cf93-488e-a12d-5c194ecf95c2 --host gameserver --port
13050
```

Auch der Server mit grafischer Oberfläche ruft Ihr Programm mit diesen Parametern auf. Ihr Programm wird also nur von der grafischen Oberfläche richtig gestartet, wenn es die Parameter richtig verarbeitet.

Für die meisten Programmiersprachen gibt es Bibliotheken, die die Kommandozeilenparameter nach diesem Schema verarbeiten können, sie müssen diese Funktion also nicht unbedingt selbst implementieren.

Weiterhin ist es wichtig, den Interpreter in der ersten Zeile des Scripts anzugeben, da das Script nicht von einer Shell aufgerufen wird. Ein start.sh Script sieht also in etwa so aus:

```
#!/bin/sh①
chmod +x hauptprogramm_dateiname②
./hauptprogramm_dateiname "$@"③
```

① Script soll von /bin/sh interpretiert werden, es ist also ein einfaches Shell-Script.

② Die Binärdatei wird ausführbar gemacht (das ist nötig, da in einem ZIP-Archiv das AusührbarAttribut nicht gespeichert wird).

③ Die Binärdatei wird aufgerufen und alle Parameter, die das Script bekommen hat, werden weitergereicht ("$@").

Die start.sh muss in UTF-8 und mit UNIX(LF) Zeilenenden kodiert sein. Andere Kodierungen führen zu Fehlern bei der Ausführung auf dem Server. In Notepad++ kann die Kodierung einfach in dem Tab Kodierung angepasst werden, die Zeilenenden in Bearbeiten > Format Zeilenende.

Bei kompilierten Sprachen müssen die Computerspieler für 64bit Linux kompiliert werden, bei interpretierten Sprachen muss ein passender Interpreter auf dem Wettkampfsystem vorhanden sein. Weiterhin müssen Abhängigkeiten wie z.B. genutzte Bibliotheken vorhanden sein oder mitgeliefert werden.

Der saubere Programmierstil

Eventuell hat manche(r) schon erlebt, dass man von einer/m Bekannten ein Stück Programmcode bekommen hat, den man gar nicht versteht. Oft versteht man sogar nach einiger Zeit seine eigenen Codezeilen nicht mehr. Meistens liegt das gar nicht an den komplizierten Algorithmen, sondern an einen schlechten Programmierstil. Deshalb gibt es für alle Programmiersprachen sog. Style Guides, also Regeln für den Aufbau von Quelltexten.

Mit den *Java Code Conventions* stammt der bekannteste Style Guide für Java von seinen Entwicklern. Dieser soll hier ein wenig nähergebracht werden.

Allgemeiner Dokumentaufbau

- Keine Zeile sollte länger als 80 Zeichen sein. Gerade in Zeiten großer Breitbildschirme ist das wohl eine der schwierigsten Regeln überhaupt. Man muss aber davon ausgehen, dass nicht jeder, der den Code lesen will, auch einen ähnlich breiten Bildschirm hat. Außerdem ist meistens die Schriftgröße zum Drucken so eingestellt, dass höchstens 80 Zeichen in eine Zeile passen
- In einer Klasse sollten immer als erstes die globalen Variablen, dann die Konstruktoren und als letztes die Methoden auftauchen
- Klassennamen sollten im *CamelCase* geschrieben werden, also jedes Teilwort wird mit einem Großbuchstaben geschrieben (z.B. GameHandler)
- Für Variablen- und Methodennamen wird der *lower CamelCase* verwendet, bei dem nur das erste Teilwort mit einem kleinen Buchstaben beginnt (z.B. eigenerSpieler)
- Für Konstantenbezeichner werden ausschließlich große Buchstaben benutzt (z.B. int ANZAHL_SPIELER = 2;)
- Jede Zeile sollte nur eine Anweisung enthalten.

Die Schnittstelle zum Server

Der Spielleiter kommuniziert mit den Computerspielern über eine Netzwerkverbindung. Dadurch ist man aus technischer Sicht komplett flexibel, was die Wahl der Programmiersprache angeht. Die Computerspieler müssen lediglich das Kommunikationsprotokoll erfüllen, das du bei uns auf der Webseite (**https://cutt.ly/programmierwettbewerb**) separat herunterladen kannst.

Anfängern wird allerdings davon abgeraten, einen komplett eigenen Client zu schreiben. Es ist deutlich einfacher, auf dem bereitgestellten SimpleClient aufzubauen (Den SimpleClient um eine eigene Strategie erweitern). Damit muss man sich nicht um die Kommunikation kümmern, sondern kann sich direkt auf die Spiellogik konzentrieren. Außerdem wird vom Institut für Informatik die beste Unterstützung für Java geboten.

Einführung in XML

Die Kommunikation zwischen Spielleiter und Computerspieler wird mittels XML-Nachrichten realisiert. XML ist eine Auszeichnungssprache, d.h. eine Sprache, die nicht nur die Daten selbst, sondern auch Informationen über die Interpretation oder Bearbeitung liefert. Der Vorteil dieser Sprache liegt darin, dass sie sowohl vom Computer als auch vom Menschen gut gelesen werden kann. Dieser Artikel soll den Aufbau einer XML-Datei beschreiben.

Tags

Die Grundelemente von XML sind *Tags*. Ein Tag liefert Informationen über die Art der Daten, die verarbeitet werden sollen. In XML wird ein Tag gebildet, indem man den Tagnamen zwischen spitze Klammern setzt. Dabei kennt XML drei verschiedene Tag-Arten:

- Öffnendes Tag: <Tag>
- Schließendes Tag: </Tag>
- Leeres Tag: <Tag />

Der Schrägstrich bedeutet, dass das Tag geschlossen wird. Durch den Schrägstrich am Ende wird das soeben geöffnete Tag direkt wieder geschlossen.

Zwischen den öffnenden und schließenden Tag steht die Information, die mitgeteilt werden soll.

Hinweis: XML unterscheidet strikt zwischen Groß- und Kleinschreibung.

Bildungsregeln

Die Tags dürfen nicht beliebig in Dokumenten verwendet werden. Es gelten hier die folgenden Regeln:

- Zu jedem öffnenden Tag muss ein schließendes Tag existieren.
- Man kann Tags ineinander schachteln. Die einzelnen Tags dürfen sich jedoch nicht überkreuzen.
- Es darf nur ein Root-Tag geben, d.h. es gibt auf oberster Ebene genau ein Tag, in dem alle anderen enthalten sind.

Beispiel für korrekte XML-Syntax

```
<addiere>
    <komplexe_zahl>
        <realteil>3.5</realteil>
        <imaginaerteil>4.2</imaginaerteil>
    </komplexe_zahl>
    <komplexe_zahl>
        <realteil>1</realteil>
        <imaginaerteil>6.9</imaginaerteil>
    </komplexe_zahl>
</addiere>
```

Beispiele für fehlerhafte XML-Syntax

- Fehlerhaft, da es mehrere Elemente auf oberster Ebene gibt:

```
<ueberschrift>
    Beispieldokument
</ueberschrift>
<text>
    Dies ist ein<unterstrichen>Beispieltext</unterstrichen>
    <absatz />
    Noch mehr Text
</text>
```

- Fehlerhaft, da Tags sich kreuzen:

```
<dokument>
    <ueberschrift>
        Beispieldokument
    </ueberschrift>
    <text>
        <kursiv>Dies <unterstrichen>ist </kursiv>ein Beispieltext</unterstrichen>
        <absatz />
        Noch mehr Text
    </text>
</dokument>
```

Attribute

Man kann im Tag auch Attribute einfügen, in denen Informationen übertragen werden:

```
<Tag attribut="wert">
```

Auf diese Weise lässt sich das Beispiel mit den komplexen Zahlen etwas übersichtlicher gestalten:

```
<addiere>
    <komplexe_zahl realteil="3.5" imaginaerteil="4.2" />
    <komplexe_zahl realteil="1" imaginaerteil="6.9" />
</addiere>
```

Der Header

Ganz wichtig ist noch die erste Zeile in einem XML-Dokument. Aus ihr kann das Computerprogramm erfahren, wie er mit den Daten umzugehen hat (z.B. welcher Zeichensatz benutzt wird). Dieser Header sieht einem Tag sehr ähnlich:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
```

Man nennt so ein Tag, auch *Verarbeitungsinformation*.

Theoretisch darf es in einem XML-Dokument mehrere Verarbeitungsinformationen geben, für die Software-Challenge muss man aber nur den Header kennen.

Kommentare

Man kann in sein XML-Dokument auch Kommentare einfügen, die beim Einlesen dann ignoriert werden:

```
<!-- Ich bin ein XML-Kommentar -->
```

Es darf beliebig viele solcher Kommentare geben und sie dürfen nur zwischen Tags stehen.

Technische Daten für die Ausführung der Computerspieler

Für alle im Wettkampfsystem ausgetragenen Spiele laufen die Computerspieler auf den Servern des Wettkampfsystems.

Betriebssystem:	64 Bit Linux
Prozessor:	Ein Kern von einem **Intel Xeon E5-2620 v4, 2,1 GHz**
Arbeitsspeicher:	1,5 GB

Log-Ausgabe

Die Computerspieler laufen im Wettkampfsystem ohne eine grafische Oberfläche, sie können also keine Fenster oder ähnliches anzeigen. Der Versuch eines Computerspielers, so etwas trotzdem zu tun, wird wahrscheinlich zum Absturz des Computerspielers führen.

Die Computerspieler können jedoch Text auf die beiden Standard-Ausgabedatenströme "stdout" und "stderr" schreiben. Diese Ausgaben finden sich dann in den Log-Dateien wieder, die nach Beenden eines Spiels über das Wettkampfsystem verfügbar sind.

Lesen von Daten

Zugriff auf das Internet ist nicht möglich. Schreiben auf die Festplatte ist möglich, es kann jedoch nicht auf Daten, die in früheren Spielen geschrieben wurden, zugegriffen werden.

Weitere Ausführungsumgebung

Der Computerspieler wird in einem sogenannten "Docker Container" ausgeführt, welcher die verfügbaren Bibliotheken und Programme bestimmt. Folgende Container-Images können genutzt werden:

Bezeichnung	Image-Name	Beschreibung
Java 8	openjdk:8u151-jre	Open Source Java Platform, Standard Edition, Version 1.8.0. Für alle Computerspieler auf Basis des Java SimpleClients.
Ruby 2	ruby:2.4.2 mit installiertem Software-Challenge-Client-Gem	Ruby Interpreter, Version 2.4.2. Für alle Computerspieler auf Basis des Ruby SimpleClients.
Python 3	python:3.6.3	Python Interpreter, Version 3.6.3. Für selbst entwickelte Computerspieler in Python.

Mono 5.4 (C# .NET)	mono:5.4.1.6	Mono Laufzeitumgebung, Version 5.4.1.6. Für selbst entwickelte Computerspieler basierend auf dem Microsoft .NET Framework.

Wenn dein Computerspieler eine speziellere Umgebung benötigt (zum Beispiel ein hier nicht angebotener Interpreter), nehmt bitte frühzeitig Kontakt mit uns auf (**startupbridge@fh-wedel.de**).

Die Weboberfläche

Das Wettkampfsystem ist die Plattform, auf der die Computerspieler der einzelnen Teams gegeneinander antreten. Die Teams können sich dabei nicht nur im Wettkampf, sondern auch in Freundschaftsspielen, mit ihren Gegnern messen. Außerdem liefert er alle Informationen rund um den Wettkampf, wie z.B. die Ergebnisse der einzelnen Spieltage oder die Anzahl der Mitglieder in den einzelnen Teams.

Die Weboberfläche ist unter der URL **https://contest.software-challenge.de/saison/29** erreichbar. Im Wettkampfsystem findet die gesamte Wettkampfverwaltung statt. Hier können die Clients abgegeben, getestet und aktiviert werden.

Freundschaftsspiele

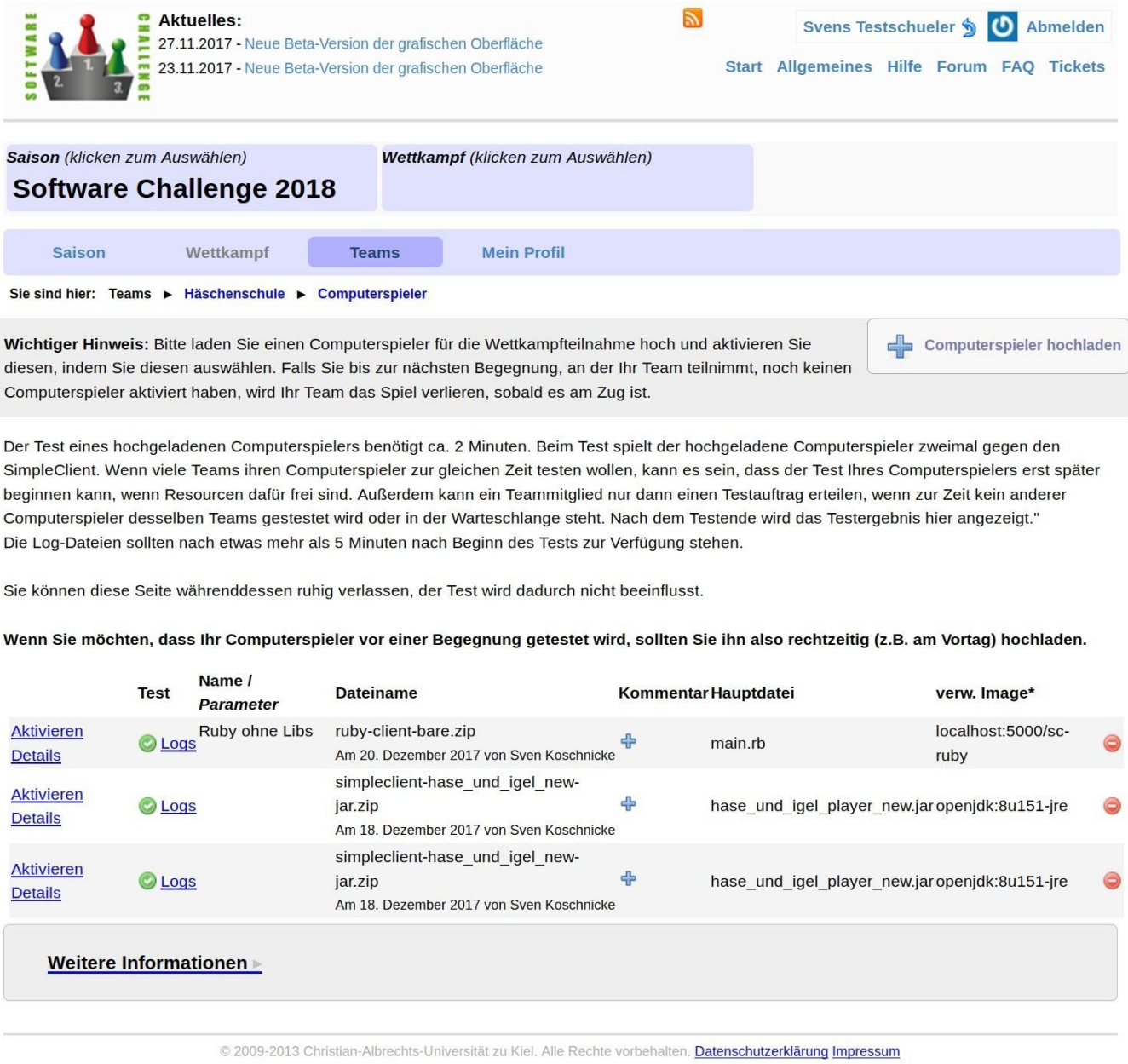
Um sich schon vor dem Wettkampf einen Eindruck von der Stärke des eigenen Teams zu machen, kann man mit seinen Gegnern Freundschaftsspiele absolvieren.

Um ein Freundschaftsspiel zu spielen, gibt es zwei Möglichkeiten: Man fordert einen (oder alle) Gegner heraus oder man nimmt eine Herausforderung an.

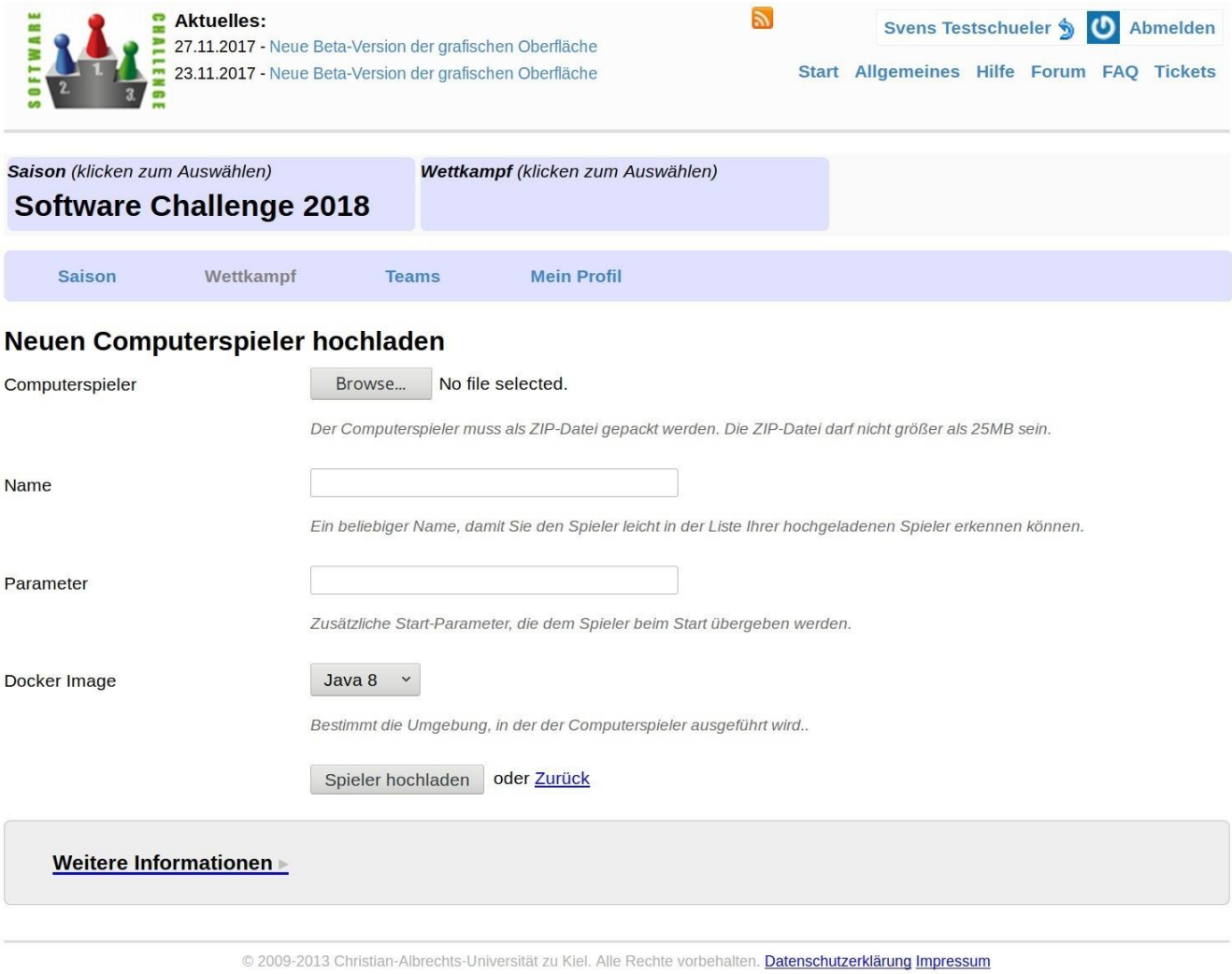
Durch einen Klick auf das Ergebnis eines gespielten Freundschaftsspiels kann man es sich im Detail anschauen.

Computerspieler abgeben

In dieser Rubrik kann man die Computerspieler hochladen und verwalten. Bevor man einen Computerspieler hochlädt, muss der Computerspieler abgabefertig gemacht werden.



Oben am rechten Rand befindet sich die Schaltfläche, um neue Computerspieler hinzuzufügen. Diese führt zu einem Formular, mit dem man das ZIP-Archiv des Computerspielers hochladen kann.



Man kann dem Spieler einen **Namen** geben, damit man ihn in der Liste der hochgeladenen Spieler später besser erkennt. Man kann auch noch zusätzliche **Parameter** festlegen, die dem Computerspieler beim Start übergeben werden. Dies ist nützlich, wenn der Computerspieler verschiedene Spielstrategien unterstützt und man diese per Parameter auswählen kann. Dann muss man den Computerspieler nur einmal hochladen (die Parameter kann man auch später verändern). Die Angabe eines Namens und von Parametern ist optional.

Als letztes muss noch die Umgebung (das **Docker Image**) gewählt werden, in der der Computerspieler auf dem Wettkampfsystem ausgeführt werden soll. Verwendet der Computerspieler eine der beiden offiziell unterstützten Programmiersprachen Java und Ruby, kann hier einfach das entsprechende ausgewählt werden. Ansonsten hängt es von der gewählten Programmiersprache ab, ob eine passende Umgebung angeboten wird. Sollte sich nichts finden, installieren wir gern etwas passendes nach.

Wurde ein Spieler erfolgreich hochgeladen, befindet er sich in der Liste aller Spieler des Teams. Hier kann man mit dem Link "Testen" seine Turnierfähigkeit prüfen. Der Spieler spielt dann zweimal gegen den SimpleClient. Ein Haken in einem grünen Kreis symbolisiert einen erfolgreichen Test. Das Logbuch über den Testlauf kann mit dem Link "Logs" aufgerufen werden. Unter Umständen muss noch die richtige Startdatei eingestellt werden. Dafür kann man rechts in der Spalte "Hauptdatei" auf den entsprechenden Link klicken und im folgenden Dateimenü die richtige Startdatei auswählen.

Mit dem '+' kann man einen Kommentar an den Client heften, so dass man ihn besser von den anderen unterscheiden kann.

Mit dem Link "Aktivieren" markiert man den Spieler als denjenigen, der das nächste Spiel auf dem Wettkampfsystem spielen soll. Dies kann ein Freundschaftsspiel oder ein Spiel des Wettkampfes sein.

Hinweis: Es nimmt der jeweils aktive Computerspieler am Spieltag teil. Die Frist für das Aktivieren eines Clients, der an einem Spieltag teilnehmen soll, endet am Spieltag um 0 Uhr. Bei späterer Aktivierung kann nicht garantiert werden, dass der neue statt des bisherigen Clients am Spieltag teilnimmt. Ist an einem Spieltag kein Computerspieler aktiviert, nimmt das Team nicht an der Begegnung dieses Spieltages teil und die betreffenden Spiele zählen als verloren.