Reversi: Server und manueller Spieler

Plenum Programmierpraktikum

2006-12-14

Kodierung der Textdateien

- Einstellung für alle Projekte: Window ▶ Preferences ▶ Workspace ▶ Text file encoding. Auf UTF-8 stellen.

- Kann pro Projekt bzw. Datei überschrieben werden.

PlayerInterface: Änderungen

- Eindeutige Namensgebung: Gruppennummer zur Disambiguisierung:

```
public static final Pattern LEGAL_NAME = Pattern.compile(
                        "^[0-9][0-9] [A-ZÄÖÜa-zäöü0-9 !?.-]+$");
```

- Timeout für *alle* Aufrufe verwenden. Verhindert DoS-Attacken.

- `trackMove()` wird zu:
  - trackAndComputeMove (neu): Normalbetrieb.
  - trackMove (alt): Gegner zieht mehrmals, Spielende.
  - computeMove (alt): Spieler fängt an.

  Damit können Spieler diese Operation nicht zum Vorausrechnen ausnutzen.

Neu: ProtocolConstants

- Standardisierter Zeilentrenner, nicht `println()` verwenden.
  `ProtocolConstants.NEWLINE`.
  - `BufferedReader.readLine()` ist OK.
  - `flush()` nicht vergessen!
- Standardisiertes Text-Encoding.
  `ProtocolConstants.CHARSET`.

Reversi

Referenzregeln

- http://de.wikipedia.org/wiki/Reversi

  - „Sind bei Spielende noch nicht alle Felder belegt, werden die freien Felder dem Gewinner gutgeschrieben.“
  - „Wenn ein Spieler keine Steine mehr hat, aber noch ziehen kann, erhält er welche von seinem Gegenspieler, und zwar so lange, bis dieser selbst wieder ziehen kann.“ ☺

- KReversi (im CIP-Pool).

- Regel-Überraschungen und -Fragen können auch gerne ins Forum gestellt werden.

Turniere

- Erstes Turnier: Am ersten Montag nach der vorlesungsfreien Zeit (8. Jan 2007). Teilnahme freiwillig.
- Tutoren starten Eure Implementierungen.
- Ich gebe im Plenum das Ranking bekannt.
- Danach gibt es noch zwei weitere Turniere jeweils anstatt eines Plenums und im CIP-Pool(!).

Turnierregeln

- Turnierregeln werden enorm kompliziert, sobald es „unentschieden“ als Spielausgang gibt.

- Beispiel: http://en.wikipedia.org/wiki/Swiss_system_tournament

Reversi-Turnierregeln

- Alle Spiele einer Runde finden parallel statt.
- Jeder Spieler A spielt zweimal gegen jeder anderen Spieler B, wobei einmal A das Spiel anfängt, einmal B.
- Ranking: Der $Score$ eines Spielers setzt sich zusammen aus
  - $Points$ (pro Spiel: 2 Punkte für's Gewinnen, 1 Punkt für ein Unentschieden, 0 Punkte für's Verlieren) und
  - $Pieces$ (die Anzahl der insgesamt erworbenen Steine).

  Gerankt wird nach Points, mit den Pieces als zweites Sortierkriterium (engl. $tie\ breaker$).

Reversi-Turnierregeln: Fragen

- Wie viele Runden gibt es?
- Wie berechnet man optimale Spielerpaarungen?

Berechne Runden für n Spieler

- n gerade: $n - 1$ Runden.

- n ungerade: Eine Runde mehr (also n Runden; jeder muss einmal pausieren).
  $\Rightarrow$ „Ghost“ einführen, dessen Gegner bekommt ein „Bye“: er muss pausieren, hat das Spiel aber gewonnen.

- Algorithmus: Die Spieler werden in zwei Reihen angeordnet. Spieler, die übereinander stehen, spielen gegeneindander. Nach jeder Runde wird im Uhrzeigersinn gewechselt, nur Spieler 1 ändert seine Position nicht.

- Wer fängt jeweils das Spiel an? Wir verdoppeln die Rundenzahl und lassen in der ersten Hälfte den obenstehenden Spieler anfangen, in der zweiten den untenstehenden.

Quelle: http://www.devenezia.com/downloads/round-robin/

Berechne Runden

Beispiel: $n = 5$ Spieler. Rundenanzahl: 5 mit Ghost.

5 Runden, die Teilnehmer drehen sich im Uhrzeigersinn, die 1 bleibt fest ("-" ist der Ghost).

1	2	3
-	5	4

1	-	2
5	4	3

1	5	-
4	3	2

1	4	5
3	2	-

1	3	4
2	-	5

Bemerkung: Die Mindestanzahl der Runden ist leicht zu beweisen, mit diesem Algorithmus hat man auch bei der Höchstanzahl eine Chance.

Apache Ant

- Ein sogenanntes „Build Tool“ (vgl. $make$), das einem dabei hilft, Übersetzungs- und Generierungsaufgaben zu automatisieren.

- Die Build-Anweisungen sind in einer XML-Datei `build.xml` definiert.
  $\Rightarrow$ können entweder über Eclipse oder über die Kommandozeile ausgeführt werden.

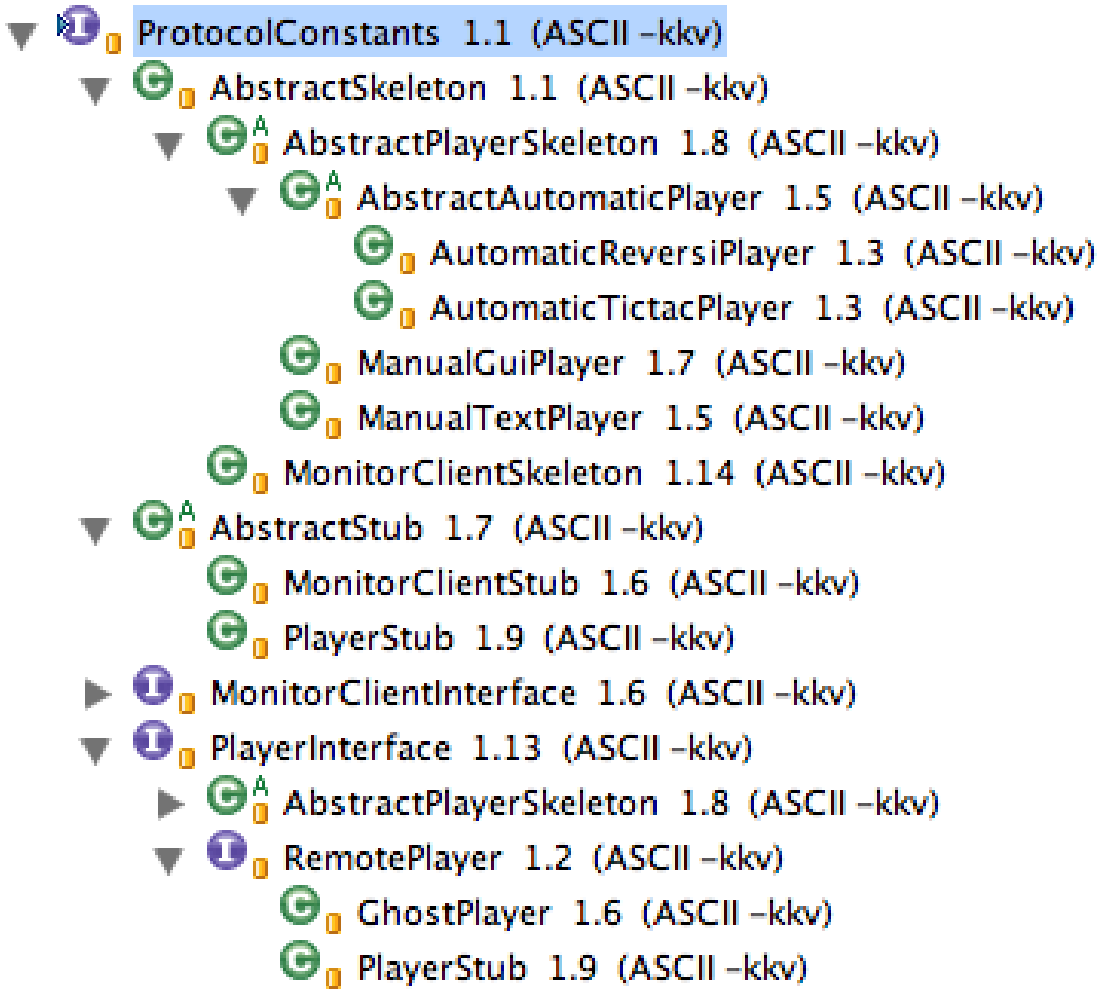
- Beispiel-Target “jarfile”: Baue ein JAR-File meiner Anwendung:
  - Soll auch ohne Eclipse funktionieren.
  - Bibliotheken sollen mit eingepackt werden.

Apache Ant: Build-File-Struktur

- Properties: Variablendefinitionen. (Wo befindet sich der Quellcode? Wo sind die Bibliotheken?)
- Targets: Eine der oben erwähnten Anweisungen.
- Tasks: Aufgaben, die im Rahmen eines Targets ausgeführt werden.
- Ant Hompage: http://ant.apache.org/
- "Top 15 Ant Best Practices": http://www.onjava.com/pub/a/onjava/2003/12/17/ant_bestpractices.html

`java07: build.xml`

Stub and Skeleton



enum PieceColor

```
BLACK, WHITE;
public PieceColor invert() // ...
public static PieceColor parseProtocolString(String string) {
    if (string.equals(NULL_PROTOCOL_STR)) {
        return null;
    } else {
        return PieceColor.valueOf(string);
    }
}
public static String toProtocolString(PieceColor pieceColor) {
    if (pieceColor != null) {
        return pieceColor.name();
    } else {
        return NULL_PROTOCOL_STR;
    }
}
```

AbstractStub

Für beliebige Stubs, nicht nur für Player.

```
public abstract class AbstractStub implements ProtocolConstants {
    public AbstractStub(Socket socket) throws ProtocolException {
        // ...
    }
    public void close() {
        // ...
    }
    protected String invoke(String output) {
        // ...
    }
    protected void invokeVoid(String output) throws ProtocolException {
        // ...
    }
}
```

RemotePlayer

```
/**
 * These are stub-only methods. Define them so that we
 * can use mock objects for testing and offline mode.
 */
public interface RemotePlayer extends PlayerInterface {
    public boolean isActive();
    public void close();
}
```

PlayerStub extends AbstractStub implements RemotePlayer

```
public PlayerStub(String serverRules, Socket socket)
        throws ProtocolException {
    super(socket);
    String playerRules = getRules();
    if (! serverRules.equalsIgnoreCase(playerRules)) {
        throw new ProtocolException(String.format(
                "Rules don't match: server is %s, player is %s",
                serverRules, playerRules));
}}
public String getRules() throws ProtocolException {
    String rules = invoke(GET_RULES);
    // Check for valid rule ID now, check for server/client match later
    if (!(rules.equals(RULES_TICTACTOE) || rules.equals(RULES_REVERSI))) {
        throw new ProtocolException("Unknown rule ID: "+rules);
    }
    return rules;
}
```

PlayerStub extends AbstractStub implements RemotePlayer

```
public String getName() {
    String name = invoke(GET_NAME);
    if (! LEGAL_NAME.matcher(name).matches()) {
        throw new ProtocolException("Illegal name: "+name);
    }
    return name;
}
public void openGame(PieceColor color, String opponentName)
        throws ProtocolException {
    invokeVoid(OPEN_GAME+SEP+PieceColor.toProtocolString(color)
            +SEP+opponentName);
}
```

AbstractSkeleton

```
public class AbstractSkeleton implements ProtocolConstants {
    public AbstractSkeleton(String hostAddress, int port)
        throws IOException {
        // ...
    }
    protected void close() throws IOException {
        // ...
    }
    protected String readln() throws IOException {
        // ...
    }
    protected void println(String line) {
        // ...
    }
}
```

AbstractPlayerSkeleton

```
public abstract class AbstractPlayerSkeleton extends AbstractSkeleton
        implements PlayerInterface {
    public static Pattern regex(String name, int arity) {
        return Pattern.compile(
                    name + Strings.repeat(arity, SEP+"([^"+SEP+"]*)"));
    }
    public static final Pattern RE_OPEN_GAME = regex(OPEN_GAME, 2);
    // ...
```

AbstractPlayerSkeleton

```
public AbstractPlayerSkeleton(String rules, String hostAddress,
        String name) throws IOException {
    this.rules = rules;
    this.name = name;
    this.socket = new Socket(hostAddress, LoginServer.LOGIN_SERVER_PORT);
}
public String getName() {
    return this.name;
}
public String getRules() {
    return this.rules;
}
// ...
```

Wie werden diese Methoden aufgerufen?

AbstractPlayerSkeleton

```
public void startInteraction() throws IOException, ProtocolException {
    String line;
    while((line = readln()) != null) {
        Matcher matcher;

        matcher = RE_OPEN_GAME.matcher(line);
        if (matcher.matches()) {
            openGame(PieceColor.parseProtocolString(
                                matcher.group(1)), matcher.group(2));
            println(VOID_RESPONSE);
        }
        // ...
    }
    close();
}
```

Andere Coding Themen

Don't Repeat Yourself (DRY)

Duplizierter Code ist böse.

`java07`: package `dry`

Tell, Don't Ask

Ich vergebe Aufträge, ich frage nicht nach den Werten und tue es selbst.

`java07`: package `template`

User-Stories

- Server (neueste Protokoll-Version, Reversi-Regeln)
  - Es gelten die in diesen Folien dargelegten Turnierregeln.
  - Der Server läßt am Anfang freies Einloggen zu.
  - Mit Anfang des ersten Turniers kann man sich nicht mehr (neu) einloggen.
  - Nach dem 1. Turnier kann man weitere Turniere starten. Neu einloggen ist weiterhin nicht möglich, dazu muss man den Server beenden und neu starten.
  - Wenn ein Spieler die Verbindung verliert (wegen eines Absturzes o.ä.), soll er sich jederzeit wieder anmelden können.

- Manueller Spieler (neueste Protokoll-Version, Reversi-Regeln)

- Ant-Script, das (für die Tutoren) ein Archiv der Klassendateien erzeugt.

Abnahme: in einer Woche.