

Allgemeine Informatik 2

Programmierprojekt

Wintersemester 18/19



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Prof. Dr. Johannes Fürnkranz

Hien Quoc Dang, Wolfgang Brabänder, Felix Gail, Philipp Malkmus

V1.0 (12. Dezember 2018)

Allgemeine Hinweise zum Projekt

Das Programmierprojekt ist auf den Lehrinhalt der Allgemeinen Informatik 2 aus dem Wintersemester 2018/2019 abgestimmt. Es wird empfohlen, dass Sie das Projekt in der aus der Vorlesung bekannten Entwicklungsumgebung BlueJ implementieren. Sie können auch Eclipse oder eine andere IDE Ihrer Wahl verwenden, jedoch muss das Projekt **unter BlueJ lauffähig sein** und auch die entsprechenden Dateien enthalten.

Die einzelnen Aufgaben bauen aufeinander auf, und folgen auch dem Vorlesungsstoff. Es macht daher Sinn, die Aufgaben in der angegebenen Reihenfolge zu bearbeiten.

Bitte lesen Sie die Aufgabenstellungen genau durch! Sollten Sie sich nicht exakt an die Aufgabenstellung halten, können wir Ihre Abgabe unter Umständen nicht werten. Insbesondere verwenden Sie genau die angegebenen Namen für Klassen, Methoden usw. Sollte ein Objekt nicht genau spezifiziert sein (z.B. *Klassen-Variablen*), können Sie selbst benennen. Sie können bei Bedarf auch eigene zusätzliche Methoden implementieren, so lange die angegebenen Methoden vorhanden sind. Methoden- und Klassenköpfe dürfen natürlich nicht verändert werden. Eingetragene Elternklassen oder Exceptions in Methodenköpfen dürfen also nicht entfernt oder hinzugefügt werden.

Beachten Sie, dass Sie die GUI des Programms zu Beginn noch nicht verwenden können. Nützen Sie die in BlueJ gebotenen Möglichkeiten, um Ihr Programm zu testen und ggf. zu debuggen. Sie können zum Beispiel Instanzen mit unterschiedlichen Konfigurationen anlegen, und auf diesen die Methoden der Objekte aufrufen. Für Fortgeschrittene stellen wir auch Unit-Tests zur Verfügung, um Feedback zur Richtigkeit Ihrer Implementation zu geben. Falls diese Tests die Richtigkeit Ihrer Implementation nicht zufriedenstellend zeigen, können Sie weitere Unit-Tests implementieren. Hierzu können Sie existierende Unit-Test Klassen erweitern oder Ihre eigenen Klassen Anlegen. Die Implementierung der Tests wird nicht bewertet (Es gibt demnach ebenso kein Punktabzug, falls Unit-Tests fehlerhaft oder gar nicht implementiert sind).

Beachten Sie auch aktuelle Hinweise zum Programmierprojekt auf <https://moodle.informatik.tu-darmstadt.de/course/view.php?id=455>

Verwendbare Java-Bibliotheken

Das Projekt ist ohne weiteres mit den in der Vorlesung besprochenen Klassen und Methoden (z.B. Collection-Interface, IO mit stdlib.jar, Exceptions, etc.) lösbar. In Fällen, in denen zusätzliche Klassen benötigt werden, werden diese angegeben.

Wenn Sie darüber hinausgehende Standard-Bibliotheken verwenden möchten, dürfen Sie das tun, jedoch müssen Sie in der Lage sein, deren Funktionsweise bei der Abgabe zu erklären.

Bewertung

Es sind insgesamt 20 Punkte zu erreichen, die auf die Klausur, bei der es bis zu 80 Punkte zu erzielen gibt, angerechnet werden. Das heißt, dieses Projekt macht 20% Ihrer Note aus. Teilpunkte sind zwar möglich, allerdings üblicherweise nicht innerhalb von Teilaufgaben, diese sind "atomar".

Nicht kompilierbare Abgaben werden mit 0 Punkten bewertet. Geben Sie daher nur Lösungen ab, die zumindest übersetzt werden können (auch wenn einzelne Aufgaben vielleicht fehlen).

Ihre Programme müssen auch ordentlich kommentiert sein. Das bedeutet einerseits, dass zumindest jeder Klasse und Methode eine entsprechende javadoc-Beschreibung beigelegt sein muss (siehe z.B. Beispiel beim Konstruktor der CardPool-Klasse, Anzeige bei "Documentation" oben rechts im Editor von BlueJ), sowie andererseits auch, dass einzelne Anweisungsblöcke im Code (z.B. Schleifen o.ä.) durch Kommentare erklärt werden müssen, welche Rolle diese spielen. Unzureichend kommentierte (Teil-)Aufgaben werden nicht mit der vollen Punkteanzahl bewertet.

Allgemeine Informatik 2 – Programmierprojekt WS 18/19

Wir gehen davon aus, dass Sie das Projekt mit BlueJ bearbeiten. Sie können auch Eclipse oder eine andere IDE Ihrer Wahl verwenden, das Projekt muss jedoch direkt unter BlueJ lauffähig sein und auch die entsprechende README.TXT-Datei enthalten.

Zu jeder von Ihnen zu bearbeitenden Aufgabe werden Tests bereitgestellt, die zum Erreichen der vollen Punktzahl auf eine Aufgabe bestanden werden müssen. Die Tests werden während des Testats dem Projekt beigelegt. *Einige Aufgaben bauen aufeinander auf. In diesen Fällen müssen beide Aufgaben vollständig bearbeitet worden sein um die Tests erfolgreich auszuführen.*

Das Projekt kann in Zweiergruppen bearbeitet werden, **jedoch muss jeder Student/jede Studentin** einzeln ein Testat ablegen. Im wesentlichen wird dort ein Tutor einige Fragen zum Code stellen, um sicherzustellen, dass die **gesamte Abgabe** verstanden wurde.

Bonuspunkte

Durch einige Teilaufgabe können Sie einen Bonuspunkt erhalten, die Abzugspunkte an anderer Stelle ausgleichen können. Diese Aufgaben sind aber z.T. aufwändiger und ohne detaillierte Anleitungen, daher etwas schwieriger zu lösen. Es ist jedoch nicht möglich, mehr als die maximale für das Projekt vorgesehene Punktzahl von 20 Punkten zu erreichen.

Abgabeformalitäten

Tragen Sie in der im Projekt enthaltenen Datei README.TXT (in BlueJ das Symbol links oben auf der Arbeitsfläche) Ihren Namen, Ihre Matrikelnummer und Ihre TU-ID (so wie bei Zweiergruppen die Daten Ihres Partners!) ein. Packen Sie Ihr komplettes Projektverzeichnis in eine Standard-ZIP-Datei, die als Namen Ihre Matrikelnummer trägt, also z.B. 1234567.zip. Keine ZipX-, 7z- oder RAR-Archive! Bei Zweiergruppen reicht eine der Matrikelnummern. Das Projekt wird über Moodle abgegeben, eine andere Form der Abgabe, oder eine zu späte Abgabe wird nicht akzeptiert und somit auch nicht gewertet!

Wichtige Daten

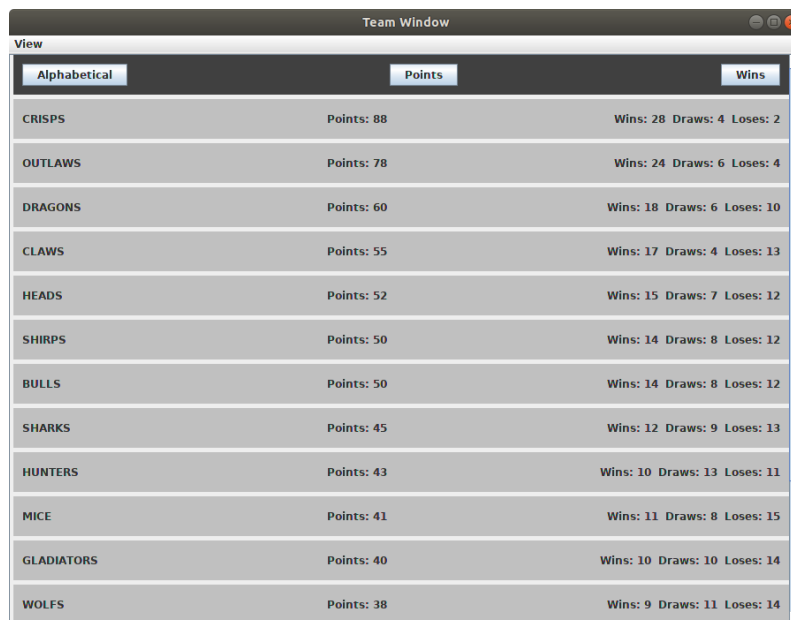
Projekt-Abgabe: 27. 1. 2019
Testate: 4.2. – 15.2.2019

Plagiarismus

Der Fachbereich Informatik misst der Einhaltung der Grundregeln der wissenschaftlichen Ethik großen Wert bei. Zu diesen gehört auch die strikte Verfolgung von Plagiarismus. Mit der Abgabe ihrer Lösung bestätigen Sie, dass Sie der alleinige Autor des gesamten Materials sind. Bei Unklarheiten zu diesem Thema finden Sie weiterführende Informationen auf https://www.informatik.tu-darmstadt.de/studium_fb20/im_studium/studienbuero/plagiarismus/ oder sprechen Sie Ihren Betreuer an. Das bedeutet nicht nur, dass Abschreiben (auch verfremdet) verboten ist, sondern auch Abschreiben lassen und Lösungen vergleichen, egal, ob das durch Weitergabe von Programmcode oder mündlich erfolgt. Lassen Sie Ihr Notebook/ Ihren Rechner/ Ihren USB-Stick nicht unbeaufsichtigt! Selbstverständlich nutzen wir zusätzlich zu einer manuellen Prüfung eine zuverlässige, auf Java-Code spezialisierte Plagiatschecker-Software.

Einleitung

In diesem Programmierprojekt geht es darum, einen Manager für Fußballturniere zu implementieren. Große Teile sind schon vorgegeben, Ihre Aufgabe ist es, sie zu einem funktionstüchtigen Programm zu vervollständigen. Wir nehmen an, daß ein Turnier aus einer Liga besteht, in der zunächst alle Teams gegeneinander spielen, was dann in einer wie in Abbildung 1 gezeigten Tabelle resultiert.



	Points	Wins
CRISPS	Points: 88	Wins: 28 Draws: 4 Loses: 2
OUTLAWS	Points: 78	Wins: 24 Draws: 6 Loses: 4
DRAGONS	Points: 60	Wins: 18 Draws: 6 Loses: 10
CLAWS	Points: 55	Wins: 17 Draws: 4 Loses: 13
HEADS	Points: 52	Wins: 15 Draws: 7 Loses: 12
SHIRPS	Points: 50	Wins: 14 Draws: 8 Loses: 12
BULLS	Points: 50	Wins: 14 Draws: 8 Loses: 12
SHARKS	Points: 45	Wins: 12 Draws: 9 Loses: 13
HUNTERS	Points: 43	Wins: 10 Draws: 13 Loses: 11
MICE	Points: 41	Wins: 11 Draws: 8 Loses: 15
GLADIATORS	Points: 40	Wins: 10 Draws: 10 Loses: 14
WOLFS	Points: 38	Wins: 9 Draws: 11 Loses: 14

Abbildung 1: Screenshot der Listenansicht des fertigen Projekts

Diese Tabelle ist nach mehreren Kriterien (alphabetisch, nach Punkten, nach der Anzahl der Gewinne) sortierbar. Diese Phase wird nicht simuliert, sondern die Tabelle aus einer Datei eingelesen und angezeigt.

Die besten Teams gelangen danach in eine KO-Phase, in welcher ein Gewinner der Saison ermittelt wird. Dazu müssen Sie in der letzten Aufgabe eine einfache Lösung zur Erstellung des Spielplans entwickeln.

Aufgabe 1 Klasse Team erstellen (Elementare Konzepte, Vererbung)

2 Punkte

Definieren Sie eine Klasse Team, die von der bereits implementierten Klasse AbstractTeam erbt und folgende Attribute intern (von außen nicht sichtbar) speichert:

- Einen String mit dem Namen des Teams.
- Jeweils eine int für die Anzahl der Siege, Unentschieden und Niederlagen des Teams.

Weiters lösen Sie folgende Aufgaben:

- a) Schreiben Sie zwei Konstruktoren für die Team-Klasse. Der erste soll in der hier aufgelisteten Reihenfolge Name, Siege, Unentschieden und Niederlagen als Argumente mit den oben angegebenen Typen übergeben bekommen und in den Attributen der Klasse speichern. Der zweite Konstruktor erhält nur den Namen des Teams und initialisiert alle anderen Werte mit 0.
- b) Implementieren Sie auch alle in AbstractTeam definierten abstrakten get-Methoden, die die Inhalte dieser Datenkomponenten auslesen.
- c) Implementieren Sie auch für jedes der drei Integer-Felder eine add-Methode, die die jeweilige interne Variable um eins erhöht und den resultierenden Wert auch als Ergebnis zurückliefern. Für eine Mannschaft x, die bereits 3 Siege hat, soll der Aufruf von x.addWin() also 4 zurückliefern (und intern die 3 durch die 4 ersetzen).

- d) Die ebenfalls zu implementierende Methode `getPoints` soll die Gesamtpunktzahl eines Teams zurückgeben. Für jeden Sieg erhält ein Team drei Punkte, für ein Unentschieden einen Punkt, und für eine Niederlage keine Punkte.

Verwenden Sie die in BlueJ vorgesehen Möglichkeiten zum Testen Ihrer Implementierung (Anlegen einer Instanz, gezielter Aufruf von Methoden, etc.). Sie können auch die von `AbstractTeam` geerbte Methode `toString` verwenden (oder durch eine eigene überschreiben), um den Inhalt eines `Team`-Objekts auf einen Blick zu sehen.

Hinweis: Im Rest des Programms werden immer Objekte des Typs `AbstractTeam` verwendet. Sie brauchen das nicht zu ändern, da natürlich jedes `Team` durch Vererbung auch der Klasse `AbstractTeam` angehört.

Aufgabe 2 Klasse `League` erstellen (*Collections, Hash-Tabellen, IO*)

4 Punkte

Der nächste Schritt ist die Implementierung einer Klasse zur Verwaltung einer Menge von Teams in einer Liga (`League`).

- a) Erstellen Sie dazu eine Klasse `League`, die wiederum die Klasse `AbstractLeague` erweitert.

Die Klasse soll Teams in einer Hash-Tabelle abspeichern, mit deren Hilfe man über den Namen eines Teams auf das geeignete `Team`-Objekt (und damit auf die zugehörigen Statistiken) zugreifen kann. Überlegen Sie sich dazu eine geeignete Lösung.

- b) Implementieren Sie die Methode `loadTeams(String)`. Die Methode bekommt als Parameter den Dateipfad einer CSV-Datei. Diese enthält in jeder Zeile den Namen des Teams, gefolgt von der Anzahl der Siege, Unentschieden und Niederlagen. Diese Werte sind durch Kommas separiert (der Dateityp `.csv` steht für "comma-separated values" und kann auch von gängigen Spreadsheet-Programmen wie Excel verarbeitet werden). Eine Beispiel-Datei `standings.csv` finden Sie im Verzeichnis `resources`, einen Auszug daraus sehen Sie in Abb. 2. Ihr Programm muß aber natürlich auch mit anderen auf diese Art aufgebauten Dateien funktionieren.

```
ANGELS , 9 , 10 , 15
CRISPS , 28 , 4 , 2
SHIRPS , 14 , 8 , 12
HEADS , 15 , 7 , 12
WOLFS , 9 , 11 , 14
```

Abbildung 2: Auszug aus der Datei `Team-CSV`.

Erstellen Sie aus jeder Zeile ein `Team`-Objekt und speichern Sie diese in Ihrer in a) gewählten Lösung. Vergessen Sie nicht, dass eventuell bereits enthaltenen Teams gelöscht bzw. ersetzt werden sollen. Zum Zerlegen einer Zeile wählen Sie geeignete Methoden der Klasse `String`.

Hinweis: Sie können zum Einlesen die `In`-Klasse aus der `stdlib`-Bibliothek nutzen. Kann die Datei nicht gefunden werden, oder hat die Datei nicht das richtige Muster, soll die Menge der abgespeicherten Teams leer bleiben.

- c) Implementieren Sie die in der abstrakten Klasse `AbstractLeague` vorgesehene Methode `getTeams()`, die eine (unsortierte) Liste aller Teams zurückgeben soll. Sie müssen diese geeignet aus Ihrer Datenstruktur auslesen. Wenn keine Teams abgespeichert sind, soll ein leeres Listen-Objekt zurückgegeben werden.

Hinweis: Die gleichnamige Methode `'public List<Team> getTeams(Comparator<Team> comp)'`, die einen Komparator als Argument erhält, können Sie vorerst ignorieren.

- d) Implementieren Sie ebenso die in der abstrakten Klasse `AbstractTeam` vorgesehene Methode `addResult`, die ein Spielergebnis zu den momentanen Teams hinzufügt. Übergeben werden der Name der beiden Teams und wie viele Tore sie jeweils erzielt haben. Das Team, das mehr Tore erzielt hat, erhält einen Sieg gut geschrieben, das andere eine Niederlage. Bei Gleichstand erhalten beide Teams ein Unentschieden. Verwenden Sie dazu die `win`-Methoden, die Sie in Aufgabe 1 geschrieben haben.

Hinweis: Sie können annehmen, dass beide Mannschaften bereits in der Tabelle existieren.

- e) Schreiben Sie eine Methode `printTeams()`, die die momentanen Teams auf dem Bildschirm ausgibt. Das Format der Ausgabe können Sie frei wählen (z.B. einfach durch Aufruf der `toString()`-Methode der einzelnen Teams). Diese Methode wird im Rest des Programms nicht weiter verwendet, Sie dient primär zu Ihrer eigenen Kontrolle.

Wenn Sie die Aufgaben gelöst haben, sollten Sie nun der Lage sein, durch Aufruf der `main`-Methode der Klasse `TeamManager` eine Liste aller Teams aus Ihrer Datei wie in Abb. 1 angezeigt zu bekommen.

Bonus: Ausnahmebehandlung

1 Punkte

Implementieren Sie die Klasse `MalformedFileException`, die von `RuntimeException` erbt und die geworfen wird, wenn das definierte Muster in der in b) einzulesenden Datei verletzt wird.

Aufgabe 3 Ordnen von Teams (Interfaces, Comparators)

3 Punkte

Diese und die nächste Aufgabe beschäftigen sich mit der Implementierung und Nutzung von Sortier-Algorithmen. Um Teams sortieren zu können müssen wir sie zuerst vergleichen können. Setzen Sie dazu zunächst die folgenden Teilaufgaben um:

- Vervollständigen Sie die Komparator-Klasse `PointsComparator`, welche das in der Vorlesung besprochene Interface `java.util.Comparator` für die `Team`-Klasse implementiert. Die dabei zu implementierende Methode `'public int compare(Team t1, Team t2)'` soll zwei Teams anhand der anhand ihrer Punktzahl (Methode `getPoints` aus Aufgabe 1) vergleichen. Wie in der Vorlesung besprochen, soll `compare` einen positiven Wert zurückgeben, wenn `t1` weniger Punkte als `t2` hat, 0 wenn beide Teams gleich viele Punkte haben, und einen negativen Wert wenn `t1` mehr Punkte als `t2` hat.
- Nun möchten wir noch "alphabetisch" als natürliche Ordnung für `Team`-Objekte festlegen. Erweitern Sie dazu die in Aufgabe 1 erstellte Klasse `Team` sodass sie das in der Vorlesung besprochene Interface `java.util.Comparable` implementiert. Es soll dadurch eine alphabetische Ordnung aufgrund des Namens der Teams erfolgen.
- Um die in b) erstellte natürliche Ordnung nützen zu können, modifizieren sie Ihre in Aufgabe 2 erstellte `getTeams()`-Methode dahingehend, dass sie diese Ordnung nützt. Sie können beispielsweise die Teams vorübergehend in einem Objekt eines geeigneten `Collection`-Typs für eine sortierte Sammlung ablegen, und dieses dann in eine Liste umwandeln und zurückgeben.
- Implementieren Sie zuletzt noch die zweite Komparator-Klasse `WinsComparator`, die zwei Teams aufgrund der gewonnenen und verlorenen Spiele vergleicht ohne dabei Punkte zu berechnen. Die `compare`-Methode dieser Klasse vergleicht zwei Teams zunächst nach der Anzahl der gewonnenen und verlorenen Spiele. Das Team, das mehr Spiele gewonnen hat, ist zuerst zu reihen. Bei einer gleichen Anzahl von Siegen wird das Team mit weniger Niederlagen zuerst gereiht. Wenn hier ebenfalls Gleichstand erzielt wird, sollen die Teams nach ihrer natürlichen Ordnung gereiht werden (siehe Aufgabe b)).

Aufgabe 4 Sortieren (Sortier-Algorithmen)

3 Punkte

Implementieren Sie den in der Vorlesung erwähnten Algorithmus Bubble-Sort, der in mehreren Durchläufen durch eine Liste immer zwei benachbarte Elemente vergleicht und ggf. vertauscht, bis in einem Durchlauf keine Vertauschungen mehr durchgeführt wurden. Mehr Informationen finden Sie u.a. in Wikipedia¹ oder Nabla.²

Überschreiben Sie dazu in der Klasse `Team` die von `AbstractTeam` geerbte Methode `getTeams(Comparator<Team>)`, die ihrerseits wieder `getTeams()` überladet. Die Methode soll wie `getTeams()` eine Liste von Teams zurückgeben, diese jedoch zuerst mit Bubble-Sort sortieren. Wird kein Komparator übergeben (`null`) muss die Methode dasselbe Resultat wie `getTeams()` zurückliefern.

Nach der Fertigstellung dieser Aufgabe sollten nun in der Lage sein, die Sortier-Buttons der GUI zu verwenden.

Hinweis: Sie müssen Bubble-Sort natürlich selbst implementieren. Aufrufe von `Collections.sort()` oder ähnlichen Methoden sind hier nicht erlaubt.

¹ <https://de.wikipedia.org/wiki/Bubblesort>

² <https://nabla.algo.informatik.tu-darmstadt.de/catalog/6>

Aufgabe 5 GUI zum Hinzufügen eines Ergebnisses (IO, GUI)

4 Punkte

Wir möchten nun Funktionalität hinzufügen, die es erlaubt, die Tabelle um ein neues Ergebnis zu erweitern. Sie sollen dazu eine geeignetes User Interface designen. Dazu modifizieren Sie den markierten Teil im Konstruktor der Klasse `ResultsPanel`, der ein `Team`-Objekt übergeben bekommt. Sie erbt von der Klasse `JPanel` und ist bereits im Menü unter „Add Result“ eingebunden, es fehlt jedoch noch das Design.

- a) Es stehen Ihnen in der Klasse schon einige Komponenten zur Verfügung, die Sie verwenden und in der GUI anordnen müssen. Diese sind:
- `JComboBox teamSelector1, teamSelector2` sind Auswahlboxen für zwei Mannschaften. Mit der Methode `Object getSelectedItem()` können Sie die jeweilige Auswahl auslesen.
 - `TextField scoreField1, scoreField2` sind Textfelder, in denen Sie die erzielten Tore für jede Mannschaft eintragen können, die mit der Methode `String getText()` ausgelesen werden können.
 - `JButton addButton` liest die Daten aus den Feldern aus.
 - `JLabel statusLabel` zeigt eine Status-Nachricht an. Die Nachricht kann durch die Methode `void setText(String)` verändert werden.

Integrieren Sie diese Komponenten in Ihr Design. Um ein verständliches Design zu erstellen können Sie auch noch weitere, eigene Komponenten verwenden. Ein Beispiel-Design sehen Sie in Abbildung 3.

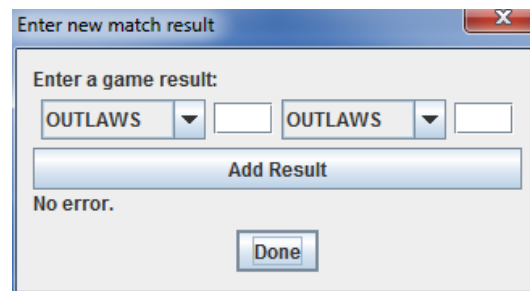


Abbildung 3: GUI-Beispiel des `AddTeamPanel`s

- b) Die Funktionalität des Designs müssen Sie durch einen Listener für den `addButton` realisieren. Dieser Listener liest mit Hilfe der oben angegebenen Funktionen die Eingaben aus den anderen Feldern aus und integriert sie durch Aufruf der in 2.d) realisierten Methode `addResult` in die beiden beteiligten Teams. Es empfiehlt sich auch eine passende Status-Nachricht (z.B. dass das ein Resultat eingelesen wurde oder eventuell auch dass ein Eingabefehler vorlag) zu setzen.

Hinweis: Es ist Ihnen überlassen, welches Layout sie verwenden, solange es am Ende verständlich und benutzbar ist. Verändern Sie aber nicht die Deklarationen der vorgegebenen Komponenten oder die allgemeine Struktur der Klasse (z.B. den Konstruktor).

Bonus: Speichern der Tabelle

1 Punkte

Im Moment ist es im Menü "File" nur möglich eine CSV-Datei zu laden. Wenn Sie die Änderungen auch speichern möchten, können Sie analog dazu eine "Save" Methode realisieren, die z.B. bei Auswahl eines entsprechenden Menü-Punktes die Datei nach den Änderungen wieder ausschreibt.

Aufgabe 6 Erstellen der Baumstruktur (Bäume, Rekursion)

4 Punkte

Die ersten acht Teams qualifizieren sich für eine K.O.-Runde, in der der endgültige Sieger ermittelt wird. Die Funktion ist im Menü "View → Tournament → Tree" bereits vorgesehen, muss aber von Ihnen noch vervollständigt werden.

Insbesondere muss eine Baumstruktur erstellt, die drei Ebenen tief ist (also 8 Blätter hat). An den Blättern werden die 8 Teams in einer vorgegebenen Reihenfolge eingetragen. Durch Aufruf des Menüs wird dann eine Eingabe-Maske generiert, die es erlaubt, den jeweiligen Gewinner anzuklicken, und als Gewinner an die nächst-höhere Ebene des Baums weiterzumelden (vgl. Abbildung 4).

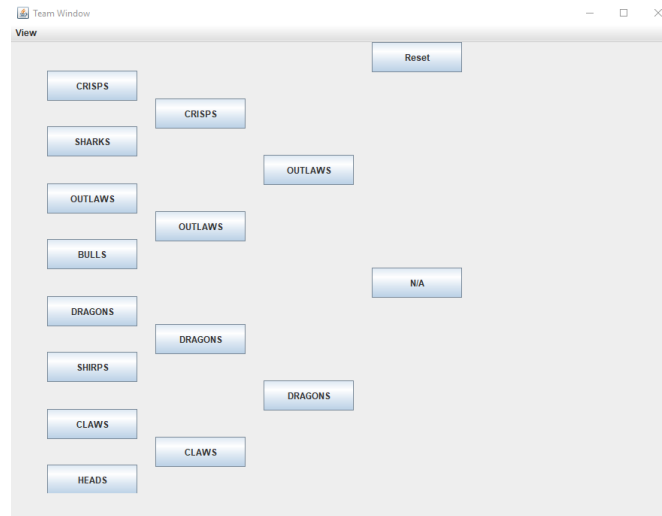


Abbildung 4: Screenshot der KO-Phase des fertigen Projekts

- a) Erstellen Sie die Klasse `TournamentNode`, die alle in `AbstractTournamentNode` beschriebenen Funktionalitäten implementiert. Jeder `TournamentNode` besteht (zumindest) aus Verweisen auf seinen linken und rechten Nachfolger, sowie auf seinen Eltern-Knoten, und speichert das ihm zugeordnete Team ab. Internen Knoten ist zu Beginn kein Team zugeordnet (Wert null), beim Füllen des Baums wird dem Knoten dann der Gewinner aus dem Spiel der beiden Mannschaften in den Nachfolge-Knoten zugeordnet. Die Blatt-Knoten werden mit der Methode `initializeTree` aus Aufgabe b) mit Mannschaften gefüllt.
- Erstellen Sie einen Konstruktor für einen `TournamentNode`, der seinen Eltern-Knoten (repräsentiert durch einen `AbstractTournamentNode`) übergeben bekommt, diesen abspeichert, und alle anderen Komponenten mit null initialisiert.
 - Schreiben Sie die angegebenen get und set-Methoden `getLeftChild`, `getRightChild`, `getParent`, und `setChildren`.
 - Die Methode `getWinner` liefert das in diesem Knoten abgespeicherte Team (kann auch null sein). Das Gegenstück dazu, `getLoser`, liefert den Verlierer zurück (also denjenigen der beiden Kind-Knoten dessen Team nicht in diesem Knoten abgespeichert wurde). An den Blättern soll `getLoser` null als Ergebnis liefern.
 - Die Methode `setWinner` soll das übergebene Team im aktuellen Knoten abspeichern, allerdings nur, wenn beide Nachfolger-Knoten bereits ihre Gewinner abgespeichert haben (also `getWinner` für diese Knoten nicht null liefert) und einer der beiden das übergebene Team ist. Sollte es sich um einen Blatt-Knoten handeln (zumindest einer der beiden Nachfolger existiert nicht), wird der Knoten auf jeden Fall abgespeichert. Die Methode liefert `true`, wenn neue Information abgespeichert wurde.
- b) Die Klasse `TournamentTree` repräsentiert einen Turnier-Baum, indem der Knoten der Wurzel in `root` abgespeichert. Ausserdem erhält sie im Konstruktor noch eine (nach Punkten sortierte) Liste von Teams übergeben, die sie intern als `teams` speichert. Zu guter Letzt verwendet sie noch einen `SCHEDULE`, der angibt, welche Teams aus der Liste gegeneinander spielen sollen. Der in der Klasse vorgegebene `SCHEDULE` ist `{0, 7, 3, 4, 2, 5, 1, 6}`, was bedeutet, dass Team 0 gegen Team 7 spielen soll, der Sieger dann gegen den Sieger von Team 3 und Team 4, und der Sieger daraus dann gegen das beste Team aus 2, 5, 1, und 6, das analog dazu ermittelt wird.

Ihre Aufgabe ist es nun, die Methode `'void initializeTree(AbstractTournamentNode n, int[] schedule)` rekursiv zu implementieren. Der Basisfall ist, wenn der übergebene Array nur mehr aus einem Element `i` besteht. In diesem Fall soll ein Blattknoten (ohne Nachfolge-Knoten) erzeugt werden und das `i`-te Element der Liste `teams` darin abgespeichert werden. Ansonsten wird ein interner Knoten mit zwei Nachfolgern erzeugt, wobei die erste Hälfte des Schedules an den linken und die zweite Hälfte an den rechten Nachfolger übergeben wird.

Hinweis: Sie können davon ausgehen, dass die Länge der Liste `schedule` eine Potenz von 2 ist. Zum Teilen des Arrays können Sie die Methode `Arrays.copyOfRange(a, i, j)` verwenden. Sie liefert eine Kopie des Teils des Arrays `a`, der an der Position `i` beginnt und bis (exklusive) `j` läuft. Sie müssen auch den auskommentierten Aufruf von `initializeTree` im Konstruktor aktivieren.