Name: _____

Abiturprüfung 2023

Informatik, Leistungskurs

Aufgabenstellung:

Um bei Sportturnieren den unterschiedlichen Leistungsstärken der Teams gerecht zu werden, sollen möglichst Teams gegeneinander spielen, die ungefähr gleich stark sind. Dazu werden die Teams nach jeder Runde in einer Reihenfolge angeordnet, die dem Erfolg des Teams im Turnier entspricht. Anhand der Reihenfolgen werden die Spiele nach folgendem Prinzip ermittelt:

- Bei Beginn des Turniers werden alle Teams in zufälliger Reihenfolge angeordnet.
- Nach jeder Partie wird die Reihenfolge der Teams anhand ihrer Punktzahlen angepasst: Die Reihenfolge der Teams ist absteigend nach deren Punktzahl geordnet. Für einen Sieg erhält ein Team `1` Punkt, für eine Niederlage `-1` Punkt. Bei einem Unentschieden erhalten beide Teams `0` Punkte.
- Um für ein Team `A` einen Gegner zu bestimmen, wird das Team `B` als Gegner ausgewählt, das in der Reihenfolge möglichst nah beim Team `A` ist.
- Um zu vermeiden, dass immer wieder die gleichen Teams gegeneinander spielen, wird eine Maximalzahl gleicher Begegnungen festgelegt: Zwei Teams sollen höchstens so häufig wie in dieser Zahl festgelegt gegeneinander spielen.

Folgendes Softwaremodell ist ein erster Entwurf für die Verwaltung eines solchen Turniers:

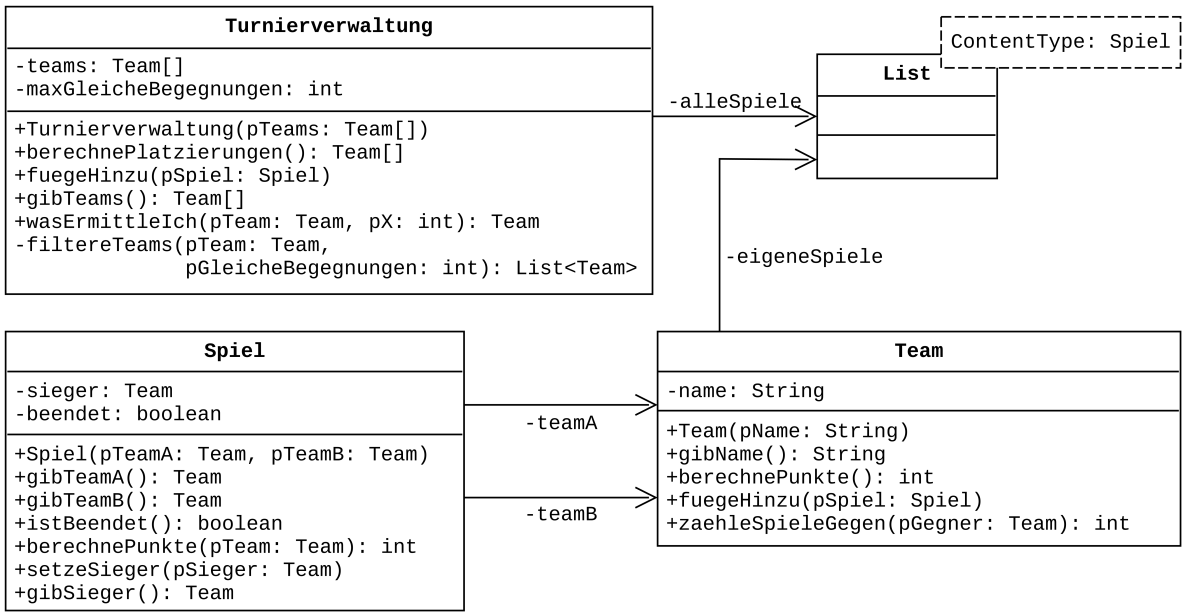


Abbildung 1: Teilmodellierung des Turniers

Name: _____

a) *Analysieren Sie das Implementationsdiagramm und erläutern Sie die dargestellten Assoziationen im Sachkontext.*

Erläutern Sie anhand des Implementationsdiagramms und der Dokumentation im Anhang, wie laut dem Modell ein Spiel hinzugefügt und wie der Sieger eingetragen wird.

(4 + 4 Punkte)

b) Die private Methode `filtereTeams` der Klasse `Turnierverwaltung` filtert die Teams heraus, die höchstens eine bestimmte Anzahl von Spielen gegen das übergebene Team gespielt haben. Das als Parameter übergebene Team ist nicht im Ergebnis der Methode enthalten.
Wird für `pTeam null` übergeben oder ist der Wert von `pGleicheBegegnungen` kleiner als `0`, so wird eine leere Liste zurückgegeben. Die Methode hat den folgenden Methodenkopf:

```
private List<Team> filtereTeams(Team pTeam,
                                int pGleicheBegegnungen)
```

Entwickeln Sie einen Algorithmus, mit dem das Filtern entsprechend der oben beschriebenen Vorgabe durchgeführt werden kann.

Implementieren Sie die Methode.

(4 + 6 Punkte)

Name: _____

c) Der Quellcode des Projekts enthält in der Klasse `Turnierverwaltung` folgende nicht dokumentierte Methode:

```
 1 public Team wasErmittleIch(Team pTeam, int pX) {
 2   List<Team> auswahl
 3       = filtereTeams(pTeam, pX);
 4   auswahl.toFirst();
 5   int punkte = pTeam.berechnePunkte();
 6   Team g = auswahl.getContent();
 7   int d = Math.abs(g.berechnePunkte() - punkte);
 8   auswahl.next();
 9   while (auswahl.hasAccess()) {
10     Team temp = auswahl.getContent();
11     int diff = Math.abs(temp.berechnePunkte()
12                         - punkte);
13     if (diff < d) {
14       d = diff;
15       g = temp;
16     }
17     auswahl.next();
18   }
19   return g;
20 }
```

Hinweis: Die Methode `Math.abs(int a)` liefert den Absolutbetrag einer Integerzahl `a`. Ist beispielsweise `a = -5`, also negativ, so ist die Rückgabe `5`. Ist `a = 3`, also positiv, ist die Rückgabe `3`.

Die Methode soll mithilfe der Beispieldaten in der Anlage (Wert des Attributs `pX = 0` und Team `SF Forst`) analysiert werden.

Analysieren Sie die Methode `wasErmittleIch`*, indem Sie sie auf die Beispieldaten anwenden und die Rückgabe angeben.*

Erläutern Sie die zugrundeliegende Strategie des Algorithmus.

Erläutern Sie, was die Methode im Sachkontext ermittelt.

Analysieren und erläutern Sie, an welcher Stelle des Quellcodes ein Laufzeitfehler durch einen Zugriff auf ein nicht existentes Objekt (eine sogenannte `NullPointerException`*) entstehen kann.*

(4 + 3 + 2 + 4 Punkte)

Name: _____

d) Um die Software unter anderem für verschiedene Sportarten einsetzen zu können, soll die Software flexibler gestaltet werden. Deswegen soll das Modell um folgende Punkte verändert werden:

1. In verschiedenen Sportarten werden Sieg, Unentschieden und Niederlage unterschiedlich bepunktet. Für jedes Turnier sollen diese Punkte zu Turnierbeginn festgelegt werden können.
2. Für jedes Team soll jeweils angefragt werden können, gegen welche Teams es bereits gespielt hat.

Erweitern Sie das Modell aus Abbildung 1 um die angegebenen Erweiterungen.

Erläutern Sie, wie Ihr Modell die Anforderungen umsetzt.

Hinweis: Stellen Sie nur die Änderungen im Modell im Vergleich zu Abbildung 1 dar.

(6 + 5 Punkte)

Name: _____

e) Bei der Implementierung der Software entsteht die Frage, wie die Spiele für die nächste Runde von der Software ermittelt werden sollen. Zur Vereinfachung soll die Anzahl der bereits gegeneinander gespielten Spiele zweier Mannschaften hier nicht berücksichtigt werden. Außerdem wird von einer geraden Anzahl von Teams ausgegangen.

Die Software soll dafür sorgen, dass für jedes Spiel zu erwarten ist, dass es möglichst ausgeglichen sein wird. Dazu sollen Teams gegeneinander spielen, die ähnlich erfolgreich im Turnier sind: Wenn zwei Teams gegeneinander spielen sollen, soll die Differenz ihrer Punkte möglichst gering sein.

Dabei gibt es zwei Vorschläge:

(i) Beginne in der Mitte der Rangfolge:
- Wähle das erste der beiden mittleren Teams.
- Dieses Team ist in der Reihenfolge von zwei Teams umgeben: Wähle von den beiden Teams das als Gegner aus, zu dem die Punktdifferenz geringer ist.
- Ist die Punktdifferenz gleich, wähle das untere Team als Gegner.
- Entferne die beiden gewählten Teams aus der Rangfolge.
- Wiederhole diesen Prozess, bis nur noch zwei Teams ungeplant sind.
- Sind nur noch zwei Teams ungeplant, lasse diese beiden gegeneinander spielen.

(ii) Beginne bei der aktuellen Rangfolge beim ersten Team:
- Wähle das erste Team gegen das zweite Team als Spiel.
- Entferne beide Teams aus der Rangfolge.
- Wiederhole diesen Prozess, bis die Rangfolge leer ist.

Ermitteln Sie für beide Vorschläge die Spiele und die Differenz der Punkte für jedes Spiel anhand der vorliegenden Beispieldaten im Anhang.

Beurteilen Sie welcher Ansatz zu besseren Spielpaarungen führt.

(4 + 4 Punkte)

Zugelassene Hilfsmittel:

- Taschenrechner (grafikfähiger Taschenrechner / CAS-Taschenrechner)
- Wörterbuch zur deutschen Rechtschreibung

Name: _____

Anlage: Beispieldaten

Teams (in dieser Reihenfolge im Feld `teams`):

```
SF Forst
SG Heide
TV Feld
SSV Bergen
TuS Wiese
SV Wald
```

Spiele:

Runde 1:

```
SF Forst – SG Heide:          Sieger ist SF Forst
TV Feld – SSV Bergen:         Sieger ist TV Feld
TuS Wiese – SV Wald:          Unentschieden
```

Runde 2:

```
SF Forst – TV Feld:           Unentschieden
TuS Wiese – SG Heide:         Sieger ist TuS Wiese
SV Wald – SSV Bergen:         Unentschieden
```

Aktueller Punktestand:

Platzierung	Team	Punkte
1.	SF Forst	1
2.	TV Feld	1
3.	TuS Wiese	1
4.	SV Wald	0
5.	SSV Bergen	-1
6.	SG Heide	-2

Ergebnis von `filtere(pTeam, pX)`, wobei es sich bei dem Team um `SF Forst` handelt und `pX` den Wert `0` besitzt:

```
SSV Bergen
TuS Wiese
SV Wald
```

Name: _____

Anhang

Dokumentationen der verwendeten Klassen

Die Klasse `Team`

Objekte der Klasse `Team` repräsentieren die zu einem Team vorhandenen Daten. Zu jedem Team in einem Turnier soll immer nur genau ein Objekt existieren. Ein Objekt der Klasse verwaltet Referenzen auf die Spiel-Objekte, an denen das Team beteiligt ist.

Dokumentation der Klasse `Team`

`Team(String pName)`
Ein Objekt der Klasse wird initialisiert. Die übergebene Zeichenkette wird gespeichert. Das Team hat keine eigenen Spiele gespeichert.

`String gibName()`
Die Methode liefert den Namen des Teams zurück.

`int berechnePunkte()`
Die Methode liefert die Punkte des Teams auf Basis der eigenen Spiele, die bereits beendet sind. Die Rückgabe entspricht der Summe der Einzelbewertungen jedes beendeten Spiels für das Team. Wurde noch kein eigenes Spiel beendet, wird `0` zurückgegeben.

`void fuegeHinzu(Spiel pSpiel)`
Das durch den Parameter referenzierte Objekt der Klasse `Spiel` wird zu den eigenen Spielen hinzugefügt.

`int zaehleSpieleGegen(Team pGegner)`
Die Methode liefert die Anzahl der gespielten Spiele, die das als Parameter übergebene Team gegen dieses Team gespielt hat. Wird `null` für den Parameter übergeben, so wird `-1` zurückgegeben.

Name: _____

Die Klasse `Turnierverwaltung`

Objekte der Klasse `Turnierverwaltung` verwaltet alle Teams und alle Spiele eines Turniers. Die Teams werden in der Reihenfolge verwaltet, in der sie sich für das Turnier angemeldet haben.

Dokumentation der Klasse `Turnierverwaltung`

`Turnierverwaltung(Team[] pTeams)`
Ein Objekt der Klasse wird initialisiert. Die im Feld übergebenen Teams werden in einer zufälligen Reihenfolge angeordnet. Es existieren keine Spiele.

`Team[] berechnePlatzierungen()`
Die Methode liefert alle im Turnier vorhanden Teams in einem Feld. Das Feld ist absteigend nach der Punktzahl der Teams sortiert. Sind zwei Teams punktgleich, ist die Reihenfolge zufällig.

`void fuegeHinzu(Spiel pSpiel)`
Das übergebene Spiel wird der Liste aller Spiele hinzugefügt. Außerdem wird beiden Teams das übergebene Spiel hinzugefügt. Dabei wird davon ausgegangen, dass die beiden im Spiel referenzierten Objekte der Klasse `Team` nicht identisch sind und von dem Objekt der Klasse `Turnierverwaltung` referenziert werden.

`Team[] gibTeams()`
Die Methode liefert die Teams in der Reihenfolge, in der sie sich zum Turnier angemeldet haben.

Über die öffentlichen Methoden hinaus verfügt die Klasse über folgende private Methode:
Dokumentation dieser privaten Methode der Klasse:

`List<Team> filtereTeams(Team pTeam, int pGleicheBegegnungen)`
Es wird eine Liste zurückgegeben, die alle Teams enthält, die bisher in dem Turnier höchstens den als den Wert von `pGleicheBegegnungen` mal gegen das übergebene Team gespielt haben. Referenziert der Parameter `pTeam null` oder ist der Wert des Parameters `pGleicheBegegnungen` kleiner als `0`, so wird eine leere Liste zurückgegeben.

Name: _____

Die Klasse `Spiel`

Ein Objekt der Klasse `Spiel` verwaltet die beteiligten Teams, ob das Spiel bereits beendet wurde, und das siegende Team des Spiels, sofern es eins gibt.

Dokumentation der Klasse `Spiel`

`Spiel(Team pTeamA, Team pTeamB)`
Ein Objekt der Klasse wird initialisiert. Die beiden übergebenen Teams werden in dem Spiel gespeichert. Das Spiel gilt als nicht beendet. Es ist kein Sieger eingetragen.

`Team gibTeamA()`
Die Methode liefert das Team A des Spiels.

`Team gibTeamB()`
Die Methode liefert das Team B des Spiels.

`boolean istBeendet()`
Ist das Spiel beendet, wird `true` zurückgegeben, andernfalls wird `false` zurückgegeben.

`int berechnePunkte(Team pTeam)`
Ist das Spiel nicht beendet oder ist kein Sieger gespeichert (das Spiel also als Unentschieden gewertet), so wird der Wert `0` zurückgegeben. Ist das im Parameter referenzierte Team nicht an dem Spiel beteiligt, wird der Wert `-1` zurückgegeben.

`void setzeSieger(Team pSieger)`
Das Spiel wird als beendet gewertet. Das übergebene Team wird als Sieger gesetzt. Wird `null` übergeben, gilt das Spiel als unentschieden. Es wird davon ausgegangen, dass entweder `null` oder ein beteiligtes Team übergeben wird.

`Team gibSieger()`
Die Methode liefert den Sieger des Spiels. Gibt es keinen Sieger, weil das Spiel unentschieden oder noch nicht beendet ist, wird `null` geliefert.

Name: _____

Die generische Klasse `List`

Objekte der generischen Klasse `List` verwalten beliebig viele, linear angeordnete Objekte vom Typ `ContentType`. Auf höchstens ein Listenobjekt, aktuelles Objekt genannt, kann jeweils zugegriffen werden. Wenn eine Liste leer ist, vollständig durchlaufen wurde oder das aktuelle Objekt am Ende der Liste gelöscht wurde, gibt es kein aktuelles Objekt. Das erste oder das letzte Objekt einer Liste kann durch einen Auftrag zum aktuellen Objekt gemacht werden. Außerdem kann das dem aktuellen Objekt folgende Listenobjekt zum neuen aktuellen Objekt werden.
Das aktuelle Objekt kann gelesen, verändert oder gelöscht werden. Außerdem kann vor dem aktuellen Objekt ein Listenobjekt eingefügt werden.

Dokumentation der Klasse `List<ContentType>`

`List()`

Eine leere Liste wird erzeugt.

`boolean isEmpty()`

Die Anfrage liefert den Wert `true`, wenn die Liste keine Objekte enthält, sonst liefert sie den Wert `false`.

`boolean hasAccess()`

Die Anfrage liefert den Wert `true`, wenn es ein aktuelles Objekt gibt, sonst liefert sie den Wert `false`.

`void next()`

Falls die Liste nicht leer ist, es ein aktuelles Objekt gibt und dieses nicht das letzte Objekt der Liste ist, wird das dem aktuellen Objekt in der Liste folgende Objekt zum aktuellen Objekt, andernfalls gibt es nach Ausführung des Auftrags kein aktuelles Objekt, d. h., `hasAccess()` liefert den Wert `false`.

`void toFirst()`

Falls die Liste nicht leer ist, wird das erste Objekt der Liste aktuelles Objekt. Ist die Liste leer, geschieht nichts.

`void toLast()`

Falls die Liste nicht leer ist, wird das letzte Objekt der Liste aktuelles Objekt. Ist die Liste leer, geschieht nichts.

Name: _____

`ContentType getContent()`

Falls es ein aktuelles Objekt gibt (`hasAccess() == true`), wird das aktuelle Objekt zurückgegeben. Andernfalls (`hasAccess() == false`) gibt die Anfrage den Wert `null` zurück.

`void setContent(ContentType pContent)`

Falls es ein aktuelles Objekt gibt (`hasAccess() == true`) und `pContent` ungleich `null` ist, wird das aktuelle Objekt durch `pContent` ersetzt. Sonst bleibt die Liste unverändert.

`void append(ContentType pContent)`

Ein neues Objekt `pContent` wird am Ende der Liste eingefügt. Das aktuelle Objekt bleibt unverändert. Wenn die Liste leer ist, wird das Objekt `pContent` in die Liste eingefügt und es gibt weiterhin kein aktuelles Objekt (`hasAccess() == false`). Falls `pContent` gleich `null` ist, bleibt die Liste unverändert.

`void insert(ContentType pContent)`

Falls es ein aktuelles Objekt gibt (`hasAccess() == true`), wird ein neues Objekt `pContent` vor dem aktuellen Objekt in die Liste eingefügt. Das aktuelle Objekt bleibt unverändert. Falls die Liste leer ist und es somit kein aktuelles Objekt gibt (`hasAccess() == false`), wird `pContent` in die Liste eingefügt und es gibt weiterhin kein aktuelles Objekt. Falls es kein aktuelles Objekt gibt (`hasAccess() == false`) und die Liste nicht leer ist oder `pContent == null` ist, bleibt die Liste unverändert.

`void concat(List<ContentType> pList)`

Die Liste `pList` wird an die Liste angehängt. Anschließend wird `pList` eine leere Liste. Das aktuelle Objekt bleibt unverändert. Falls es sich bei der Liste und `pList` um dasselbe Objekt handelt, `pList == null` oder eine leere Liste ist, bleibt die Liste unverändert.

`void remove()`

Falls es ein aktuelles Objekt gibt (`hasAccess() == true`), wird das aktuelle Objekt gelöscht und das Objekt hinter dem gelöschten Objekt wird zum aktuellen Objekt. Wird das Objekt, das am Ende der Liste steht, gelöscht, gibt es kein aktuelles Objekt mehr (`hasAccess() == false`). Wenn die Liste leer ist oder es kein aktuelles Objekt gibt (`hasAccess() == false`), bleibt die Liste unverändert.

Unterlagen für die Lehrkraft

Abiturprüfung 2023

Informatik, Leistungskurs

1. Aufgabenart

Analyse, Modellierung und Implementation kontextbezogener Problemstellungen mit Schwerpunkt auf den Inhaltsfeldern Daten sowie ihre Strukturierung und Algorithmen

2. Aufgabenstellung[1]

siehe Prüfungsaufgabe

3. Materialgrundlage

entfällt

4. Bezüge zum Kernlehrplan und zu den Vorgaben 2023

Die Aufgaben weisen vielfältige Bezüge zu den Kompetenzerwartungen und Inhaltsfeldern des Kernlehrplans bzw. zu den in den Vorgaben ausgewiesenen Fokussierungen auf.
Im Folgenden wird auf Bezüge von zentraler Bedeutung hingewiesen.

1. *Inhaltsfelder und inhaltliche Schwerpunkte*
   Daten und ihre Strukturierung
   - Objekte und Klassen
     - Entwurfsdiagramme und Implementationsdiagramme
     - Lineare Strukturen
       Array bis zweidimensional
       Lineare Liste (Klasse `List`)

   Algorithmen
   - Analyse, Entwurf und Implementierung von Algorithmen
   - Algorithmen in ausgewählten informatischen Kontexten

   Formale Sprachen und Automaten
   - Syntax und Semantik einer Programmiersprache
     - Java

2. *Medien/Materialien*
   - entfällt

[1] Die Aufgabenstellung deckt inhaltlich alle drei Anforderungsbereiche ab.

5. Zugelassene Hilfsmittel

- GTR (grafikfähiger Taschenrechner) oder CAS (Computer-Algebra-System)
- Wörterbuch zur deutschen Rechtschreibung

6. Modelllösungen

Die jeweilige Modelllösung stellt eine mögliche Lösung bzw. Lösungsskizze dar. Der gewählte Lösungsansatz und -weg der Schülerinnen und Schüler muss nicht identisch mit dem der Modelllösung sein. Sachlich richtige Alternativen werden mit entsprechender Punktzahl bewertet (Bewertungsbogen: Zeile „Sachlich richtige Lösungsalternative zur Modelllösung“).

Teilaufgabe a)

Im Diagramm sind die vier Assoziationen `alleSpiele`, `eigeneSpiele`, `teamA` und `teamB` dargestellt.
Die Assoziation `alleSpiele` modelliert den Sachverhalt, dass ein Objekt der Klasse `Turnierverwaltung` alle Spiele des Turniers in einer Liste (Datentyp `List` mit Inhaltstyp `Spiel`) verwaltet – unabhängig davon, ob ein Spiel beendet ist oder nicht.
Die Assoziationen `teamA` und `teamB` modellieren die Beteiligung der beiden Teams an einem bestimmten Spiel. Ein Objekt der Klasse `Spiel` verwaltet also über diese Referenzen zwei Objekte vom Typ `Team`.
Die Assoziation `eigeneSpiele` modelliert den Sachverhalt, dass ein Objekt der Klasse `Team` nur die Spiele verwaltet, an denen das Team beteiligt ist – entweder als Team A oder als Team B.

Ein Spiel wird hinzugefügt, indem zunächst mit zwei Referenzen auf zwei Teams des Turniers ein Objekt der Klasse `Spiel` erzeugt wird. Dieses Objekt wird der Liste des Objekts der Klasse `Turnierverwaltung` mittels der Methode `fuegeHinzu` hinzugefügt. Außerdem wird über die Methode `fuegeHinzu` der Klasse `Team` beiden Objekten mit den Referenzen `teamA` und `teamB` das Spiel zu der Liste der eigenen Spiele mit dem Bezeichner `eigeneSpiele` hinzugefügt.

Das siegende Team eines Spiels wird über den Parameter der Methode `setzeSieger` des Objekts der Klasse `Spiel` übergeben. Ist das Ergebnis eines Spiels ein Unentschieden, so wird `null` übergeben.

Teilaufgabe b)

Beim Filtern sollten zunächst die Sonderfälle betrachtet werden, dass die Parameter ungültige Werte enthalten. Tritt einer der beiden genannten Sonderfälle ein, wird null zurückgegeben, andernfalls kann das Filtern beginnen.

Für das Ermitteln der geeigneten Teams wird das Feld vollständig durchlaufen und für jedes Element geprüft, ob das aktuelle Team nicht dem übergebenen Team entspricht und die genannte Voraussetzung der Spielanzahl erfüllt. Ist dies der Fall, wird es an die später zurückgegebene Ergebnisliste angehängt.

```
private List<Team> filtereTeams(Team pTeam,
                                int pGleicheBegegnungen) {
  List<Team> ergebnis = new List<>();
  if (pTeam != null
      && pGleicheBegegnungen >= 0) {
    for (int i = 0; i < teams.length; i++) {
      Team gegner = teams[i];
      if (pTeam != gegner
          && pTeam.zaehleSpieleGegen(gegner)
             < pGleicheBegegnungen) {
        ergebnis.append(gegner);
      }
    }
  }
  return ergebnis;
}
```

Teilaufgabe c)

Es wird ein Objekt der Klasse Team mit dem Namen `TuS Wiese` zurückgegeben.

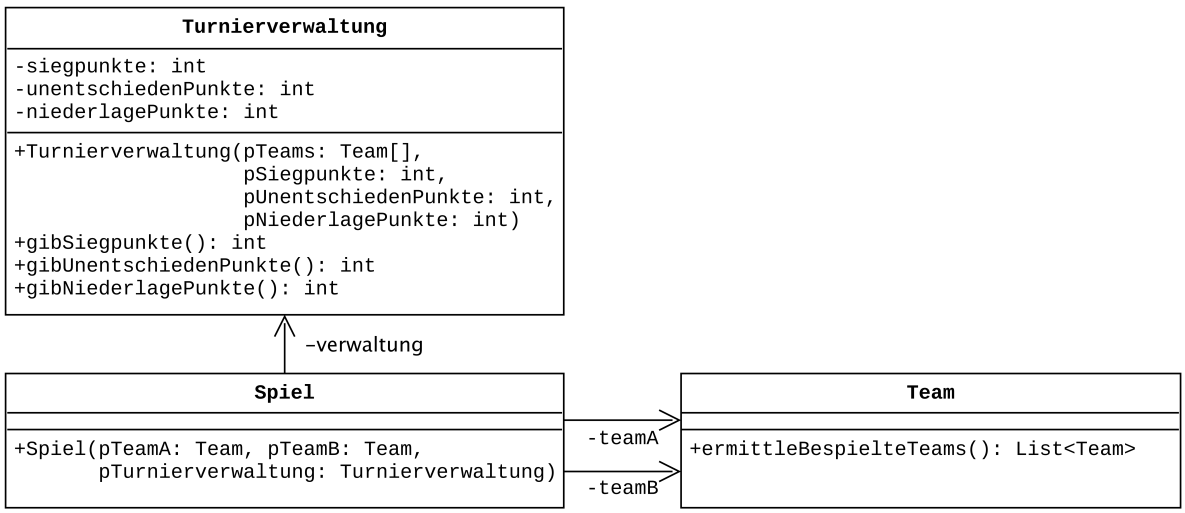
Die Methode ermittelt durch Aufruf der privaten Methode `filtere` alle Teams, die noch nicht gegen das übergebene Team `SF Forst` gespielt haben (`pX = 0`).

Sofern die Liste nicht leer ist, wird mittels eines Durchlaufs durch die Liste das Minimum des Punkteabstands zum übergebenen Team gesucht. Dazu wird zunächst das erste Element als das mit geringstem Abstand angenommen, bevor die Liste der möglichen Gegner durchlaufen wird und das Team mit der kleinsten absoluten Punktdifferenz ermittelt wird.

Die Methode ermittelt eins der Teams, die die günstigsten Gegner für das übergebene Team sind, deren Punktabstand zum übergebenen Team also am geringsten ist.

Eine Nullpointer-Exception kann in folgendem Fall auftreten:
In Zeile 7, wenn die Methode `filtere` eine leere Liste liefert, referenziert `g` (nach der Zuweisung in Zeile 6) `null`. Damit kann der Methodenaufruf von `berechnePunkte` in Zeile 7 nicht ausgeführt werden.

Teilaufgabe d)



1. Für ein Turnier kann beim Erzeugen eines Objekts der Klasse `Turnierverwaltung` über die entsprechenden Parameter der Wert der Attribute `siegpunkte`, `unentschiedenPunkte` und `niederlagePunkte` einmalig festgelegt werden und über die entsprechenden Methoden `gibSiegpunkte`, `gibUnentschiedenPunkte` und `gibNiederlagePunkte` angefragt werden. Da die Punktauswertung für die Teams in der Klasse `Spiel` vorgenommen wird, wird beim Erzeugen eines Objekts der Klasse `Spiel` eine Referenz auf die Turnierverwaltung übergeben.
2. Da jedes Objekt der Klasse `Team` die eigenen Spiele referenziert, müssen keine weiteren Daten gespeichert werden. Über die Methode `ermittleBespielteTeams` kann für jedes Objekt der Klasse `Team` ermittelt werden, gegen welche Teams bereits gespielt wurde.

Teilaufgabe e)

Vorschlag 1:

```
TuS Wiese – TV Feld            0
SV Wald – SSV Bergen           1
SF Forst – SG Heide            3
```

Vorschlag 2:

```
SF Forst – TV Feld             0
TuS Wiese – SV Wald            1
SSV Bergen – SG Heide          1
```

Beim ersten Vorschlag besteht im Gegensatz zum zweiten Vorschlag das Risiko, dass das Team mit den meisten Punkten gegen das Team mit den wenigsten Punkten spielt. Ein solches Spiel steht diametral gegen die Forderung, dass ungefähr gleich erfolgreiche Teams gegeneinander spielen.

Beim zweiten Vorschlag ist die Auswahl für jedes einzelne Team möglicherweise nicht optimal, es handelt sich jedoch stets um hinreichend ausgewogene Spiele, da einerseits eine große Punktdifferenz zwischen zwei benachbarten Teams nicht zu erwarten ist. Sollte andererseits doch eine große Punktdifferenz existieren, muss sie auf jeden Fall aufgelöst werden. Für den ersten Ansatz können dies jedoch Spiele mit einer größeren Punktdifferenz sein als beim zweiten Ansatz.

Damit ist der zweite Vorschlag als geeigneter zu bewerten.