



Artemis 6.4.0

Course Overview

ge64baw

Courses > Praktikum: Grundlagen der Programmierung WS22/23 > Exercises > W10H02 - Pinguin Schneeballschlacht

W10H02 - Pinguin Schneeballschlacht

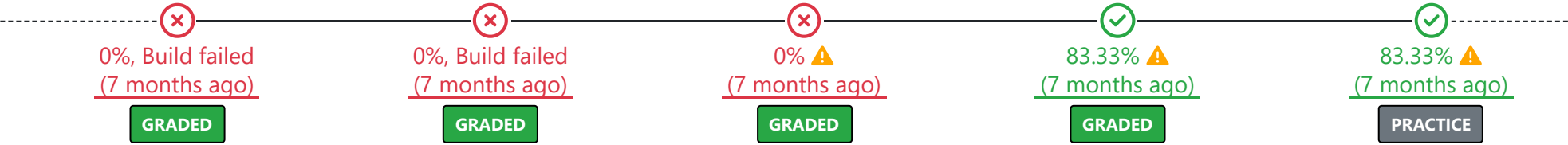
HausaufgabeMedium

Submission due: 7 months ago

Points: 5 of 6Assessment: automatic ?Complaint due: 7 months ago

83.33% (7 months ago)GRADED

Recent results:



Show all results

Tasks:

## Pinguin Schneeballschlacht

DAS sportliche Großereigniss des Jahres steht an: die Pinguin Schneeballschlacht WM. Bei diesem Ereigniss kämpfen Teams aus allen Teilen der Welt und von jeder möglichen Pinguin-Organisation um die Vorherrschaft auf dem Mt. Pingu und ein nicht unerhebliches Preisgeld.

Bei den Pinguin Schneeballschlachten ist jedes Team aus einer variablen Anzahl an Pinguinen (je nach Disziplin) aufgebaut, die sich in drei Gruppen Aufteilen: Angreifer, Verteidiger und Unterstützer. Die Angreifer versuchen ihr Team näher an die Spitze zu des Berges zu bringen, während die Verteidiger ihnen Rückendeckung vor anderen Teams unter ihnen geben. Damit Angreifern und Verteidigern die Schneebälle nicht ausgehen, werden sie von den Unterstützern begleitet, die in Windeseile einen Ball nach dem nächsten zusammendrücken.

Wie du inzwischen weißt, ist das Forschungsbudget der PUM immer sehr knapp bemessen, weshalb sie die diesjährige WM unbedingt gewinnen will. Daher haben sich die Forscher der Fakultät für internationale Wettkämpfe bereits ein System einfallen lassen, mit dem sie das Können ihrer Spieler auf jeder Position messen können. Außerdem haben sie durch genaue Beobachtungen bei den wöchentlichen Spieleabenden Ab- und Zuneigungen zwischen den Pinguinen erkannt und in einen Synergiewert umgewandelt.

Der letzte Schritt zum Erfolg fehlt aber noch. Aus den gesammelten Daten der verfügbaren Pinguine müssen für die verschiedenen Disziplinen Teams gebildet werden. Und genau für diese Aufgabe brauchen die Forscher wiederum deine Hilfe!

**Update 09.01.:** Hinweis zum Speicherverbrauch und der zugehörige Zulip Post wurden hinzugefügt.

### Aufgabe

**Hinweis:** Einschränkungen zu Eingabeparametern und weitere Hinweise zur Aufgabe findest du unter der Aufgabenbeschreibung.

Im Template erhältst du bereits die Klasse `Penguin`. Jeder `Penguin` hat natürlich einen Namen (`name`), seine Fähigkeitswerte je Position (`attack`, `defence`, `support`) und Synergiewerte zu anderen `Penguins` (`synergies`). Außer den Synergiewerten werden alle Attribute im Konstruktor initialisiert. Der Synergiewert zweier `Penguins` kann mittels der statischen Methode `setSynergy` gesetzt und mittels `getSynergy` abgefragt werden. Alle Fähigkeits- und Synergiewerte werden als Ganzzahl (`int`, bzw. `Integer`) abgespeichert und können sowohl positive, als auch negative Werte annehmen. Falls zwei `Penguins` keine spezifische Synergie aufweisen, ist ihr Synergiewert 0.

Das Template enthält außerdem die Klasse `Lineup`, die eine Teamzusammenstellung repräsentiert. Neben den `Penguins` pro Gruppe wird außerdem die Größe der Gruppe gespeichert. Zusätzlich besitzt ein `Lineup` noch Werte für die Gesamtfähigkeit (`teamSkill`), Gesamtsynergie (`teamSynergy`) und Gesamtbewertung (`teamScore`) des Teams, das dadurch repräsentiert wird. Berechnet werden diese Werte in der Methode `computeScores`, die du als erstes Implementieren musst. Deine zweite Aufgabe ist es dann, die Methode `computeOptimalLineup` zu implementieren. Diese Methode berechnet aus der gegebenen Menge aus `Penguins` und der geforderten Aufstellung (Anzahl an Angreifern, Verteidigern und Unterstützern) eine optimales `Lineup` und gibt dieses zurück.

computeScores 2 of 2 tests passing

Wie bereits erwähnt soll `computeScores` die Werte für `teamSkill`, `teamSynergy` und `teamScore` berechnen.

- `teamSkill` entspricht der Summe aller relevanten Skill-Werte der `Penguins` des `Lineups`. Ein Skill-Wert ist genau dann relevant, wenn der `Penguin` der entsprechenden Gruppe zugeteilt ist, also z.B. `attack` für einen `Penguin` in `attackers`.
- `teamSynergy` entspricht der Summe der Synergien aller `Penguin`-Paare, die mit dem `Lineup` gebildet werden können. Außerdem werden Synergien von `Penguin`-Paaren aus der gleichen Gruppe doppelt gewichtet. Besteht ein `Lineup` also z.B. aus den `Penguins` a1, a2 (beide Angreifer) und d1 (Verteidiger), dann gilt:  $teamSynergy = 2 * syn(a1,a2) + syn(a1, d1) + syn(a2, d1)$
- `teamScore` berechnet sich einfach aus der Summe von `teamSkill` und `teamSynergy`

computeOptimalLineup 2 of 2 tests passing

Wie oben beschrieben berechnet `computeOptimalLineup` aus den gegebenen Parametern ein optimales `Lineup`. Ein `Lineup` ist genau dann optimal, wenn es den mit den zur Verfügung stehenden `Penguins` und den Gruppengrößen maximal möglichen `teamScore` hat.

## Einschränkungen und Anmerkungen

- Fähigkeits- und Synergiewerte sind im Bereich [Integer.MINVALUE, Integer.MAXVALUE] möglich.
- Ein **Penguin** hat Synergien zu 0 oder mehr anderen **Penguins**. Die Synergie zu sich selbst ist immer 0.
- Die Parameter **numberXXX** der Methode **computeOptimalLineup** sind alle  $\geq 0$ .
- Die Summe der Parameter **numberXXX** der Methode **computeOptimalLineup** ist  $\leq$  der Anzahl der übergebenen **Penguins** (**players.size()**)
- Das übergebene Set **players** ist nicht modifizierbar, du kannst also nur lesend auf das Set zugreifen.
- Methoden und Konstruktoren wird nie **null** übergeben.
- Falls es mehrere optimale **Lineups** gibt, werden alle akzeptiert.
- In dieser Aufgabe wird es nie zu Integer Under- oder Overflows kommen (solange nicht unnötigt multipliziert wird).
- Die Laufzeit ist für diese Aufgabe nicht besonders wichtig. Solange das große Beispiel für **computeOptimalLineup** nicht timeoutet bist du auf der sicheren Seite. Lokal solltest du je nach PC für das Beispiel nicht viel mehr als 30 Sekunden benötigen. Beachte aber, dass die Build-Clients über begrenzten Arbeitsspeicher verfügen (siehe dazu auch [diese](#) Zulip Ankündigung). Solange die Public-Tests durchlaufen, bist du aber auch hier auf der sicheren Seite.
- Gegebene Attribute, Konstruktoren und Methodensignaturen dürfen nicht geändert werden. Hilfsmethoden und neue Attribute sind erlaubt. Letztere dürfen auch im Konstruktor initialisiert werden.

## Beispiele

**Disclaimer:** Ähnlichkeiten zu realen Pinguinen sind rein zufällig.

Die Beispiele findest du auch im Template der main. Sie werden außerdem mit den PublicTests überprüft.

▼ **computeScores:** kleines Beispiel

Beispiel:

```
Penguin jonas = new Penguin("Jonas", 10, 0, 0);
Penguin anatoly = new Penguin("Anatoly", 10, 10, 0);
Penguin julian = new Penguin("Juilan", 10, 10, 0);
Penguin simon = new Penguin("Simon", 0, 0, 10);
Penguin.setSynergy(jonas, anatoly, 10);
Penguin.setSynergy(jonas, julian, 5);

Lineup l0 = new Lineup(Set.of(jonas, anatoly), Set.of(julian), Set.of(simon));
System.out.println(l0);
```

Ausgabe:

```
Line-up: 2 - 1 - 1
Team Score: 65
Team Skill: 40
Team Synergy: 25

Attackers:
  Jonas [10,0,0]
  Anatoly [10,10,0]

Defenders:
  Juilan [10,10,0]

Supporters:
  Simon [0,0,10]
```

▼ **computeScores:** großes Beispiel

Beispiel:

```
Penguin eve = new Penguin("Eve", 9151, 5, 11);
Penguin enrico = new Penguin("Enrico", 97, 103, 3499);
Penguin hanna = new Penguin("Hanna", 6367, 331, 337);
Penguin sachmi = new Penguin("Sachmi", 103, 5701, 109);
Penguin jasmine = new Penguin("Jasmine", 233, 5737, 239);
Penguin jakob = new Penguin("Jakob", 307, 313, 3559);

Penguin.setSynergy(eve, hanna, 30);
Penguin.setSynergy(enrico, jakob, 77);
Penguin.setSynergy(sachmi, jasmine, 121);
Penguin.setSynergy(jasmine, jakob, 34);
Penguin.setSynergy(eve, sachmi, 1);

Lineup l1 = new Lineup(Set.of(eve, hanna), Set.of(sachmi, jasmine), Set.of(enrico, jakob));
System.out.println(l1);
```

Ausgabe:

```
Line-up: 2 - 2 - 2
Team Score: 34505
Team Skill: 34014
Team Synergy: 491

Attackers:
  Eve [9151,5,11]
  Hanna [6367,331,337]

Defenders:
  Sachmi [103,5701,109]
  Jasmine [233,5737,239]

Supporters:
  Enrico [97,103,3499]
  Jakob [307,313,3559]
```

▼ computeOptimalLineup: kleines Beispiel

Beispiel:

```
Penguin eric = new Penguin("Eric", 10, 0, 0);
Penguin nils = new Penguin("Nils", 10, 10, 0);
Penguin felix = new Penguin("Felix", 10, 10, 0);
Penguin thomas = new Penguin("Thomas", 0, 0, 10);

Penguin.setSynergy(eric, nils, 20);
Penguin.setSynergy(eric, felix, 5);

Lineup l2 = Lineup.computeOptimalLineup(Set.of(eric, nils, felix, thomas), 2, 1, 1);
System.out.println(l2 + "\n");
```

Ausgabe:

```
Line-up: 2 - 1 - 1
Team Score: 85
Team Skill: 40
Team Synergy: 45

Attackers:
  Nils [10,10,0]
  Eric [10,0,0]

Defenders:
  Felix [10,10,0]

Supporters:
  Thomas [0,0,10]
```

▼ computeOptimalLineup: großes Beispiel

Beispiel:

```
Penguin jan = new Penguin("Jan", -101, 177013, 777);
Penguin georg = new Penguin("Georg", 9001, -25984, 66);
Penguin anton = new Penguin("Anton", 300, 5180, -20000);
Penguin johannes = new Penguin("Johannes", 0, 314, 2792);
Penguin konrad = new Penguin("Konrad", 420, 8008, 911);
Penguin max = new Penguin("Max", 1337, -161, 69);
Penguin oliver = new Penguin("Oliver", 1, 271, 2319);
Penguin robin = new Penguin("Robin", 13, 34, 666);
Penguin laura = new Penguin("Laura", -37, 577, 1459);
Penguin lukas = new Penguin("Lukas", -79, 549, 1123);

Penguin.setSynergy(georg, max, 1137);
Penguin.setSynergy(max, oliver, 33);
Penguin.setSynergy(max, konrad, 9);
Penguin.setSynergy(georg, anton, 2187);
Penguin.setSynergy(oliver, anton, 1138);
Penguin.setSynergy(jan, lukas, 883);
Penguin.setSynergy(jan, laura, 787);
Penguin.setSynergy(johannes, oliver, 420);
Penguin.setSynergy(johannes, jan, 69);

Lineup l3 = Lineup.computeOptimalLineup(Set.of(jan, georg, anton, johannes, konrad, max, oliver, robin, laura, lukas), 2, 3, 5);

System.out.println(l3);
```

Ausgabe:

```
Line-up: 2 - 3 - 5
Team Score: 217118
Team Skill: 208898
Team Synergy: 8220


Attackers:
  Max [1337,-161,69]
  Georg [9001,-25984,66]

Defenders:
  Jan [-101,177013,777]
  Anton [300,5180,-20000]
  Konrad [420,8008,911]

Supporters:
  Lukas [-79,549,1123]
```

Exercise details

Release date:	Jan 5, 2023 18:30
Start date:	Jan 5, 2023 18:30
Submission due:	Jan 22, 2023 18:00
Complaint due:	Jan 29, 2023 18:00

Every student is allowed to complain once per exercise. In total 1000 complaints are possible in this course. You still have **998** complaints left. 

You have already submitted a complaint 7 months ago **Complaint was rejected**

Response to your complaint 7 months ago:

Ein herzliches Servus an alle :)  
nachdem ich mir die Tests angeschaut habe, habe ich tatsächlich  
bemerkt, dass die letzten Tests wegen einem sehr kleinen Fehler in  
meinem Code durchgefallen haben.

Hi Mostafa,  
Ich gebe dir Recht, dass es ein sehr kleiner Fehler war. Aber ich kann  
keine volle Punktzahl für fehlerhafte Codes geben. Von mir aus würde  
ich noch einen halben Punkt zurückgeben, aber es gibt leider nur ganze

How useful is this feedback to you?

