Praktikumsaufgaben für VW5

Programmierung interaktiver Systeme im Wintersemester 2023/24
Prof. Dr. Martin Weigel

Eis- und Elektro-Typen · **Legendäre Vögel** · **Generic Pokedex**

> Verwenden Sie für diese Praktikumsaufgaben den Code aus Übung 04.

1 Eis- und Elektro-Typen

Implementieren Sie zwei neue Interfaces `IceType extends PokemonType` und `ElectricType extends PokemonType`. Beide sollen die Stärken und Schwächen des jeweiligen Types mit einer default-Methode abbilden.

- Eis-Typen (IceType) sind stark gegen Flug- und Gras-Typen. Sie sind schwach gegen Wasser-Typen.
- Elektro-Typen (ElectricType) sind stark gegen Flug- und Wasser-Typen. Sie sind schwach gegen Gras-Typen.

Passen Sie die existierenden Interfaces in `Type.java` so an, dass diese die Stärken und Schwächen gegenüber den neuen Typen wiederspiegeln.

Fügen Sie dem Code aus der Übung zwei neue Pokemonklassen hinzu: `Glaziola implements IceType` und `Pikachu implements ElectricType`.

2 Legendäre Vögel

In der ersten Generation von Pokemon gibt es drei legendäre Vögel, welche je zwei Typen besitzen:

- *Arktos* ist ein Flug und Eis-Typ
- *Lavados* ist ein Flug und Feuer-Typ
- *Zapdos* ist ein Flug und Elektro-Typ

Implementieren Sie diese Pokemon-Klassen. Achten Sie beim Implementieren von `isWeakAgainst(Pokemon other)` darauf, dass die legendären Vögel keine Schwächen gegenüber der eigenen Typen besitzen. Zum Beispiel hat Zapdos trotz des Flug-Typs keine Schwäche gegenüber Elektro-Pokemon. Die legendären Vögel können aber Stärken gegen andere Pokemon vom gleichen Typ besitzen.

3 Generic Pokedex

Generische Klassen erlauben das Implementieren von einer Funktionalität für verschiedene Dateitypen. In dieser Aufgabe sollen Sie einen `Pokedex<T>` implementieren. Ein Pokedex ist eine Sammlung an Pokemon, welche ein Pokemontrainer besitzt. Sie sollen einen Pokedex mit Tausch-Funktionen für ein Spiel entwickeln. Da diese Funktionalität in vielen Spielen gebraucht wird (z.B. einem Pokemon-Kartenspiel oder Pokemon-Snap), soll die Klasse generisch implementiert werden, um mit verschiedensten Datentypen nutzbar zu sein.

1. Erstellen Sie ein Interface „Nameable“. Dieses soll eine Methode besitzen, welche einen String mit einem eindeutigen Namen der Klasse zurück gibt. Implementieren Sie dieses Interface in der Pokemon-Klasse mit `return name;`.

2. Erstellen Sie eine Klasse `Pokedex<T extends Nameable>`. Implementieren Sie die folgenden generischen Methoden:
   - `void add(T nameable)` fügt das übergeben Objekt dem Pokedex hinzu, falls es noch nicht vorhanden ist.
   - `void swap(String name, Pokedex<T> other, String otherName)` tauscht zwei Objekte. Das Objekt mit dem Namen `name` soll in den `other`-Pokedex verschoben werden. Das Objekt das mit dem Namen `otherName` im Pokedex `other` soll in den eigenen Pokedex (d.h. `this`) verschoben werden.
   - `Set<T> getUniqueObjectsOf(Pokedex<T> other)` gibt ein Set mit allen Objekten aus `other` aus, welche nicht im eigenen Pokedex vorhanden sind.

> ⓘ Sie dürfen innerhalb von `Pokedex` die Objekte in einer Collection oder Map speichern.

Erstellen Sie zum Testen eine Main-Methode, welche zwei `Pokedex<Pokemon>`-Objekte erstellt und mit Inhalten füllt. Zeigen Sie mithilfe eines Code-Beispieles, dass Ihre implementierten Methoden funktionieren. Sie können dazu die `toString()`-Funktion ähnlich wie in der `Team`-Klasse umschreiben.