Übungen: Entwurf und Entwurfsmuster

AUFGABEN MIT LÖSUNG

Aufgabe 1 (UML-Klassendiagramm)

Entwerfe ein UML-Klassendiagramm ohne Operationen, das für folgendes Anwendungsbeispiel als konzeptionelles Datenmodell dient: Es soll ein Mietwagenverleih verwaltet werden.

- Es werden Automobile unterschiedlicher Typen an verschiedenen Mietstationen an Kunden vermietet.
- Zu jedem Automobil werden Kennzeichen, Laufleistung sowie der Typ erfasst.
- Je nach Automobiltyp gibt es Unterschiede in der Kraftstoffart, Leistung (PS) und Höchstgeschwindigkeit. Spezielle Typen sind PKW und LKW, wobei nur für PKW die Anzahl der Sitzplätze und die Getriebeart (automatische oder manuelle Schaltung) erfasst werden. Für LKW werden dagegen das Laderaumvolumen und die Nutzlast angegeben. Spezielle PKW sind Cabrios, Vans und Geländewagen.
- Zu einem Kunden wird dessen Name, Geburtsdatum und als Teil des Kunden eine Anschrift erfasst. Eine Anschrift wird je Kunde gespeichert, d.h. mehrere Kunden teilen sich nicht eine Anschrift. Die Anschrift besteht aus Straße, PLZ, Ort und Land. Zusätzlich wird festgehalten, ob es sich um einen Premiumkunden handelt.
- Ein Kunde kann beliebig viele Automobile mieten. Zum Zeitpunkt der Anmietung wird in der Ausleihe zunächst nur der Automobiltyp erfasst und erst bei der Abholung ein konkretes Automobil. Ein Automobil wird im Laufe der Zeit natürlich mehrfach verliehen.
- Zusätzlich werden zu einer Ausleihe der vereinbarte Preis sowie die Daten und Orte für Abholung und Rückgabe gespeichert.
- Die Orte sind dabei Mietstationen, zu denen je eine Bezeichnung und eine Anschrift (s. oben) festgehalten werden, sowie eine Information darüber, ob es sich um eine Flughafenstation handelt.
- Zu einer Mietstation wird auch erfasst, welche Automobiltypen hier grundsätzlich angeboten werden und welche Automobile gerade am Standort sind. Ein Automobil kann sich max. an einer Mietstation zurzeit befinden (oder an keiner, wenn es aktuell verliehen ist).
- Zu einer Ausleihe wird zusätzlich erfragt, ob der Kunde ein Navigationssystem wünscht. Falls im angemieteten Automobil keines verbaut ist wird vor der Übergabe ein mobiles Navigationssystem installiert.

Aufgabe 2 (UML-Klassendiagramm)

Entwerfe ein UML-Klassendiagramm, das für folgendes Anwendungsbeispiel als konzeptionelles Datenmodell dient: Es sollen Snowboarder verwaltet werden, die über mehrere Jahre an verschiedenen Wettbewerben teilnehmen:

Datenstrukturen (Klassen, Attribute, Assoziationen, Vererbung):

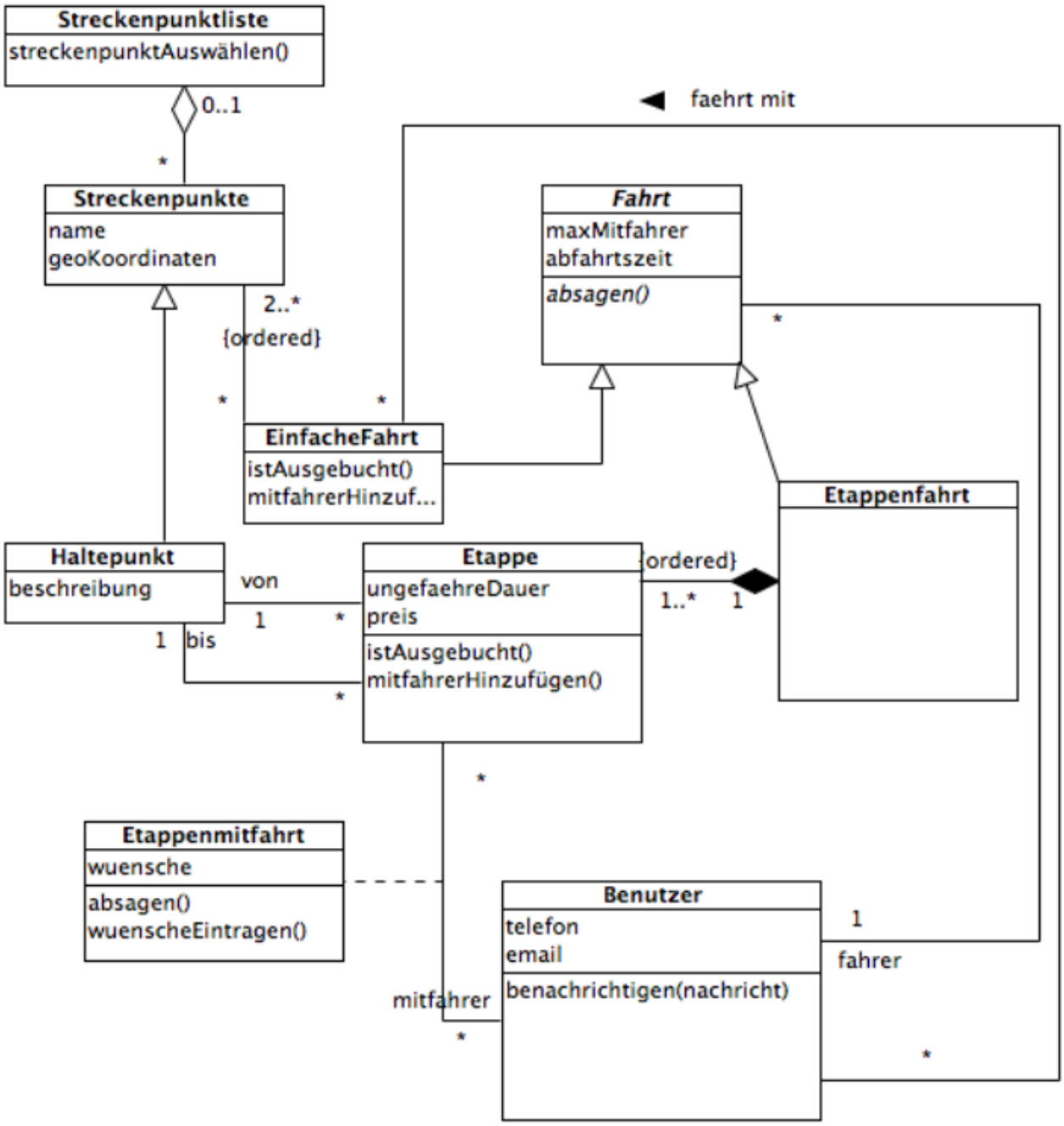
- Snowboarder haben einen Namen, einen Vornamen und einen Geburtstag. Jedem ist eine eindeutige Person-ID zugeordnet. Zusätzlich soll für jeden das heimatliche Skigebiet gespeichert werden.
- Unter den Snowboardern gibt es Pros. Diese haben eine eigene Lizenznummer. Für diese werden zusätzlich ihr aktueller "Best-Trick" und die Weltcup-Punkte je Saison gespeichert.
- Pros haben Sponsoren. Diese werden mit Namen und dem zur Verfügung stehenden Budget pro Saison gespeichert. Der Geldbetrag, mit der ein Pro von einem Sponsor für eine Saison gefördert wird, soll ebenfalls festgehalten werden.
- Die Sponsoren sind gleichzeitig auch die Veranstalter der Wettkämpfe. Dabei wird jeder Wettkampf von genau einem Sponsor ausgetragen. Wettkämpfe werden mit ihrem Namen und der jeweiligen Saison identifiziert. Zusätzlich soll die Höhe des Preisgeldes und u.U. die erforderliche Anzahl von Teilnehmern angegeben werden.
- Es soll aufgenommen werden, welcher Snowboarder sich bei welchem Wettkampf mit einer bestimmten Punktzahl für welchen anderen Wettkampf qualifiziert hat.

Verhalten (Operationen):

- Ein Wettkampf stellt eine Methode zur Anmeldung für einen Snowboarder bereit, deren Rückgabe ein boolscher Wert ist. Die erfolgreich angemeldeten Teilnehmer werden in einer Liste am Wettkampf gespeichert.
- Ein Sponsor kann einen Pro für eine Saison mit einem bestimmten Geldbetrag fördern.

Aufgabe 3 (UML-Klassendiagramm)

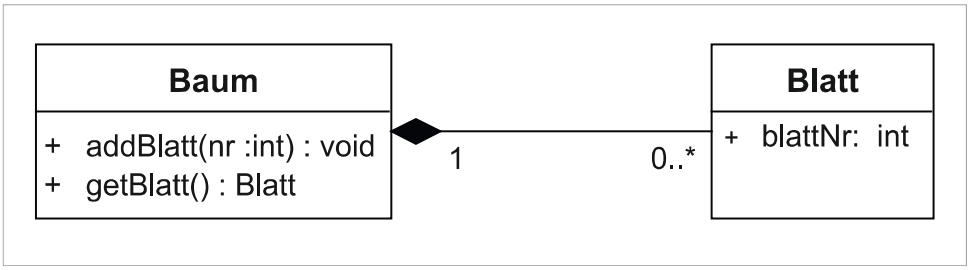
Beschreiben Sie das folgende Klassendiagramm.



Aufgabe 4 (UML-Klassendiagramm)

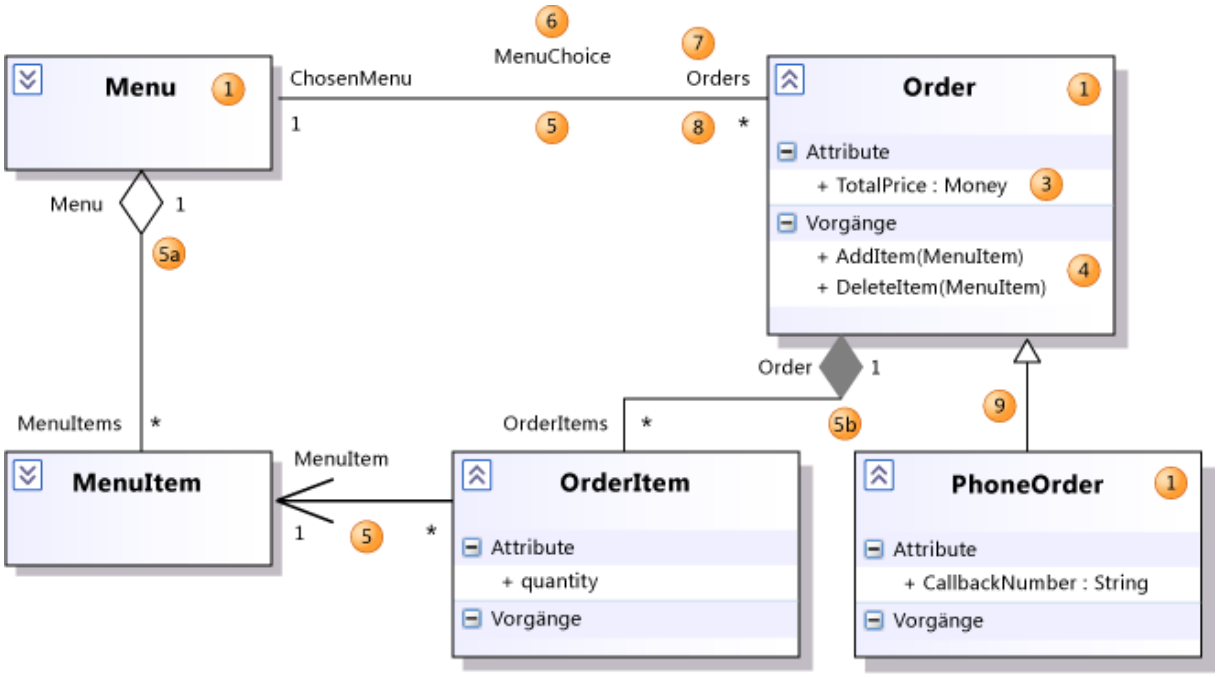
Setzen Sie folgenden Ausschnitt aus einem Klassendiagramm in JAVA um.

Was würde sich im Falle einer Aggregation ändern?



Aufgabe 5 (UML-Klassendiagramm)

Setzen Sie folgenden Ausschnitt aus einem Klassendiagramm in JAVA um.



Aufgabe 6 (Entwurfsmuster - Adapter)

Bestehendes Interface:

```
interface Shape {
    public int calculateArea(int r);
}
```

Eine genutzte (eigene) Implementierung:

```
class Square implements Shape {
    @Override
    public int calculateArea(int r) {
        return r * r;
    }
}
```

Wir wollen nun weiterhin die Circle Klasse als 3rd-party-Software nutzen:

```
class Circle {
    public double calculateCircularArea (int r) {
        return 3.14 * r * r;
    }
}
```

1. Schreiben Sie eine Adapterklasse `CircleToShapeAdaptor`
    a. als Klassenadapter
    b. als Objektadapter

    und eine entsprechende Main-Methode, die Objekte der Klasse `Square` und `CircleToShapeAdaptor` sinnvoll verwendet.

2. Warum muss eine solche Adapter-Klasse erstellt werden und kann die Klasse `Circle` nicht direkt verwendet werden?

3. Benennen Sie Komponenten korrekt mit: *Adapter, Dienst, Ziel, Klient*.

4. Konstruieren Sie das entsprechende UML-Klassendiagramm.

Aufgabe 7 (Entwurfsmuster – Composite)

Für die Darstellung eines Dateimanagers gilt folgendes: Verzeichnisse und ZIP-Dateien sind Dateien, die Dateien enthalten können. Einfache Dateien enthalten keine anderen Dateien und sind so genannte „Blätter“. Eine Datei soll in der Lage sein, ihre Größe zurückzugeben. Außerdem soll man eine Datei löschen können. Modellieren Sie die beschriebene Struktur in einem Klassendiagramm unter Verwendung des geeigneten Entwurfsmusters *Composite*. Erstellen Sie des Weiteren eine entsprechende JAVA-Implementierung mit einem Beispiel-Client.

Aufgabe 8 (Entwurfsmuster – Decorator)

Für unten beschriebenes Szenario ist nach dem Entwurfsmuster *Decorator* Folgendes zu erstellen:

a. Das entsprechende UML-Klassendiagramm
b. Die entsprechende JAVA-Implementierung

Basis-Szenario

Ein Café möchte seine unterschiedlichen Kaffeebestellungen, bspw.

- Coffee „Guatemala“
- Coffee „Own-brand", milk
- Coffee „Brazil", milk, mocha, cream

modellieren. Es soll die Möglichkeit bestehen, alle Zutaten (*milk, mocha, cream*) mit allen Kaffeesorten (= Beverage) *Guatemala, Own-Brand, Brazil* zu kombinieren. Der Gesamtpreis und die Beschreibung sollen via Methoden `printDescription()` und `getPrice()` abgefragt werden. Preise sind wie folgt:

milk = 0.3 Euro; mocha = 0.6 Euro; cream = 0.5 Euro*;* Coffee „Guatemala“ = 2.5 Euro; Coffee „Brazil“ = 3.0 Euro; Coffee „Own-Brand“ = 2.0 Euro

Zusatzaufgaben

1. Erweitern Sie das oben genannte Szenario wie folgt: Für die Zutat *cream* soll es optional ein *topping* geben. Die Klasse `Cream` ergänzt somit seine Superklasse um einen zusätzlichen Zustand *topping* sowie um eine zusätzliche Methode `getToppingPrice(...)`. Letztere gibt den Preis für die möglichen Toppings zurück (*cherry = 0.2 Euro, cinemon = 0.1 Euro, sirup = 0.15 Euro)*. Die Aus- bzw. Rückgabe der geerbten Methoden `printDescription()` und `getPrice()` der Klasse `Cream` soll entsprechend abhängig vom Zustand *topping* erfolgen.
2. Wie würde eine Implementierung ohne das Entwurfsmuster *Decorator* aussehen? Gegeben seien bspw. die oben genannten Kombinationen (1) – (3)?

Aufgabe 9 (Entwurfsmuster – Observer)

Gegeben sei folgendes Szenario: Für einen Verlag sollen Abonnenten ab- und angemeldet werden können. Der FAZ-Verlag versendet die jeweils aktuelle Zeitung an seine Abonnenten Familie Fischer und Familie Meier. Bei jeder neuen Zeitung soll eine Benachrichtigung aller Abonnenten erfolgen. Entwerfen Sie für dieses Szenario das entsprechende UML-Klassendiagramm nach dem Entwurfsmuster Observer und implementieren Sie die entsprechenden Klassen und einen Beispielclient in JAVA. Verwenden Sie dabei

a. das Push-Modell
b. das Pull-Modell

Hinweis: Modellieren Sie die aktuelle Zeitung – den Subjekt-Zustand – hier mit Hilfe einer separaten Klasse Zeitung.

Aufgabe 10 (Entwurfsmuster – Abstract Factory)

Implementieren Sie die JAVA Klassen für folgendes Beispiel:

Spielesammlung

- Abstraktes Produkt 1*: Spielbrett*
  Konkrete, abgeleitete Produkte sind Mühlebrett, Damebrett, Halmabrett, …

- Abstraktes Produkt 2: Spielfigur
  Konkrete, abgeleitete Produkte sind Holzstein, Hütchen, …

- Abstrakte Fabrik
  Spielfabrik, die zusammengehörige Teile (Spielbrett und Spielfiguren) eines Gesellschaftsspiels erstellt. Konkrete, davon abgeleitete Fabriken sind *Mühlefabrik*, *Damefabrik*, *Halmafabrik*.

Was müssten Sie ändern, um ein neues Spielbrett (bspw. *Go*) mit neuen Spielsteinen (*PlastiksteinFlach*) einzuführen?

AUFGABEN OHNE LÖSUNG

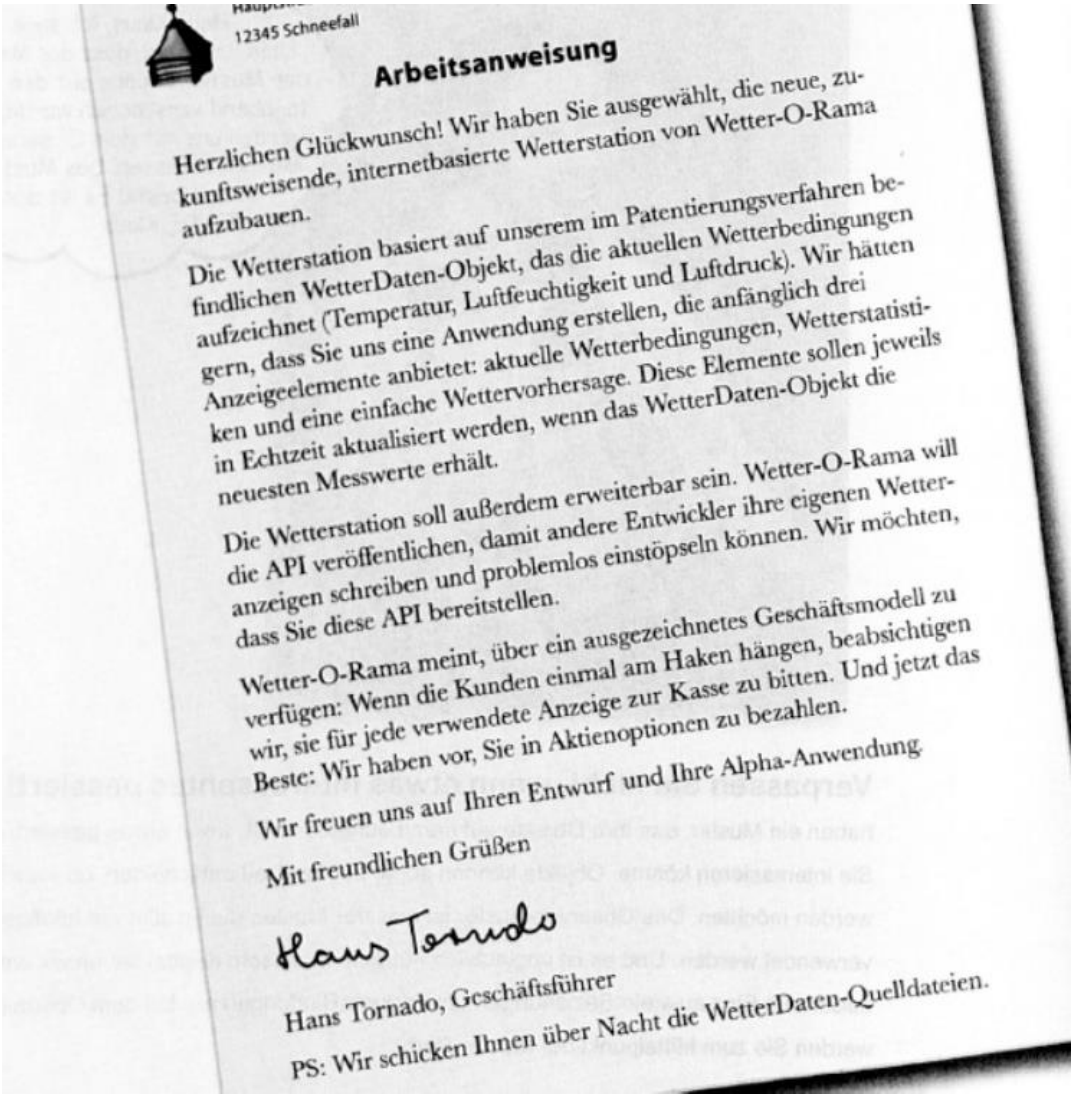
Aufgabe 1 (Entwurfsmuster – Abstract Factory)

Frau Meier möchte gerne einen Weihnachtsbaum aufstellen. Dieser verfügt üblicherweise über eine Beleuchtung (Methode `getBeleuchtung`) und über Weihnachtsbaumschmuck (Methode `getBaumschmuck`). Da Frau Meier eine auf Sicherheit bedachte Dame ist, setzt sie Kerzen ausschließlich in Kombination mit Weihnachtsbaumkugeln ein. Lametta verziert dem hingegen nur mit einer elektronischen Lichterkette den Weihnachtsbaum.

Erstelle entsprechend dem Erzeugungsmuster *Abstract Factory* ein UML Klassendiagramm.

Aufgabe 2 (Entwurfsmuster – Observer)

Entwerfe ein UML Klassendiagramm unter Verwendung des Observer Entwurfsmusters für folgenden Sachverhalt:



12345 Schneefall

Arbeitsanweisung

Herzlichen Glückwunsch! Wir haben Sie ausgewählt, die neue, zukunftsweisende, internetbasierte Wetterstation von Wetter-O-Rama aufzubauen.

Die Wetterstation basiert auf unserem im Patentierungsverfahren befindlichen WetterDaten-Objekt, das die aktuellen Wetterbedingungen aufzeichnet (Temperatur, Luftfeuchtigkeit und Luftdruck). Wir hätten gern, dass Sie uns eine Anwendung erstellen, die anfänglich drei Anzeigeelemente anbietet: aktuelle Wetterbedingungen, Wetterstatistiken und eine einfache Wettervorhersage. Diese Elemente sollen jeweils in Echtzeit aktualisiert werden, wenn das WetterDaten-Objekt die neuesten Messwerte erhält.

Die Wetterstation soll außerdem erweiterbar sein. Wetter-O-Rama will die API veröffentlichen, damit andere Entwickler ihre eigenen Wetteranzeigen schreiben und problemlos einstöpseln können. Wir möchten, dass Sie diese API bereitstellen.

Wetter-O-Rama meint, über ein ausgezeichnetes Geschäftsmodell zu verfügen: Wenn die Kunden einmal am Haken hängen, beabsichtigen wir, sie für jede verwendete Anzeige zur Kasse zu bitten. Und jetzt das Beste: Wir haben vor, Sie in Aktienoptionen zu bezahlen.

Wir freuen uns auf Ihren Entwurf und Ihre Alpha-Anwendung.

Mit freundlichen Grüßen

Hans Tornado

Hans Tornado, Geschäftsführer

PS: Wir schicken Ihnen über Nacht die WetterDaten-Quelldateien.

Die Beobachter-Klassen sind neben den möglichen Drittanbieter-Anzeigen zunächst die Anzeigen für die aktuellen Wetterbedingungen, die Wetterstatistik und die Wettervorhersage.

Implementiere die Klassen in Java unter Verwendung von `java.util.Observable` und `java.util.Observer`.