Eine Kraftwerksklasse, ihre Implementierung und Instantiierung

3. Praktikumstermin, X-Gruppe

Praktika sind Prüfungsvorleistungen, die zum genannten Termin erbracht werden müssen. Bei Verhinderung durch Krankheit ist eine ärztliche Bescheinigung der Arbeitsunfähigkeit vorzulegen. Dieses Deckblatt ist zum Praktikumstermin ausgefüllt mitzubringen und unterschrieben abzugeben. Das Programm (C++-Projekt) wird während des Praktikums zum angegebenen Zeitpunkt im Informatik-Labor abgenommen.

Name	Vorname	Matrikelnummer	Ihre Unterschrift

Zu Beginn dieses Praktikumtermins müssen Sie Basiskenntnisse zur Thematik „Klassen, Objekte, Vererbung“ nachweisen und den prinzipiellen Lösungsweg erläutern können.

Ziel

Das Programm des zweiten Praktikums soll umstrukturiert werden. Von der Basisklasse (oder Elternklasse) `cKraftwerk` wird die Klasse `cWindkraftanlage` als Kindklasse (Spezialisierung der Basisklasse) abgeleitet. Durch die Ableitung einer Klasse entsteht eine spezialisierte Klasse, welche (mit Einschränkungen) die Elementvariablen und Elementfunktionen der Basisklasse erbt, aber darüber hinaus zusätzliche Eigenschaften und Funktionalitäten besitzen, welche die Basisklasse nicht hat.

Einige Elementvariablen („Eigenschaften“ oder „Attribute“) und einige Elementfunktionen („Methoden“) der früheren Kraftwerksklasse werden jetzt alleiniger Bestandteil der neuen, abgeleiteten Klasse beziehungsweise eines Unterobjekts davon. Die reale Windkraftanlage setzt sich selbst wiederum aus anderen realen Objekten zusammen, zum Beispiel einem Generator, einem Rotor und einem Wechselrichter. Im Bezug auf unsere Klassenbeziehungen – siehe Abbildung 1 – stammen diese Objekte natürlich wieder aus eigenen Klassen. Sie sind eingebettete Objekte und damit Elemente (Member) der Klasse `cWindkraftanlage`. Diese Beziehung ist allerdings keine Vererbung („Eine Windkraftanlage *ist* ein Kraftwerk.“), sondern eine Member-Beziehung („Eine Windkraftanlage *hat* einen Generator.“).

Welche weiteren Member-Objekte könnte eine Windkraftanlage haben?

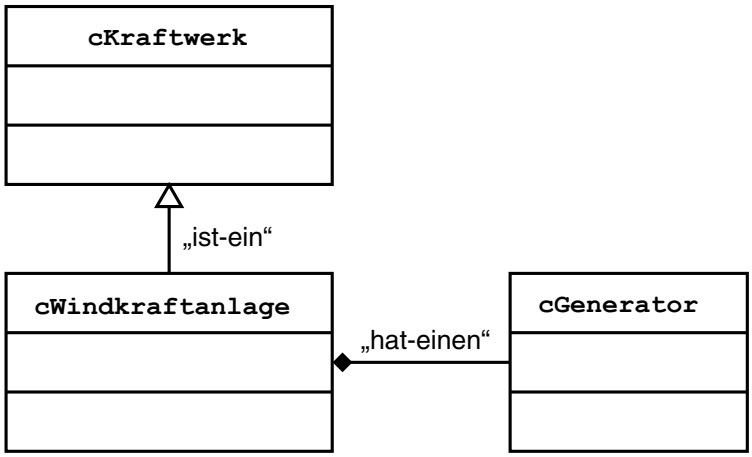

Abbildung 1: Beziehungen zwischen Klassen

Wäre es denkbar, dass Member-Objekte selbst wieder aus Objekten zusammengesetzt sind? Nennen Sie ein Beispiel!

Das Programm ist als Konsolen-Anwendung mit der Entwicklungsumgebung Turbo C++ zu realisieren.

main-Funktion

In `main` sollen einige wenige Daten einer einzigen Windkraftanlage in das Programm eingegeben, verarbeitet und in modifizierter Form wieder ausgegeben werden:

```
int main(void)
{
  //Variablen für das Einlesen der Daten der Windkraftanlage:
  //Zeichenketten: standort
  //Gleitkomma-Variablen: leistung, kosten, co2

  //Ausgabe auf cout: "Eingabe von Parametern einer Windkraftanlage"
  //Eingabe von cin: typ, ort, baukosten, co2

  //Anlegen einer Objektvariablen der Klasse cKraftwerk
  //Ausgabe der Objekteigenschaften: Aufruf der Elementfunktion gibAus

  //Anlegen einer Objektvariablen der Klasse cWindkraftanlage
  //ein Jahr lang 0,5 MW produzieren
  //Ausgabe der Objekteigenschaften: Aufruf der Elementfunktion gibAus

  system("Pause");
  return 0;
}//main
```

Klassen

Die Klassen und ihre Elemente sehen in vereinfachter Darstellung so aus (zum leichteren Verständnis hier wie in C++ notiert, nicht völlig gemäß dem UML-Standard):

cKraftwerk
(-) string typ; (-) string ort;
(+) cKraftwerk(string typ_, string ort_); (+) string getStandort(void); (+) void gibAus(void);

cWindkraftanlage

(-) double baukosten; //in EUR
(-) double co2; //CO2-Verursachung beim Bau in kg
(-) double energie; //bislang eingespeiste Energie in kWh
(-) cGenerator generator; //Member-Objekt

(+) cWindkraftanlage(...<Welche Parameter sind sinnvoll?>...);
(+) void gibAus(void);
(+) void produziere(double leistung, double dauer);
//weitere Energie einspeisen [kW, h]
(+) double co2Bilanz(void); //Gramm CO2 pro kWh

cGenerator

(-) double maxP_MW; //max. Leistung in MW
(-) double aktP_MW; //aktuelle Leistung in MW
(-) int drehzahl; //UpM gerundet auf ganze Zahl

(+) cGenerator(double maxP_MW_, double aktP_MW_=0.0, int drehzahl_=0); (+) double getMaxP(void);

Elementfunktionen

Alle Elementfunktionen sollen in den jeweiligen .cpp-Dateien implementiert werden, Nur `getStandort` soll im Klassenrumpf innerhalb der Header-Datei implementiert werden.

Klasse `cKraftwerk`

- Der Konstruktor initialisiert alle Elementvariablen.
- `getStandort` liefert die Elementvariable `ort` zurück.
- `gibAus` schreibt die Elementvariablen auf `cout` und benutzt dazu `getStandort`.

Klasse `cWindkraftanlage`

- Der Konstruktor initialisiert alle Elementvariablen einschließlich des Memberobjekts und ruft den Konstruktor der Basisklasse auf. Initialisierliste!
- `produziere` erhöht den Wert von `energie` um `leistung` mal `dauer`.
- `co2Bilanz` gibt `1000.0*co2/energie` zurück.

- `gibAus` schreibt den Standort, die Maximalleistung, die CO_2-Bilanz und die Kosten der bisher eingespeisten Energie [EUR/kWh] auf `cout`.

Klasse `cGenerator`

- Der Konstruktor initialisiert alle Elementvariablen. Default-Parameter!
- `getMaxP` gibt `maxP_MW` zurück.

Strukturierung des Quellcodes

Legen Sie für jede der drei Klassen getrennte „Units“ aus .cpp- und .h-Dateien an. Denken Sie an die nötigen `#include`-Befehle. Insbesondere muss `Windkraftanlage.cpp` sowohl `"Kraftwerk.h"` wie auch `"Generator.h"` inkludieren.

Hinweise

Daten für Kraftwerke finden Sie zum Beispiel in der ersten Praktikumsaufgabe. Die Baukosten für das dort angeführte Windkraftwerk können Sie mit $2{,}8 \cdot 10^6$ EUR und die CO_2-Emissionen für den Bau mit $4 \cdot 10^6$ kg ansetzen.