OPR Praktikum
Übung: Alarm

Thomas Mahr

17. Dezember 2019

1 Lernziele

- Abstrakte Funktionen verwenden
- Schnittstellen verwenden
- Dynamische Speicherverwaltung verwenden

2 Voraussetzungen

Kapitel *Polymorphie*.

3 Aufgabe: Alarm

1. Implementieren Sie die in Abb. 1 gezeigten Klassen, so dass die vorgegebene `main()`-Funktion die unten gezeigte Ausgabe liefert.
2. Ergänzen Sie die Datei *main.cpp* um die in der `main()`-Funktionen verwendeten Funktionen `alarmFuerAlleAnzeigen` und `alarmFuerAlleZuruecksetzen` zur Anzeige bzw. Zurücksetzung des Alarmzustands für alle im `vector` übergebenen alarmierbaren Objekte.
3. Ergänzen Sie die Datei *main.cpp* um die notwendigen **`#include`**-Anweisungen.

Anforderungen an die Implementierung:

1. Vermeiden Sie globale Variablen.
2. Teilen Sie den Code der Klassen `BewegungsDetektor`, `DetektorImpl`, `Flutlicht`, `GeraeuschDetektor` und `Sirene` auf jeweils eigene *.h*- und *.cpp*-Dateien auf.
3. Verwenden Sie, falls möglich, Vorwärtsdeklarationen anstelle von **`#include`**-Anweisungen.
4. Verzichten Sie auf **`using namespace`** `std;` in den *.h*-Dateien. Greifen Sie stattdessen über `std::` auf Elemente des Namensraums `std` zu.
5. Sie dürfen die `main()`-Funktion nicht ändern.

Vorgegebene `main()`−`Funktion`:

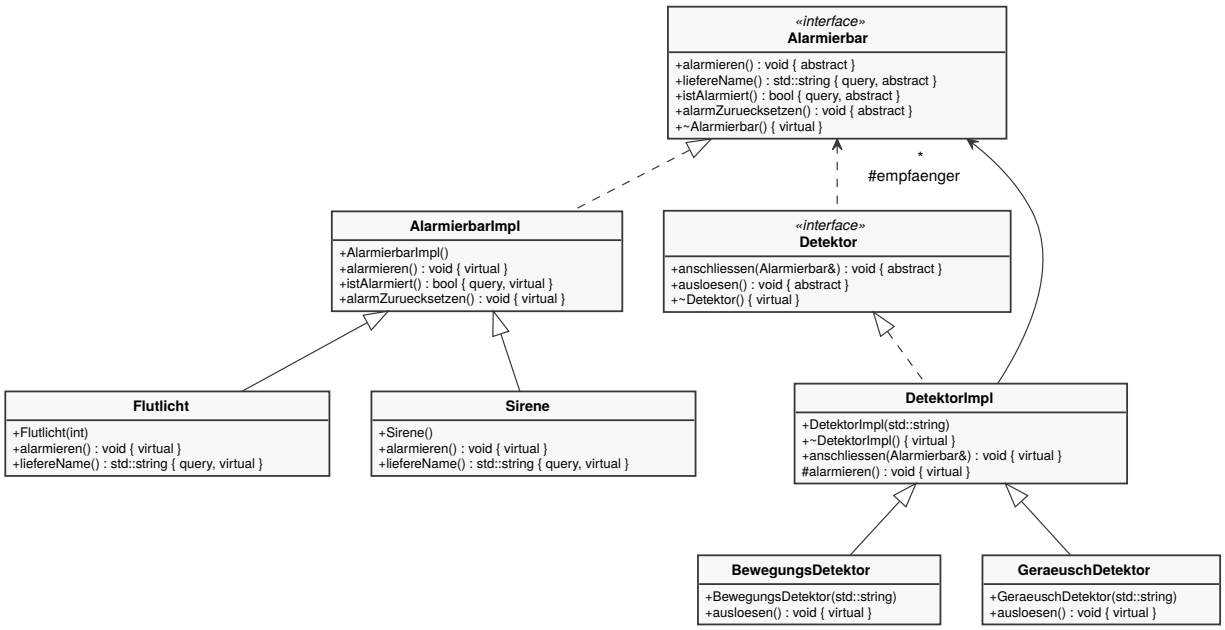

Abbildung 1: UML-Klassendiagramm

```
1  int main() {
2    cout << "*** Alarmsystem aufbauen\n";
3
4    vector<Alarmierbar*> vectorAlarmierbar;
5    Detektor* detektor[2];
6
7    detektor[0] = new BewegungsDetektor("BD1");
8
9    Sirene sirene;
10   vectorAlarmierbar.push_back(&sirene);
11   detektor[0]->anschliessen(sirene);
12
13   int lumen = 5000;
14   const int N_FLUTLICHT = 3;
15   Flutlicht* flutlicht[N_FLUTLICHT];
16   for(int i=0; i<N_FLUTLICHT; i++) {
17     flutlicht[i] = new Flutlicht(lumen*(i+1));
18     vectorAlarmierbar.push_back(flutlicht[i]);
19     detektor[0]->anschliessen(*flutlicht[i]);
20   }
21   detektor[1] = new GeraeuschDetektor("GD1");
22   detektor[1]->anschliessen(sirene);
23   detektor[1]->anschliessen(*flutlicht[1]);
24
25   cout << "*** Alarmsystem testen\n";
26   detektor[0]->ausloesen();
27   alarmFuerAlleAnzeigen(&vectorAlarmierbar);
28   alarmFuerAlleZuruecksetzen(&vectorAlarmierbar);
29   alarmFuerAlleAnzeigen(&vectorAlarmierbar);
```

```
30    detektor[1]->ausloesen();
31    alarmFuerAlleAnzeigen(&vectorAlarmierbar);
32
33    cout << "*** Alarmsystem abbauen\n";
34    delete detektor[0];
35    delete detektor[1];
36    for(auto f : flutlicht) {
37      delete f;
38    }
39  }
```

Listing 1: main.cpp

Das Programm muss diese Ausgabe liefern:

```
*** Alarmsystem aufbauen
Detektor BD1 aufbauen
Sirene Nr. 1 an BD1 anschliessen
Flutlicht Nr. 1 an BD1 anschliessen
Flutlicht Nr. 2 an BD1 anschliessen
Flutlicht Nr. 3 an BD1 anschliessen
Detektor GD1 aufbauen
Sirene Nr. 1 an GD1 anschliessen
Flutlicht Nr. 2 an GD1 anschliessen
*** Alarmsystem testen
BewegungsDetektor BD1 detektiert Bewegung
Sirene Nr. 1 geht an (heul)
Flutlicht Nr. 1 erstrahlt mit 5000 Lumen
Flutlicht Nr. 2 erstrahlt mit 10000 Lumen
Flutlicht Nr. 3 erstrahlt mit 15000 Lumen
Sirene Nr. 1: Alarm ist an
Flutlicht Nr. 1: Alarm ist an
Flutlicht Nr. 2: Alarm ist an
Flutlicht Nr. 3: Alarm ist an
Sirene Nr. 1: Alarm ist aus
Flutlicht Nr. 1: Alarm ist aus
Flutlicht Nr. 2: Alarm ist aus
Flutlicht Nr. 3: Alarm ist aus
GeraeuschDetektor GD1 detektiert Geraeusch
Sirene Nr. 1 geht an (heul)
Flutlicht Nr. 2 erstrahlt mit 10000 Lumen
Sirene Nr. 1: Alarm ist an
Flutlicht Nr. 1: Alarm ist aus
Flutlicht Nr. 2: Alarm ist an
Flutlicht Nr. 3: Alarm ist aus
*** Alarmsystem abbauen
Detektor BD1 abbauen
Sirene Nr. 1 abmelden
Flutlicht Nr. 1 abmelden
Flutlicht Nr. 2 abmelden
Flutlicht Nr. 3 abmelden
Detektor GD1 abbauen
Sirene Nr. 1 abmelden
Flutlicht Nr. 2 abmelden
```