SEN-Übung 2.4

Schett Matthias

13. April 2013

1. Aufgabe 1

1.1. Lösungsidee

Die Klasse WeatherStation soll eine Wetterstation abbilden.
Es gibt 3 Member: mName, mCelsius, mHumidity.
Weiters werden folgende Methoden verwendet:

1. WeatherStation(std::string const &name=, double celsius=0, double humidity=0);
2. std::string const &GetName() const;
3. void SetName(std::string const &name);
4. double GetCelsius() const;
5. void SetCelsius(double c);
6. double GetFahrenheit() const;
7. void SetFahrenheit(double f);
8. double GetHumidity() const;
9. void SetHumidity(double h);
10. void Print(std::ostream &out) const;

ad 1 Der Konstruktor nimmt die Werte für die 3 Member an und weist sie mittels der Elementinialierliste zu. Sollte nichts angegeben werden, gibt es Defaultwerte.

ad 2 GetName() ist ein einfacher Getter und gibt mName zurück.

ad 3 SetName() setzt mName auf name.

ad 4 GetCelsisus() ist ein einfacher Getter der mCelsius zurückgibt.

ad 5 SetCelsius() setzt mCelsius auf den Wert von c.

ad 6 GetFahrenheit() gibt mCelsius in Fahrenheit umgerechnet zurück. Zur Berrechnung der Fahrenheit wird folgende Formel verwendet:

$$T_F = T_C * 1.8 + 32$$

ad 7 SetFahrenheit() setzt mCelsius auf den in Celsisus umgerechneten Wert von f. Zur Berechnung wird folgende Formel verwendet:

$$T_C = \frac{(T_F - 32)}{1.8}$$

ad 8 GetHumidity() gibt den Wert von mHumidity zurück.

ad 9 SetHumidity setzt den Wert von mHumidity auf den Wert von h.

ad 10 Print() kümmert sich um die Ausgabe. Diese gibt die Werte der drei Member auf dem Übergebenen std::ostream aus.

1.2. Code

Der Code befindet sich im Anhang unter Wetterstation Header und Wetterstation Implementierung. Der Testreiber ist im Anhang unter Testreiber A1 zu finden.

1.3. Testfälle

Der Testreiber testet die Funktionnalität indem er zuerst ein Objekt erstellt und anschließend sämtliche Methoden aufruft.

```
Name: Erste Wetterstation Temperatur: 32.3 Luftfeuchtigkeit:
    28.3
Getter und Setter Test
Setting Name to Hans
GetName = HansSetting Temperature to 18.3C
GetTemp = 18.3
Setting Humidity to 78
GetHumid = 78
Setting Temperature to 18.3F
GetTemp in F = 18.3
GetTemp in C = -7.61111
```

2. Aufgabe 2

2.1. Lösungsidee

Die Klasse Weatherstations bildet eine Verwaltung für Wetterstationen ab, dazu gibt es die Member:
mMaxNumber (die maximale Anzahl an Wetterstationen), mNumberOfStations (die aktuelle Anzahl an Wetterstationen) und ein dynamischen Array für Wetterstationen mStationArray.
Weiters gibt es noch folgende Funktionen:

1. WeatherStation & searchForColdest() const throw(WeatherException&);
2. WeatherStation & searchForWarmest() const throw(WeatherException&);
3. WeatherStations(size_t maxNr=10);
4. WeatherStations(WeatherStations const &ws);
5. WeatherStations();
6. WeatherStations &operator =(WeatherStations const &ws);
7. bool Add(WeatherStation const &ws);
8. size_t GetNrStations() const;
9. void PrintAll(std::ostream &out) const;
10. void PrintColdest(std::ostream &out) const;
11. void PrintWarmest(std::ostream &out) const;

ad 1 Die Funktion searchForColdest() ist eine Private Funktion und sucht aus mStationArray die Wetterstation mit der niedrigsten Temperatur und gibt sie zurück. Sollte mStationArray leer sein wird eine Exception geworfen um den Methodenaufruf abzubrechen.

ad 2 Die Funktion searchForWarmest() ist eine Private Funktion und sucht aus mStationArray die Wetterstation mit der höchsten Temperatur und gibt sie zurück. Sollte mStationArray leer sein wird eine Exception geworfen um den Methodenaufruf abzubrechen.

ad 3 Der Konstruktor weißt mMaxNumber maxNr zu und erstellt ein dynamisches Array mit der Anzahl von maxNr Einträgen, zusätzlich wird mNumberOfStations auf 0 gesetzt. Sämtliche Operationen, werden mittels Elementinitalisierliste durchgeführt.

ad 4 Der Copy Konstruktor kopiert alle Werte von ws in das neue Objekt. Bei mStationArray wird eine tiefe Kopie (also das Kopieren der einzelnen Werte, nicht der Adressen) erstellt.

ad 5 Der Destruktor gibt den Speicher des dynamisch erstellten Arrays mittels delete [] wieder frei.

ad 6 Der Zuweisungsoperator kopiert alle Werte von ws in das neue Objekt. Bei mStationArray wird eine flache Kopie (die Adresse wird kopiert) erstellt.

ad 7 Add() fügt eine neue Wetterstation in das mStationArray falls noch ein Platz frei sein sollte. Bei Erfolg gibt die Methode true zurück.

ad 8 GetNrStations() gibt mNumberOfStations zurück.

ad 9 PrintAll() ruft für jedes Element in mStationArray die Print() Funktion auf. Mittels out kan angegeben werden auf welchen Stream geschrieben werden soll.

ad 10 PrintColdest() ruft den Wert von searchColdest() auf dem Stream out aus.

ad 11 PrintWarmest() ruft den Wert von searchWarmest() auf dem Stream out aus.

2.2. Code

Der Code befindet sich im Anhang unter Wetterstations Header und Wetterstations Implementierung. Der Testreiber befindet sich im Anhang unter Testreiber A2 Zusätzlich wird der Code aus Aufgabe 1 mitverwendet.

2.3. Testfälle

Der Testreiber testet die Funktionnalität indem er zuerst ein Objekt erstellt, vier Wetterstationen versucht hinzuzfügen und anschließend sämtliche Methoden aufruft.

```
Test Add
Neuanlage erfolgreich
Neuanlage erfolgreich
Neuanlage erfolgreich
Wetterstation nicht hinzugefuegt
Test PrintAll
Name: WS1 Temperatur: 18.3 Luftfeuchtigkeit: 99
Name: WS2 Temperatur: 6.3 Luftfeuchtigkeit: 12
Name: WS3 Temperatur: -17 Luftfeuchtigkeit: 14
Test Getter
Should return 3: 3
Test copy constr
Coldest of original: Name: WS3 Temperatur: -17
    Luftfeuchtigkeit: 14

Coldest of copy: Name: WS3 Temperatur: -17 Luftfeuchtigkeit:
    14

Test Assignment operator
Warmest of original: Name: WS1 Temperatur: 18.3
    Luftfeuchtigkeit: 99

Warmest of copy: Name: WS1 Temperatur: 18.3 Luftfeuchtigkeit:
    99
```

A. Wetterstation Header

```
1  ////////////////////////////////////
2  // Workfile    : WeatherStation.h
3  // Author      : Matthias Schett
4  // Date        : 12-04-2013
5  // Description : WeatherStation class
6  // Remarks     : -
7  // Revision    : 0
8  ////////////////////////////////////
9
10 #include <string>
11 #include <iostream>
12
13 class WeatherStation{
14
15 private:
16     std::string mName; // Name of the weather station
17     double mCelsius; // Temperature in degree Celsius
18     double mHumidity; // Humidity in percent
19
20 public:
21     // Constructor
22     WeatherStation(std::string const &name="", double celsius
          =0, double humidity=0);
23
24     // Accessor methods
25     std::string const &GetName() const;
26     void SetName(std::string const &name);
27     double GetCelsius() const;
28     void SetCelsius(double c);
29     double GetFahrenheit() const;
30     void SetFahrenheit(double f);
31     double GetHumidity() const;
32     void SetHumidity(double h);
33     // Print the data of the weather station (Name, C,F,
          humidity)
34     void Print(std::ostream &out) const;
35 };
36
37 inline std::ostream &operator << (std::ostream &out,
       WeatherStation & ws){
38     ws.Print(out);
39     return out;
```

```
40 }
```

B. Wetterstation Implementierung

```
1  ////////////////////////////////////
2  // Workfile     : WeatherStation.cpp
3  // Author       : Matthias Schett
4  // Date         : 12-04-2013
5  // Description  : WeatherStation class
6  // Remarks      : -
7  // Revision     : 0
8  ////////////////////////////////////
9
10 #include "WeatherStation.h"
11 using namespace std;
12
13 double const convertFactor = 1.8;
14 size_t const convertConstant = 32;
15
16 WeatherStation::WeatherStation(string const &name, double
       celsius, double humidity) : mName(name), mCelsius(celsius)
       , mHumidity(humidity){
17 }
18
19 // Accessor methods
20 std::string const &WeatherStation::GetName() const{
21     return mName;
22 }
23
24 void WeatherStation::SetName(std::string const &name){
25     mName = name;
26 }
27
28 double WeatherStation::GetCelsius() const{
29     return mCelsius;
30 }
31
32 void WeatherStation::SetCelsius(double c){
33     mCelsius = c;
34 }
35
36 double WeatherStation::GetFahrenheit() const{
37     // TF = TC * 1.8 + 32
38     return (mCelsius * convertFactor + convertConstant);
```

```
39 }
40
41 void WeatherStation::SetFahrenheit(double f){
42     // TC = (TF - 32) / 1.8
43     mCelsius = (f - 32) / 1.8;
44 }
45
46 double WeatherStation::GetHumidity() const{
47     return mHumidity;
48 }
49
50 void WeatherStation::SetHumidity(double h){
51     mHumidity = h;
52 }
53
54
55 // Print the data of the weather station (Name, C,F, humidity)
56 void WeatherStation::Print(std::ostream &out) const{
57     out << "Name: " << GetName() << " Temperatur: " <<
          GetCelsius() << " Luftfeuchtigkeit: " << GetHumidity()
           << endl;
58 }
```

C. Testreiber A1

```
1  ////////////////////////////////////
2  // Workfile    : Main.cpp
3  // Author      : Matthias Schett
4  // Date        : 12-04-2013
5  // Description : WeatherStation class
6  // Remarks     : -
7  // Revision    : 0
8  ////////////////////////////////////
9
10 #include <iostream>
11 #include "WeatherStation.h"
12
13 using namespace std;
14
15 int main(){
16
17     WeatherStation ws1 ("Erste Wetterstation", 32.3, 28.3);
18
19     ws1.Print(cout);
```

```
20
21      cout << "Getter und Setter Test" << endl;
22      cout << "Setting Name to Hans" << endl;
23      ws1.SetName("Hans");
24      cout << "GetName = " << ws1.GetName();
25      cout << "Setting Temperature to 18.3C" << endl;
26      ws1.SetCelsius(18.3);
27      cout << "GetTemp = " << ws1.GetCelsius() << endl;
28      cout << "Setting Humidity to 78" << endl;
29      ws1.SetCelsius(78);
30      cout << "GetHumid = " << ws1.GetCelsius() << endl;
31
32      cout << "Setting Temperature to 18.3F" << endl;
33      ws1.SetFahrenheit(18.3);
34      cout << "GetTemp in F = " << ws1.GetFahrenheit() << endl;
35      cout << "GetTemp in C = " << ws1.GetCelsius() << endl;
36
37      cin.get();
38      return 0;
39  }
```

D. Wetterstations Header

```
1  ////////////////////////////////////
2  // Workfile    : WeatherStations.h
3  // Author      : Matthias Schett
4  // Date        : 12-04-2013
5  // Description : WeatherStation Manager class
6  // Remarks     : -
7  // Revision    : 0
8  ////////////////////////////////////
9
10 #include "WeatherStation.h"
11 #include <iostream>
12 #include <exception>
13
14 class WeatherException : public std::exception
15 {
16 public:
17     WeatherException(const char* errMessage):errMessage_(
       errMessage){}
18     // overriden what() method from exception class
19     const char* what() const throw() { return errMessage_; }
20
21 private:
22     const char* errMessage_;
23 };
24
25 class WeatherStations {
26
27 private:
28
29     // member variables
30     WeatherStation *mStationArray;
31     size_t mNumberOfStations;
32     size_t mMaxNumber;
33
34     // private helper methods
35     // Search for the coldes station
36     WeatherStation & searchForColdest() const throw(
       WeatherException&);
37     // Search for the warnest station
38     WeatherStation & searchForWarmest() const throw(
       WeatherException&);
39
```

```
40 public:
41     // Constructor
42     WeatherStations(size_t maxNr=10);
43
44     // Copy constructor
45     WeatherStations(WeatherStations const &ws);
46
47     // Destructor
48     ~WeatherStations();
49
50     // Assignment operator
51     WeatherStations &operator =(WeatherStations const &ws);
52
53     // Adds a new weather station at the end (if possible)
54     bool Add(WeatherStation const &ws);
55
56     // Returns the number of weather stations
57     size_t GetNrStations() const;
58
59     // Prints all weather stations
60     void PrintAll(std::ostream &out) const;
61
62     // Prints coldest/warmest weather station
63     void PrintColdest(std::ostream &out) const;
64     void PrintWarmest(std::ostream &out) const;
65
66 };
```

E. Wetterstations Implementierung

```
1  ////////////////////////////////////////
2  // Workfile    : WeatherStations.cpp
3  // Author      : Matthias Schett
4  // Date        : 12-04-2013
5  // Description : WeatherStation Manager class
6  // Remarks     : -
7  // Revision    : 0
8  ////////////////////////////////////////
9  #include "WeatherStations.h"
10
11
12 // Constructor
13 WeatherStations::WeatherStations(size_t maxNr) : mMaxNumber(
       maxNr), mNumberOfStations(0), mStationArray(new
```

```
    WeatherStation[maxNr]){
}

// Copy constructor
WeatherStations::WeatherStations(WeatherStations const &ws){
    mNumberOfStations = ws.GetNrStations();
    mMaxNumber = ws.mMaxNumber;
    mStationArray = new WeatherStation[mMaxNumber];
    for(size_t i = 0; i < mNumberOfStations; i++){
        mStationArray[i] = ws.mStationArray[i];
    }
}

// Destructor
WeatherStations::~WeatherStations(){
    delete [] mStationArray;
    mStationArray = 0;
}

// Assignment operator
WeatherStations &WeatherStations::operator =(WeatherStations
    const &ws){
    mStationArray = ws.mStationArray;
    mMaxNumber = ws.mMaxNumber;
    mNumberOfStations = ws.GetNrStations();

    return *this;
}

// Adds a new weather station at the end (if possible)
bool WeatherStations::Add(WeatherStation const &ws){
    if(mNumberOfStations < mMaxNumber){
        mStationArray[mNumberOfStations] = ws;
        ++mNumberOfStations;
        return true;
    }
    return false;
}

// Returns the number of weather stations
size_t WeatherStations::GetNrStations() const{
    return mNumberOfStations;
}

// Prints all weather stations
```

```
57  void WeatherStations::PrintAll(std::ostream &out) const{
58      for(size_t i = 0; i < mNumberOfStations; i++){
59          mStationArray[i].Print(out);
60      }
61  }
62
63  // Prints coldest/warmest weather station
64  void WeatherStations::PrintColdest(std::ostream &out) const{
65      try{
66          searchForColdest().Print(out);
67      } catch(WeatherException& e){
68          std::cout << e.what() << std::endl;
69      }
70  }
71  void WeatherStations::PrintWarmest(std::ostream &out) const{
72      try{
73          searchForWarmest().Print(out);
74      } catch(WeatherException& e){
75          std::cout << e.what() << std::endl;
76      }
77  }
78
79  WeatherStation & WeatherStations::searchForColdest() const{
80      double temperature;
81      size_t pos = 0;
82
83      if(mNumberOfStations > 0){
84          temperature = mStationArray[0].GetCelsius();
85          for(size_t i = 1; i < mNumberOfStations; i++){
86              if(temperature > mStationArray[i].GetCelsius()){
87                  temperature = mStationArray[i].GetCelsius();
88                  pos = i;
89              }
90          }
91          return mStationArray[pos];
92      } else {
93          throw WeatherException("No station are defined");
94      }
95  }
96
97  WeatherStation & WeatherStations::searchForWarmest() const{
98      double temperature;
99      size_t pos = 0;
100
101     if(mNumberOfStations > 0){
```

```
102         temperature = mStationArray[0].GetCelsius();
103         for(size_t i = 1; i < mNumberOfStations; i++){
104             if(temperature < mStationArray[i].GetCelsius()){
105                 temperature = mStationArray[i].GetCelsius();
106                 pos = i;
107             }
108         }
109         return mStationArray[pos];
110     } else {
111         throw WeatherException("No station are defined");
112     }
113 }
```

F. Testreiber A2

```
1  ////////////////////////////////////
2  // Workfile    : Main.cpp
3  // Author      : Matthias Schett
4  // Date        : 12-04-2013
5  // Description : WeatherStation Manager class
6  // Remarks     : -
7  // Revision    : 0
8  ////////////////////////////////////
9
10 #include <iostream>
11 #include "WeatherStations.h"
12
13 using namespace std;
14
15 int main(){
16
17     WeatherStation ws1 ("WS1", 18.3, 99);
18     WeatherStation ws2 ("WS2", 6.3, 12);
19     WeatherStation ws3 ("WS3", -17, 14);
20
21     WeatherStations wsManager (3);
22
23     cout << "Test Add" << endl;
24
25     wsManager.Add(ws1) ? cout << "Neuanlage erfolgreich" <<
           endl : cout << "Wetterstation nicht hinzugefuegt" <<
           endl;
26     wsManager.Add(ws2) ? cout << "Neuanlage erfolgreich" <<
           endl : cout << "Wetterstation nicht hinzugefuegt" <<
```

```
        endl;
27      wsManager.Add(ws3) ? cout << "Neuanlage erfolgreich" <<
            endl : cout << "Wetterstation nicht hinzugefuegt" <<
            endl;
28      wsManager.Add(ws1) ? cout << "Neuanlage erfolgreich" <<
            endl : cout << "Wetterstation nicht hinzugefuegt" <<
            endl;
29
30      cout << "Test PrintAll" << endl;
31
32      wsManager.PrintAll(cout);
33
34      cout << "Test Getter" << endl;
35
36      cout << "Should return 3: " << wsManager.GetNrStations()
            << endl;
37
38      cout << "Test copy constr" << endl;
39
40      WeatherStations wsManager1 (wsManager);
41
42      cout << "Coldest of original: ";
43      wsManager.PrintColdest(cout);
44
45      cout << endl << "Coldest of copy: ";
46      wsManager1.PrintColdest(cout);
47
48      cout << endl << "Test Assignment operator" << endl;
49
50      WeatherStations wsManager2 = wsManager;
51
52      cout << "Warmest of original: ";
53      wsManager.PrintWarmest(cout);
54
55      cout << endl << "Warmest of copy: ";
56      wsManager2.PrintWarmest(cout);
57
58      cin.get();
59      return 0;
60  }
```