Dirk Praetorius
Michele Ruggeri

Wintersemester 2014/15
16.12.2014

Übungen zur Vorlesung
Einführung in das Programmieren für TM

Serie 12

Aufgabe 12.1. Implement the `get` and `set` methods of the class

```
class Fraction {
   long numerator;
   unsigned long denominator;
public:
   Fraction();
   Fraction(long numerator, unsigned long denominator);
   setNumerator(long z);
   setDenominator(unsigned long n);
   double getValue();
};
```

The method `getValue` should return the floating-point value of the fraction. Take care of the fact, that the denominator has to be nonzero. Additionally, implement the constructor `Fraction()` that sets the numerator to 0 and the denominator to 1. Save your source code as `fraction.cpp` into the directory `serie12`.

Aufgabe 12.2. Write a class `Stopwatch` that simulates a stopwatch. The stopwatch consists of two buttons: If the first button is pressed, then the time measurement starts. If the button is pressed again, then the time measurement stops. The second button is used to reset the time to zero. To realize this situation, implement the methods `pushButtonStartStop` (first button) and `pushButtonReset`. Implement another method that prints out the time formatted in the style `hh:mm:ss.xx`, e.g., if the measured time is two minutes, then the output should be `00:02:00.00`. Save your source code as `Stopwatch.{hpp,cpp}` into the directory `serie12`.
Hint: Use the data-type `clock_t` and the function `clock()` from the library `time.h`. It makes sense to use a variable `isRunning` of type `bool`. If the first button is pressed, then this variable is either set to `true` or `false`.

Aufgabe 12.3. Write a class `University`. This class should contain the members `numStudents`, `city`, and `name` as well as the methods `graduate`, and `newStudent`. If the method `graduate` is called, the number of students gets decreased by one, whereas if `newStudent` is called, the number of students increases by one. All data-members should be declared as `private`! Therefore, you have to implement `get` and `set` methods. Save your source code as `University.{hpp,cpp}` into the directory `serie12`.

Aufgabe 12.4. Write a class `Name` which contains two members, `firstName` and `surname` of type `string`. Implement the set-method `setName` that has one string variable as input parameter, and splits the input in first name and surname automatically. Note that the input can contain multiple first names. Furthermore, write a method `printName` which prints out the whole name on the monitor. In case of multiple first names, the output should be shortened as follows: The name `Max Maxi Mustermann` should be printed out as `Max M. Mustermann`. Save your source code as `name.{hpp,cpp}` into the directory `serie12`.

Aufgabe 12.5. Write a class `Client` that stores a list of deposits. Furthermore, the class should contain an object of the class `Name` from Exercise 12.4. Implement methods for adding and deleting deposits. Moreover, write a function that computes the assets of all deposits. Think of other useful functions. Save your source code as `client.{hpp,cpp}` into the directory `serie12`.

Aufgabe 12.6. Write a class `Deposit` with members `accountNumber`, `assets`, and `ratePerCent`. Moreover, implement `set` and `get` methods for the members `accountNumber`, `assets`. To change the assets, write a method `drawMoney` and `placeOnDeposit`. Note that with this deposit you are not allowed to draw more money than is given, i.e., the member `assets` must be positive. The rate per cent as well as the account number must also be positive. Finally, implement the method `calculateAssets`. Save your source code as `Deposit.{hpp,cpp}` into the directory `serie12`.

Aufgabe 12.7. Write a `Makefile` for the exercises of this sheet. It should contain:

- The compilation of all solved exercises.
- The generation of a library and an example of its usage.

Aufgabe 12.8. What is the output of the following code? Explain why!

```
#include <iostream>
#include <string>
using namespace std;

class T1 {
   string t1;
public:
   T1(string val) { cout << "I am constructor of " << val << endl; t1=val; }
   T1() { cout << "I am constructor of default" << endl; t1="default"; }
   ~T1() { cout << "I am destructor of " << t1 << endl; }
};

int main() {
   T1 bert("bert");
   T1 bob;
   T1 def("bob");
   return 0;
}
```