Object-oriented Programming I, Rotation 2
Jacobs University Bremen
Dr. Kinga Lipskoch

Course: CH08-320142
October 19th, 2017

Assignment 5 - Inheritence, `static` Members and Operator Overloading

- The problems of this assignment must be solved in C++.
- The TAs are grading solutions to the problems according to the following criteria: https://grader.eecs.jacobs-university.de/courses/320142/2017_2r2/Grading-Criteria-C++.pdf

Problem 5.1 *`Hexagon` class* (1 point)

Presence assignment, due by 18:30 h today

Download the files:
`https://grader.eecs.jacobs-university.de/courses/320142/cpp/Shapes.h`
`https://grader.eecs.jacobs-university.de/courses/320142/cpp/Shapes.cpp`

Write a class called `Hexagon` which is derived from the `RegularPolygon` class. A hexagon has a side and a color. Provide methods for computing the perimeter and the area of a hexagon. For each property appropriate setter and getter methods need to be provided (i.e., it should not be possible to manipulate data directly). The class should also a parametric constructor, a copy constructor and a destructor.

Note, that the area of a hexagon of side `t` can be computed by using the formula $\frac{3\sqrt{3}}{2}t^2$.

Name the files `Shapes.h` and `Shapes.cpp`. Then write a testing program `testHexagon.cpp` that:

a) creates a red hexagon that has the side of 2,

b) creates a yellow hexagon that has the side of 10,

c) creates a copy for the second hexagon using the copy constructor, and

d) computes the perimeter and area of all three hexagons and prints the results on the screen.

Problem 5.2 *`TournamentMember` class* (2 points)

Imagine that you are in the design stage of a software system for handling the data of the participants of a major soccer tournament. As different roles will be present (players, coaches, referees, etc.) you are required to develop a class handling the data of a generic league member. For each person the following data is at least needed

- first name as character array (31 characters including `'\0'`)
- last name as character array (31 characters)
- date of birth as character array (11 characters, storage format is `yyyy-mm-dd`)

In addition the whole team is located somewhere, so `location` is an additional `static` property of the class.

Design and implement a class for holding these data. In addition add at least two other general properties to this class.

The class, which will be called `TournamentMember`, should provide constructors and destructors and also a copy constructor which creates a correct copy of the original object. The class should also provide `inline` setter and getter methods (either inside or outside of the class). Moreover, in order to carry out the functionality of the application, the following methods are required:

- a method which prints the information of a tournament member to the screen,
- a method which changes the `location`.

Also all constructors and destructors should print a short informational message on the screen, such that you can see which is being called when.
You should provide three files: a header file named `TournamentMember.h` with the declaration of the class, a file named `TournamentMember.cpp` with its definition, and an additional file called `testTournamentMember.cpp` with a `main()` function which tests the functionality of the class.
The needed data can be initialized in the code from the `main()` function.

Problem 5.3 *`Player` class* (1 point)

A `Player` class should be derived from the `TournamentMember` class. It holds additional properties such as number, position, number of goals scored, and whether the player is left-footed or right-footed. For each property appropriate setter (except the number of goals scored) and getter methods need to be provided as inline methods, and it should not be possible to manipulate data directly. An appropriate constructor to set all properties on creation should be provided as well as a copy constructor that creates a correct copy of a player, and a destructor. Also all constructors and destructors should print a short informational message on the screen such that you can see which is being called when.
Also the following methods are required:

- a method which prints the information of a player on the screen,
- a method which increments the number of goals scored by a player.

Add code to the files `TournamentMember.h`, `TournamentMember.cpp`, and write a testing program named `testPlayer.cpp` that tests the functionality of the `Player` class. Create three players with different properties. Then move all players to the location `"Bremen"`.
The needed data can be initialized in the code from the `main()` function.

Problem 5.4 *`Referee` class* (3 points)

Write a class named `Referee` derived from the `TournamentMember` class. This class should have the following additional properties:

- `int yellowCardCount;`
- `Player *yellowCardList[50];`
- `int redCardCount;`
- `Player *redCardList[50];`

And the following methods should be implemented:

- `bool addToYellowCardList(Player *p);`
- `bool addToRedCardList(Player *p);`

If the player `p` is not yet on the yellow card list then it should be added to it, but if the player `p` is already on the yellow card list then the player should be removed from the yellow card list and should be added to the red card list. If the player `p` is not yet on the red card list then it should be added to it, but if the player `p` is already on the red card list then nothing should happen. Both methods should return `true` if the adding was successful and `false` if not.
Add code to the files `TournamentMember.h`, `TournamentMember.cpp`, and write a testing program named `testReferee.cpp` that tests the functionality of the `Referee` class. Create a referee and some players. Your code should illustrate the referee actions for adding players to the list of players with yellow cards and to the list of players with red cards.
You can assume that the input or the setting of the data will be valid.

Problem 5.5 *Fractions I* (2 points)

- As a starting point, use the files `fraction.h, fraction.cpp, testfraction.cpp` (which you can download from:
  `https://grader.eecs.jacobs-university.de/courses/320142/cpp/fraction.h`,
  `https://grader.eecs.jacobs-university.de/courses/320142/cpp/fraction.cpp`,
  `https://grader.eecs.jacobs-university.de/courses/320142/cpp/testfraction.cpp`).
- Replace the method `print()` by an overloaded operator << such that you can use `cout <<` for printing a fraction on the screen.
- Overload the operator >> such that you can enter from the keyboard a fraction using `cin >>`. Check the validity of the input (you can assume that the numerator and denominator will be numbers).
- Overload the operators * and / for computing the multiplication and division of two fractions.
- In your testing program you should then be able to enter two fractional numbers (using `cin >>`), then the product and quotient are printed on the screen (one per line using the overloaded operator and `cout <<`).

Use the suggestions from slide 36 (Lecture 5&6) for choosing to write member methods or friend functions.
You can assume that the input will be valid.

How to submit your solutions

- Your source code should be properly indented and compile with `g++` without any warnings (You can use `g++ -Wall -o program program.cpp`). Insert suitable comments (not on every line ...) to explain what your program does.
- Please name the programs according to the suggested filenames (they should match the description of the problem) in Grader.
  Each program **must** include a comment on the top like the following:

```
/*
   CH08-320142
   a5_p1.cpp
   Firstname Lastname
   myemail@jacobs-university.de
*/
```

- You have to submit your solutions via *Grader* at
  `https://grader.eecs.jacobs-university.de`.
  If there are problems (but **only** then) you can submit the programs by sending mail to
  `k.lipskoch@jacobs-university.de` **with a subject line that begins with CH08-320142**.
  It is important that you do begin your subject with the coursenumber, otherwise I might have problems to identify your submission.
- Please note, that after the deadline it will not be possible to submit any solutions. It is useless to send late solutions by mail, because they will not be accepted.

This assignment is due by Tuesday, October 24th, 10:00 h.