Introduction to Computer Science I

Summer Sessions 2021 - COM SCI31-1 - STAHL

Project 5

Programming Assignment 5
BaseballTeam

Time due: 9:00 PM, Wednesday August 4th

Introduction

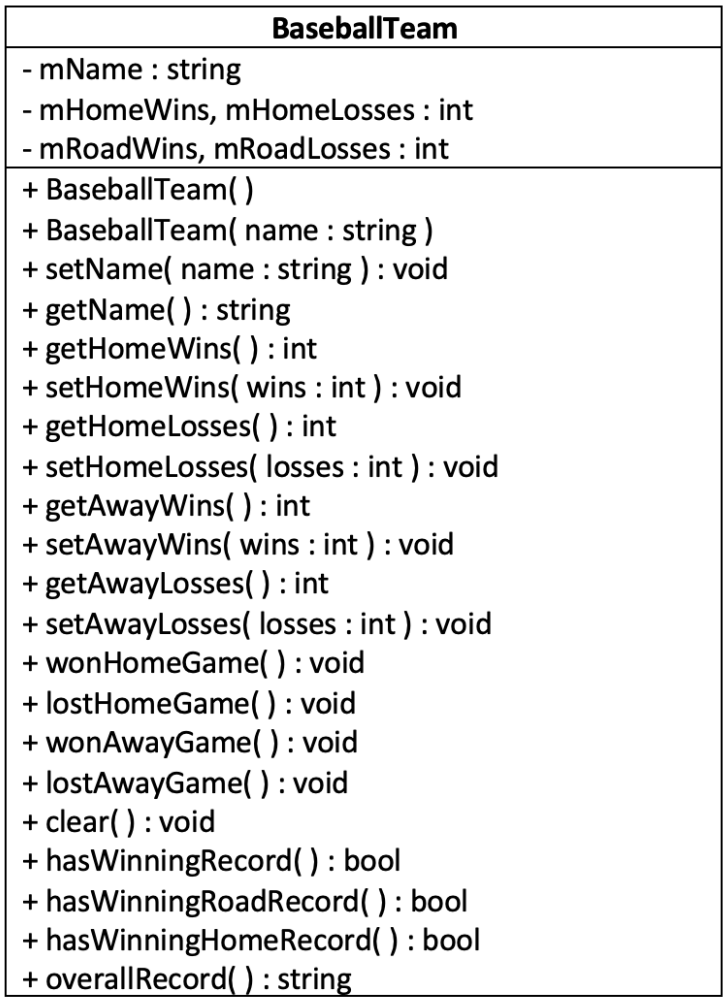
I happen to be a very big Los Angeles Dodgers fan, especially after their recent World Series victory. In normal years, Major League Baseball organizes a full 162-game season between teams grouped into Leagues. Teams play games both in their local home stadium as well as "on the road" in away games. When two teams play a game, the team that scores more wins the game. Games cannot end tied. Since typically home fans are so very supportive and road fans are not, a home record is often maintained separated from road record.

Your task

Your assignment is to produce a two classes that work together to simulate a BaseballTeam and a League. In an effort to help you, the design of these two classes will be discussed here. In addition, some sample code has been provided to assist you with this task. Various UML diagrams are shown below to communicate the code you need to create. Please follow the steps outlined below and don't jump ahead until you have finished the earlier step.

First, create the class `BaseballTeam`. This class represents a baseball team. Each BaseballTeam object has a name, a number of road wins, road losses, home wins and home losses. Your class should provide public getter and setter operations for each of these different data members. Only a name is passed to its constructor call. Individual game wins and losses are adjusted by calls to `.wonHomeGame( )`, `.lostHomeGame( )`, `.wonAwayGame( )` or `.wonAwayGame( )`. Wins and losses get reset to zero by calls to `.clear( )`. A BaseballTeam has a winning record if

total number of wins (both home and away) equals or exceeds its total number of losses (both home and away). A winning road record is similarly determined but considers only at the road/away games played. A winning home record is similarly determined but considers only at the home games played. Calls to `.overallRecord( )` should return a string with the format "*www-lll*" where *www* is a three digit number of wins (padded with zeros if necessary) and *lll* is a three digit number of losses (padded with zeros if necessary). Please review the class diagram shown here:



For this project, you will create both a .h and .cpp for this class. Write some sample driver code in your main( ) and create assertions to verify that your accessor methods are all working properly. Some sample code is shown below to further document how this class should work.

Next, create the `League` class. This class represents a league of baseball teams. Each League has a name which gets passed to its constructor call. Through the League's `.play( ... )` method, the final game score is reported for two teams playing a game. Both team's home or away wins/losses should get updated based on that score. As stated earlier, games will always have a winner so the score will never wind up tied. In addition to reporting the results of a single game, a League's `.season( ... )` method can be used to report a full season's set of wins and losses. The passed team should have its home and away wins/losses should get updated based on those values. A call to `.pennantWinner( ... )` is used to select the team with the best overall record from the five passed team arguments. HINT: Since a team's `.overallRecord( )` is formatted in a particular way, the best record will be the team whose formatted string sorts largest with `operator >` or, alternatively, you could call the int getter operations. Please review the class diagram shown here:

League
- mName : string
+ League(name : string) + setName(name : string) : void + getName( ) : string + play(away : BaseballTeam&, home : BaseballTeam&, awayScore : int, homeScore : int) + season(team : BaseballTeam&, homewins : int, homelosses : int, roadwins : int, roadlosses : int) + pennantWinner(team1 : BaseballTeam, team2 : BaseballTeam, team3 : BaseballTeam, team4 : BaseballTeam, team5 : BaseballTeam) : BaseballTeam

For this project, you will create both a .h and .cpp for this class. Write some sample driver code in your main( ) and create assertions to verify that your accessor methods are all working properly. Some sample code is shown below to further document how this class should work.

You are free to create additional public and private methods and data members as you see fit. However, the test cases will only be driving the public methods of the two classes diagrammed here.

The source files you turn in will be these classes and a main routine. You can have the main routine do whatever you want, because we will rename it to something harmless, never call it, and append our own main routine to your file. Our main routine will thoroughly test your functions. You'll probably want your main routine to do the same. If you wish, you may write additional class operation in addition to those required here. We will not directly call any such additional operations directly.

The program you turn in must build successfully, and during execution, no method may read anything from cin. If you want to print things out for debugging purposes, write to cerr instead of cout. When we test your program, we will cause everything written to cerr to be discarded instead — we will never see that output, so you may leave those debugging output statements in your program if you wish.

Please read the posted FAQ for further assistance.

Additionally, I have created a testing tool called CodeBoard to help you check your code. CodeBoard enables you to be sure you are naming things correctly by running a small number of tests against your code. Passing CodeBoard tests is not sufficient testing so please do additional testing yourself. To access CodeBoard for Project 5, please click the link you see in this week named CodeBoard for Project 5. Inside the files named BaseballTeam.h, BaseballTeam.cpp, League.h and League.cpp, copy and paste the versions you have developed. CodeBoard uses its own main( ) to run tests against your code so you won't get any opportunity to provide a main( ) function. Click Compile and Run. However please be aware that no editing changes can be saved inside CodeBoard. In this anonymous user configuration, CodeBoard is read-only and does not allow for saving changes.

In an effort to assist CS 31 students with the contents of your .zip file archive, Howard has created a Zip File Checker which will echo back to you the contents of your .zip file. Please use this file checker to ensure you have named all your files correctly.

Programming Guidelines

Your program must *not* use any function templates from the algorithms portion of the Standard C++ library or use STL <list> or <vector>. If you don't know what the previous sentence is talking about, you have nothing to worry about. Additionally, your code must *not* use any global variables which are variables declared outside the scope of your individual functions.

Your program must build successfully under both Visual C++ and either clang++ or g++.

The correctness of your program must not depend on undefined program behavior.

What you will turn in for this assignment is a zip file containing the following 6 files and nothing more:

1. The text files named **BaseballTeam.h** and **BaseballTeam.cpp** that implement the BaseballTeam class diagrammed above, the text files named **League.h** and **League.cpp** that implement the League class diagrammed above, and the text file named **main.cpp** which will hold your main program. Your source code should have helpful comments that explain any non-obvious code.

2. A file named **report.doc** or **report.docx** (in Microsoft Word format), or **report.txt** (an ordinary text file) that contains in addition **your name** and **your UCLA Id Number**:
   - A brief description of notable obstacles you overcame

- A list of the test data that could be used to thoroughly test your functions, along with the reason for each test. You must note which test cases your program does not handle correctly. (This could happen if you didn't have time to write a complete solution, or if you ran out of time while still debugging a supposedly complete solution.) Notice that most of this portion of your report can be written just after you read the requirements in this specification, before you even start designing your program.

How nice! Your report this time doesn't have to contain any design documentation.

As with Project 3 and 4, a nice way to test your functions is to use the assert facility from the standard library. As an example, here's a very incomplete set of tests for Project 5. Again, please build your solution incrementally. So I wouldn't run all these tests from the start because many of them will fail until you have all your code working. But I hope this gives you some ideas....

```
#include <iostream>
#include <string>
#include <cassert>

#include "BaseballTeam.h"
#include "League.h"

using namespace std;

int main()
{
   // sample test code
   BaseballTeam team( "Your Team" );
   assert( team.getName( ) == "Your Team" );
   assert( team.overallRecord( ) == "000-000" );
   assert( team.hasWinningRecord( ) );
   assert( team.hasWinningRoadRecord( ) );
   assert( team.hasWinningHomeRecord( ) );

   team.wonHomeGame( );
   team.lostAwayGame( );

   assert( team.overallRecord( ) == "001-001" );
   assert( team.hasWinningRecord( ) );
   assert( ! team.hasWinningRoadRecord( ) );
   assert( team.hasWinningHomeRecord( ) );

   team.clear( );
   assert( team.overallRecord( ) == "000-000" );

   League league( "Your League" );
   assert( league.getName() == "Your League" );
   league.season( team, 15, 10, 10, 15 );
   assert( team.overallRecord( ) == "025-025" );
   assert( team.hasWinningRecord( ) );
   assert( !team.hasWinningRoadRecord( ) );
   assert( team.hasWinningHomeRecord( ) );

   cout << "all tests passed!" << endl;

   return 0;
}
```

By August 4th, there will be links on the class webpage that will enable you to turn in your zip file electronically. Turn in the file by the due time above. Give yourself enough time to be sure you can turn something in, because we will not accept excuses like "My network connection at home was down, and I didn't have a way to copy my files and bring them to a SEASnet machine." There's a lot to be said for turning in a preliminary version of your program and report early (You can always overwrite it with a later submission). That way you have something submitted in case there's a problem later.

G31 Build Commands

```
g31 -c BaseballTeam.cpp
g31 -c League.cpp
g31 -c main.cpp
```

```
g31 -o runnable    main.o    BaseballTeam.o    League.o
./runnable
```

Submission status

Submission status	No attempt
Grading status	Not graded
Due date	Wednesday, 4 August 2021, 9:00 PM PDT
Time remaining	20 hours 25 mins
Last modified	-
Submission comments	▸ Comments (0)

Add submission

You have not made a submission yet.

◄ Notes From Class - Monday

Jump to...

Project 5 FAQ ►