TECHNOLOGICAL UNIVERSITY DUBLIN

TALLAGHT CAMPUS

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRONIC ENGINEERING

PROJECT REPORT

YEAR 2

ACADEMIC YEAR 2020/2021

**SOFTWARE DEVELOPMENT 2 ASSIGNMENT ON A KARATE SCHOOL MANAGEMENT SYSTEM**

Name: Tim Jäger

Student number: x00170190

Submission Date: /05/2021

tudublin.ie

Module: PROJ H2066, SWPR H2002

Programmes: TA\_ERAHUM\_B

1. Description of your project application area.
   1. IDK
   2. More
2. A basic class, with access member functions, Constructors, Static member data
   1. Give all the fully working and tested code and it should be fully commented. State at the top if there are any problems with the code – such as it does not work fully.

This code is fully functional without known bugs. Testing code can be requested from the version control database.

User.h

|  |
| --- |
| 1. #pragma once 2. /\*\* 3. \* Author: Tim Jäger 4. \* E-Mail: tim.jager2001@gmail.com 5. \* 6. \* Description: 7. \* This file contains a user class, used to store 8. \* the details of a karate practitioner. 9. \* 10. \* External dependencies: 11. \* - iostream 12. \* - ctime 13. \* - Vector 14. \* - iomanip 15. \* - string 16. \* 17. \* User-defined dependencies: 18. \* - TJ (namespace) 19. \* - Activity 20. \* - Rank 21. \*/ 22. #include <iostream> 23. #include "TJ.h" 24. #include <ctime> 25. #include <vector> 26. #include <iomanip> 27. #include <string> 28. #include "Activity.h" 29. #include "Rank.h" 30. class User { 31. private: 32. static long count; // Amount of users that have been initialized 33. long UUID; // This is set during the object initialization, hence the set function is private 34. std::string name; // This is the users real name, first and last name 35. TJ::simpleDate dob; // Date of birth 36. Rank rank; // The karate rank, in our style this is a set of every rank you ever got, see Rank.h for details 37. std::vector<Activity> activities; // This stores all mandatory/voluntary karate activities 39. /\* Private getters and setters \*/ 40. void setUUID(long UUID); 41. public: 42. User(); 43. User(std::string name, long clubID, TJ::simpleDate dob); // New practitioners have no rank, so this sets an empty rank 44. User(std::string name, long clubID, TJ::simpleDate dob, Rank rank, std::vector<Activity> activities); 45. ~User(); // This is currently the default destructor 47. /\* Getters and setters \*/ 48. long getUUID(); // Only getter is public for the afore mentioned reason 49. void setName(std::string name); 50. std::string getName(); 51. void setDob(TJ::simpleDate dob); 52. TJ::simpleDate getDob(); 53. void setRank(Rank rank); 54. void giveRank(RankEntry rankEntry); // giveRank(...) adds a rank to the list 55. void giveRank(RankEnum rank, std::string examiner); 56. void giveRank(RankEnum rank, TJ::simpleDate date, std::string examiner); 57. Rank getRank(); 58. void setActivities(std::vector<Activity> activities); 59. void addActivity(Activity activity); // This adds an activity to the list 60. std::vector<Activity> getActivities(); 61. }; |

User.cpp: functions “void getName()” and “User()”:

|  |
| --- |
| std::string User::getName() {  return this->name;  }  User::User() {  User::count++;  UUID = User::count;  this->name = "Unnamed";  this->dob = { 1,1,1 };  } |

* 1. Explain how the code behaves.
  2. Explain the topic and how the code covers the topic. Why it is a good example for your project area.

I have combined points b and c, as they are too closely related in this case.

Above is the header file, which can be recognised by the “.h” extension. This contains all that is needed to answer this question. If you want the implementation file, which is the “.cpp” file, you can find it attached at the end. To use a class, you can call on a function by using “ClassName::functionName(…)” or you can create an object, using code like below.

|  |
| --- |
| User user; |

Here the variable “user” is an instance of “User”, which is acting as a type. Functions like:

|  |
| --- |
| user.getName() |

Will use the “getName()” function as specified in “User.h” and “User.cpp”. In this case it will return “this->name” which means the variable “name” stored inside the object “user”.

In C++ and object oriented programming in general, there is also something called accessibility. In this case it is implemented using the keywords in the class. In any class the date written behind a “private:” keyword, will only be accessible to anything inside that class (exceptions may apply for things like inheritance, which is discussed later on). The “public:” keyword means that anything outside the function can also access it. For this the most recent keyword counts. Therefore in the aforementioned code “void setUUID(long UUID);” is a private member and for instance can’t be used in the “main()” function, while “long getUUID()” is public and can be used in the “main()” function.

Constructors are the functions that get called when you declare an object. When running this line:

|  |
| --- |
| User user; |

The object “user” is declared and the function “User()” in the “User” class will be called. Usually this is to initialize variables and prepare everything for further use. If you look at its content in part “a.”, you can see that the variable “User::count” is increased by 1, that the UUID is set to “User::count” and so on.

Then there is static member data, this data is scope independent, with the exception of its name. This is specified by writing the keyword “static” in front of the variable. In the user class this is “static long count;”. When called anywhere this static data will give the same result. In this case the accessibility is private, therefore a limitation has been posed, that only objects of the same class can access it. But if I were to create multiple users, like I do in “main.cpp”:

|  |
| --- |
| std::vector<User> users; |

Here, when “User::count” is set to 3 in one user, it will be like that for all of them. And if we leave the scope in which this data was created or deleted, it will stay the same, next time you call that variable again, until the program shuts down.

In my opinion the User class is a good example for my project area, as every karate school has members, which are part of the computer system. It also does not cover to much data outside of the question.