به نام خدا

گزارش تمرین سری 4

برنامه نویسی پیشرفته(1)

رضا مرادی 9623100

- **مقدمه**

در ابتدا برای اینکه تمامی کلاس ها باهم مرتبط باشند و برنامه بدون اینکه ارور بدهد اینکلود های لازم شده است به این صورت که از forward declaration استفاده است به طور مثال در فایل ER.h از سینتکس class Prof; به جای اینکلود کردن استفاده کرده ایم به جای اینکه اینلکود مستقیم بکنیم زیرا با این کار ممکن است در یک لوپ گیر کند ولی برای کلاس هایی که از کلاس دیگر ارث میبرند اینکار به صورت مستقیم انجام شده است مثلا اگر کلاس student از کلاس person ارث میبرد به جای نوشتن سینتکس قبلی از "#include "person.h استفاده شده زیرا کامپایلر در همان لحظه ی ورود به هدر فایل کلاس child باید بتواند تمام ممبر ها و متد های کلاس parent را زیر نظر داشته باشد به همین ترتیب نیز در برخی فایل های cpp نیز اینکلود کلاس های لازم انجام شده است زیرا کلاسی مانند course باید بتواند به صورت مستقیم به تمام کلاس های دیگر مانند ... & person student prof lab دسترسی کامل داشته باشد.

- **ER.cpp**

نیاز به توضیح خاصی ندارد دیفالت کانستراکتور مقادیر اولیه را برابر nullptr در نظر میگیرد تا بعدا مقدار دهی شوند و دیتسراکتور به صورت دیفالت در فایل ER.h در نظر گرفته شده چون آبجکت خاصی را قرار نیست پاک کنیم.

- **Theorical.cpp**

نیاز به توضیح خاصی ندارد دیفالت کانستراکتور مقادیر اولیه را برابر صفر در نظر میگیرد و دیفالت کانستراکتور کلاس ER را نیز کال میکند تا از کد کپی جلوگیری شود تا بعدا مقدار دهی شوند و دیتسراکتور به صورت دیفالت در فایل theorical.h در نظر گرفته شده چون آبجکت خاصی را قرار نیست پاک کنیم در فایل theorical.h توابع calculate_final و scoring به صورت override تعریف شده اند چون در کلاس parent موجود هستند و به صورت virtual تعریف شده اند.

- **Lab.cpp**

مانندtheorical.cpp عمل میکند و نیاز به توضیح خاصی ندارد.

- **Person.cpp**

چون دارای متغیر داینامیک courses است در دیستراکتور باید دیلیت شود زیرا courses از نوع ** است و new میشود تابع get_mean را نیز ;0 return میکنیم (در کلاس های child مقدار درست را ریترن میکند)

- **Student.cpp**

نیاز به توضیح خاصی ندارد دیفالت کانستراکتور مقادیر اولیه را برابر nullptr در نظر میگیرد تا بعدا مقدار دهی شوند و دیستراکتور به صورت دیفالت در نظر گرفته شده چون آبجکت خاصی را قرار نیست پاک کنیم تابع get_mean به صورت override تعریف شده چون در کلاس parent موجود است.

- **Prof.cpp**

مانند student نیاز به توضیح خاصی ندارد دیفالت کانستراکتور مقادیر اولیه را برابر nullptr در نظر میگیرد تا بعدا مقدار دهی شوند و دیستراکتور به صورت دیفالت در نظر گرفته شده چون آبجکت خاصی را قرار نیست پاک کنیم تابع get_mean به صورت override تعریف شده چون در کلاس parent موجود است.

- **Course.cpp**

خط 9 تا 14- متغیر های که مقدار ندارند را nullptr ست میکنیم.

خط 16 تا 20- چون ممبر های students و educational_report از نوع ** هستند و new میشوند باید در اینجا یعنی destructor پاک شوند.

خط 22 تا 30- مقادیر ورودی درون ممبر های کلاس ریخته میشوند.

خط 32 تا 48- مانند قسمت قبل است با این تفاوت که یک prof باید اضافه شود برای اینکار course قبلی قابل استفاده نیست پس یک course با تعداد یکی بیشتر درست میکنیم تا prof جدید را نیز داشته باشد در آخر لازم نیست مقدار tempCourse را دیلیت کنیم به دو دلیل یکی اینکه اگر اینکار را انجام دهیم به ارور بر میخوریم و دیگری اینکه این متغیر از نوع پوینتر است و وقتی درون متغیر کلاس ریخته میشود در واقع آدرس متغیر کلاس را دارد با دیلیت کردن این متغیر، متغیر کلاس پاک میشود!!

خط 50 تا 80- مانند قبل است با این تفاوت که این پروسه را لازم است برای student نیز تکرار کنیم.

خط 82 تا 169- دو حالت مختلف را برای اضافه کردن و حذف کردن در نظر میگیریم بدین صورت که مقدار متغیر AR تحت یک if این کار را انجام میدهد سپس باید روند قبل را برای اضافه کردن student جدید و theorical جدید نیز انجام دهیم و چون theorical از ER ارث میبرد این روند را برای ER نیز انجام دهیم و در آخر این روند را برای course انجام میدهیم تا course جدید حاوی اطلاعات صحیح باشد و در else باید به جای اضافه کردن حذف کنیم با این تفاوت که وقتی برای ER این کار را انجام میدهیم برای کلاس theoricalدیگر نیاز نیست چون به صورت اتوماتیک پاک میشود.

خط 172 تا 262- مانند قبل است با این تفاوت که به جای theorical کلاس lab جایگزین میشود.

بقیه ی کلاس نیز واضح است.

در ادامه تمامی تست ها نیز پاس میشوند.

```
root@c545c1f929d0:/usr/src/app# make && ./main
make: 'main' is up to date.
RUNNING TESTS ...
[==========] Running 7 tests from 1 test suite.
[----------] Global test environment set-up.
[----------] 7 tests from APHW4Test
[ RUN      ] APHW4Test.InitiAmirzing0
[       OK ] APHW4Test.InitiAmirzing0 (0 ms)
[ RUN      ] APHW4Test.virtual_calculate_final_test1
[       OK ] APHW4Test.virtual_calculate_final_test1 (0 ms)
[ RUN      ] APHW4Test.max_test2
[       OK ] APHW4Test.max_test2 (0 ms)
[ RUN      ] APHW4Test.average_test3
[       OK ] APHW4Test.average_test3 (0 ms)
[ RUN      ] APHW4Test.max_test4
[       OK ] APHW4Test.max_test4 (0 ms)
[ RUN      ] APHW4Test.max_test5
[       OK ] APHW4Test.max_test5 (0 ms)
[ RUN      ] APHW4Test.get_mean6
[       OK ] APHW4Test.get_mean6 (0 ms)
[----------] 7 tests from APHW4Test (0 ms total)

[----------] Global test environment tear-down
[==========] 7 tests from 1 test suite ran. (4 ms total)
[  PASSED  ] 7 tests.
<<<SUCCESS>>>
root@c545c1f929d0:/usr/src/app#
```

s