Homework 04

> This homework and content were created by 黃漢軒 (109590031), please feel free to ask me if you have any questions.
>
> Email: t10950031@ntut.org.tw／MS Teams 黃漢軒

⚠ Due: 11:59 p.m., 08 / 11 / 2022 ⚠

Goal

This homework has these goal:

- Know how to interactive to the other Class.
- Practice the opeator overloading.
- Enhance the technique of constructor and destructor.

Folder Structure Tree

- You should finish the unit test written by you.
- you can split the unit test into multiple file, just remember include all of it into `ut_main.cpp` (see course repo).
- The `topping.h` and `ut_topping.h` already completed.

While your project has been built by `makefile`, the structure tree should be the same as the following section.

```
bin/
├─ ut_all
src/
├─ drink.h
├─ topping.h
test
├─ The unit test file written by you, like ut_drink.h, ut_sample.h...
├─ ut_topping.h
├─ ut_main.cpp
makefile
```

Problem Content

When you already finish the class of Drink, you thought that the topping is string-type object.

You feel so boring because we should describe the topping by Class, not only string.

You ask Uriah to complete the Topping Class to press ahead the FoodPolarBeer project.

You think when the topping has been added to the drink, it will change the price and the sweetness level of drink.

Uriah already finished the *Topping* class for you to use it on the function *addTopping* and *getToppingByIndex*.

In this homework, you should complete the following task.

Task

In this task, we continue use the source code in HW#3, you can paste the section of code, or just rewrite it.

See the source code we provide.

- The function you should modify
  - `Drink::Drink(std::string name, double sweetness_level)`
    - You should change the constructor to `Drink::Drink(std::string name, double sweetness_level, int price)`
  - `Drink::addTopping(std::string topping)`
    - You should change the parameter of function to `Drink::addTopping(Topping topping)`
  - `Drink::getTopping(int index)`
    - You should change the return-type of function to `Topping` class.

- The function you should implement
  - `Drink::getPrice()`
    - The price should be setup by user, if not, throw the exception.
    - You should setup the price on the constructor or setter.
  - `Drink::operator=(const Drink &other)`
    - You can use it to do the *Copy Assignment* on the Drink.

- Please notice that:
  - When the *Topping* add to the *Drink*, it should be increase the sweetness level and the price, see the Sample section.
  - The strict should be as same as HW#3.
  - You can see the Sample section to know the detail of function.
  - You should test the function, see the Test section.
  - You can ignore the situation that when the topping added to drink, the `sweetness_level` will out of range we strict.

Sample

Assume we have a variable `drink`, which is constructed by the Class *Drink* you implement.

Sample #1

Check all the data by getter.

```
/* Setup name to "Signature Black Tea with Milk"（熟成歐蕾）. */
/* Setup sweetness level to 0.3 */
/* Setup price to 45 */

drink.getName();  // It will return "Signature Black Tea with Milk"
drink.getSweetnessLevel(); // It will return 0.3
drink.getPrice(); // It will return 45.
```

Sample #2

Add a topping, and it should change sweetness level and price.

```
/* Setup name to "Signature Black Tea with Milk"（熟成歐蕾）. */
/* Setup sweetness level to 0.3 */
/* Setup price to 45 */

drink.addTopping(Topping("Bubble", 0.15, 5));
drink.getName();  // It should return "Signature Black Tea with Milk"
drink.getSweetnessLevel(); // It will return 0.45, because 0.3 + 0.15 = 0.45.
drink.getPrice(); // It should return 50, because 45 + 5 = 50.
```

Sample #3

Using copy assignment to copy the data to the object, and use the getter to check value is correct.

```
Drink some_drink("Black Tea", 0.2, 15);
Drink drink;
drink = some_drink;

drink.getName();  // It should return "Black Tea"
drink.getSweetnessLevel(); // It will return 0.2
drink.getPrice(); // It should return 15.
```

Sample #4

Using *getToppingByIndex* to get the value of *Topping*

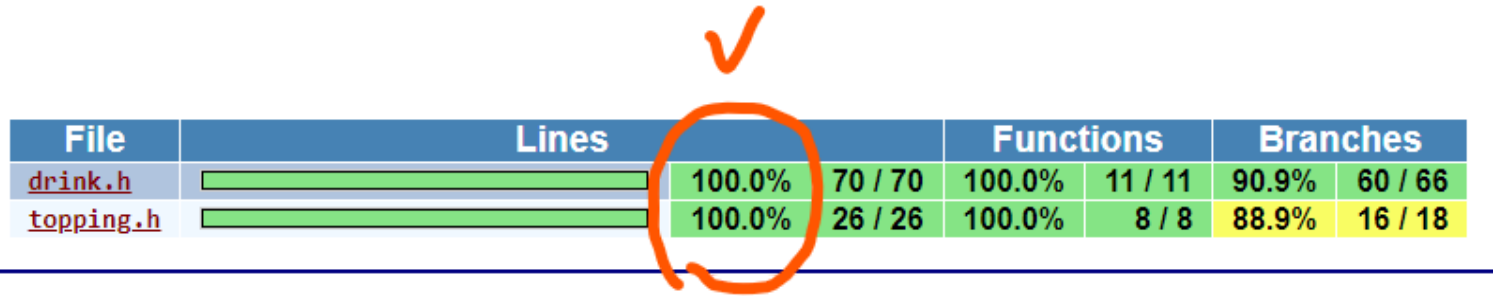
```
/* Setup name to "Signature Black Tea with Milk"(熟成歐蕾). */
/* Setup sweetness level to 0.3 */
/* Setup price to 45 */

drink.addTopping(Topping("Bubble", 0.15, 5));
Topping topping = drink.getToppingByIndex(0);
topping.getName(); // It should return "Bubble"
topping.getSweetnessLevel(); // It should return 0.15
topping.getPrice(); // It should return the price of topping, which is 5
```

Test

In this homework, you should use `gcovr` tool to make sure your *code coverage* in `/src` is all above of \90%\.

- If your lines of *code coverage* is below of \90%\, you will receive `FAILURE` in the HW Job.



File	Lines		Functions		Branches	
drink.h	100.0%	70 / 70	100.0%	11 / 11	90.9%	60 / 66
topping.h	100.0%	26 / 26	100.0%	8 / 8	88.9%	16 / 18

You will get the 35% score if HW Job passed, otherwise, you will lose the 35% score if HW Job failed.

See course slide (`OOP_gcovr.pptx`) to know how to install and how to use it .

Notice

- Use nullptr if you want to have a null pointer, which is a special pointer that doesn't point to anything.
- Use `ASSERT_EQ` to test integer, `ASSERT_NEAR` to test floating-point number, `ASSERT_THROW` to test exception.
- You should neither add bin foler to your git, nor add a file with the name of '.gitignore' in bin folder (see our class repo).
- Some situation you will lose score:
  - **You lose 5 points for each test that has memory leak. You can check memory leak with `valgrind` cmd.**
  - **You will lost 10% if your bin folder contains compiled ut_all in git repo.**

Meme