Fix the create_sandwich_array should throw exception if count is negative.
Everything TA authored 1 week ago

Name	Last commit	Last update
README.md	Fix the create_sandwich_array should throw ...	1 week ago
test_image.png	Setup hw#8	2 weeks ago

README.md

Homework 8

> This homework was 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., 05 / 01 / 2023 ⚠️

Goal

This homework has these goals:

- Know how to use simple factory.
- Know how to use static container.
- Collect the previous homework into one simple homework.
- [Optional but strongly suggest] Know how to use set container in C++.

Folder Structure Tree

- You should finish the unit test written by you.
- You can split the unit test into multiple files, just remember to include all of it into `ut_main.cpp` (see course repo).

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

```
bin/
├─ ut_all
src/
├─ beef_sandwich.h
├─ sandwich_factory.h
├─ sandwich.h
├─ sweet_sandwich.h
test/
├─ <some test file>
├─ ut_main.cpp
makefile
```

Problem Content

In this time, Uriah is going to run a sandwich shop.

Since he doesn't want to make sandwich by himself, he want to make a sandwich factory.

Sandwich

Uriah will sell two type of sandwich: `SweetSandwich` and `BeefSandwich`.

- The `SweetSandwich` have `price` attribute to record the price and the `ID` attribute to record the `ID` of `SweetSandwich`.
- The `BeefSandwich` have `price` attribute to record the price and the `ID` attribute to record the `ID` of `BeefSandwich`.

The both sandwich should store the ID in the container, used to record the **exist** sandwich with specific flavor.

The sandwich base class should store all the ID of **exist** sandwich, when any flavor of sandwich has been made.

See the task below.

Factory

You are going to create a sandwich factory `SandwichFactroy` .

The sandwich factory can do these things:

- Create sandwich and return the sandwich pointer.
- Create many sandwich and return the sandwich vector array.

See the task below.

Task

In this task, you should create 4 class: `SweetSandwich` , `BeefSandwich` , inheritance `Sandwich` , and `SandwichFactroy` .

The public member listed in below.

- `class Sandwich`
  - `Sandwich(int price, int id)`
    - The constructor to create Sandwich.
    - If the price is negative, you should throw the `std::invalid_argument` exception.
  - `int get_price()`
    - Return the price of Sandwich.
  - `int get_id()`
    - Return the ID of Sandwich.
  - `static bool record_has_specific_id(int id)`
    - Check the specific ID is exist in record.
  - `static int get_size_of_record_container()`
    - Return the size of the record.
- `class BeefSandwich` , inheritance `Sandwich`
  - `BeefSandwich(int price, int id)`
    - The constructor to create BeefSandwich.
    - If the price is negative, you should throw the `std::invalid_argument` exception.
  - `static bool record_has_specific_id(int id)`
    - Check the specific ID is exist in record.
  - `static int get_size_of_record_container()`
    - Return the size of the record.
- `class SweetSandwich` , inheritance `Sandwich`
  - `SweetSandwich(int price, int id)`
    - The constructor to create BeefSandwich.
  - `static bool record_has_specific_id(int id)`
    - Check the specific ID is exist in record.
  - `static int get_size_of_record_container()`
    - Return the size of the record.
- `class SandwichFactory` , should use template (`SandwichFactory<T>`)
  - `static T* create_sandwich(int price, int id)`
    - Create the sandwich pointer with static member.
    - Return specific type of sandwich pointer.
  - `static vector<T*> create_sandwich_array(int price, int count, std::vector<int> id_list)`
    - Create the specific count of sandwich with specific ID.
    - Throw the `std::invalid_argument` exception if the count is negative.
    - Return specific type of sandwich array.

Two different type of sandwich should not appear repeated ID.

(i.e. Not exists the situation when BeefSandwich with ID 6 and SweetSandwich with ID 6.)

You should **delete the default constructor** and **implement destructor** in every class.

For every sandwich class, if a sandwich has been constructed, you should record the ID in the sandwich and the specific flavor sandwich.

Also, if a sandwich has been destructed, you should remove the ID in the sandwich and the specific flavor sandwich.

Sample

```
/* Pre create these sandwich */
bf1 = new BeefSandwich(25, 0);
sw1 = new SweetSandwich(15, 1);
bf2 = new BeefSandwich(45, 2);
sw2 = new SweetSandwich(25 3);

/* Check record in sandwich */
BeefSandwich::record_has_specific_id(0); // TRUE
SweetSandwich::record_has_specific_id(1); // TRUE
BeefSandwich::record_has_specific_id(2); // TRUE
SweetSandwich::record_has_specific_id(3); // TRUE
Sandwich::record_has_specific_id(0); // TRUE
Sandwich::record_has_specific_id(1); // TRUE
Sandwich::record_has_specific_id(2); // TRUE
Sandwich::record_has_specific_id(3); // TRUE

/* Check record size in sandwich */
BeefSandwich::get_size_of_record_container(); // 2
SweetSandwich::get_size_of_record_container(); // 2
Sandwich::get_size_of_record_container(); // 4

/* Delete some sandwich and check record */
delete sw1;
delete bf1;

/* Check record in sandwich */
BeefSandwich::record_has_specific_id(0); // FALSE
SweetSandwich::record_has_specific_id(1); // FALSE
BeefSandwich::record_has_specific_id(2); // TRUE
SweetSandwich::record_has_specific_id(3); // TRUE
Sandwich::record_has_specific_id(0); // FALSE
Sandwich::record_has_specific_id(1); // FALSE
Sandwich::record_has_specific_id(2); // TRUE
Sandwich::record_has_specific_id(3); // TRUE

/* Check record size in sandwich */
BeefSandwich::get_size_of_record_container(); // 1
SweetSandwich::get_size_of_record_container(); // 1
Sandwich::get_size_of_record_container(); // 2
```

```
SweetSandwich* sweet_sandwich = SandwichFactory<SweetSandwich>::create_sandwich(15, 32767);
// It should return a SweetSandwich pointer with price 15 and ID 32767.
```

```
vector<SweetSandwich*> sweet_sandwich_set = SandwichFactory<SweetSandwich>::create_sandwich_array(15, 5, {33, 44, 55, 66, 9

// It should return a SweetSandwich pointer array.
// sweet_sandwich_set[0]: price 15, ID 33
// sweet_sandwich_set[1]: price 15, ID 44
// sweet_sandwich_set[2]: price 15, ID 55
// sweet_sandwich_set[3]: price 15, ID 66
// sweet_sandwich_set[4]: price 15, ID 99
```

The way to impement the ID record

It will be very convenient to use `std::set` to implement the record. See this.

Add the value

```
some_set.insert(value)
```

Erase the value

```
some_set.erase(value)
```

Check the value in the set

```
some_set.find(value) != some_set.end();
```

Check the size of set

```
set_set.size()
```

Test

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

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

File	Lines			Functions		Branches	
beef_sandwich.h		100.0%	10 / 10	100.0%	4 / 4	100.0%	2 / 2
sandwich.h		100.0%	17 / 17	100.0%	6 / 6	100.0%	4 / 4
sandwich_factory.h		100.0%	9 / 9	100.0%	2 / 2	100.0%	7 / 7
sweet_sandwich.h		100.0%	10 / 10	100.0%	4 / 4	100.0%	2 / 2

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

See the course slide (`OOP_gcovr.pptx`) to know how to install and how 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 integers, `ASSERT_NEAR` to test floating-point numbers, and `ASSERT_THROW` to test exceptions.
- You should neither add a bin folder to your git nor add a file with the name '.gitignore' in the bin folder (see our class repo).
- In some situations you will lose score:
  - **You lose 5 points for each test that has a memory leaks. You can check memory leak with `valgrind` cmd.**

    ```
    valgrind --track-origins=yes --leak-check=all <executable_file>
    ```

  - **You will lose 10% if your bin folder contains compiled ut_all in the git repo.**

Meme