Object-Oriented Programming
Assignment 5
Project

403410033 資工三 曾俊宏

June 12, 2017

1 Concurrent Development

Concurrent development sounds good in theory, but in reality it's pretty hard to achieve.

By defining all of the interfaces first, then starts writing code, we can divide the work among teammates. We can call the method even if it's not implemented yet, knowing that it'll eventually be working in the end.

Having interfaces also have an advantage that we can easily swap the implementations. For example, in our team, class Rational have 2 different implementations. We can easily swap them and test for correctness.

The downside of concurrent development is that teammates are procrastinating too much. When this happens, the development will be slowed down significantly, since testing can't be done at all.

Take our team for example, no one was working on the project when I am already on it. So I decided to implement all interfaces and write a testbench. Testbench is a pretty good idea for making you confident that all of the code your teammate eventually turnin won't break the codebase, especially when they are under deadline pressure.

2 What are the advatanges of OOP principles (encapsulation, inheritance, and polymorphism) used in this work?

Encapsulation: The implementation details are hidden from the user. When Rational class uses BigInt, it doesn't need to know, for example, how addition is implemented. All we need to know is how to correctly use it, e.g. how to invoke BigInt methods, what will be returned. The major advantage is that everybody just need to take care of their respective jobs, and trust the components that you use will give you the correct result.

Inheritance: Not used.

Polymorphism: Not used.

3 What did you learn from this experience of team programming?

Testbench is pretty awesome! It gives you confidence as you develop and make modification to the code you wrote. Better yet, when your teammate turn in their code, you can test it quickly and if all the tests passed you can be very certain that the code they wrote is correct.

4 Teammate Evaluation

4.1 黄子俞

Score: 10

An average teammate.

Accomplished tasks, but it's too close to deadline so there isn't a lot of time to discuss with him.

4.2 童彦淇

Score: 5

An average teammate.

She's in charge of the BigInt part, which is the core of the entire program. But she started working on the day before deadline, which is a pretty bad idea.

Worse yet, the code she turned in was incorrect. (To be exact, only the addition is correct. I am kind of glad that I decided to take on her part myself in the beginning, or our team won't have functional codebase at all)

5 Notes on code

The codebase is fully tested on MacOS 10.12.5, using g++ compiler 7.1.0

5.1 Makefile

make normal: Produces an executable file named *normal*, which runs the sample test case provided on the spec.

make test: Produces an executable file named *test*. This is the testbench that I used for testing the correctness of codebase.

5.2 Codebase

The folder *Teammate* contains the code written by my teammates. The code in *Rational* folder is working and correct (authored by 黄子俞). The code in *BigInt* folder is incorrect, although it can be compiled successfully (authored by 童彦淇).

The folder *Codebase* contains the final version of our project. Compile using the aforementioned method should work.

The code file *main.cpp* contains the sample test case.