Lecture 9 / Tutorial 8
Software Contracts

Design by contracts
Programming by contracts
Today…

1. Sign a contract
2. Design by contract
3. Programming by contract
4. Summary
5. Questions and Answers
1. Sign a Contract

Having done one module, to swap with other team, you can sign a contract with other teams:

- Name of Team A: ……………………. 
- Name of Team B: ……………………. 
- Team A is responsible for the …………… module 
- Team B is responsible for the …………… module 
- Terms on functionalities and qualities 
- Terms on intellectual properties: license 
- Terms on compensation for failures 
- And so on … 
- Signature
2. Design by contracts

- Why design contracts? Verification and Validation ________ checks whether the end-product meets the customer requirements
  ________ check whether the product of current phase preserves the requirements of the product of the previous phase
- In object-oriented software construction, a design contract consists of such obligations
  - Pre-conditions and post-condition for a _________
  - Invariants for a _________
- Inheritance can extend the design contracts
  - precondition of A.foo() implies precondition of B.foo()
    ________ extends _________
  - postcondition of C.bar() implies postcondition of D.bar()
    ________ extends _________
  - invariant of E implies invariant of F
    ________ extends _________

Reference
3. Programming by contracts

How to guarantee the design contracts?

Today we show three techniques:

• Assertions
• Unit tests
• Class wrappers
3.1 Assertions

- Assertions are *debug* statements inserted into the normal statements to check on the conditions

```java
float division(float a, float b) {
    assert(______);
    float c;
    // c = f(a, b)
    assert(______);
    return c;
}

class number {
    int n;
    // invariant: n>0
    void inc() { assert(______)... assert(______); }
    void dec() { assert(______)... assert(______); }
}
```

- Assertions can be _____________ before the code is released
3.2 Unit tests

• One can guarantee the correctness through unit tests, for example:

  - `junit.framework.Assert.assertTrue("output matches input", nodiff);`
  - `junit.framework.Assert.assertEquals("output matches input", output, expected_output);`
  - `junit.framework.Assert.assertNotNull("output matches input", object);`
  - *And so on*
3.3 Class wrappers

• Having a class wrapper is more convenient

• Example

```java
class Number {
    NumberImpl n;
    float division (float a, float b) {
        assert (b != 0);
        float c = n.division(a, b);
        assert (c*a == b);
        return c;
    }
}
```

• Question: The _______ design pattern is used in the above example

• Advantages over assertions and unit tests
  – Better than assertions: _________________________________
  – Better than unit tests: _________________________________
    _________________________________

• Reference

• http://www.ddj.com/documents/s=1640/ddj0503f/
4. Summary

- What is “design by contracts”
- How to implement the contracts
- Think about how to enforce your customer contracts with your developer contracts?
- Questions and answers…
Project information

On Web Service Deployment

- What’s more
  - We have a course forum
    http://seawolf.cdf.toronto.edu:9192/ece450

- If you want to deploy the web service in the lab
  - We have a Tomcat/MySQL server in the Linux Lab of CDF
  - Production http://werewolf.cdf.toronto.edu:9192/production
  - Sand box: http://werewolf.cdf.toronto.edu:9192/sandbox
  - Put your binary files into
    - /u/yijun/.ece450/production
    - /u/yijun/.ece450/sandbox
  - Ask me to create a mysql database for you if necessary