University of Washington
CSE 403 Software Engineering
Spring 2009

Final Exam

Friday, June 5, 2009

Name: _____

Initials: _____

UW net id: _____

UW id number: _____

This quiz is closed book, closed notes. You have **50 minutes** to complete it. It contains 25 questions and 11 pages (including this one), totalling 100 points. Before you start, please check your copy to make sure it is complete. Turn in all pages, together, when you are finished. **Write your initials on the top of *ALL* pages.**

Please write neatly; we cannot give credit for what we cannot read.

Good luck!

Page	Max	Score
2	14	
3	7	
4	8	
5	15	
6	6	
7	16	
8	14	
9	8	
11	12	
Total	100	

1 True/False

(2 points each) Circle the correct answer. T is true, F is false.

1. **T/F** A representation invariant should be part of the specification of every non-trivial class.

2. **T/F** In general, designing a system such that its UML class/dependence diagram has more edges is good.

3. **T/F** Assertions should never be enabled in production code.

4. **T/F** Bottom-up testing is not an effective way to reveal architectural design flaws.

5. **T/F** Top-down testing can be time-consuming, because you need to write stubs for many modules.

6. **T/F** One drawback of bottom-up system testing (as compared to top-down system testing) is that when a test fails, there are more places you have to look for the bug.

7. **T/F** Design patterns typically increase the amount of code that needs to be written to accomplish a specific purpose.

2 Multiple choice

8. (4 points) Which of the following reasons are valid ones for choosing a top-down process? (Circle all that apply.)

   (a) A top-down process is more time consuming because of the unit tests.

   (b) Developers can present a demo of the project to the management faster than using a bottom-up process.

   (c) In a top-down design, if an error is detected it's always because a lower-level module is not meeting its specifications (because the higher-level ones are already been tested).

   (d) A top-down process makes it possible to detect performance problems faster

   (e) A top-down process makes it easier to fix a global conceptual problem

9. (3 points) The tests that run every night during the project development or before a programmer checks in any changes to the global repository are normally characterized as:

   (a) Regression tests

   (b) Integration tests

   (c) Validation tests

   (d) System tests

   (e) None of the above

10. (4 points) Which of these statements about black-box tests are commonly true? (circle all that apply)

    (a) Black-box tests generally aim to provide as much coverage of the lines of code in a module as possible, because coverage is usually positively-correlated with test quality.

    (b) Black-box tests can be used to find aliasing errors that occur when two different formal parameters of a method both refer to the same object.

    (c) Black-box tests can be effective at detecting representation exposure.

    (d) In practice, it is often difficult to re-use the same black-box test when crucial parts of the implementation change, even when the specification remains the same.

    (e) Black-box tests are able to partition the input space into finer (smaller) partitions than glass-box tests.

11. (4 points) Circle all of the below that are advantages of sketching a proposed user interface (as opposed to using a drawing program or building a prototype).

    (a) It is easier to make small changes to a large drawing

    (b) It is faster

    (c) It looks less polished to users, encouraging them to suggest changes

    (d) It makes it easier to ignore irrelevant details

3 Short answer

Answer each question with **one sentence or phrase**.

12. (3 points) What is the most important reason for comparing a procedure to a specification?

---

13. (3 points) What is the most important reason for comparing one specification to another?

---

14. (3 points) In Java, `Integer` is a subtype of `Number`.
`List<Integer>` is not a subtype of `List<Number>`. Why? (Don't give a Java-specific answer; explain why Java's choice is the right one.)

---

15. (3 points) `List<Number>` is not a subtype of `List<Integer>`. Why?

---

16. (3 points) For physical objects, maintenance is required to repair the effects of wear and tear. For non-buggy software, what is the most frequent cause that requires "maintenance"?

---

17. (6 points) Give the weakest precondition for the following code, with respect to the postcondition $x > y$. Assume that `p` is `boolean` and `x` and `y` are `int`.

```
p = x>y;
if (p) {
  x++;
} else {
  y = x + y;
}
```

Answer (show work below for partial credit): _____

18. (8 points) Compare incremental (per-checkin) code reviews to comprehensive (whole-module, but now whole-system) code reviews.

    Give two benefits of incremental code reviews.

    - _____
    - _____

    Give two benefits of comprehensive code reviews.

    - _____
    - _____

    Give two differences in the mechanics of how they are typically performed.

    - _____
    - _____

19. (4 points) What are two aspects of a software system that are explicitly omitted from a UML class diagram?

    - _____
    - _____

20. (4 points) Name the two key advantages of factory methods, when compared to constructors. (Use no more than 10 words each.)

    - _____
    - _____

21. (6 points) Exhaustive testing (testing every value in the input domain reveals every defect, but is impractical. Partition testing splits the input domain into parts, and chooses just one test for each of the parts. Partition testing reveals every defect, under what condition? (One sentence or phrase)

    _____

    What might happen if some of the partitions are too large?

    _____

    What might happen if some of the partitions are too small?

    _____

22. (8 points) State 3 distinct benefits of writing tests before writing the code.

    - _____
    - _____
    - _____

    Is this approach more applicable to black-box tests, more applicable to clear-box tests (aka glass-box or white-box tests), or equally applicable to both?

    _____

4 Long answer

23. (8 points) Suppose that each procedure in my program has a specification. I wish to prove the whole program is correct; that is, each procedure satisfies its specification.

    If the program has no recursion, it is easy to reason *modularly* — that is, one procedure at a time. Here is how you could do that.

    (a) First, prove correctness of each procedure at the leaves of the procedure call tree (the ones that call no other procedure).

    (b) Next, prove correctness of the procedures that call only the leaf procedures. During this process, it is valid to rely on the specifications of the leaf procedures, and assume that each leaf procedure call does exactly what its specification says.

    (c) Now, continue up the procedure call tree.

    Now, suppose that the program contains *recursion*. When reasoning about a procedure call (including a self-call) it would be circular reasoning to assume that the procedure is correct if I have not already proved it — that is, to assume that the procedure is correct in order to prove it correct.

    What approach do you suggest in this circumstance? Explain why it works and any potential downside. Use no more than 5 sentences. (It is possible for a 1- or 2-sentence answer to get full credit.)

5 Reasoning

(This information is for questions 24–25 on page 5.)

Ben Bitdiddle joins the Megasensible Corporation's famous Flunk spreadsheet group. Bens first assignment is to refactor the Flunk source base to eliminate duplicate code that had accumulated over the years. Ben finds 4 implementations for the function that copies the first `n` elements from List `src` to List `dest`.

```
void partialcopy(int n, List<Integer> dest, List<Integer> src)
```

Fortunately for Ben, Megasensible only hires CSE 403 graduates, and all the implementations have specifications.

(Note: in this codebase, arrays are indexed starting with 1, not 0.)

```
Specification A
  requires: n > 0
  modifies: dest
  throws: ArrayOutOfBoundsException if src.length < n
  effects: for i=1..n  dest[i]post = src[i]pre
  returns: nothing

Specification B
  requires: n > 0
  modifies: src, dest
  throws: ArrayOutOfBoundsException if src.length < n
  effects: for i=1..n  dest[i]post = src[i]pre
  returns: nothing

Specification C
  requires: n > 0
  modifies: dest
  throws: nothing
  effects: for i=1..min(n, src.length):  dest[i]post = src[i]pre
           for i=src.length+1..n:  dest[i]post = 0
  returns: nothing

Specification D
  requires: n > 0 and src.length >= n
  modifies: dest
  throws: nothing
  effects: for i=1..n:  dest[i]post = src[i]pre
  returns: nothing
```

24. (9 points) In the following diagram, for all pairs of specifications, draw an arrow from X to Y (X→Y) if and only if X is stronger than Y. (There are 12 possible arrows that you might draw in the diagram, for each direction among the 6 pairs of specifications.)

A B

D C

25. (3 points) According to the specifications, which method(s) should Ben choose to replace all the others?

_____

End of quiz.

Make sure that you have written your initials on the top of ALL pages.