University of Washington
CSE 403 Software Engineering
Spring 2011

Final exam

Friday, June 3, 2011

Name: _____

CSE Net ID (username): _____

UW Net ID (username): _____

This exam is closed book, closed notes. You have **50 minutes** to complete it. It contains 34 questions and 7 pages (including this one), totaling 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	24	
3	20	
4	14	
5	17	
6	14	
7	11	
Total	100	

1 True/False

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

1. **T / F** The spiral lifecycle model is always better than the outdated waterfall model.
2. **T / F** Including the users/customers in the process of developing requirements will yield a better product.
3. **T / F** Including the users/customers in the process of designing the architecture will yield a better product.
4. **T / F** Recall that the null object pattern performs a no-op at each method call where null would otherwise be dereferenced. One advantage is that this is more efficient than checking, at each method invocation, whether the receiver (the `this` object on which the method is being invoked) is null.
5. **T / F** Another advantage of the null object pattern is that it makes the code shorter and its behavior easier to understand, compared to other approaches to addressing null pointer errors.
6. **T / F** The piece table is an example of a lazy data structure, in which as few changes as possible are performed.
7. **T / F** It is possible for a single "piece", or part of the original file, to appear multiple times in a piece table.
8. **T / F** When the writes to a data structure have spatial locality (they are nearby one another in memory), then they can be made faster.
9. **T / F** There exist datasets for which a hash table has $O(n)$ performance for insertion, deletion, and searching — the same as for a linked list.
10. **T / F** An example of refactoring is adding new features to satisfy a customer requirement discovered after a project is shipped.
11. **T / F** For every refactoring that can improve the design of a software system, undoing or reversing the refactoring can also improve the design of the *same* system.
12. **T / F** Refactoring is a risk: it incurs a cost now, in return for potential payoff later.

13. **T / F** The `map` in MapReduce is *different* from the `map` in a functional programming language.

14. **T / F** The purpose of a code review is to examine one specific diff or commit — sometimes after but usually before it is incorporated into the main code repository.

15. **T / F** An advantage of a GUI is that it permits an experienced user to work faster.

16. **T / F** Suppose you are able to find the true revealing sub-domains for a software system, and you design unit tests for it accordingly. Later, if you change your implementation, it is also necessary to update your tests so that they still cover the revealing sub-domains, which may have changed.

17. **T / F** It is theoretically possible for testing to prove the absence of bugs.

18. **T / F** Unit tests should be written both before and after code is written.

19. **T / F** Boundary testing can catch off-by-one errors, but not null pointer exceptions.

20. **T / F** Interviewers generally prefer to hire applicants who can answer questions quickly, without having to ask a lot of clarifying questions first.

21. **T / F** Interviewers are looking to see you get to the best/most optimal solution on any problem they give you.

22. **T / F** If you are having issues with a team member, you should inform your management about the potential problem.

2 Multiple choice

23. (5 points) Which of the following are reasons for performing a code review? Circle all that apply.

    (a) improve the design early in the lifecycle
    (b) improve the design late in the lifecycle
    (c) educate new team members
    (d) find bugs
    (e) check code coverage
    (f) ensure conformance to code style, such as indentation
    (g) evaluate the programmer for promotion
    (h) increase the "bus number" of the code

24. (3 points) Which usually uses more memory?

    (a) depth-first search
    (b) breadth-first search
    (c) neither: usually the same memory usage

25. (3 points) Over the lifetime of a typical successful software project, what percentage of effort is spent on maintenance (as opposed to initial development)?

    (a) 10%
    (b) 50%
    (c) 90%

26. (3 points) In the standard design encouraged by object-oriented languages, which is easier to add?

    (a) new operations on existing objects
    (b) new objects that support existing operations
    (c) equally easy/hard

3 Short answer

27. (3 points) To implement the singleton pattern often (but not always) requires using what other pattern?

    _____

28. (6 points) In one phrase each, state the two key limitations of constructors in Java.

    (a) _____

    (b) _____

29. (4 points) Your goal is to build a highly reliable system, so you run three independently-developed programs (each developed to the same spec) on three separate computers, and use the majority answer. Why doesn't this significantly improve your reliability? Explain in one sentence.

    _____

    _____

    _____

30. (4 points) Under what circumstances does a GUI show an hourglass/clock/spinning ball? Answer in one phrase or sentence. Be specific.

    _____

    _____

    _____

31. (8 points) Consider a wrapper whose implementation logs each call that occurs.

    In no more than 2 sentences each, explain when the wrapper should be considered a decorator (and why), and when that *same* wrapper should be considered a proxy (and why).

    - Decorator: _____

    - Proxy: _____

32. (6 points) It is cheaper and faster to fix known bugs before you write new code. Why? In one phrase or sentence each, give three reasons. Give reasons that are as different from one another as possible.

    - _____

    - _____

    - _____

33. (3 points) In one phrase, what is MapReduce's biggest advantage over older parallel database systems?

34. (8 points) Recall the architecture of MapReduce, which uses two routines with the following signatures:

    - map(k1, v1) → list of ⟨k2, v2⟩
    - reduce(k2, list of v2) → list of v3

    Write pseudocode for an anagram generator. An anagram of a word is a dictionary word that can be obtained by rearranging the letters in the original word. For example, "team" is an anagram of "meat" (and vice versa), but "aemt" is not an anagram because it is not a real word.

    The anagram generator takes as input a dictionary — a list of words. Assume that each k1 is a word. You will not need to use v1; it can be a dummy value.

    The anagram generator produces as output a list of records, where each record maps a word to a list of all its anagrams (including itself). So, the type of v3 is a pair ⟨word, list of words⟩.

    You should only need 1–2 lines of pseudo-code for each of map and reduce. Your pseudocode can be high-level, but must be precise.

    (a) What is the type of k2? _____

    (b) What is the type of v2? _____

    (c) What is the pseudocode for map?

    (d) What is the pseudocode for reduce?