COMP 4081

Exam 2

Fall 2019

Name: _____, _____

Last name First name

Rules:

- No potty breaks.
- Turn off cell phones/devices.
- Closed book, closed note, closed neighbor.

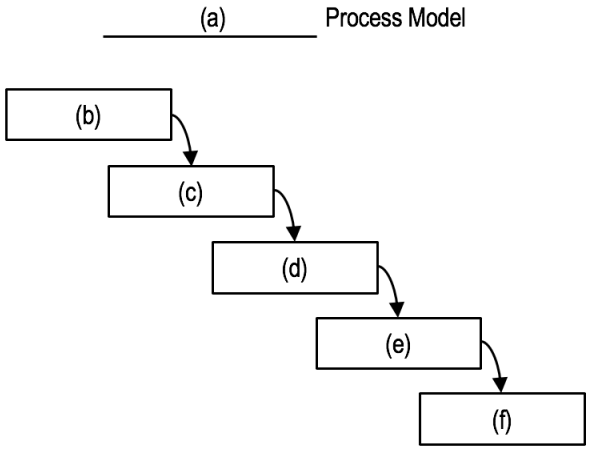
- WEIRD! Do not write on the backs of pages. If you need more pages, ask me for some.

Reminders:

- Verify that you have all pages.
- Don't forget to write your name.
- Read each question carefully.
- Don't forget to answer every question.

Two process model diagrams are given below; however, a number of labels are missing from them. Using the terms in Figure 1, fill in the correct labels for each diagram. Note that some terms may appear in one diagram and not the other, and some terms may not appear in either diagram. Fill in only one term per label. (Note that, for sake of simplicity, a few labels have been omitted entirely from the diagrams. That is, there is no slot given for these labels, and they don't appear in the list of terms.)

1. [5%]



a) _____

b) _____

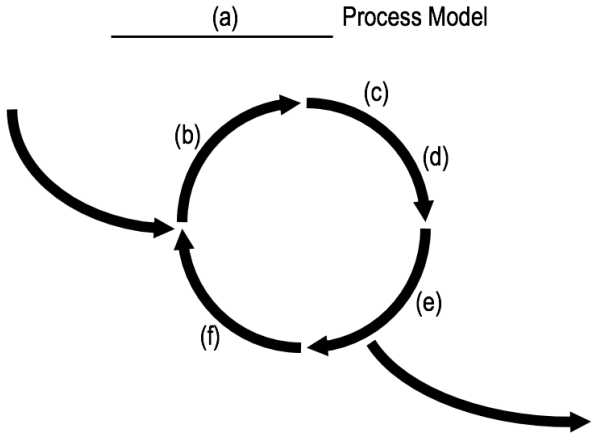
c) _____

d) _____

e) _____

f) _____

2. [5%]



a) _____

b) _____

c) _____

d) _____

e) _____

f) _____

3. [5%]

Which of the above process models (Q1 or Q2) copes better with unstable requirements? Explain why one of the process model handles it poorly, and the other handles it well.

4. [2%]

How long should it generally take to complete a lap of the inner loop of the second process model (Q2)?

5. [2%]

Which term below best fits the following definition?

> *An organization or structure imposed on the tasks and activities involved in developing a software product.*

a) Software verification

b) Software design

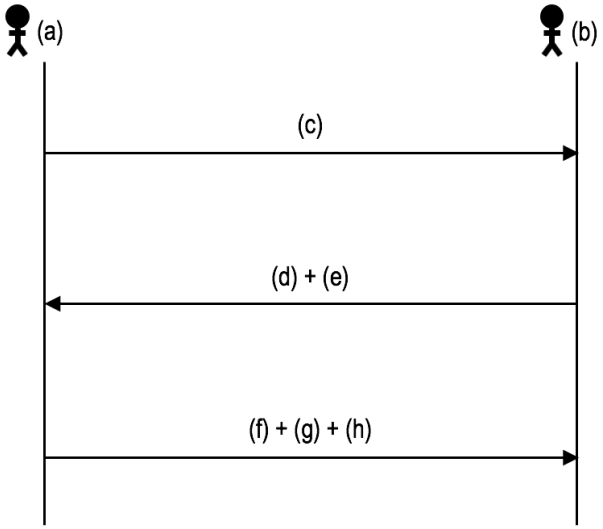
c) Software configuration management

d) Software version control

e) Software engineering process

6. [5%]

Given below is a sequence diagram depicting how people interact in the development process taught in class; however, a number of labels are missing from the diagram. Using the terms in Figure 2, fill in the correct labels for each diagram. Note that some terms may appear multiple times in the diagram and some may not appear at all. Fill in only one term per label. (Note that this diagram, although still able to capture the core ideas, has been changed somewhat from the one shown in class—don't let that fool you!)



a) _____

b) _____

c) _____

d) _____

e) _____

f) _____

g) _____

h) _____

Fill in the blanks below.

7. [2%]

The _____ the estimate, the _____ likely it is to be accurate.

8. [2%]

Two ways to create more accurate estimates are to…

(1) Use the "wisdom of the _____".

(2) Use _____ performance.

9. [2%]

Thinking back to the previous question (Q8), what specific technique did we discuss in class that uses the first way (i.e., the "wisdom" one) to estimate user stories?

Fill in the blanks below.

10. [2%]

To exhaustively test a component, you must create a test for every possible _____.

11. [2%]

_____-box testing emphasizes achieving certain levels of code coverage.

12. [2%]

_____-box testing emphasizes covering boundary conditions.

13. [3%]

_____ tests target individual modules/components in isolation, whereas

_____ tests target groups of interacting modules/components.

14. [4%]

Consider the two test cases in Figure 3. Each of the test cases is missing part of its assertion command. Tell what instruction should be filled in for each blank (a and b), and for each instruction, tell why it is the correct instruction.

(a)

(b)

15. [8%]

Draw a control-flow graph (CFG) for the function in Figure 4. In addition to the usual CFG features, label the basic blocks B1, B2, B3, etc., and label the flows of control F1, F2, F3, etc. Don't forget to include entry and exit points.

Use the CFG you created for the previous question (Q15) to answer the following questions.

16. [6%]

Fill in the table below with a test suite that provides statement coverage. In the Input column, use only the value `[]` or `[3]` for each array. In the Covers column, list the labels (B1, B2, B3, etc.) of the basic blocks covered by each test case. Your test suite must use the minimum number of test cases to achieve this level of coverage. Some rows in the table may be left blank.

Input		Expected Output	Covers
array1	array2		

17. [6%]

Fill in the table below with a test suite that provides branch coverage. In the Input column, use only the value `[]` or `[3]` for each array. In the Covers column, list the labels (F1, F2, F3, etc.) of the flows of control covered by each test case. Your test suite must use the minimum number of test cases to achieve this level of coverage. Some rows in the table may be left blank.

Input		Expected Output	Covers
array1	array2		

18. [6%]

Fill in the table below with a test suite that provides path coverage. Before you fill in the table, first list all the paths to be covered, and label each path P1, P2, P3, etc. You need only cover executions that involve at most 1 iteration of each loop (if there are any). In the Input column, use only the value `[]` or `[3]` for each array. In the Covers column, list the path labels covered by each test case. Your test suite must use the minimum number of test cases to achieve this level of coverage. Some rows in the table may be left blank.

Input		Expected Output	Covers
array1	array2		

For each of the following questions, imagine that the while loop in the function was accidentally changed as shown. Which of the above test suites (statement, branch, path) would have detected the mistake?

19. [2%]

```
while i+1 < array1.length
```

20. [2%]

```
while i < 1
```

21. [2%]

Which of the following best defines the term *coupling*?

a) Design pattern for connecting two objects via an intermediate object

b) Another name for an inheritance relationship

c) A measure of how well focused an object is at doing one thing

d) The extent to which one object depends on other objects

e) The ratio of objects to functions in a system

22. [5%]

Why is too much coupling bad with respect the changeability of a software design?

23. [2%]

Which of the following design patterns encapsulates how a set of objects interact?

a) Coupler

b) Indirector

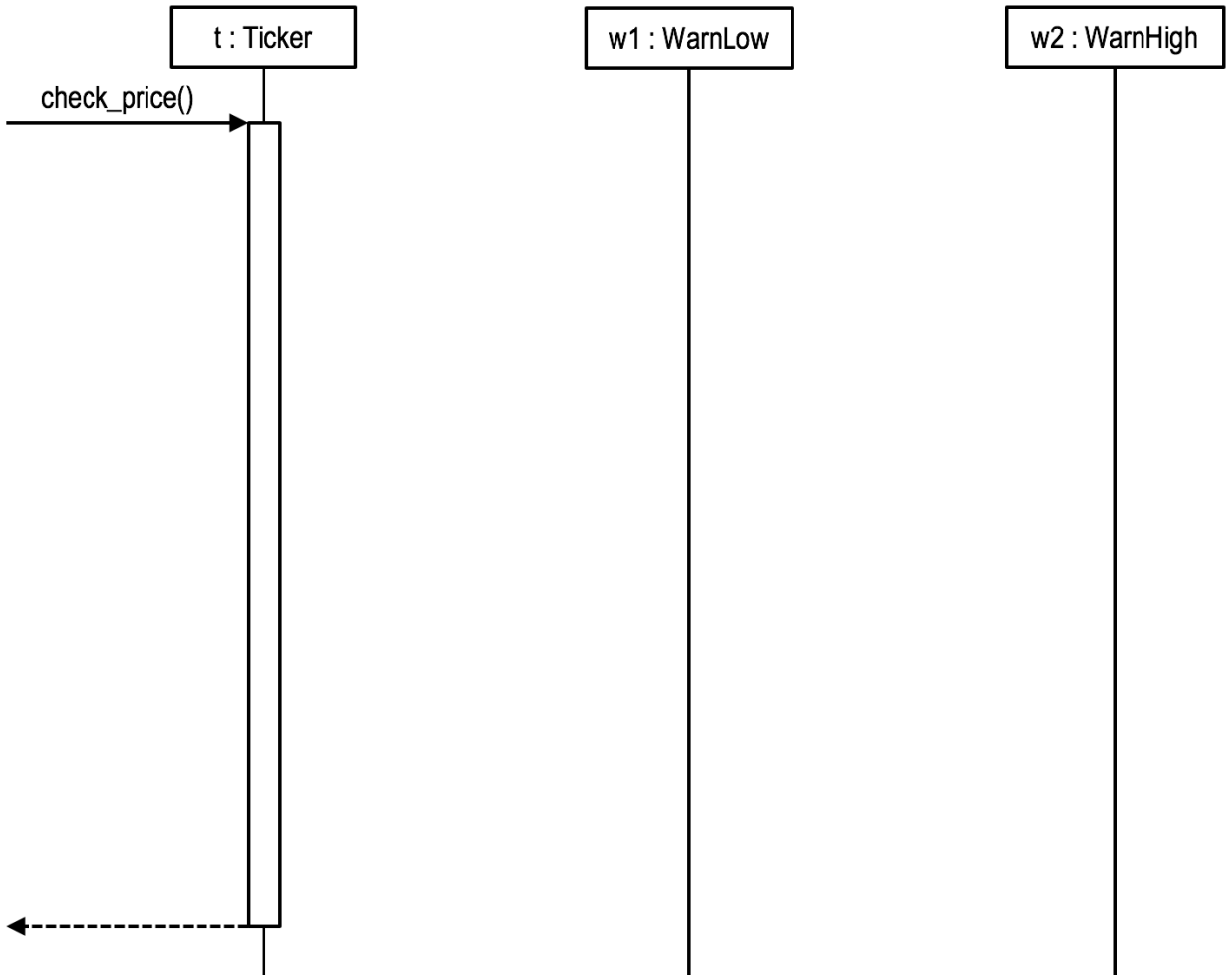
c) Mediator

d) Observer

e) Publish-Subscribe

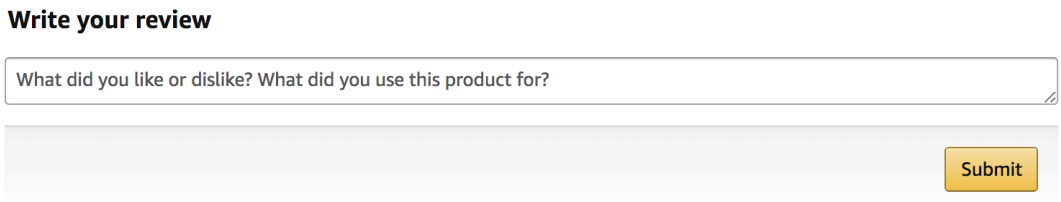
24. [10%]

Consider the application of the Observer Pattern in Figure 5. In the application, there is a stock ticker that can check the price of a stock. Warners observe the ticker, and send notifications to a user if the prices go above or below certain thresholds. The partially completed sequence diagram below depicts a Ticker object (*t*) and two Warner objects (*w1* and *w2*). The Warner objects are already attached to the Ticker object (although it is not depicted explicitly in the sequence diagram). Complete the sequence diagram such that, as per the Observer Pattern, it shows the method calls and returns triggered by the ticker making a price check. Show only calls to methods that are depicted in the class diagram.



25. [3%]

Consider the Amazon website feature for writing a product review:



And the feature that displays the reviews user leave:

In general, how would one attempt to make an XSS attack against this sort of feature? Explain in plain language how and why the attack would work.

26. [3%]

How would one prevent the above attack? Explain in plain language how and why the countermeasure would work.

Figures

- Design
- Evaluation
- Implementation
- Iterative
- Maintenance
- Requirements
- Testing (Verification)
- Version Control
- Waterfall

Figure 1. List of possible terms in process model.

- Customer
- Developer
- Estimates
- Priorities
- User Stories

Figure 2. List of possible terms in sequence diagram.

```
# == Schema Information
#
# Table name: line_items
#
#  id                  :integer          not null, primary key
#  quantity            :integer
#  created_at          :datetime         not null
#  updated_at          :datetime         not null
#  order_id            :integer
#  item_description_id :integer
#

require 'test_helper'

class LineItemTest < ActiveSupport::TestCase

  test "line item should be valid" do
    one = line_items(:one)
    __________(a)__________ one.valid?
  end

  test "quantity must be greater than zero" do
    one = line_items(:one)
    one.quantity = -1
    __________(b)__________ one.valid?
  end

end
```

Figure 3. Test cases with missing assertions.

```
def sum_arrays(array1, array2)
  if array1.length != array2.length
    return nil
  end
  result = []
  i = 0
  while i < array1.length
    result << array1[i] + array2[i]
    i = i + 1
  end
  return result
end
```

Figure 4. Function that sums two arrays. If the lengths of the arrays differ, the function should return `nil`. To the best of my knowledge, this function is correct.

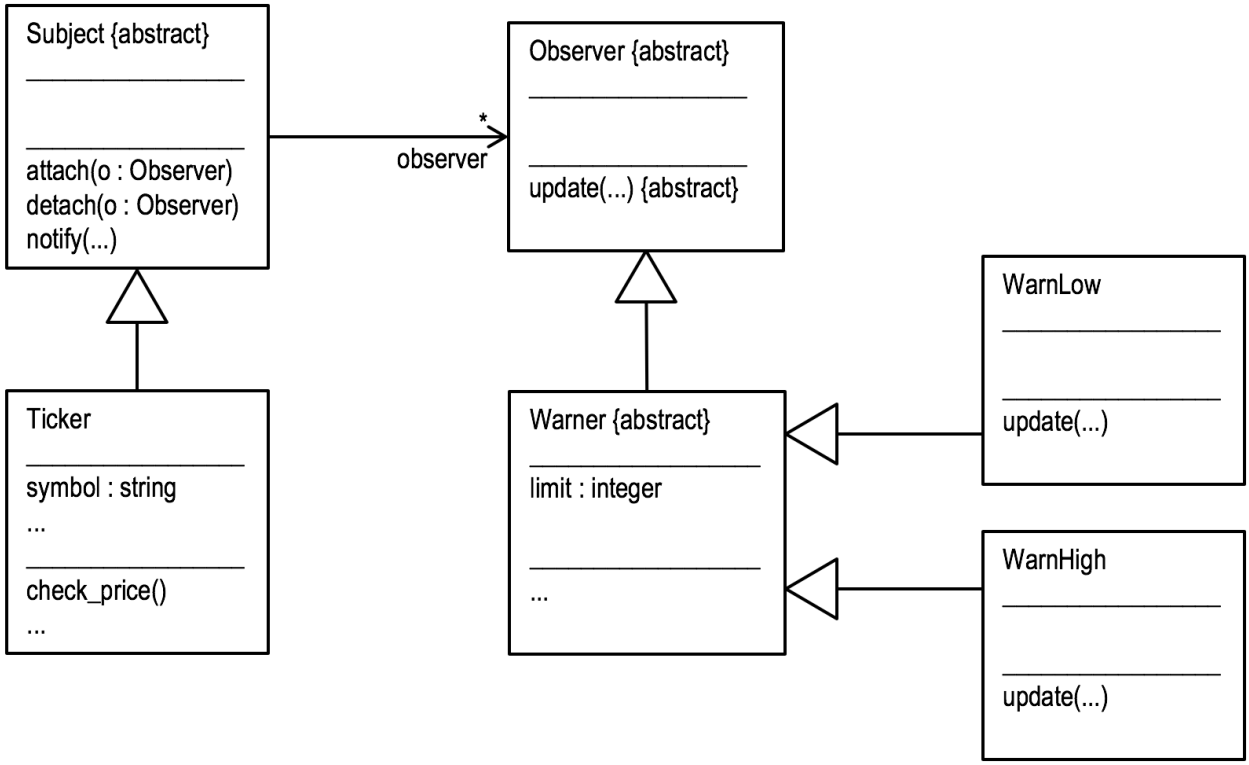


Figure 5. Application of Observer Pattern for a stock ticker application that warns users when a stock price goes above or below a certain amount.