Final Review

CSCE 247 - Lecture 26 - 04/29/2019

We Will Cover

- You have a final next Monday
  - May 6, 4:00 - 6:30 PM
- Topics:
  - Architecture
  - OO Design
  - Design Patterns
  - Class/Sequence Diagrams
  - Structural Testing
  - Dependability
  - Statistical Testing
  - Code Smells/Refactoring
- There is a practice exam on the course site.
- Let’s go over it!

Question 1

Which of the following make sense as classes (rather than objects) in a class diagram?

1. Homework Assignment
2. Manton Matthews
3. Group 5's Assignment 5
4. Person

Which of the following coverage criteria **always** requires more test cases than the others?

1. Statement Coverage
2. Branch Coverage
3. Path Coverage
4. None of the above

Question 1 - True/False

- Requirements-based test cases help the writer clarify the requirements.
- An Object is an instantiation of a Class.
- The goal of testing is to remove defects from the software.
- The use of global variables generally increases coupling.
- An oracle is needed to determine whether a test succeeded.
- Testing can be used to demonstrate that a program is free of faults.
- Path coverage is generally impossible to achieve, but if we could, we would expose all faults in the program.

Question 2

Describe the key difference between black-box testing and white-box testing.

Question 2 - Solution

Black-box testing treats the program as a machine that accepts input and issues output, with no visibility into its internal workings.

- Tests are based on requirements and specifications.
- You do not know what classes or methods are in the code, and you do now know what objects exist at runtime.

White-box involves testing the independent logic paths with full knowledge of the source code. You do not have full knowledge of the intended functionality (white box tests cannot look for unimplemented code).

Question 3

Building a weather monitoring application.

Generates three displays: current conditions, weather statistics, simple forecast.

Design system using either visitor or observer pattern.

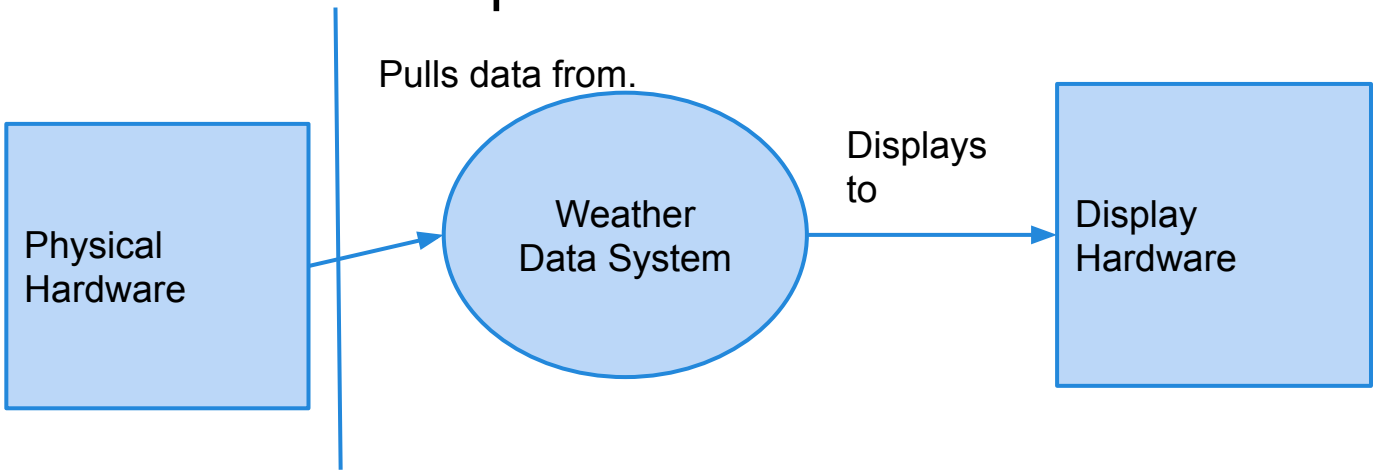
Provided:

Humidity Sensor

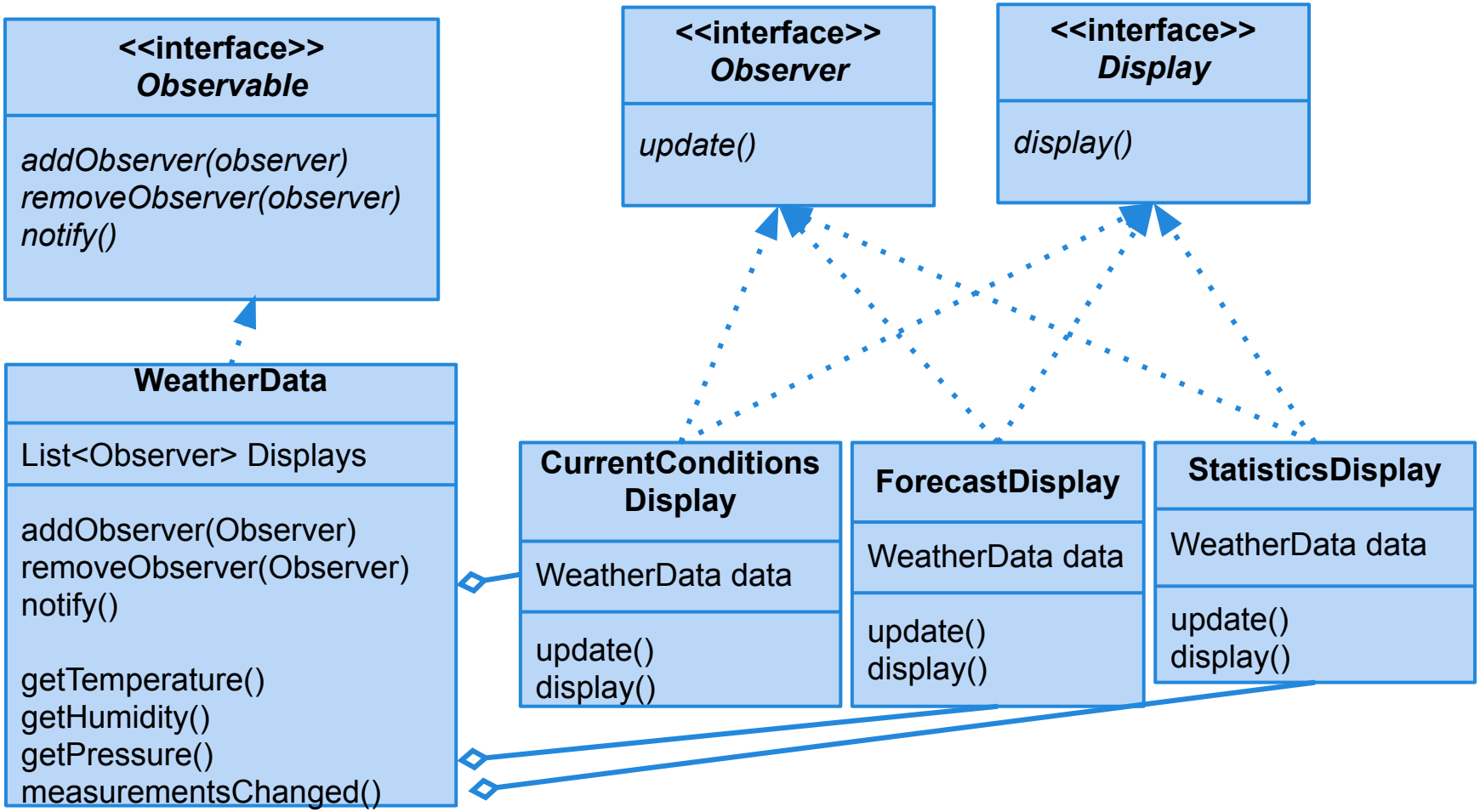
Temperature Sensor

Pressure Sensor

To Implement:



Question 3 - Solution



Question 4

We considered architectural styles, including pipe and filter, layered, and repository.

Suppose that you are to design an automotive system whose subsystems (a-h) are enumerated below. For each style discussed above, choose a subsystem from the set below and describe why this style would be an appropriate structuring mechanism - and why - or describe why this style does not apply to any of the subsystems.

a. On-star communications: manages communications with satellite
b. sensor management: turns noisy sensor data into useful information
c. motion control: operates the motors and provides position and velocity
d. Image processing system to identify highway lanes
e. UX vehicle management involving touch screen
f. Health/status monitoring: checks status of all other subsystems to ensure correct operation
g. Collision avoidance system
h. Dashboard displays

Question 4 - Solution

- Pipe-and-filter:
  - Sensor management; there are several de-noising and sensor fusion transforms that are straightforward to describe using pipe-and-filter architectural styles. Similar arguments could be made for communications with filters for compression / encryption.
- Layered:
  - There is a natural layering between supervisory control, navigation control, and motion control. Communications systems themselves are often layered as well (see TCP runs over IP which runs over physical comms)
- Repository:
  - Health and status monitoring is often performed using a repository architecture, where vehicle health from many systems is aggregated into one place. Also could be used as a data-plane underlying many of the systems mentioned here.

Question 5

Are path coverage and exhaustive testing the same thing? Motivate your answer.

Question 5 - Solution

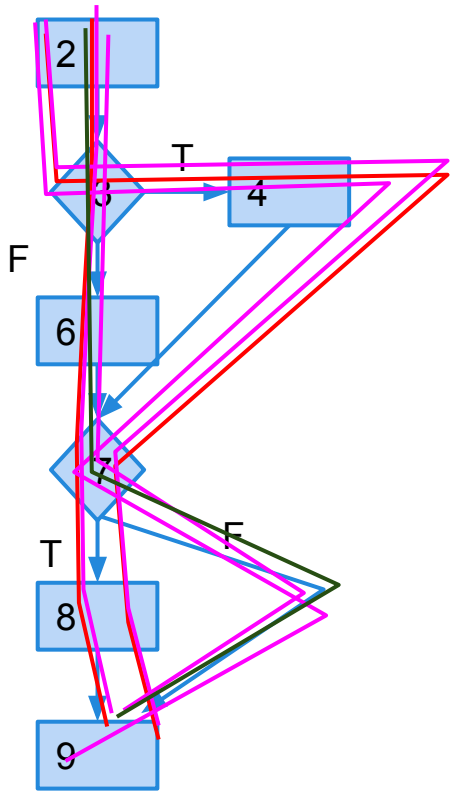
- No. Path coverage “only” requires that every path is exercised; it does not require that every input is tested.
- One can provide path coverage without testing every instance of the inputs that would take you down that path.
  - Problems with divide-by-zero, null-pointer-dereferencing, etc. might not be caught.

Question 6

- Draw the control-flow graph for this method.
- Develop test input that will provide statement coverage.
- Develop test input that will provide branch coverage.
- Develop test input that will provide path coverage.

```
int findMax(int a, int b, int c) {
     int temp;
     if (a>b)
          temp=a;
     else
          temp=b;
     if (c>temp)
          temp = c;
     return temp;
}
```

Question 6 - Solution




```
1. int findMax(int a, int b, int c) {
2.   int temp;
3.   if (a>b)
4.        temp=a;
5.   else
6.        temp=b;
7.   if (c>temp)
8.        temp = c;
9.   return temp;
10. }
```

Statement:
(3,2,4), (2,3,4)

Branch:
(3,2,4), (3,4,1)

Path:
(4,2,5), (4,2,1), (2,3,4), (2,3,1)

Question 6 - Solution

- Modify the program to introduce a fault such that even path coverage could miss the fault.

Use (a >b+1) instead of (a>b) and the test input from the last slide: (4,2,5), (4,2,1), (2,3,4), (2,3,1) will not reveal the fault.

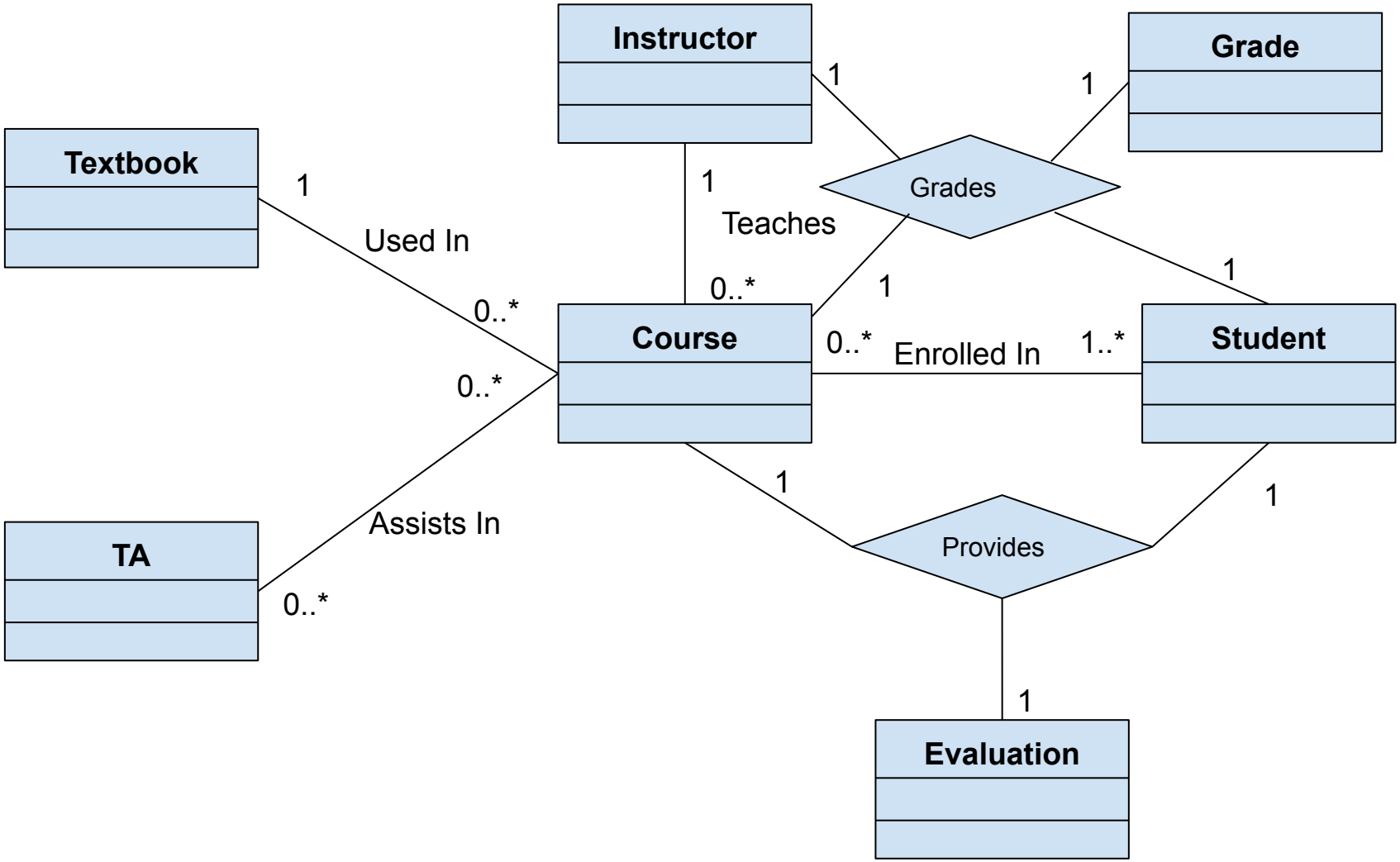
```
int findMax(int a, int b, int c) {
     int temp;
     if (a>b)
          temp=a;
     else
          temp=b;
     if (c>temp)
          temp = c;
     return temp;
}
```

Question 7

Students at USC can be enrolled in more than one class at the time. There is also an option to not be enrolled in any classes (under special circumstances). We do not offer classes with no students at all.
To allocate teaching effort, there is one instructor assigned to each class. Some instructors might not teach any class. Each class uses a textbook (a book that can be used in other classes also). Depending on class size, there are TAs assisting in the class. A small class gets no TAs, a large class might get several TAs. When all is done in the class, the instructor assigns the student a grade for the course. In return, each student must fill out a course evaluation form for the course.

Develop the **class diagram** for the situation described above.

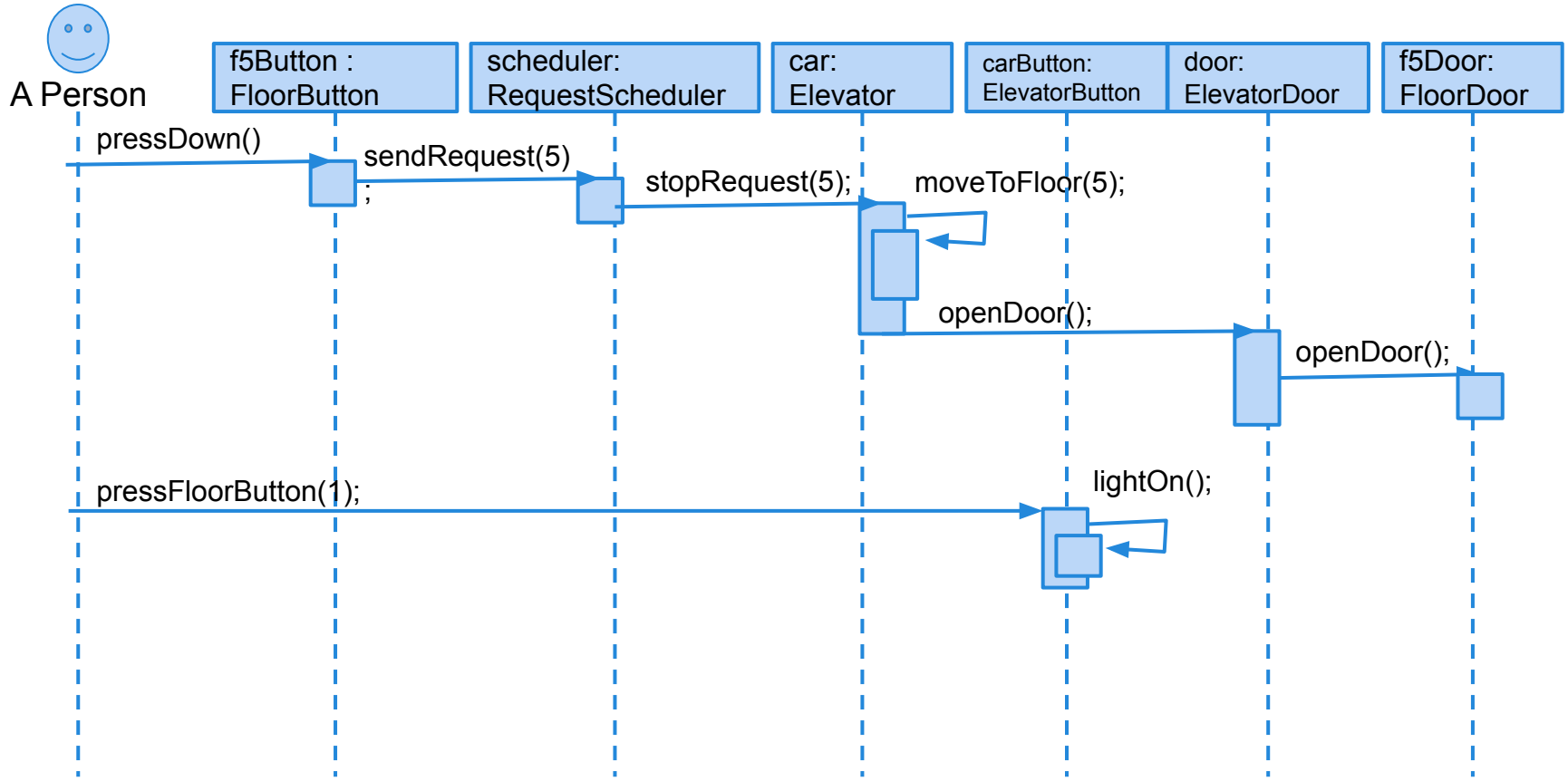
Question 7 - Solution



Question 8 - Scenario 1

Scenario 1 (Requesting a Ride Down):

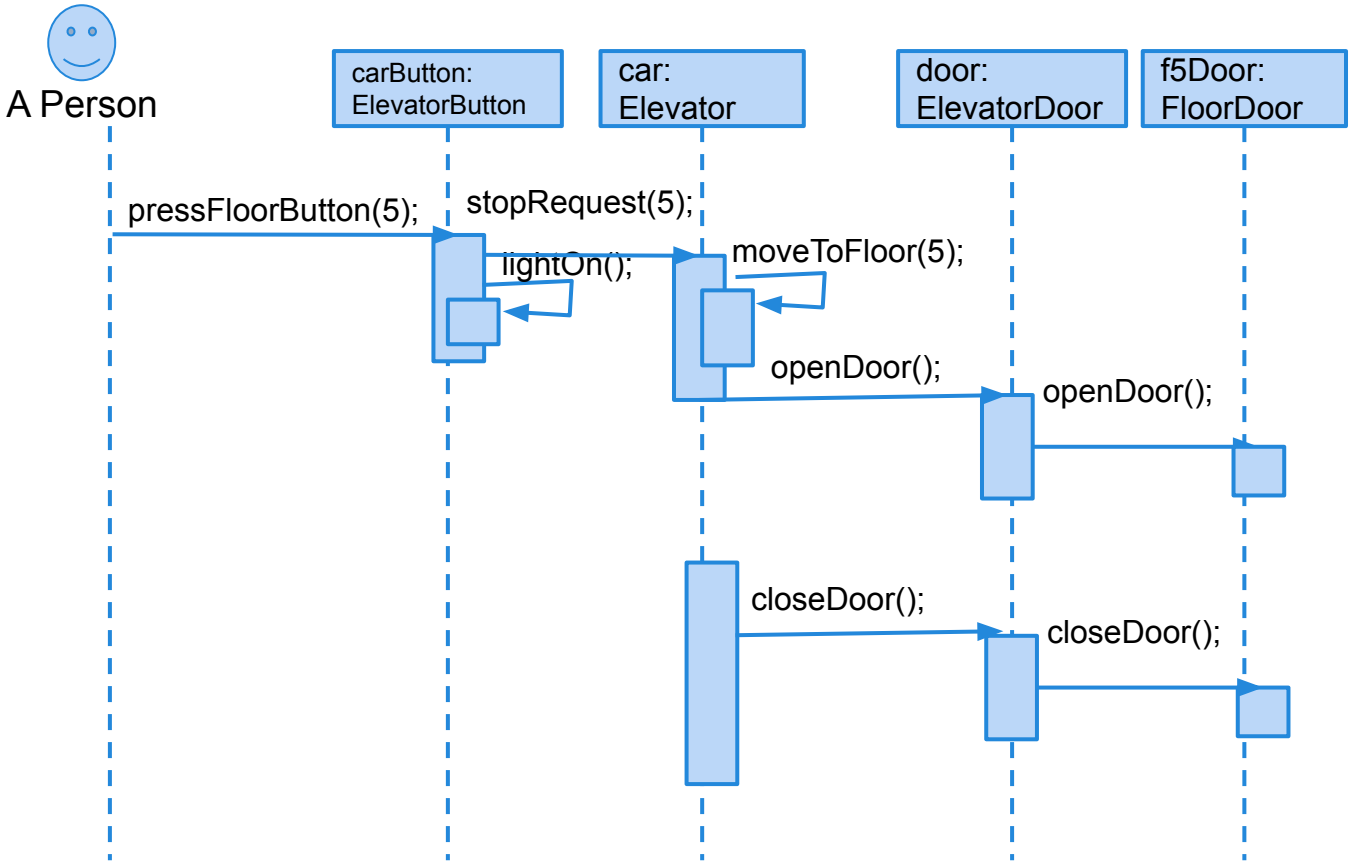
A person approaches the elevator on the fifth floor. She wants to go down so she presses the “down” button next to the elevators. She waits until an elevator arrives and the doors open. She enters the elevator and presses the elevator button for the ground floor (floor 1). The light next to the button for the first floor is lit.



Question 8 - Scenario 2

Scenario 2 (Getting Off at a Floor):

A person is standing in the elevator with the door closed. The person pushes the elevator button for floor 5 (and there are no other requests). The elevator stops at the fifth floor, opens the doors, and the person steps out. The elevator doors close.



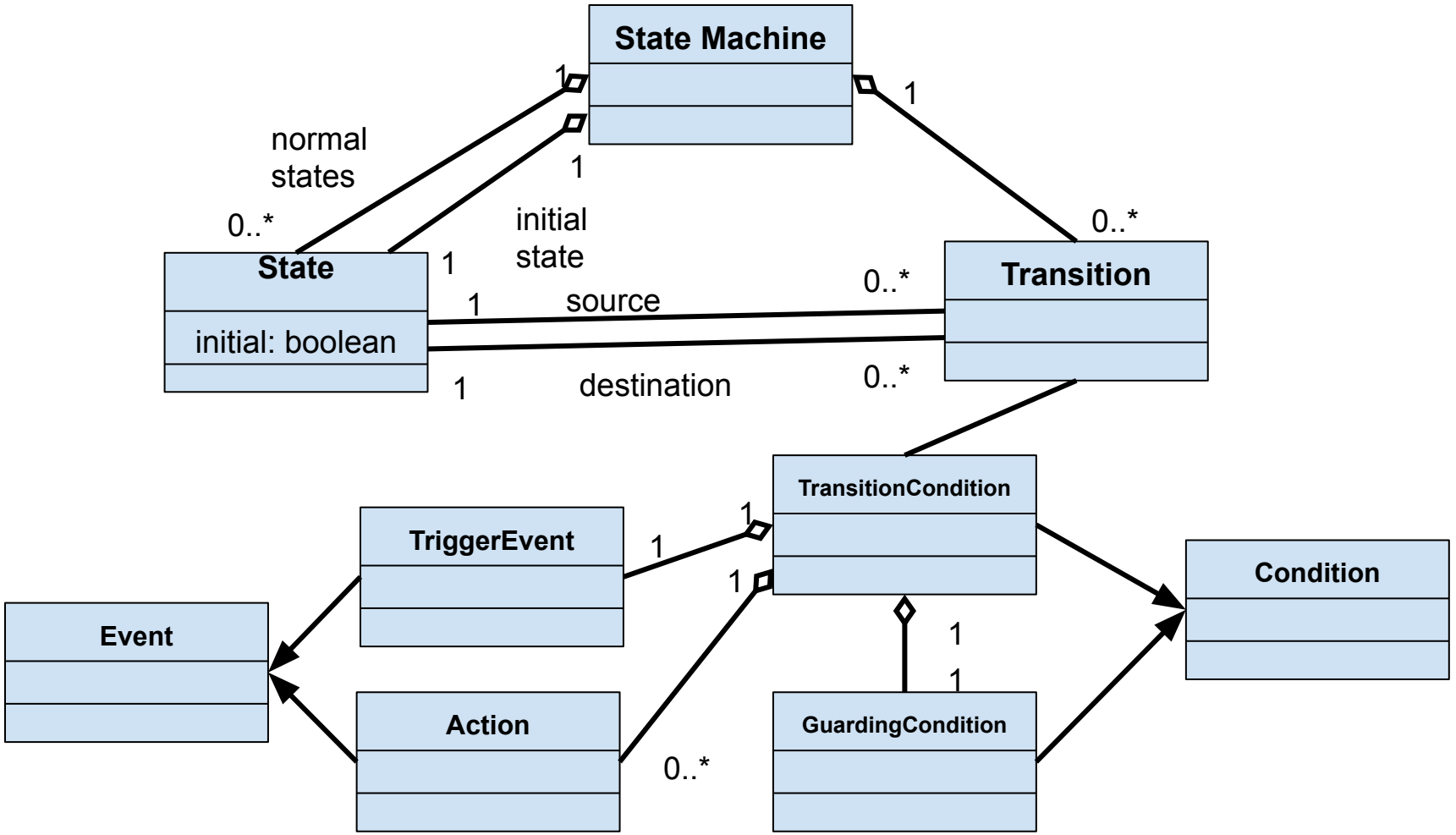
Question 9

You are developing software that will simulate and execute finite state machines.

- A state machine consists of states and transitions.
  - One state is special and designated to be the initial state (this is where we always start). Besides this, the initial state is just like all other states.
  - The transitions have transition conditions associated with them. A transition condition consists of a trigger event, a guarding condition, and a possibly empty set of actions (actions are events generated as a result of taking the transition).

Develop the Class Diagram for this software.

Question 9 - Solution



Question 10

You are building a web store that you feel will unseat Amazon as the king of online shops. Your marketing department has come back with figures stating that - to accomplish your goal - your shop will need an **availability** of at least 98.5%, a **probability of failure on demand** of less than 0.1, and a **rate of fault occurrence** of less than 2 failures per 8-hour work period.

You have recently finished a testing period of one week (seven full 24-hour days). During this time, 972 requests were served to the page. The product failed a total of 64 times. 37 of those resulted in a system crash, while the remaining 27 resulted in incorrect shopping cart totals. When the system crashes, it takes 3 minutes to restart it.

Question 10

You have recently finished a testing period of one week (seven full 24-hour days). During this time, 972 requests were served to the page. The product failed a total of 64 times. 37 of those resulted in a system crash, while the remaining 27 resulted in incorrect shopping cart totals. When the system crashes, it takes 3 minutes to restart it.

1. What is the rate of fault occurrence?
2. What is the probability of failure on demand?
3. What is the availability?
4. What additional information would you need to calculate the mean time between failures?
5. Is the product ready to ship? If not, why not?

Question 10 - Solution

1. *64/168 hours = 0.38 per hour = 3.04 per work period*
2. *64/972 = 0.066*
3. *37*3 = 111 minutes downtime. 111/10080 minutes = 0.011. Avail = 98.9%*
4. *The times that failures occurred. You know how many, but not when.*
5. *No. Avail is good, ROCOF is not.*

Question 11

During development and maintenance, some organizations track “bad fixes” - a bug fix that introduces new faults in the software when the original fault is corrected. The ratio of bad fixes to “good fixes” can be measured.

- For example, the ratio of bad fixes to good fixes could be 1% (there is one bad fix for every 100 good fixes).
- In some troubled projects the bad fix ratio might be over 100%!

What effect will a bad fix ratio of >100% have on software quality?
What do you think would be the main contributor to a very high bad fix ratio? Justify your answer.

Question 11 - Solution

A bad fix ratio over 100% means that more faults are being added than are being fixed. Software quality will deteriorate.

Poorly-structured software is likely to be the culprit. With low cohesion, high coupling, or hard-to-understand algorithms, it is hard to track down the real source of a fault (may only make a partial fix) and easy to introduce new faults (hard to determine the effect of a fix on other parts of the program).

Question 12

A class diagram in UML is generally used during design, but can also be a useful tool in the requirements elicitation stage of a software development project. Discuss briefly how class diagrams might be used in this stage of development.

Question 12 - Solution

UML class diagrams are useful for visualizing entities and their relationships at any level of abstraction.

- Relationships between data items in the problem domain.
- Clarify the relationships between concepts (credit cards and customers, students and professors, students and grades, etc).
- Model can serve as the foundation for natural language requirements by structuring the problem domain.
  - Cleaner and easier to write specifications because we can relate to the diagram.

Question 13

When performing reliability (statistical) testing, an operational profile is absolutely essential for the test-data selection. Why? What is the effect of an inaccurate operational profile?

Question 13 - Solution

Since the reliability metrics are designed to measure the reliability of a system under normal operating conditions it is essential to know what “normal” operating conditions are.

- This is what the operational profile is supposed to capture.
- If the reliability is assessed using a profile that does not accurately capture the real operating conditions, the measure is meaningless.

Any other questions?

Thank you for a great semester!