Congratulations! You passed!
TO PASS 88% or higher

Keep Learning
Retake the assignment in **1h**

GRADE
94.44%

Quality in Design

LATEST SUBMISSION GRADE
94.44%

1. The logical (intended) dependency is that observer depends on subject. **1 / 1 point**

   - (•) True
   - ( ) False

   ✓ Correct

2. Up until now, the great state of Foo has held a lottery to help fund education in the state. The corporation tasked with the drawing of these numbers (the non-televised ones) is XYZ Numerical Tasking. You are the technical lead for the system which handles the drawings for the state's lotteries. One of your developers, with a mathematical tilt, comes to you with a proposal: change how the random numbers are generated. **1 / 1 point**

   He suggests that the generation of random numbers could be better. NASA has released a random number generator which has been proven to be better than the one used by the company. He suggests that you make the change.

   One of your more senior developers notes that using the new generator will require a change. He suggests that the team connect the existing lotto system, as seen in Figure 1, to the new generator (Figure 2) using the Facade pattern.

   Figure 1

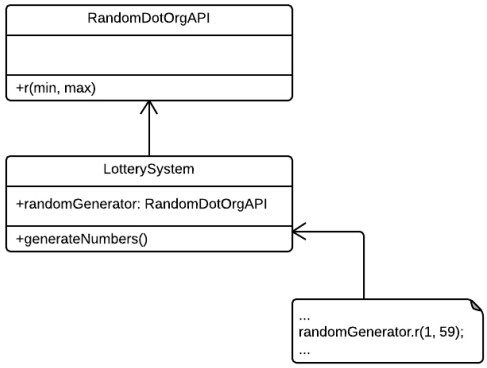
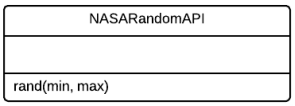
   

   Figure 2

   Facade, which seeks to simplify complex processes for a client by providing a API which hides the complicated calls/work needed, doesn't seem to fit.

   What pattern does?

   Adapter

   ✓ Correct

3. Within the Custodial Management System, there are are a variety of items that are being watched at any given time. Certain items can receive notifications that a response is needed immediately (rather than just tracking whether the regular upkeep has been done that week/month/etc.). These items may then need one of several possible actions: rapid cleaning, bodily fluid containment and/or removal, addressing traction/slip-danger, urgent repair, etc. Each of these actions fit the same profile (action) but not necessarily the same steps. **1 / 1 point**

   Which pattern would fit well as a solution to this systems needs?

   Observer

   ✓ Correct

4. Early in our proof-of-concept work on the new game we are building, we effectively hard-coded the game to ask the user what action to take next, for both players. In this case, both users are tied into a single account, so that we can play against ourselves while we build out the game. **1 / 1 point**

   Now, however, we want to add a new player type to the system, EasyDifficultyBot. While the "player" actions fit the same high-level format for both humans and our AI, the details of making it happen vary pretty widely.

   In order to allow for this change, we want to make use of a pattern that will allow this kind of expansion while minimizing changes to the existing code.

What pattern would be best here?

Strategy

✓ Correct

5. Consider the statement "In the Strategy pattern, the context should hold references to the ConcreteStrategies." 1 / 1 point

Under what circumstances would this be true? Select all that apply.

- [ ] Under no circumstances should the Context hold references to the ConcreteStrategies
- [x] When the context is tasked with deciding which strategy to use

✓ Correct

- [ ] When the client is tasked with deciding which strategy to use
- [ ] When using a Factory object to create Strategies
- [ ] When the Strategy base class uses a static method to return the correct derived type

6. In the Observer pattern, one difficulty is that the Subject cannot pass an instance of itself to the Observer being updated, due to circular dependencies. 1 / 1 point

- ○ True
- ◉ False

✓ Correct

7. Once a class has been had an Adapter pattern solution applied, it cannot be adapted again. 1 / 1 point

- ○ True
- ◉ False

✓ Correct
Please review the Adapter lecture.

8. Factories are no longer useful when you apply the Strategy Pattern to a family of algorithms. 1 / 1 point

- ○ True
- ◉ False

✓ Correct

9. Observer is an implementation of the Dependency Inversion principle. 1 / 1 point

- ◉ True
- ○ False

✓ Correct

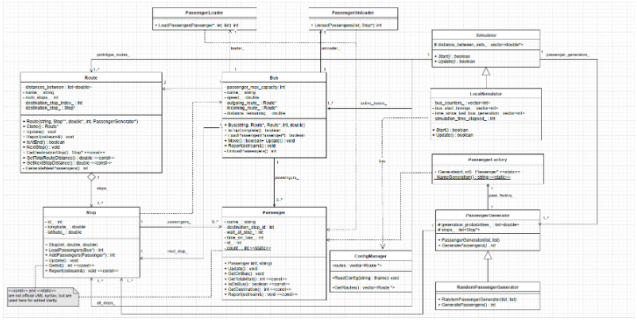
10. Strategy Pattern helps maintain the Open/Closed Principle. 1 / 1 point

- ◉ True
- ○ False

✓ Correct

11. Notes: For questions 11-14, the diagram is a standard UML Class Diagram. You can view a larger version by right clicking and selecting "View image" (or similar) or by holding on the image and selecting "Open image in new tab" (or similar) if you're on mobile. Use up to 3 significant digits (e.g. .5 for 1/2 and .333 for 1/3 are acceptable). 1 / 1 point

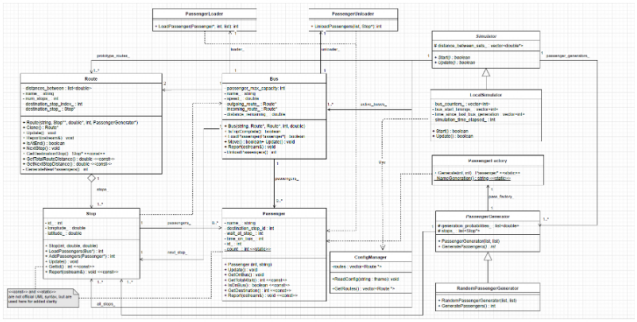
For the following class diagram, calculate the Instability value for the Bus class.



0.714

✓ Correct

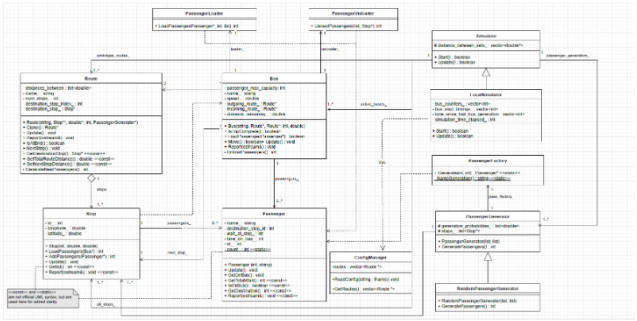
12. For the following class diagram, calculate the Instability value for the Stop class. Use up to 3 significant digits (e.g. .5 for 1/2 and .333 for 1/3 are acceptable). **1 / 1 point**



0.333

✓ Correct

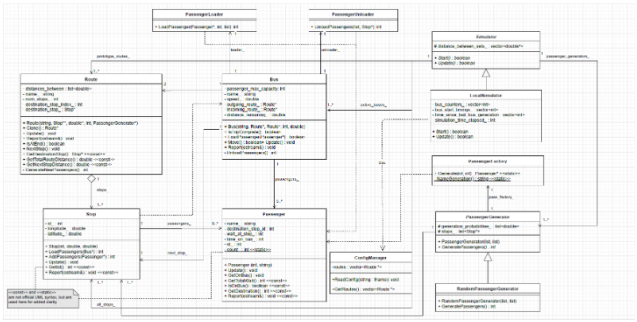
13. For the following class diagram, calculate the Instability value for the PassengerUnloader class. Use up to 3 significant digits (e.g. .5 for 1/2 and .333 for 1/3 are acceptable). **1 / 1 point**



0.5

✓ Correct

14. For the following class diagram, calculate the Instability value for the Simulator class. Use up to 3 significant digits (e.g. .5 for 1/2 and .333 for 1/3 are acceptable). **0 / 1 point**



0.5

! **Incorrect**

See the Quality Metrics lectures

15. While calculating LCOM4, we ignore constructors and destructors. Constructors and destructors frequently set and clear all variables in the class, making all methods connected through these variables, which increases cohesion artificially. **1 / 1 point**

LCOM4 = 1 indicates a cohesive class, which is the "good" class.

LCOM4 >= 2 indicates a problem. The class should be split into so many smaller classes.

LCOM4 = 0 happens when there are no methods in a class. This is also a "bad" class.

Calculate the value of the LCOM4 measurement for the following code:

```
#include "src/route.h"

Route::Route(std::string name, Stop ** stops, double * distances,
    int num_stops, PassengerGenerator * generator) {
  //Constructors ignored in LCOM4 calculation
}

void Route::Update() {
  GenerateNewPassengers();
  for (std::list<Stop *>::iterator it = stops_.begin();
       it != stops_.end(); it++) {
    (*it)->Update();
  }
}

bool Route::IsAtEnd() const {
  return destination_stop_index_ >= num_stops_;
}

void Route::NextStop() {
  destination_stop_index_++;
  if (destination_stop_index_ < num_stops_) {
    std::list<Stop *>::const_iterator iter = stops_.begin();
    std::advance(iter, destination_stop_index_);
    destination_stop_ = *iter;
  } else {
    destination_stop_ = (*stops_.end());
  }
}

Stop * Route::GetDestinationStop() const {
  return destination_stop_;
}

double Route::GetTotalRouteDistance() const {
  int total_distance = 0;
  for (std::list<double>::const_iterator iter = distances_between_.begin();
       iter != distances_between_.end();
       iter++) {
    total_distance += *iter;
  }
  return total_distance;
}

double Route::GetNextStopDistance() const {
  std::list<double>::const_iterator iter = distances_between_.begin();
  std::advance(iter, destination_stop_index_-1);
  return *iter;  // resolving the iterator gets you the Stop * from the list
}

int Route::GenerateNewPassengers() {
  // returning number of passengers added by generator
  return generator_->GeneratePassengers();
}
```

2

Correct

16. Cyclomatic complexity is calculated by the formula: 1 / 1 point

$M = E - N + 2P$,

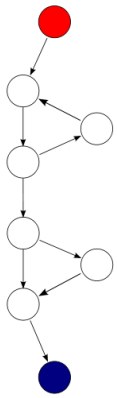
where

E = the number of edges of the graph.

N = the number of nodes of the graph.

P = the number of connected components.

Calculate the cyclomatic complexity for the following control flow graph:



3

Correct

17. Cyclomatic complexity is calculated by the formula: 1 / 1 point

$M = E - N + 2P$,

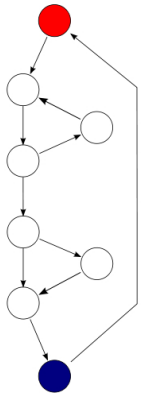
where

E = the number of edges of the graph.

N = the number of nodes of the graph.

P = the number of connected components.

Calculate the cyclomatic complexity for the following control flow graph:



4

Correct

18. Cyclomatic complexity is calculated by the formula: 1 / 1 point

$M = E - N + 2P$,

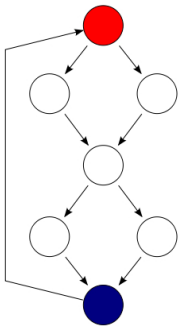
where

E = the number of edges of the graph.

N = the number of nodes of the graph.

P = the number of connected components.

Calculate the cyclomatic complexity for the following control flow graph:



4

Correct