PART I. OO Design Principles. Choose the one best answer for each question.

1. Consider the classic MVP paradigm. Which of the following GRASP principles does not apply to MVP designs?
   A. Controller
   B. Low Coupling
   C. High Cohesion
   D. Creator

Questions 2-4 deal with the code snippet below, a method in the University class.

```
public void calculateReportFor (User u) {
    AccountList list = u.getAccounts();
    int sum = 0;
    int transCount = 0;
    for (Account a : list) {
        sum += a.getBalance();
        TransactionList tl = a.getTransactions();
        for (Transaction t : tl) {
            transCount += 1;
        }
    }
    /* more code follows to format and print the report calculations*/
}
```

2. Given the code above, which of the following principles are violated?
   A. None, the code is well designed.
   B. Tell Don't Ask
   C. Protected Variations
   D. Information Expert
   E. Both B and D are correct

3. Given the code above, it complies with the Law of Demeter.
   A. True
   B. False
   C. Law of Demeter cannot be applied to that code segment

4. Given the code above, which of the following principles are violated?
   A. High Cohesion
   B. Low Coupling
   C. Protected Variations
   D. Pure Fabrication

5. When OO designers say that "Switch Statements are Evil", they are using which of the following principles to justify that statement?

A. Protected Variations
B. Open-Closed Principle
C. Polymorphism
D. Pure Fabrication
E. All of the above answers are correct
F. A, B and C are correct.

6. Following the OO design principles (GRASP and SOLID) are most useful for:

A. Every OO project you will do from now on
B. Only projects over 10,000 lines of code
C. Projects that will have to be maintained and evolved for long periods of time
D. All the above are true.
E. None of the above are true.

7. During a code review, you see code like the following:

```
public void validateAccount(Account a) {
    if (a instanceOf SavingsAccount) {// handle that kind of account }
    else  // do something for all other kinds of account
```

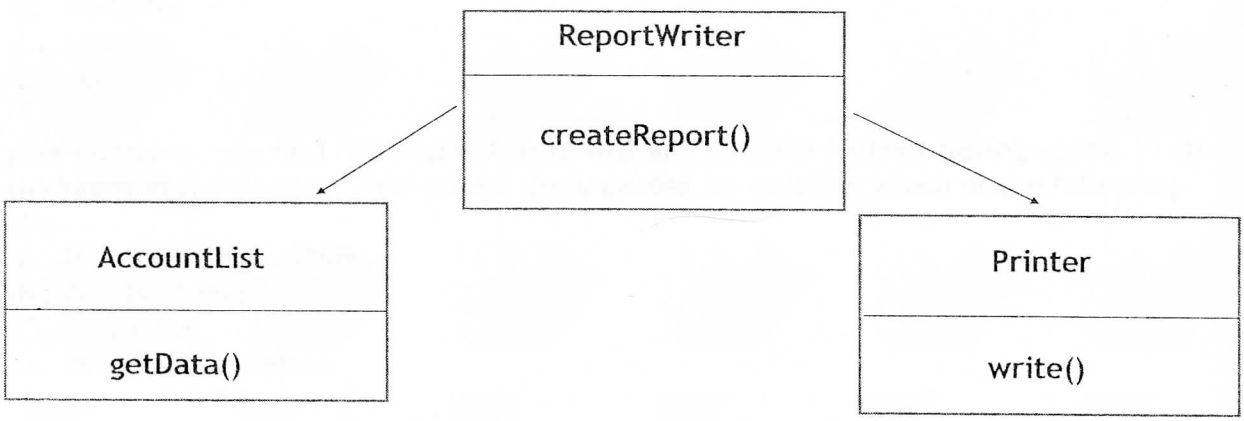
This code:

A. Complies with the PureFabrication principle
B. Violates the PureFabrication principle
C. Complies with the Liscov Substitution Principle
D. Violates the Liscov Substitution Principle
E. None of the above are correct

8. Your classmate who has not had 2340 is trying to design a simple application. They explain that they have a nice UI designed, but they do not know what to do next. Which of the following design principles would most help them take the next step?

A. Creator
B. Polymorphism
C. Interface Segregation
D. Controller

Consider the following diagram:



9. This design could be helped most by applying the following correction:

A. Use dependency inversion. Introduce a Source and Destination interface which Printer and AccountList could implement.

B. Refactor the design such that the ReportWriter class does not need AccountList or Printer, it just does the work itself.

C. Do nothing the design is just fine as it is

D. Use dependency injection. Change the createReport method to take parameters which specify the interface providing information and the interface outputting the report.

E. Both A and D are correct.

10. For the Single Responsibility Principle, which of the following GRASP principles is **least** supportive?

A. Controller

B. High Cohesion

C. Information Expert

D. Protected Variation

PART II. OO Design Patterns. Choose the one best answer to the question.

1. The observer pattern helps us achieve all the following design principles except:
   A. Low Coupling
   B. Open-Closed
   C. Indirection
   D. Pure Fabrication
   E. Liscov Substitution

2. The Factory pattern can help us achieve all the following except:
   A. Elimination of all switch statements
   B. Reuse of objects to avoid construction costs
   C. Creating objects to comply with Open-Closed principle
   D. Caching of database connections

3. The command pattern uses a Stack to manage the undo and redo lists.
   A. True
   B. False
   C. The command pattern has nothing to do with redo and undo.

4. Design patterns are always the best choice and should be used for every situation we encounter as developers.
   A. True
   B. False

5. In your design, you notice that for all the different accounts, the only thing that is different in each subclass is the applyInterest() method. Additionally, your customer has told you that they need the ability to change accounts from one type into another at runtime. The best solution for you would be:
   A. State Pattern
   B. Mediator Pattern
   C. Strategy Pattern
   D. Find another customer
   E. Find another job

6. In Java, the Listener system in Swing is an example of which design pattern?
   A. State
   B. Mediator
   C. Handler
   D. Observer

7. In your company, you find it is impossible to test any package without having access to all the other packages in the design. This is most likely caused by violating which of the following principles:
   A. Interface Segregation
   B. Acyclic Dependency
   C. Controller
   D. Protected Variation

PART III. User Interface Evaluation. Pick the one best answer to the question.

1. Which of the following describes the technique which would check whether your interface meets the Microsoft Usability Guidelines?
   A. Conformance Evaluation
   B. Cognitive Walkthrough
   C. Heuristic Evaluation
   D. Think Aloud Evaluation

2. Which of the following evaluations is most dependent on understanding the demographics of the projected user base?
   A. Conformance Evaluation
   B. Cognitive Walkthrough
   C. Heuristic Evaluation
   D. Think Aloud Evaluation

3. Which of the following evaluations would give you a sense of how likely a new user could operate your system without reading a manual first?
   A. Conformance Evaluation
   B. Cognitive Walkthrough
   C. Heuristic Evaluation
   D. Think Aloud Evaluation

4. The only technique we covered which used real users as opposed to developers was:
   A. Conformance Evaluation
   B. Cognitive Walkthrough
   C. Heuristic Evaluation
   D. Think Aloud Evaluation

5. When conducting a Conformance Evaluation it is important to:
   A. Have the latest standards document
   B. Have a fully functioning User Interface
   C. Have a test subject that has not seen the system before
   D. There is no such thing as a Conformance Evaluation

6. The designer's mental model and the user's mental model need to be the same if the system will be intuitive.
   A. True
   B. False

7. Greying out a menu item unless conditions in the application allow that function is an example of:
   A. Natural Mapping
   B. Affordance
   C. Constraints
   D. Mental Models

PART IV. Miscellaneous Topics. Pick the one best answer for the question.

Consider the following code snippet for 1 questions 1-3:

```
public int findMax(List<Integer> nums) {
    if (nums == null || nums.isEmpty()) return Integer.MIN_INTEGER;
    if (nums.size() == 1) return  nums.get(0);
    int max = nums.get(0);
    for (int i = 0; i < nums.size(); ++i)
        if (max < nums.get(i)) max = nums.get(i);
    return max;
```

. Your new intern gives you the following test cases:
nums = [ ]
nums = [5]
nums = [3,4,5,6]

A. These test cases will achieve statement coverage
B. You will need one more test case to ensure statement coverage, nums = null;
C. You will need one more test case where at least one number is negative.
D. None of the above are correct.

. To achieve branch coverage, you need the following additional test cases beyond question 1.
A. nums = null, nums = [8, 3, 4, 9]
B. nums = null, nums = [-2, 3, 4, 5]
C. nums = null
D. nums = [8,3,2,1]
E. None, question 1 also provides branch coverage

. If you could achieve branch coverage without using negative numbers, and you add an dditional test with some negative numbers then:
A. You are wasting development time on a useless test
B. You are testing for plausible errors
C. You could not get branch coverage without having negative numbers

. You are writing a number guessing game where the user has to enter a number between 1 and 0 (inclusive) and enter a 99 to quit the game. Using equivalence partitions, you would need the ollowing inputs:
A. You cannot test this spec using partitions.
B. -4, 1, 20, 44, 99, 1000
C. -543, 14, 55, 99, 1300
D. 1, 15, 20, 99

esting the game from question 4 using boundary conditions would require t

A. You cannot use boundary conditions for this situation
B. 0, 1, 20, 21, 98, 99, 100
C. 0,1, 20, 21
D. -1, 0, 20, 21, 98, 99

hich of the following is NOT true about testing?

A. Passing all our tests shows the code is defect-free.
B. We should test all possible inputs to ensure we have no defects.
C. Testing should begin after the entire application is finished.
D. White-box testing to branch coverage is sufficient to find all defects.
E. All the above are false.