CSC/ECE 517 Final Exam, Spring 2016, review versio (10577312)

Due: Sat Dec 8 2018 11:59 PM EST

Question | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10

Description

On this test are 10 questions of various types. There is no individual time limit for any of these questions.

Instructions

There are 96 points on this test. The maximum score is 100. If you earn more than 90 points, your final score will be 90.

Rules

1. You are welcome to consult your own book, notes etc. as well as look up online resources to help you with answering the questions. But be careful not to spend too much time looking up answers on the Internet! It may be more effective to start working with what you already know.
2. You are not allowed to talk to another person, or receive help from someone through IM, phone or other similar devices, during the test.
3. You are welcome (and encouraged) to save (by hitting the "save" button at the bottom of the screen). This will allow you to resume your test even after you close the main browser window. However, please be advised that no extra time will be given to complete the exam, even if you have your exam "paused."
4. We strongly suggest that you use one browser (e.g., Chrome) to take the test, and another browser (e.g., Firefox) to browse the web. **Also, please clear cookies and your browser cache before starting the exam.** This will help avoid situations in which work can be saved but not submitted.
5. You can "submit" ONLY ONCE! Do not submit prematurely (but please do "save" frequently).

1. Question Details Design patterns 1 [3605287]

(2 points each) Which design pattern is most useful for each of the following situations?

(a) In the future, you are implementing a system for time travel. It is to allow you to travel to any era for your life, and later, you can return to the present, or to another era you have been to. Once you return, everything is as it was when you left.

Memento ✖ Memento

(b) Here is a physical analogy to a design pattern. Which pattern? When you call Microsoft tech support, you deal with a Level 1 consultant. If (s)he can't answer your question, eventually you are routed to a Level 2 consultant. If this consultant can't answer your question, you are transferred to a Level 3 consultant.

Chain of Responsibility ✖ Chain of Responsibility

(c) Suppose we are designing a hierarchy of cars for an ordering system. The system has a notion of a basic car. Instead of being a basic car, the vehicle we are ordering could also be a sports car, a minivan, or an SUV. Various features can be added to each of these cars: premium sound system, rear video camera, collision avoidance, etc. Cars with each set of features behave in a certain way. But we don't want to create separate subclasses for each combination of cars and features.

Decorator ✖ Decorator

(d) A calculator program allows the user to enter an operand, or specify an arithmetic operation to be performed. After each operation is performed, the user can then specify operands and operators for the the next calculation.

Interpreter ✖ Interpreter

If you are unsure of any answers, you may explain them below for possible credit.

Key: .

Solution or Explanation

Prototype Observer Chain of Responsibility (also Template Method was allowed) Adapter

2. Question Details Design patterns 2 [3603471]

(2 points each) Which pattern is most useful for each of the following situations? (*Note:* The same pattern will not be used more than once.)

(a) Allow users to scan over and access the elements of a collection without knowing the underlying representation of the collection.

Iterator ✗ Iterator

(b) You are tasked with designing an application for scheduling a conference room. This process uses web services from the facilities team for heating, IT team for audiovisual and with security to grant access to the rooms at the booked time. However, to save employees' time you create simple interface which takes care of this in the background. What pattern is used here.

Facade ✗ Facade

(c) You are designing a web interface for both computers and mobile devices. Depending upon the type of the device, appropriate interfaces for the underlying platform should be invoked.

Abstract Factory ✗ Abstract Factory

(d) A design pattern that requires making the constructor private.

Singleton ✗ Singleton

If you are unsure of any answers, you may explain them below for possible credit.

Key: .

3. Question Details Design Patterns 3 [3603472]

(2 points each) Which GoF pattern is most useful for each of the following situations? (*Note:* The same pattern will not be used more than once.)

(a) In our system, the mobile site is different from the desktop site. However, some users wish to see the desktop site on their mobile devices. When such a request is made, we handle the request and show the user the desktop site. What pattern are we using here.

Adapter ✗ Adapter

(b)You are developing a game, and your task is to change an actress's physical positions based on her previous conditions. For instance, If actress is standing, and user presses "Up", she should jump. If in same position user presses down she should sit. If actress is sitting, and user presses "Up", she should stand up. What design pattern should be used here.

State ✗ State

(c) We are building Taypals add a payment option feature. A user can add a credit card, debit card, or bank account. The processing steps differ depending on the given payment option.

Strategy ✗ Strategy

(d)

```
                class Car {
                  //At ABC showroom a car can be an Audi/BMW, can be a an automatic or manual transmissio
                  // Can have airbags or not. And it may be a convertible.
        }
        class myPattern{
                    void setMake(String Make)      {//set the make}
                    void setTransmissionType()     {//make the system remember the transmission type}
                    void setConvertible()          {//make the system remember this is a convertible)  }
                    void setWithAirbags()          {//make the system remember this car comes with airbags}

        }
```

To build a car the operator will send each option step by step and then construct the final car with the right options.

Builder ✗ Builder

If you are unsure of any answers, you may explain them below for possible credit.

Key: .

4. Question Details Statements about design patterns [3886583]

(2 points each)

(a) Select the **correct** statement.

- ○ Flyweight pattern implementations can be memory intensive, hence its use should be limited.
- ○ In the Visitor pattern, a new operation cannot be defined without changing the classes of the elements on which it operates.
- ○ The Mediator pattern allows objects to interact with each other directly.
- ◉ The Bridge pattern decouples a set of implementations from set of objects using them. ✖

(b) Select a **false** statement about the observer design pattern. (Subject in this question is used for object observed.)

- ◉ Using this pattern, Subjects and Observers can push messages back and forth to each other. ✖
- ○ This pattern defines a one-to-one dependency between objects.
- ○ The Observer is responsible for registration, not the subject.
- ○ There is high coupling between observer base class and subject.

(c) Select a **false** statement about the Proxy design pattern.

- ○ Proxies can be used to invoke remotely located services.
- ○ The Proxy pattern provides control of an object by providing a placeholder for it.
- ○ Both Decorator and Proxy patterns delegate execution of a method to the subject.
- ◉ Proxy usually implements a different interface than the interface implemented by the object it is serving as a proxy for. ✖

5. Question Details Design patterns coded [3602028]

(4 points each) Select an answer for each question from the dropdown list. You may write your reasoning in text box provided below for partial credit.

(a) Which design pattern is used in this program? Select from the dropdown list below.

```
public class ClassA{
   protected ClassB objectA;
   protected String propertyA;
   public ClassA(ClassB obj1,String obj2){
      this.objectA=obj1;
      this.propertyA=obj2;
   }
   public void MethodA(String message){
      objectB.methodA(message);
   }
   public void MethodB(String message){
      system.out.println(this.propertyA+": "+message);
   }
}

public class ClassB{
   private list listClassA;
   public ClassB(){
      listClassA=new list();
   }
   public void AddClassA(ClassA classA){
      this.listClassA.add(classA);
   }
   public void MethodA(String message, ClassA classA){
      for(Class obj:listClassA){ //iterate over each element of the collection
        if(obj!=classA)
         obj.MethodB(message);
      }
   }
}
```

Correct answer is: Mediator ✘ Mediator

(b) Which design pattern is used in this program? Select from the dropdown list below.

```
interface InterfaceA{void Method1();}

class ClassA implements InterfaceA{
   private int property1;
   public ClassA(int obj){property1=obj;}
   public void Method1(){//do something}
}
class ConcreteClass implements InterfaceA{
   private List children=new ArrayList();
   public void Add(InterfaceA obj){children.add(obj);}
   public void Method1(){
      for(int i=0;i < children.size();i++)
       ((InterfaceA) children.get(i)).Method1();
   }
}
```

Correct answer is: Visitor ✘ Visitor

(c) Which design pattern is used in this program? Select from the dropdown list below.

```
public interface InterfaceA {
   public MyType getMyType();
}
public class ClassB implements InterfaceA {
   public ClassB( InputStream in ) {
      //convert inputStream information of typeB (local type) to MyType and assign it to objectMyType.
   }
   public MyType getMyType() {
      return objectMyType;
   }
}
public class ClassC implements InterfaceA {
   public ClassC( InputStream in ) {
      //convert inputStream information of typeC (local type) to MyType and assign it to objectMyType.
   }
   public MyType getMyType() {
      return objectMyType;
   }
}
```

Correct answer is: Factory Method ✖ Factory Method

If you are unsure of any answer above, you may explain it here for partial credit.

Key: .

6. Question Details Elegance and Classes [3603386]

In each of the following questions, two different implementations are given, that have the same objective. Choose the better implementation, and tell which guideline from the list below is most appropriate.

1. Different responsibilities should be divided among different objects.
2. Encapsulation: One class should be responsible for knowing and maintaining a set of data, even if that data is used by many other classes
3. Expert pattern: The object that contains the necessary data to perform a task should be the object that manipulates the data
4. The DRY principle: Code should not be duplicated
5. Design your classes so that they can handle change
6. Code to interfaces, not classes
7. Encapsulate the concepts that vary
8. Give classes a complete interface
9. A well-formed class has a consistent interface

(a)

Example 1:

```
public class Banker {

        public updateUserEmail(User current_user, String message){
                current_user.email = message;
        }

}
public class User{

        String email;
}
```

Example 2:

```
public class Banker {

        public updateUserEmail(User current_user, String message){
                current_user.updateEmail(message);
        }

}
public class User{

        String email;
        public updateEmail( String message){
                this.email = message;
}
```

Answer: 2 ✗ 2
Guideline: 3 ✗ 3

(b) Example 1:

```
        public class Auto{

                  private String make;
                  private String color;

                  public String getColor() {
                     return color;
                  }

                  public String getMake() {
                     return make;
                  }
```

```
                public void setColor(String newColor) {
                   color = newColor;
                }

                public void setMake(String newMake) {
                   make = newMake;
                }
             }

        public class AutoDealer{

                public static void main() {
                        Auto a1 = new Auto();
                        a1.setColor ("green");
                }
        }
```

Example 2:

```
        public class Auto{

                   public String make;
                   public String color;

                   public String getColor() {
                      return color;
                   }

                   public String getMake() {
                      return make;
                   }

        }
        public class AutoDealer{

                public static void main(){
                        Auto a1 = new Auto();
                        a1.color = "green";
                }
        }
```

Answer: 1 ✖ 1
Guideline: 2 ✖ 2

c) Let us extend example 1 from the previous question. Now we need to print the details of the auto. Example 1:

```
        public class Auto{

                   private String make;
                   private String color;

                   public String getColor(){
                      return color;
                   }

                   public String getMake(){
                      return make;
                   }

                   public void setColor(String newColor){
                      color = newColor;
                   }
```

```
                public void setMake(String newMake){
                   make = newMake;
                }
             }

        public class AutoDealer{

                public static void main(){
                        Auto a1 = new Auto();
                        Auto a2 = new Auto();
                        System.out.println("Make of the car a1 is : " + a1.getMake());
                        System.out.println("Color of the car a1 is : " + a1.getColor());
                        System.out.println("Make of the car a2 is : " + a2.getMake());
                        System.out.println("Color of the car a2 is : " + a2.getColor());
                }
        }
```

Example 2:

```
        public class Auto{

                private String make;
                private String color;

                public String getColor(){
                   return color;
                }

                public String getMake(){
                   return make;
                }

                public void setColor( String newColor){
                   color = newColor;
                }

                public void setMake(String newMake){
                   make = newMake;
                }

                public void printDetails(){
                        System.out.println("Make of the car a1 is : " + this.getMake());

                        System.out.println("Color of the car a1 is : " + this.getColor());
                }
        }

        public class AutoDealer{

                public static void main(){
                        Auto a1 = new Auto();
                        Auto a2 = new Auto();
                        a1.printDetails();
                        a2.printDetails();
                }
        }
```

Answer: 2 ✖ 2
Guideline: 4 ✖ 4

Comments.

Key: .

7. Question Details Programming by contract [3604014]

(2 points each, except 4 points for (e)) This question asks about preconditions and postconditions for a method that calculates the `arcsin(x)` function.

(a) What would be a good precondition for `y = arcsin(x)`?

Key: .

(b) What would be a good postcondition for the method? *Note:* The postcondition should involve `sin`.

Key: .

(c) What is the obligation for the client in using the `arcsin` method?

Key: .

(d) What is the benefit for the client in using the `arcsin` method?

Key: .

(e) Suppose we wanted to incorporate the precondition into the postcondition. Give the new postcondition.

Key: .

Solution or Explanation

(a) $-1 \le x \le 1$ (b) $\sin(y) = x$ and $-\pi/2 \le y \le \pi/2$ (c) Pass in a value that is between −1 and 1. (d) Have the value of the arcsin returned. (e) $\sin(y) = x$ and $-\pi/2 \le y \le \pi/2$. If $x < -1$ or $x > 1$, an IllegalArgumentException is thrown.

8. Question Details Elegance and Inheritance II [3603863]

(a) *(3 points)* A Rectangle class has following structure:

```
class Rectangle {
  public:
    double getHeight();
    void setHeight(double h);
    double getWidth();
    void setWidth(double w);
};
```

A Square is made a subclass of Rectangle.
Does the above scenario violate the Liskov Substitution Principle? ---Select--- Yes

(b) How will you design the given classes considering the guidelines taught in class?
For each of the following pairs of dropdowns ...

(2 points each) In the firstdrop down, select whether -
There is an is-a relationship between/among these classes.
There is a has-a relationship between/among these classes.
There is both an is-a and a has-a relationship between/among these classes.
There is neither an is-a nor a has-a relationship between/among these classes.

(1 point each) In the second dropdown -
If it is a is-a relation, select the superclass.
If it is has-a relation, select the class which is included in the other class/classes.
If it is both an is-a and a has-a relation, select the option with the superclass first and then the class included in the other class/classes.

(i) List, ArrayList, LinkedList
---Select---
There is an is-a relationship between/among these classes
---Select--- List

(ii) Car, Mustang, Wheel
---Select---
There is both an is-a and a has-a relationship between/among these classes
---Select--- Car, Wheel

(iii) Person, Veteran, Teacher
---Select---
There is a has-a relationship between/among these classes
---Select--- Person

9. Question Details Multiple-choice potpurri [3605503]

(2 points each)

a) What is not true about Test-First Development? Select one.

- Both programmers and testers create their test cases before code development starts.
- System requirement documentation can be used while writing test cases to make sure they cover all possible behaviors.
- First write test cases; then write code so that all the test cases pass all the time.
- Test cases should cover all the use cases of the class.

b) Which statement about Git is false?

- To use the "lock, modify, unlock" approach in version-control systems, the entire project must be locked when any part of it is being worked on.
- Git uses hashes rather than names to identify files.
- A branch in git can be used to for a version of the project that contains a specific feature.
- In a git push, only new or modified files are copied from the local to the remote repository.

c) Which statement about Rails is false?

- Partial templates can be used to increase reusability.
- Active Record objects represent instances of model classes.
- "Has-one" and "belongs-to" are both one-to-one Active Record associations.
- Controllers must contain code to validate models before they are persisted in the database.

If you are unsure of any answer, you may explain it for partial credit.

Key: .

10. Question Details Review question-Overriding&Overloading [2803670]

(12 points) Fill in the blanks so that the output is "BBBACpinkD". Please select one of the variables initialized (o, auto, sedanAuto, sedan) as a part of your answer.

```
enum Color{
   black,pink,blue,gray
}

class Automobile{
  Color color;
  public Automobile() {
    color = Color.pink;
  }
  public Automobile(Color c) {
     color = c;
  }
  public void print(Object o) {
    System.out.print("A"); //version A
  }
   public void print (Automobile o){
    System.out.print("B"); //version B
  }
}

class Sedan extends Automobile{
   Sedan(Color c) {
    color = c;
   }
   public void print(Automobile o){
    System.out.print("C" + o.color); //version C
   }
   public void print(Sedan o){
    System.out.print("D"); //version D
   }
}

public class OverridePrint {
   public static void main(String[] args) {
    Object o = new Automobile();
    Automobile auto = new Automobile();
    Automobile sedanAuto = new Sedan(Color.black);
    Sedan sedan = new Sedan(Color.gray);
    auto.print( [          ] auto -or- sedanAuto -or- sedan );
    auto.print( [          ] sedanAuto -or- auto -or- sedan );
     [          ] auto .print(sedan);
    sedan.print( [          ] o );
    sedan.print( [          ] auto );
    sedan.print( [          ] sedan );
   }
}
```

Assignment Details