2009 November 23

Example Test Questions for Patterns

1.

The Composite design pattern is used to compose classes into tree structures in order to represent containment relationships. The pattern lets you treat objects and compositions of objects in the same way.

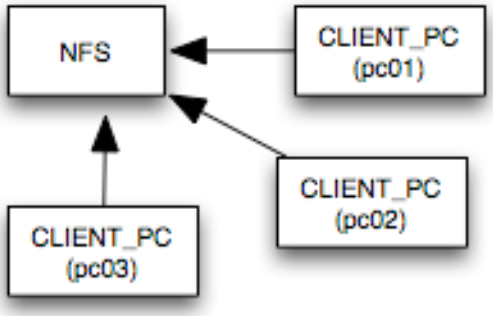
Use the Composite pattern, together with BON, to model the notion of a folder in Windows XP. Folders may be nested, and may also contain text files and binary files. Files may be opened, closed, and drawn on the screen. Folders may also have items added to and removed from them. Draw a static BON diagram modeling this notion. Show the interface of each class.

2.

In a few sentences, explain the intent and motivation of the Builder pattern. Give an example of its use, drawn using BON. Do not show the general structure of the pattern; show how it might be used in a real software system.

3.

The Observer design pattern is used to define a many-to-one dependency among objects. When one object (called a subject) changes its state, all of its dependents are notified and are updated automatically. Use the observer pattern in BON to model a network file system, which consists of a remote server and a number of client personal computers. Using BON, represent the remote server and the clients as classes and draw their interactions, so that the server and the clients satisfy the observer pattern (i.e.. clients are notified of changes to the server). You may use BON static and dynamic models, as well as simple contracts, to make your design clear.



4.

In this question you are expected to use your knowledge on design patterns to suggest a solution for a given scenario. For each of the following scenarios, state which design pattern(s), of the ones described in the course, can best solve the problem. Describe how you would use the patterns (which pattern participants are responsible for doing what) to solve the problem. Briefly identify and discuss related issues. Only high-level descriptions are required. You do not need to provide pattern details.

A A set of cities is maintained as an interconnected graph structure. A web based application needs to maintain multiple views of the distances between cities. Assume that the GRAPH class provides operations for obtaining the distance between any two cities. New roads are always constructed between cities and as a result the distances change. One view provides a table of distances between cities in kilometers, and another view maintains the same information in miles.

B In the problem of Part A, an application needs to collect various kinds of information about the cities such as the least and most crowded cities, the city with the largest mall, etc. Assume each

object representing a city is equipped with necessary operations to obtain the population, size of the largest mall, etc.

C The Canadian Government needs to set certain attributes for the Canadian provinces based on characteristics of each province and federal policies. For instance, health care and education budgets allocated to the provinces use formulas that take into account regional differences. There are many other operations whose components depend on provincial data. New operations are introduced every year and some operations may be removed.

D A pizza factory produces pizzas with various toppings. There are 20 different toppings and a customer may order any combination of toppings. Assume that each of pizza bread and each topping will be represented by a different class.

5.

A. Using Bon, give a generic static diagram of the <Name> pattern and include relevant interface features.

B. Give a scenario and object communication diagram (dynamic model) for the <Name> pattern.

6.

Consider the class PERSON below. Only features relevant to this question are shown.

```
class PERSON
feature
      house: HOUSE  ;  car: CAR  ;  stereo: STEREO

      describe is
        do house.describe  ;  car.describe  ;  stereo.describe end

      sell is
        do house.sell  ;  car.sell  ;  stereo.sell end
end
```

Our customer indicates that it is unlikely that we will need to keep track of things other than a PERSON's HOUSE, CAR, or STEREO in the future. However, we might need to add more operations similar to describe and sell. Indicate what design pattern might be useful in this situation, and why. Draw a BON diagram with expanded classes showing the features and their signatures but not the contracts. Describe, either in English or Eiffel, what the implementation of each new feature will look like.

7.

A Briefly, explain the purpose of the *Decorator* pattern. Describe the pattern in BON.

B Briefly, explain the purpose of the *Composite* pattern. Describe the pattern in BON.

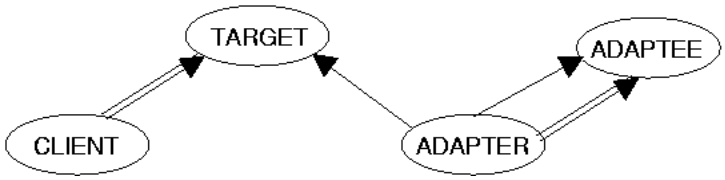
C It has been suggested that the Decorator design pattern is a degenerate instance of the Composite Design pattern. Explain what this means.

8.

Describe two key advantages of applying design patterns. Describe one typical disadvantage of applying design patterns.

9.

Answer the following questions with respect to the following Adapter pattern diagram.



A What is the responsibility of the ADAPTER class?

B What is the responsibility of ADAPTEE class?

C The relationship between ADAPTER and ADAPTEE can be either client-supplier or inheritance. Briefly explain the implications of using one against the other.

10.

The Eiffel program text, given on the last 2 pages, implements the Composite Design Pattern for a psuedo biological classification system. Base your answers for the following questions on that program text.

A Provide a BON static diagram for the given classes. Make sure to indicate if a feature is deferred or effective. Label all association and aggregation links.

B Complete the *display* feature and *invariant* clause for the COMPOSITE_GROUP class in the space provided below:

```
class COMPOSITE_GROUP

      -- rest of the code deleted

feature
  display is
      -- display all sub-groups
end  -- class COMPOSITE_GROUP
```

C In the ROOT_CLASS below complete the body of *make* in the space provided below.

```
class  ROOT_CLASS
creation make
feature
  make is
      -- Create a family with one species composed of a mammal and two  fish and then
      -- display the family

      local
            family: FAMILY
            species: SPECIES
            group: GROUP
      do


  end
end   -- class ROOT_CLASS
```

Question ?? – Reference page, you may tear off this page

```
indexing
    description: "Components having children and
                  an optional parent"

class
    COMPOSITE [T]

feature

    parent: COMPOSITE [T] is
            do
            end

    has (child: T): BOOLEAN is
        -- does 'child' belong to the composite?

    add (new_child: T) is
        -- add `new_child' to the composite

    remove (child: T) is
        -- remove child from the composite

feature {NONE}

    children: LINKED_LIST [T]

invariant

    children_not_void: children /= void

end -- class COMPOSITE
```

```
-- features of class COMPOSITE defined here

remove (child: T) is
    -- remove child from the composite
    require
        child_not_void: child /= void
    do
        from children.start
        until
            children.after or child = children.item
        loop
            children.forth
        end
        if not children.after then
            children.remove
        end

        ensure
            removed: not has (child)
        end

has (child: T): BOOLEAN is
    -- does 'child' belong to the composite?
    require
        child_not_void: child /= void
        do
            Result := children.has (child)
        end

add (new_child: T) is
    -- add `new_child' to the composite
    require
        new_child_not_void: new_child /= void
        do
            children.put_front (new_child)
        ensure
            added: has (new_child)
        end
```

Question ?? – Reference page, you may tear off this page

```
Indexing description: "A group"
deferred class GROUP
feature
    name: STRING    -- Name of this group
    display is
      -- display the group
          deferred
          end

    make (s: STRING) is
      require
          s_not_void: s /= void
      do
          name := s
      end
end -- class GROUP
```

```
Indexing
    description: "Composite group"
class
    COMPOSITE_GROUP
inherit
    COMPOSITE [GROUP]
    GROUP
      redefine make end

feature {NONE}
  make (s: STRING) is
      -- Create and initialize the children
      -- as a LINKED_LIST
      do
          Precursor (s)
          create children.make
      end

feature
  display is
    -- display all sub-groups – to be completed
       ???
    invariant Parent_child_link: ???
end -- class COMPOSITE_GROUP
```

```
Indexing description: "A mammal"
class MAMMAL
  inherit GROUP
creation make
feature
    display is
          -- Display a mammal
          do print("mammal") end
end -- class MAMMAL
```

```
Indexing description: "A species with groups"
class SPECIES
  inherit COMPOSITE_GROUP
creation make
end -- class SPECIES
```

```
Indexing description: "A fish"
class FISH
  inherit GROUP
creation make
feature
    display is
          -- Display a fish
          do print("fish") end
end -- class FISH
```

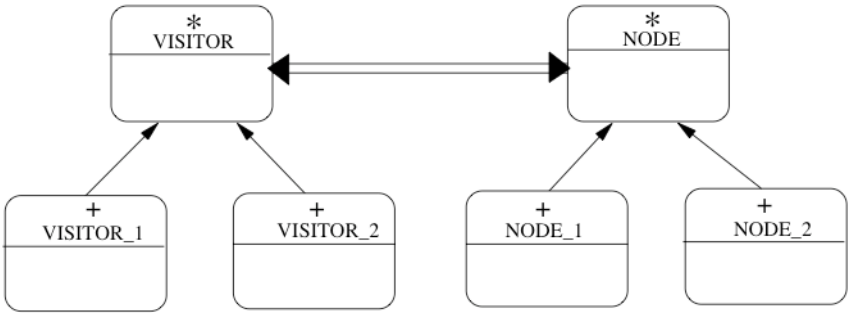
```
Indexing description: "A family of species"
class FAMILY
  inherit COMPOSITE_GROUP
creation make
end -- class FAMILY
```

11.

A In a few sentences, explain the <Name> pattern.
B Describe when the <Name>pattern would applicable?
C Give an example of the use of the <Name>pattern, written in Bon.

12.

The following is a simplified BON diagram for the Visitor pattern.



A Describe the features required in each deferred class, and a typical effective class in each hierarchy, to support the pattern.

B Suppose a class *NODE_C* is added as a subclass of *NODE*. List and describe the required changes to all of the classes affected by the addition.

C Would you advise using the Visitor Pattern if the *NODE* hierarchy changed frequently? Explain your answer.

D Describe the type of applications that are suitable for the Visitor Pattern.