**Ôn thi cuối kỳ Công nghệ phần mềm**

1. Để quản lý danh sách có thứ tự các phần tử, ta có thể dùng:

Class tổng quát hóa HashSet<T>

Class tổng quát hóa List<T>

Tùy sở thích mà dùng a hay b

Cả 3 câu trên đều không giải quyết được yêu cầu

1. Trong các mẫu thiết kế đã học, mẫu nào cung cấp 2 loại class pattern và object pattern:

Composite

Adapter

Proxy

Façade

1. Trong đại đa số mẫu thiết kế, phần tử cha đóng vai trò như

1 abstract class

1 interface sử dụng

1 class tổng quát hóa

Cả 3 câu trên đều sai

1. Để tạo 1 ứng dụng multi-platform dễ dàng, dễ nâng cấp, ta nên dùng mẫu thiết kế:

Adapter

Abstract Factory

Kết hợp các mẫu thiết kế trên

Cả 3 câu trên đều sai

1. Mẫu thiết kế phục vụ việc thêm động trách nhiệm cho đối tượng:

Virtual Proxy

Remote Proxy

Protection Proxy

Smart Proxy

1. Chọn câu đúng nhất:

Mẫu Smart Proxy là trường hợp đặc biệt của mẫu Decorator

Mẫu Decorator là trường hợp đặc biệt của mẫu Smart Proxy

2 mẫu Smart Proxy và Decorator tương đương nhau

Cả 3 câu trên đều sai

1. Khi client muốn tương tác với một họ trong tập hợp các họ đối tượng và việc chọn họ đối tượng được xác định tại thời điểm run-time, ta dùng mẫu thiết kế:

Composite

Template method

Abstract Factory

Factory method

1. Để một class chuyển quá trình thực hiện một nhiệm vụ nào đó cho một trong các class con nhưng cho phép ứng dụng xác định class con cụ thể, ta dùng mẫu thiết kế:

Template method

Factory method

Cả 2 câu a và b đều đúng

Cả 2 câu a và b đều sai

1. Mối quan hệ giữa các thành phần trong các mẫu thiết kế là:

Hiện thực và thừa kế

Kết hợp (association)

Bao gộp

Cả 3 câu trên đều đúng

1. Khi hệ thống cần độc lập với các đối tượng mà nó sinh ra và các đối tượng sinh ra thuộc cùng một class và có rất ít điểm khác biệt, ta dùng mẫu thiết kế:

Abstract factory

Method Factory

Prototype

Builder

1. Muốn xác định động tại thời điểm chạy tập các đối tượng xử lý 1 request nào đó, ta nên dùng mẫu thiết kế:

Proxy

Composite

Chain of responsibility

Cả 3 câu trên đều đúng

1. Muốn tập trung các hành vi giống nhau của các class để tránh trùng lắp, ta nên dùng mẫu thiết kế:

Prototype

Strategy

Template method

Factory method

1. Để giúp client tạo đối tượng phức hợp theo cơ chế tích lũy tăng dần, ta dùng mẫu thiết kế:

Abstract Factory

Factory method

Prototype

Builder

1. Muốn tránh việc dùng các lệnh kiểm tra số học có nhiều khả năng xảy ra, ta nên dùng mẫu thiết kế:

State

Strategy

Cả 2 câu a và b đều đúng

Cả 2 câu a và b đều sai

1. Muốn định nghĩa sự phụ thuộc 1-n giữa các đối tượng sao cho khi 1 đối tượng thay đổi trạng thái thì các đối tượng phụ thuộc được cảnh báo dấu hiệu chỉnh tự động, ta dùng mẫu thiết kế:

Command

Chain of responsibility

Observer

State

1. Mỗi mẫu thiết kế thường chứa:

1 vài loại thành phần

Từ 3-5 loại thành phần

Từ 4-10 loại thành phần

Trên 10 loại thành phần

1. Đa số các mẫu thiết kế thuộc nhóm “Creational” thuộc loại

Class pattern

Object pattern

Class pattern lẫn object pattern

Cả 3 câu trên đều sai

1. Mẫu thiết kế Command thuộc loại:

Class pattern

Object pattern

Class pattern lẫn object pattern

Cả 3 câu trên đều sai

1. Để tạo ứng dụng cho phép người dùng dễ dàng cá nhân hóa hệ thống menu-bar, ta nên dùng mẫu thiết kế:

Composite

Adapter

Command

Observer

1. Trong ứng dụng cần tạo nhiều số ngẫu nhiên ở nhiều nơi, để đảm bảo ứng dụng chỉ dùng được 1 đối tượng tạo số ngẫu nhiên duy nhất, ta dùng mẫu thiết kế:

Composite

Adapter

Command

Singleton

1. Thường hệ thống đã cung cấp đối tượng tạo số ngẫu nhiên, để đặc tả đối tượng tạo số ngẫu nhiên theo interface của ứng dụng được dễ dàng, ta dùng mẫu thiết kế:

Composite

Adapter

Command

Singleton

1. Class Singleton cần phải có tính chất nào sau đây:

Hàm khởi tạo phải có tầm vực private

Chứa tác vụ static có nhiệm vụ trả về tham khảo đến đối tượng duy nhất của class

Chứa thuộc tính static tham khảo đến đối tượng duy nhất của class

Cả 3 câu trên đều đúng

1. Biểu thức gồm nhiều phần tử cấu thành như hằng, biến, giá trị cụ thể, … Để đặc tả các thành phần cấu thành biểu thức sao cho cùng cung cấp interface duy nhất cho ứng dụng, ta dùng mẫu thiết kế:

Composite

Adapter

Command

Chain of responsibility

1. Mẫu thiết kế nào có sử dụng abstract class:

Adapter & Composite

Composite & Proxy

Proxy & Abstract Factory

Abstract Factory & Factory Method

1. Mẫu thiết kế nào có sử dụng mối quan hệ bao gộp:

Adapter & Composite

Composite & Observer

Observer & Strategy

Strategy & State

1. Mẫu thiết kế nào có sử dụng mối quan hệ kết hợp (association):

Adapter & Composite

Composite & Observer

Observer & Strategy

Adapter & Proxy

1. Hai mối quan hệ có mẫu thiết kế hỗ trợ là:

Kết hợp & bao gộp

Bao gộp & 1-n

1-n & hiện thực

Hiện thực & thừa kế

1. Để quản lý 1 họ đối tượng được dùng trong phần mềm cho mỗi lần chạy, ta có thể dùng mẫu thiết kế:

Abstract Factory

Factory Method

Cả 2 câu a và b đều đúng

Cả 2 câu a và b đều sai

1. Điều kiện cần và đủ để 1 class trở thành Singleton là:

Tất cả các constructor đều private

Ít nhất 1 constructor không tham số và phải private

Không chứa bất kỳ constructor nào

Cả 3 câu trên đều sai

1. Điều kiện cần và đủ để 1 class trở thành Prototype là:

Phải override tác vụ Clone()

Không cần làm vì hết class Object đã có tác vụ Clone()

Phải định nghĩa mới tác vụ Clone()

Cả 3 câu trên đều sai

1. Tác vụ getInstance() của class Singleton cần phải có tính chất:

Public

Static

Cả 2 câu a và b đều đúng

Cả 2 câu a và b đều sai

1. Môi trường .Net hỗ trợ trực tiếp mẫu thiết kế:

Singleton

Prototype

Strategy

Observer

1. Trong class Singleton, lệnh new 1 đối tượng thuộc class Singleton:

Phải xuất hiện chỉ 1 lần

Phải nằm ngoài thân của tất cả các tác vụ

Cả 2 câu a và b đều đúng

Cả 2 câu a và b đều sai