

OOAD Review Final Exam

Midterm Semester II (Main Questions) (Role, Use, UML Class Diagram, Code)

1. ចូរនិយាយពីត្បូនាទី (Role) របស់ Decorator Design Pattern ដោយមានភ្លាច់ខាងក្រោម។
2. ចូរនិយាយពីការប្រើប្រាស់ (Use) Adapter design pattern ដោយមានភ្លាច់ខាងក្រោម។
3. ចូរគូរ UML Class diagram ទាំងនឹងបញ្ជាផណ៌យ ដែលមានអនុវត្ត Prototype design pattern។

OOAD Review Final Exam

1. Creational Design Pattern:

- Prototype Pattern
- Factory Method Pattern
- Singleton Pattern
- Builder Pattern

2. Structural Design Pattern:

- Decorator Pattern
- Bridge Pattern
- Composite Pattern
- Adapter Pattern

3. Behavioral Design Pattern

- Strategy Pattern
- State Pattern
- Command Pattern
- Template Method Pattern
- Observer Pattern

Creation Design Pattern

1. Prototype Pattern:

1. ចូរនិយាយពីតួនាទី (Role) របស់ **Prototype** ដោយមានភ្លាប់ខាងក្រោមបញ្ជាក់។
2. ចូរនិយាយពីការប្រើប្រាស់ (Use) **Prototype** ដោយមានភ្លាប់ខាងក្រោមបញ្ជាក់។
3. ចូរគួរ UML Class diagram ទាំងនឹងបញ្ជាណាមួយ ដែលមានអនុវត្ត **Prototype**។

2. Factory Method Pattern:

1. ចូរនិយាយពីតួនាទី (Role) របស់ **Factory Method** ដោយមានភ្លាប់ខាងក្រោមបញ្ជាក់។
2. ចូរនិយាយពីការប្រើប្រាស់ (Use) **Factory Method** ដោយមានភ្លាប់ខាងក្រោមបញ្ជាក់។
3. ចូរគួរ UML Class diagram ទាំងនឹងបញ្ជាណាមួយ ដែលមានអនុវត្ត **Factory Method**។

3. Singleton Pattern:

1. ចូរនិយាយពីតួនាទី (Role) របស់ **Singleton** ដោយមានភ្លាប់ខាងក្រោមបញ្ជាក់។
2. ចូរនិយាយពីការប្រើប្រាស់ (Use) **Singleton** ដោយមានភ្លាប់ខាងក្រោមបញ្ជាក់។
3. ចូរគួរ UML Class diagram ទាំងនឹងបញ្ជាណាមួយ ដែលមានអនុវត្ត **Singleton**។

4. Builder Pattern:

1. ចូរនិយាយពីតួនាទី (Role) របស់ **Builder** ដោយមានភ្លាប់ខាងក្រោមបញ្ជាក់។
2. ចូរនិយាយពីការប្រើប្រាស់ (Use) **Builder** ដោយមានភ្លាប់ខាងក្រោមបញ្ជាក់។
3. ចូរគួរ UML Class diagram ទាំងនឹងបញ្ជាណាមួយ ដែលមានអនុវត្ត **Builder**។

Structural Design Pattern

5. Decorator Pattern:

- ចូរនិយាយពីតួនាទី (Role) របស់ **Decorator** ដោយមានភ្លាប់ខាងក្រោមបញ្ជាក់។
- ចូរនិយាយពីការប្រើប្រាស់ (Use) **Decorator** ដោយមានភ្លាប់ខាងក្រោមបញ្ជាក់។
- ចូរគួរ UML Class diagram ទាំងនឹងបញ្ជាណាមួយ ដើលមានអនុវត្ត **Decorator** ។

6. Bridge Pattern:

- ចូរនិយាយពីតួនាទី (Role) របស់ **Bridge** ដោយមានភ្លាប់ខាងក្រោមបញ្ជាក់។
- ចូរនិយាយពីការប្រើប្រាស់ (Use) **Bridge** ដោយមានភ្លាប់ខាងក្រោមបញ្ជាក់។
- ចូរគួរ UML Class diagram ទាំងនឹងបញ្ជាណាមួយ ដើលមានអនុវត្ត **Bridge** ។

7. Composite Pattern:

- ចូរនិយាយពីតួនាទី (Role) របស់ **Composite** ដោយមានភ្លាប់ខាងក្រោមបញ្ជាក់។
- ចូរនិយាយពីការប្រើប្រាស់ (Use) **Composite** ដោយមានភ្លាប់ខាងក្រោមបញ្ជាក់។
- ចូរគួរ UML Class diagram ទាំងនឹងបញ្ជាណាមួយ ដើលមានអនុវត្ត **Composite** ។

8. Adapter Pattern:

- ចូរនិយាយពីតួនាទី (Role) របស់ **Adapter** ដោយមានភ្លាប់ខាងក្រោមបញ្ជាក់។
- ចូរនិយាយពីការប្រើប្រាស់ (Use) **Adapter** ដោយមានភ្លាប់ខាងក្រោមបញ្ជាក់។
- ចូរគួរ UML Class diagram ទាំងនឹងបញ្ជាណាមួយ ដើលមានអនុវត្ត **Adapter** ។

Behavioral Design Pattern

9. Strategy Pattern:

- ចូរសិយាយពីតួនាទី (Role) របស់ **Strategy** ដោយមានភ្លាប់ខាងការណ៍បញ្ជាក់។
- ចូរសិយាយពីការប្រើប្រាស់ (Use) **Strategy** ដោយមានភ្លាប់ខាងការណ៍បញ្ជាក់។
- ចូរគួរ UML Class diagram ទាក់ទងនិងបញ្ហាណាមួយ ដែលមានអនុវត្ត **Strategy**។

10. State Pattern:

- ចូរសិយាយពីតួនាទី (Role) របស់ **State** ដោយមានភ្លាប់ខាងការណ៍បញ្ជាក់។
- ចូរសិយាយពីការប្រើប្រាស់ (Use) **State** ដោយមានភ្លាប់ខាងការណ៍បញ្ជាក់។
- ចូរគួរ UML Class diagram ទាក់ទងនិងបញ្ហាណាមួយ ដែលមានអនុវត្ត **State**។

11. Command Pattern:

- ចូរសិយាយពីតួនាទី (Role) របស់ **Command** ដោយមានភ្លាប់ខាងការណ៍បញ្ជាក់។
- ចូរសិយាយពីការប្រើប្រាស់ (Use) **Command** ដោយមានភ្លាប់ខាងការណ៍បញ្ជាក់។
- ចូរគួរ UML Class diagram ទាក់ទងនិងបញ្ហាណាមួយ ដែលមានអនុវត្ត **Command**។

12. Template Method Pattern:

- ចូរសិយាយពីតួនាទី (Role) របស់ **Template Method** ដោយមានភ្លាប់ខាងការណ៍បញ្ជាក់។
- ចូរសិយាយពីការប្រើប្រាស់ (Use) **Template Method** ដោយមានភ្លាប់ខាងការណ៍បញ្ជាក់។
- ចូរគួរ UML Class diagram ទាក់ទងនិងបញ្ហាណាមួយ ដែលមានអនុវត្ត **Template Method**។

13. Observer Pattern:

- ចូរសិយាយពីតួនាទី (Role) របស់ **Observer** ដោយមានភ្លាប់ខាងការណ៍បញ្ជាក់។
- ចូរសិយាយពីការប្រើប្រាស់ (Use) **Observer** ដោយមានភ្លាប់ខាងការណ៍បញ្ជាក់។
- ចូរគួរ UML Class diagram ទាក់ទងនិងបញ្ហាណាមួយ ដែលមានអនុវត្ត **Observer**។

ចម្លើយសម្រាប់សំណុរ

1. ចូរនិយាយពីតួនាទីរបស់ Prototype និងភ្លាប់ខាងក្រោមបញ្ជាក់។

- Prototype Pattern គឺជា Creational Design Pattern ដែលមានតួនាទីអនុញ្ញាតរវាយបង្កើត Object ដើម្បីធ្វើដោយចម្លង (Clone) ពី Object ដែលមានរូបរាង (Prototype)។

- បង្កើនលើរឿង (ការ Clone objects វាដើម្បីរឿងជាងការបង្កើតឡើង)
-

ឧទាហរណ៍ :

និយាយថាមីនមាន Document class ដែលមាន properties ដូចជា title, content, author។ ប្រសិនបើ អ្នកចង់បង្កើតឯកសារថ្មីដែលដូចនឹងឯកសារមួយ (សម្រាប់ Template ឯកសារ) អ្នកអាចប្រើ clone() ដោយមិនចាំបាច់បង្កើតឡើងពីដើម។

```
using System;

public abstract class Document : ICloneable
{
    public string Title;
    public string Content;
    public string Author;

    public object Clone()
    {
        // Shallow copy
        return this.MemberwiseClone();
    }
}
```

2. ចូរនិយាយពីការប្រើប្រាស់ (Use) Prototype ដោយមានភ្លាប់ខាងក្រោមបញ្ជាក់។

ពេលណាព្យាយងារ Prototype គូរត្រូវបានប្រើប្រាស់

- នៅពេល object ត្រូវបានបង្កើតដោយសុតស្ថាព្យាបាល។
- នៅពេលដែលសមាសភាពលើ object គឺ dynamic (ផ្តាស់ប្តូរគ្មាន runtime)។
- នៅពេលអ្នកចង់ duplicate object ដោយមិនចាំបាច់ដើងថារាជជា subclass ណាម។

ឧទាហរណ៍ ៖

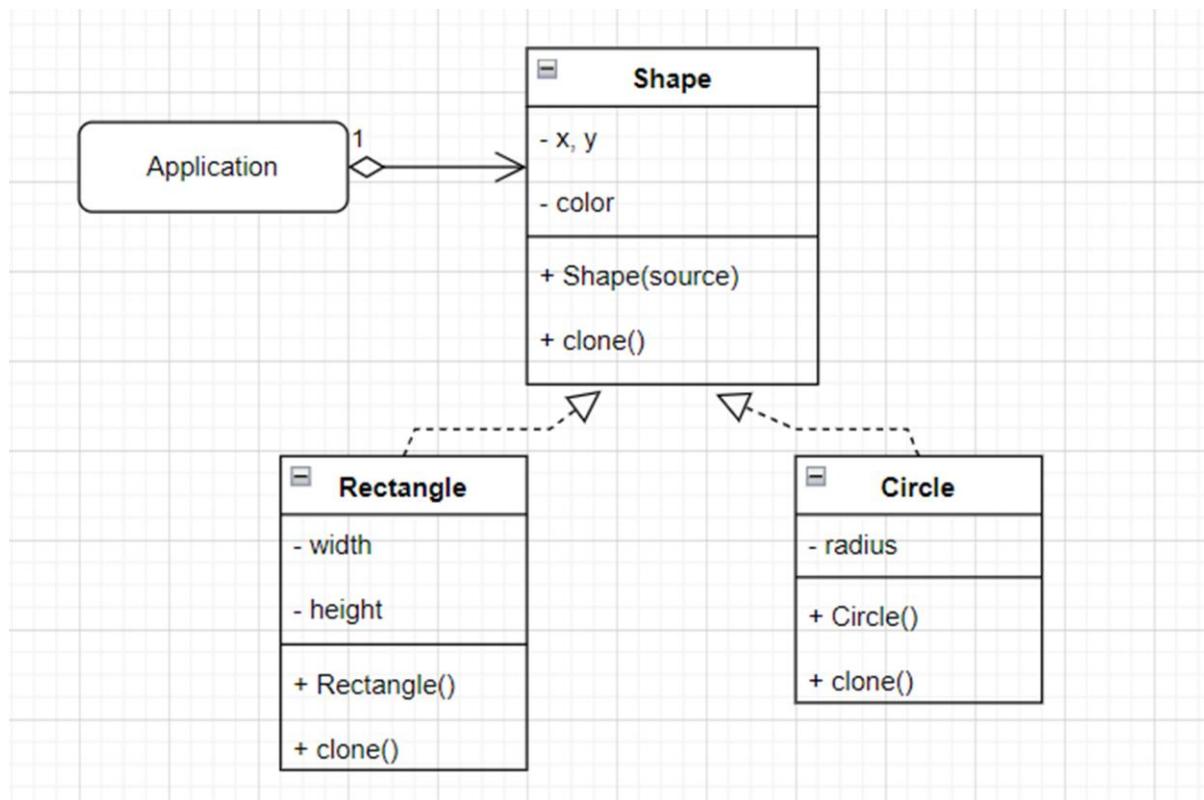
និយាយថា Game មួយមាន Enemy Object ដែលមានប្រព័ន្ធឌីជីថាមីនីតិត (AI) ស្ថិតស្ថាល្ត។ ហើយកងច់បានធ្វើកិត្តិយោង ដោយប្រើបានរបៀបណាមូដុចត្រា អ្នកអាចរបៀប Prototype ដោយ clone ពី original Enemy object។

csharp

Copy code

```
Enemy prototype = new Enemy("Orc", 150, new AI());
Enemy copy1 = (Enemy)prototype.Clone();
Enemy copy2 = (Enemy)prototype.Clone();
```

3. ចូរគូរ UML Class diagram ទាក់ទងនិងបញ្ហាណាមួយ ដែលមានអនុវត្ត Prototype ។



1. ចូរនិយាយពីក្នុងទំនាក់ទំនង (Role) និង Factory Method ដោយមានត្រូវបង្ហាញការងារ

- Factory Method Pattern គឺជា Creational Design Pattern ដែលមានក្នុងទំនាក់ទំនងដូចលម្អិត Interface ដើម្បីបង្កើត Object ប៉ុណ្ណោះនូវការរាយ subclass ផ្សេសនា តើក្រុរបង្កើត Object ពី class ណាយ។

```
Copy code
```

```
public interface IAnimal {
    void Speak();
}

public class Dog : IAnimal {
    public void Speak() => Console.WriteLine("Woof!");
}

public class Cat : IAnimal {
    public void Speak() => Console.WriteLine("Meow!");
}

public abstract class AnimalFactory {
    public abstract IAnimal CreateAnimal();
}

public class DogFactory : AnimalFactory {
    public override IAnimal CreateAnimal() => new Dog();
}

public class CatFactory : AnimalFactory {
    public override IAnimal CreateAnimal() => new Cat();
}
```

2. ចូរនិយាយពីការប្រើប្រាស់ (Use) Factory Method ដោយមានត្រូវបង្ហាញការងារ

គ្មានក្នុង Factory Method Pattern នេះពេលដែល:

- ចង់បង្កើត objects បែប dynamic ពី class ធ្វើង។
- ចង់ធ្វើសរុបការប្រើ new ដោយធ្វាល់ក្នុង client code
- ចង់រាយការបន្ថែម type ថ្មី មិនបែកលើ code ចាត់ (Open/Closed Principle)

ឧទាហរណ៍ ៖

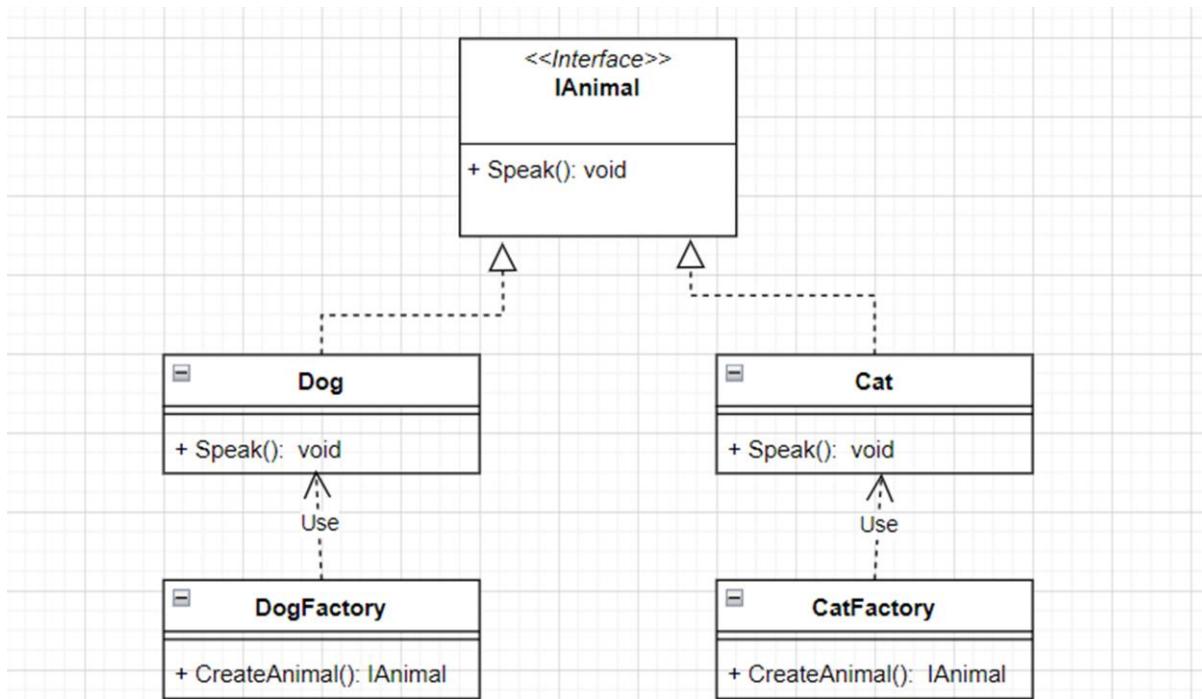
csharp

 Copy code

```
class Program {
    static void Main() {
        AnimalFactory factory = new DogFactory();
        IAnimal animal = factory.CreateAnimal();
        animal.Speak(); // Output: Woof!

        factory = new CatFactory();
        animal = factory.CreateAnimal();
        animal.Speak(); // Output: Meow!
    }
}
```

3. ចូរក្រុម UML Class diagram ទាក់ទងនិងបញ្ហាណាមួយ ដែលមានអនុវត្ត Factory Method។



1. ចូរនិយាយពីតួនាទី (Role) របស់ Singleton ដោយមានត្បាប់ខាងក្រោមខាងក្រោម

- Singleton Pattern គឺជា Creational Design Pattern ដែលមានតួនាទីដូចជា៖

- បង្កើត object តិចមួយ (one and only one instance) សម្រាប់ class មួយ។
- ផ្លូវ access point មួយតិចមួយ (global access point) ដើម្បីទូទាត់ client ទាំងអស់ទូទាត់ object នៅ។
- ការគ្រប់គ្រង instance មិនមែនអាយុយ client ទឹកបុណ្យត្រូវទេ បើនេះ class ឱ្យរារិយាល័យដែលគ្រប់គ្រងការបង្កើត និងចូលដំណើរការ។

```
public class Logger
{
    private static Logger instance;
    private static readonly object lockObj = new object();

    private Logger() {}

    public static Logger GetInstance()
    {
        if (instance == null)
        {
            lock (lockObj)
            {
                if (instance == null)
                    instance = new Logger();
            }
        }
        return instance;
    }

    public void Log(string message)
    {
        Console.WriteLine($"[LOG]: {message}");
    }
}
```

Logger មានតួនាទីដែលត្រូវការកែត្រួតពី logging messages ទាំងអស់ ឡាយប្រើពេល instance តែមួយ។

2. ចូរនិយាយពីការប្រើប្រាស់ (Use) Singleton ដោយមានត្បាប់ខាងក្រោមខាងក្រោម

- ត្រូវការធ្វើ instance មួយ (e.g. Logger, Configuration, Database connection)។
- ត្រូវការគ្រប់គ្រង access ច្បាស់លាស់ទៅកាន់ instance នៅ។
- ការបង្កើត multiple instances ការបង្កើតរបញ្ញា (e.g. database conflict, inconsistent settings)។
- ការគ្រប់គ្រងគ្មានៗ class ដែលវារឿងទៅវិញ មិនមែន client ។

```

public class DatabaseConnection
{
    private static DatabaseConnection instance;
    private static readonly object lockObj = new object();

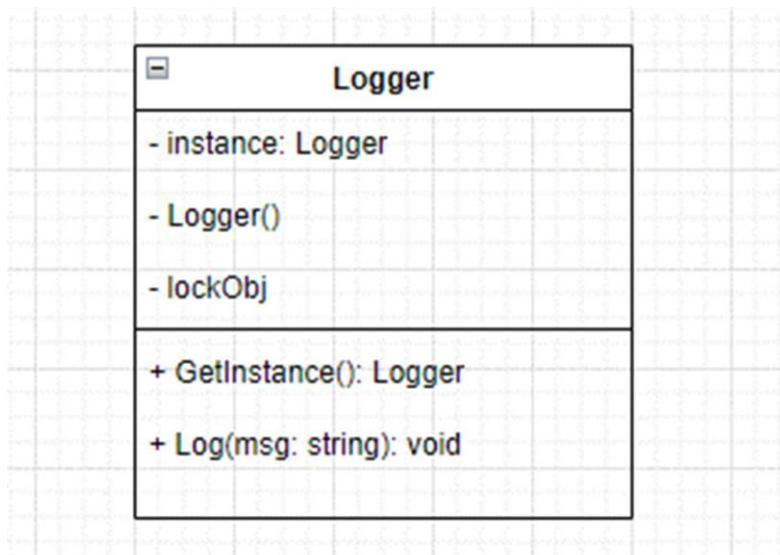
    private DatabaseConnection()
    {
        Console.WriteLine("Connected to DB");
    }

    public static DatabaseConnection GetInstance()
    {
        if (instance == null)
        {
            lock (lockObj)
            {
                if (instance == null)
                    instance = new DatabaseConnection();
            }
        }
        return instance;
    }

    public void ExecuteQuery(string sql)
    {
        Console.WriteLine($"Executing: {sql}");
    }
}

```

3. ចូរគួរ UML Class diagram ទាក់ទងនិងបញ្ជាលេមួយ ដែលមានអនុវត្ត Singleton ។



1. ចូរនិយាយពីត្បូនាទី (Role) របស់ Builder ដោយមានភ្លាម់ខាងក្រោម

- Builder Pattern គឺជា Creational Design Pattern ដែលមានត្បូនាទីដូចខាងក្រោម៖

- គ្រប់គ្រងដំណើរការបង្កើត object ស្ថិតស្ថាយ (complex objects)
- មានសារ៖ សំខាន់នៅលើលទ្ធផល object មាន property ទាំងនេះ ហើយ client មិនចង់ប្រើ constructor ដែលមាន parameter ជាប្រើប្រាស់

Step 1: បង្កើត Computer Class

```
csharp
public class Computer
{
    public string CPU { get; set; }
    public string RAM { get; set; }

    public override string ToString()
    {
        return $"CPU: {CPU}, RAM: {RAM}";
    }
}
```

Step 2: បង្កើត ComputerBuilder Class

```
csharp
public class ComputerBuilder
{
    private Computer computer = new Computer();

    public ComputerBuilder SetCPU(string cpu)
    {
        computer.CPU = cpu;
        return this;
    }

    public ComputerBuilder SetRAM(string ram)
    {
        computer.RAM = ram;
        return this;
    }

    public Computer Build()
    {
        return computer;
    }
}
```

2. ចូរនិយាយពីការប្រើប្រាស់ (Use) Builder ដោយមានភ្លាប់ខាងក្រោម

គូរប្រើ Builder Pattern នៅពេលដែល:

- Builder គឺជាសម្រាប់ការបង្កើត complex objects ដែលមាន parts ឬ components នៅក្នុង ហើយអាចត្រូវការបង្កើតបែបដូចមែន។
- វាអ្នកលែននូវ flexibility និង fine control នៃប្រព័ន្ធដែលត្រូវបានបង្កើត។
- Builder pattern អាចនាំឡាយមាន separation of concerns ពីការបង្កើតនិងការប្រើប្រាស់ object ដោយប្រើ step-by-step process។

ឧទាហរណ៍ ៖

Step 3: ប្រើ ComputerBuilder ត្រួតការបង្កើត Computer

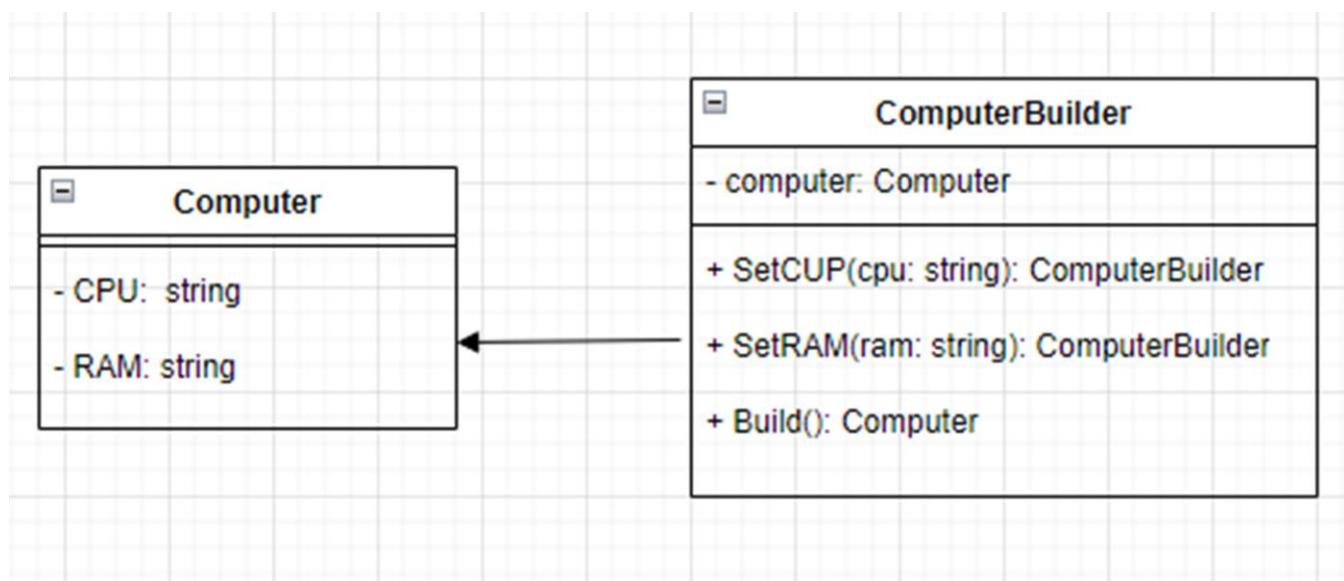
csharp

Copy Edit

```
var myPC = new ComputerBuilder()
    .SetCPU("Intel i7")
    .SetRAM("16GB")
    .Build();

Console.WriteLine(myPC);
```

3. ចូរគូរ UML Class diagram ទាំងនឹងបញ្ជាណាមួយ ដែលមានអនុវត្ត Builder ។



1. ចូរនិយាយពីតួនាទី (Role) របស់ Decorator ដោយមានភ្លាប់ខាងក្រោមបញ្ជាក់។

- Decorator Pattern គឺជា Structural Design Pattern ដែលមានតួនាទីបន្ថែមសមត្ថភាព (behavior) or state ថ្មី នៅក្នុង Object មួយដោយគ្មានការផ្តល់បញ្ជី Code នៅក្នុង class ដើម្បី

- Object មិនដឹងថាកំពុងត្រូវបាន decorate (Transparent Extension)។
- ប្រើបានសម្រាប់បន្ថែម feature ដោយមិនចាំបាច់សរសេរ subclass ប្រើបាន។
- អនុវត្តដោយ wrap object មួយជាមួយ class ថ្មី ដែលមានសមត្ថភាពបន្ថែម។

ឧទាហរណ៍៖

2. ចូរនិយាយពីការប្រើប្រាស់ (Use) Decorator ដោយមានភ្លាប់ខាងក្រោមបញ្ជាក់។

Decorator ត្រូវបានប្រើនៅពេល៖

- ចង់បន្ថែម behavior or feature ទៅក្នុង object មួយ ជាតិ dynamic
- យើងចង់ធ្វើឲ្យ object ខ្លះមាន behavior ដើម្បីងារដោយ មិនចំពាល់ជាលំដល់ others។
- យើងមិនចង់បង្កើត subclass ប្រើបានទេ

ឧទាហរណ៍៖

3. ចូរគូរ UML Class diagram ទាក់ទងនិងបញ្ហាណាមួយ ដែលមានអនុវត្ត Decorator ។

Have a good day, enjoy your day. Keep Learning.

ឧទាហរណ៍ ៖

1. ចូរនិយាយពីតួនាទី (Role) របស់ Bridge ដោយមានភ្លាប់ខាងក្រោម

- Bridge Pattern គឺជា Structural Design Pattern ដែលមានតួនាទី

- Bridge pattern មានតួនាទីក្នុងការបំបែក Abstraction ចេញពី Implementation ដើម្បីអាចអភិវឌ្ឍដោយផ្តល់ករណ្ឌ។
- ក្រឹមពេលដែលអ្នកចង់បន្ថែម version ឬ មិនចំណាត់ដល់ version មាន។
- ធ្វើសរុបតាមការបង្កើត subclass ព្រឹងពេករបោះការណាយបញ្ហាលំ Abstraction និង Implementation ទាំងពីរ

ឧទាហរណ៍៖

2. ចូរនិយាយពីការប្រើប្រាស់ (Use) Bridge ដោយមានភ្លាប់ខាងក្រោម

Decorator ត្រូវបានប្រើនៅពេល៖

- អ្នកចង់លាក់ Implementation ពី Client។
- ចង់ធ្វើសរុបតាមការចង់បណ្តាលបានបញ្ហាផ្លាស់។
- ចង់ធ្វើសរុបតាមការបង្កើត subclass ព្រឹងពេករបោះការណាយបញ្ហាលំ Abstraction និង Implementation (e.g. ដោយមិនបង្ហាញអ្នកប្រើ)។
- បង្កើតការចំប្រែ៖ runtime ជាមួយ component ធ្វើង។

ឧទាហរណ៍៖

3. ចូរគូរ UML Class diagram ទាក់ទងនិងបញ្ហាណាមួយ ដែលមានអនុវត្ត Bridge ។

Have a good day, enjoy your day. Keep Learning.

ຊາບරណ්ං

1. ចូរនិយាយពីតួនាទី (Role) របស់ Adapter ដោយមានត្រូវបង្ហាញកំណត់។

- Adapter Pattern គឺជា Structural Design Pattern ដែលមានតួនាទីដូចខាងក្រោម៖

Adapter មានតួនាទីក្នុងការបង្ហាប់ interface មួយទៅជាដែរ ដើម្បីធ្វើ client អាចប្រើប្រាស់ class ដែលរាយការ មិនបានស្រប interface ។ វាបានរៀបចំ object ដែលមាន interface មិនត្រូវត្រូវ អាចធ្វើការរួមគ្នាបាន។

ឧទាហរណ៍៖

2. ចូរនិយាយពីការប្រើប្រាស់ (Use) Adapter ដោយមានត្រូវបង្ហាញកំណត់។

Adapter គ្រប់បានប្រើប្រាស់នៅពេលដែលយើងចង់ប្រើប្រាស់ class ដើម (Adaptee) ដែលមាន interface មិនស្រប នឹង system បុត្រូវ client ដែលមានរូបរាង ។ វាជាមួយសង្ឃែរបញ្ជីការផ្តល់បញ្ជី code ដើម។

- ប្រើនៅពេល reuse legacy code
- ធ្វើឲ្យបិទិនិយប្រព័ន្ធបាន

ឧទាហរណ៍៖

3. ចូរគូរ UML Class diagram ទាក់ទងនិងបញ្ហាណាមួយ ដែលមានអនុវត្ត Adapter ។

Have a good day, enjoy your day. Keep Learning.

ຂ້າທຳກົດລົງ

Behavioral Design Pattern

9. Strategy Pattern:

- ចូរនិយាយពីត្បូនាទី (Role) របស់ **Strategy** ដោយមានភ្លាប់ខាងក្រោមបញ្ជាក់។
- ចូរនិយាយពីការប្រើប្រាស់ (Use) **Strategy** ដោយមានភ្លាប់ខាងក្រោមបញ្ជាក់។
- ចូរគួរ UML Class diagram ទាំងនេះ ដើម្បីរួចរាល់លក្ខណៈរបស់ **Strategy**។

10. State Pattern:

- ចូរនិយាយពីត្បូនាទី (Role) របស់ **State** ដោយមានភ្លាប់ខាងក្រោមបញ្ជាក់។
- ចូរនិយាយពីការប្រើប្រាស់ (Use) **State** ដោយមានភ្លាប់ខាងក្រោមបញ្ជាក់។
- ចូរគួរ UML Class diagram ទាំងនេះ ដើម្បីរួចរាល់លក្ខណៈរបស់ **State**។

11. Command Pattern:

- ចូរនិយាយពីត្បូនាទី (Role) របស់ **Command** ដោយមានភ្លាប់ខាងក្រោមបញ្ជាក់។
- ចូរនិយាយពីការប្រើប្រាស់ (Use) **Command** ដោយមានភ្លាប់ខាងក្រោមបញ្ជាក់។
- ចូរគួរ UML Class diagram ទាំងនេះ ដើម្បីរួចរាល់លក្ខណៈរបស់ **Command**។

12. Template Method Pattern:

- ចូរនិយាយពីត្បូនាទី (Role) របស់ **Template Method** ដោយមានភ្លាប់ខាងក្រោមបញ្ជាក់។
- ចូរនិយាយពីការប្រើប្រាស់ (Use) **Template Method** ដោយមានភ្លាប់ខាងក្រោមបញ្ជាក់។
- ចូរគួរ UML Class diagram ទាំងនេះ ដើម្បីរួចរាល់លក្ខណៈរបស់ **Template Method**។

13. Observer Pattern:

- ចូរនិយាយពីត្បូនាទី (Role) របស់ **Observer** ដោយមានភ្លាប់ខាងក្រោមបញ្ជាក់។
- ចូរនិយាយពីការប្រើប្រាស់ (Use) **Observer** ដោយមានភ្លាប់ខាងក្រោមបញ្ជាក់។
- ចូរគួរ UML Class diagram ទាំងនេះ ដើម្បីរួចរាល់លក្ខណៈរបស់ **Observer**។

1. ចូរនិយាយពីតួនាទី (Role) របស់ Strategy ដោយមានភ្លាមៗខាងក្រោម

Strategy Pattern គឺជា *Behavioral Design Pattern* ដែលមានភ្លាទី

→ បំបែក algorithm ឬ logic ឱ្យសម្រាប់ទៅជាអំពីជាពីរ ហើយអនុញ្ញាតឡើងប្រព័ន្ធដែលការងារត្រូវបានធានាជាមុននៅពេល runtime។

- Define family of algorithms, encapsulate each one
- ធ្វើការប្រើយុទ្ធសាស្ត្របានសេរី ដោយមិនបែងចាយទៅជាអំពី client
- ប្រើប្រាស់ polymorphism ដើម្បី បន្ថាយ behavior ដោយមិនប្រើ if-else or switch

ឧទាហរណ៍៖

2. ចូរនិយាយពីការប្រើប្រាស់ (Use) Strategy ដោយមានភ្លាមៗខាងក្រោម

Strategy Pattern ត្រូវបានប្រើនៅពេល៖

- អ្នកមាន algorithm ប្រចើនដែលអាចផ្តល់បញ្ជាផ្ទាល់ដោយអារ៉ាប់បែន្រែនភាព
- ចង់ធ្វើសរើសការងារជាបន្ទាល់ក្នុង if-else/switch statements
- ចង់រចនា system ឬក្រុមហ៊ុនដែលត្រូវបានការពារឡើងមុន

ឧទាហរណ៍៖

3. ចូរគូរ UML Class diagram ទាក់ទងនិងបញ្ហាណាមួយ ដែលមានអនុវត្ត Strategy។

Have a good day, enjoy your day. Keep Learning.

ຂ້າທຳກົດ

Have a good day, enjoy your day. Keep Learning.

1. ចូរនិយាយពីតម្លៃនាទី (Role) និង State ដោយមានត្រូវបង្ហាញរបៀបញ្ចាំ។

State Pattern គឺជា Behavioral Design Pattern ដែលមានតម្លៃនាទី

→ អនុញ្ញាតចូរ object មួយ ឬរបស់ពីការណើរបស់វា (behavior) ដោយផ្តល់ការណើ state ដ្ឋានលើទីនេះ។

ចំណាំសំខាន់៖

- បង្កើត class ដើម្បីសម្រាប់ state ដូចជា "Start", "Stop", "Pause"...
- ឬរបស់ពីការណើបាន ដោយគ្រាន់តែប្រើ state object ទៅ class ធ្លើ
- ផ្សែសភាង if-else/switch statement ដល់ state logic

ឧទាហរណ៍៖

2. ចូរនិយាយពីការប្រើប្រាស់ (Use) State ដោយមានត្រូវបង្ហាញរបៀបញ្ចាំ។

State Pattern ត្រូវបានប្រើនៅពេល៖

- Behavior និង object ធ្វើសំខាន់ប្រើការណើ internal state
- ចង់ធ្វើសភាង logic with many if-else/switch
- ចង់ទិញ code មាន structure មុតមាំ និងត្រូវការយកពេលបន្ថែម state ធ្លើ

ឧទាហរណ៍៖

3. ចូរគុរ UML Class diagram ទាក់ទងនិងបញ្ហាមួយ ដែលមានអនុវត្ត State ។

10. State Pattern:

Have a good day, enjoy your day. Keep Learning.

ຊາບරណ්ං

1. ចូរនិយាយពីតម្លៃនាទី (Role) និង Command ដោយមានភ្លាប់ខាងក្រោមបញ្ជាក់។

Command Pattern មានតម្លៃនាទី

→ បង្កើតការក្រសួង (request) ឬការប្រព័ន្ធឌីជីការណ៍មួយទៅជា object ដ៏ល្អជាយុទ្ធភាព ដែលអាចរក្សាទុននៅពេលក្រាយ។

ចំណុចសំខាន់៖

- បំបែក sender និង receiver ដើម្បីអាច manage request នៅពេលក្រាយ
- អាចធ្វើ Undo/Redo, queue commands, log history
- Encapsulate action ជា object

ខាងក្រោមនេះ

2. ចូរនិយាយពីការប្រើប្រាស់ (Use) Command ដោយមានភ្លាប់ខាងក្រោមបញ្ជាក់។

Command Pattern ត្រូវបានប្រើប្រាស់នៅពេល៖

- ចង់បំបែក logic រវាង sender និង receiver
- ចង់ធ្វើ Undo, Redo, logging, scheduling
- ចង់encapsulate ការប្រព័ន្ធឌីជី action ជា object ដូចជា queue / transaction

ខាងក្រោមនេះ

3. ចូរគូ UML Class diagram ទាក់ទងនិងបញ្ហាណាមួយ ដែលមានអនុវត្ត Command ។

Have a good day, enjoy your day. Keep Learning.

ឧទាហរណ៍

1. ចូរនិយាយពីគ្មានទៅ (Role) របស់ Template Method ដោយមានត្រូវបង្ហាគការណ៍

Template Method Pattern គឺជា *Behavioral Design Pattern* ដែលកំណត់ algorithm រួមដែលថែកចាយការងារ ដើម្បីផ្តល់មកទូទាត់ក្នុង subclass ដើម្បីបង្កើត behavior របស់វា

Role:

- កំណត់ algorithm មួយសម្រាប់ការអនុវត្តន៍វាងសំខាន់ៗ
- កំណត់ការលំអិតនៅក្នុង method មួយដែលអាចត្រូវបាន overriden បូត្រូវបានធ្វើឡាយពីច្បាប់ subclass
- ផ្តល់រាយ code នៃមំនឿងនាយកស្រួលសម្រាប់ការប្រើប្រាស់ក្រោម context ដែលផ្តល់ជូន

ឧទាហរណ៍៖

2. ចូរនិយាយពីការប្រើប្រាស់ (Use) Template method ដោយមានត្រូវបង្ហាគការណ៍

Template Method Pattern ត្រូវបានប្រើនៅពេល៖

- យើងមាន algorithm ដែលមានភាពធម្យតា ដែលអ្នកចង់កំណត់ structure តែចំណុចខ្លះទៅដាក់សម្រាប់ប្រើប្រាស់
- ចង់ធ្វើការ overriden ការអនុវត្តន៍វាងក្នុង subclass បើនឹងចង់រក្សាទុក structure តែយើងផ្តល់ជាតិ
- មានការងារដែលធ្វើជារបៀបសម្រាប់ប្រកែទូរដូចជា (ផ្តល់ជាទិន្នន័យ JSON និង XML)
- យើងចង់ប្រើ common algorithm បើនឹងអនុវត្តលម្អិតនៅក្នុង subclass។

ឧទាហរណ៍៖

3. ចូរគូរ UML Class diagram ទាំងនីងបញ្ជាណាមួយ ដែលមានអនុវត្ត template method ។

Have a good day, enjoy your day. Keep Learning.

ຊາບරណ්ං

1. ចូរនិយាយពីតួនាទី (Role) របស់ Adapter ដោយមានភ្លាប់ខាងក្រោមបញ្ជាក់។

Observer Pattern គឺជា *Behavioral Design Pattern* ដែលអនុញ្ញាតឱ្យ system ទំនាក់ទំនងជាមួយ object មួយចំនួន (observers) ដែលត្រូវបានចុះបញ្ជី ដើម្បីទទួលការរោចចេញពេលដែលដែល subject មួយមានការកំប្រឈាល (change)។

- Subject មានការ update (notify) observers នៅពេលដែលមានការកំប្រឈាល។
- Observer គឺជា object ដែលត្រូវបានចុះបញ្ជីមក្នុងជាមួយ subject ហើយបង្ហាញនូវទិន្នន័យថ្មី នៅពេលដែល subject ធ្វើការកំប្រឈាល។

ខាងក្រោម

2. ចូរនិយាយពីការប្រើប្រាស់ (Use) Observer ដោយមានភ្លាប់ខាងក្រោមបញ្ជាក់។

Observer Pattern ត្រូវបានប្រើនៅពេល៖

- យើងចង់មាន notification ជាមួយមនុស្សជាប្រើប្រាស់ នៅពេលដែល subject ធ្វើការកំប្រឈាល។
- ចង់អាយ object មួយនៅស្ថិតក្នុងភាព loosely coupled និងធ្វើការពង្រីកប្រព័ន្ធដោយមិនបង្កើតការផ្តាស់ប្តូរនៅក្នុង observer និង subject។
- ចង់តាមជានភាពផ្តាស់ប្តូររំនៅ data នៅពេលដែលមានការផ្តាស់ប្តូរ។

ខាងក្រោម

3. ចូរគូ UML Class diagram ទាក់ទងនិងបញ្ហាណាមួយ ដែលមានអនុវត្ត Adapter ។

Have a good day, enjoy your day. Keep Learning.