Problem Set 2
Object Oriented Computing
Due: 23 Sep 2009

The Strategy Pattern

ในการบ้านนี้ เราจะทำความรู้จักกับ pattern ที่มีชื่อเรียกว่า strategy pattern ซึ่งเป็น behavioral pattern ที่ค่อนข้างเรียบง่ายแต่มีประโยชน์มาก แนวคิดของ strategy pattern นี้เกี่ยวกับการแยกอัลกอริทึมจากคลาสที่ควรอยู่ออก และนำไปแบ่งเก็บไว้ในคลาสย่อยๆ เนื่องจากในปัญหาใดๆ เราอาจมีอัลกอริทึมสำหรับแก้ปัญหานั้นได้หลายแบบ ซึ่งแต่ละแบบก็มีข้อดีข้อเสียแตกต่างกันตามลักษณะของ input หากเราเก็บอัลกอริทึมทั้งหมดไว้ในคลาสเดียวจะทำให้คลาสนั้นมีสภาพของโค้ดโปรแกรมที่รกรุงรัง ทั้งนี้ ใน strategy pattern นอกจากจะทำการแบ่งอัลกอริทึมออกเป็นคลาสย่อยแล้ว ยังจัดการให้ผู้ใช้ฝั่ง client สามารถเลือกอัลกอริทึมที่จะใช้งานได้ตามลักษณะของ input ของงาน โดยมีการจัดการให้การเข้าถึงทำได้อย่างสะดวกอีกด้วย

เราจะลองคิดถึงปัญหาพื้นฐานปัญหาหนึ่ง คือปัญหาการจัดเรียงข้อมูล มีอัลกอริทึมอยู่มากมายสำหรับแก้ปัญหานี้ ตัวอย่างเช่น Quicksort, Mergesort ซึ่งเราอาจจะทราบมาว่า Quicksort นั้นมีประสิทธิภาพที่ดีสุด แต่ในความเป็นจริงแล้ว การทำงานของ Quicksort นั้นจะมีจำนวนครั้งของการเปรียบเทียบข้อมูลมากกว่าการทำงานของ Mergesort ทำให้ในบางสถานการณ์ อัลกอริทึม Mergesort ทำงานได้ดีกว่า

คำถาม 1. จากที่กล่าวมา การจัดเรียงข้อมูลของชนิดข้อมูลลักษณะใดเหมาะสำหรับใช้อัลกอริทึม Quicksort และข้อมูลลักษณะใดเหมาะแก่การใช้ Mergesort มากกว่า

การออกแบบ strategy pattern จะประกอบด้วย class หลักๆสามกลุ่ม ได้แก่ Context, Strategy Interface และ Concrete Strategies โดยสมมติว่าปัญหาของเรามีอัลกอริทึมให้เลือกใช้ได้สองแบบ เราจะทำการสร้างคลาสทั้งหมด 4 คลาสด้วยกัน ได้แก่

1. IStrategy เป็น Interface ของ Strategy ทั้งหมดโดยมีการประกาศหน้าตาของ method สำหรับเรียกใช้งาน strategy ที่ทำการ implement ต่อไปไว้ด้วย
2. StrategyA, StrategyB เป็น class ที่ implement IStrategy มาซึ่งจะมีการประกาศการทำงานของ method ใน interface IStrategy ไว้ด้วย
3. Context เป็น class ที่เก็บ object ของ IStrategy ไว้ เพื่อรักษาข้อมูลเกี่ยวกับอัลกอริทึมที่จะเรียกใช้ นั่นหมายความว่าเราจะมี object ของ IStrategy เป็นส่วนประกอบอยู่ใน class Context

คำถามที่ 2. จงแสดง UML diagram ของ Strategy Pattern ตามที่กล่าวมา

คำถามที่ 3. เพื่อทดสอบความเข้าใจ ก่อนที่จะถลำลึกไปมากกว่านี้ ให้ลองจับคู่ class ทางซ้ายกับตัวอย่างทางขวามือให้เหมาะสม

Context	Mergesort, Quicksort
Strategy	The GUI, the list generator, and the selectionof Strategy
IStrategy	Class containing the methods used for a particular sort
Algorithm	Sorting interface

ถึงตรงนี้ เราจะลองดู implementation ของ pattern นี้กันดูบ้าง ให้พิจารณาโปรแกรมต่อไปนี้

```
using System;

//The Context
class Context
{
    //Strategy aggregation
    ISortingStrategy strategy = new Mergesort();

    //Algorithm invokes a strategy method
    public void Sort()
    {
        strategy.work();
    }

    //Changing strategies
    public void SwitchStrategy()
    {
        if(strategy is Mergesort)
            strategy = new Quicksort();
        else
            strategy = new Mergesort();
    }

}
//Strategy interface
interface ISortingStrategy
{
    void work();
}
//Mergesort Strategy
class Mergesort : ISortingStrategy
{
    public void work()
    {
        Console.WriteLine("You are using Mergesort.");
    }
}
//Quicksort Strategy
class Quicksort : ISortingStrategy
{
    public void work()
    {
        Console.WriteLine("You are using Quicksort.");
    }
}

//Client
static class Program
{
    static void Main()
    {
        Context context = new Context();
        Console.WriteLine("Step1");
        context.Sort();
        Console.WriteLine("Step2");
        context.Sort();
        context.SwitchStrategy();
        Console.WriteLine("Step3");
        context.Sort();
    }
}
```

คำถามที่ 4. เมื่อโปรแกรมนี้ทำงาน ได้ผลลัพธ์เช่นไร

สังเกตว่าโปรแกรมดังกล่าวมีการประกาศ method SwitchStrategy เป็น public นั่นแสดงว่าเราเปิดโอกาสให้ผู้ใช้ฝั่ง client เปลี่ยนอัลกอริทึมที่จะใช้งานได้เอง (จริงๆแล้วใช้คำว่าเปิดโอกาสก็ไม่ถูกทีเดียว เนื่องจากหาก client ไม่มีการสั่งให้เปลี่ยนอัลกอริทึม โปรแกรมก็จะใช้อัลกอริทึมเดิมต่อไป อาจกล่าวได้ว่าเป็นการผลักภาระการเลือกอัลกอริทึมไปที่ผู้ใช้ ทั้งนี้ขึ้นอยู่กับความต้องการของระบบที่จะใช้งาน) เราสามารถปรับปรุงโปรแกรมนี้ให้ภาระการเลือกอัลกอริทึมที่จะใช้อยู่ที่ class Context ทั้งหมด

คำถามที่ 5. จงออกแบบ class Context ใหม่ ให้ client ไม่สามารถสั่งเปลี่ยนอัลกอริทึมที่จะใช้งานได้เอง แต่ให้ส่ง parameter เป็นจำนวนเต็มหนึ่งตัวมาเวลาเรียกใช้ method sort โดยจะสมมติว่าหมายถึง load ในการเปรียบเทียบข้อมูล โดย Context object จะเป็นผู้ตัดสินใจเองว่า ถ้า load ที่รับมามีค่ามากกว่า 0 จะใช้อัลกอริทึม Mergesort ในการทำงาน แต่ถ้า load มีค่าตั้งแต่ 0 ลงไป จะใช้อัลกอริทึม Quicksort แทน ตัวอย่างของ code ฝั่ง client และผลลัพธ์ที่ได้เป็นดังนี้

```
//Client
static class Program
{
    static void Main()
    {
        Context context = new Context();
        Console.WriteLine("Step1");
        context.Sort(0);
        Console.WriteLine("Step2");
        context.Sort(10);
        Console.WriteLine("Step3");
        context.Sort(-5);
        Console.Read();
    }
}


/* Output
Step1
You are using Quicksort.
Step2
You are using Mergesort.
Step3
You are using Quicksort.
*/
```