แบบฝึกหัด เรื่อง Design Patterns

01418471 Introduction to Software Engineering

โดย ผศ.ดร.อุษา สัมมาพันธ์

เรียนรู้ Design Pattern และฝึกเขียนโค้ดจากหนังสือ Head First Design Patterns (เขียนโดย E. Freeman, E. Freeman, K. Sierra, and B. Bates) คำสั่งที่ไม่มีขีดเส้นใต้จะมีโค้ดอยู่แล้วในหนังสือ ส่วนคำสั่งที่มีขีดเส้นใต้ คือส่วนที่นิสิตต้องเพิ่ม (ให้สังเกต polymorphism ในโค้ดเหล่านี้ด้วย)

ส่วนที่ 1: Adapter Pattern

1. เราจะมาเขียนโปรแกรม Duck Simulator ตามหนังสือ หน้า 499 โปรแกรมนี้จะแสดงพฤติกรรมต่าง ๆ ของเป็ด โดยจะมีเป็ดหลายชนิด เช่น Mallard duck, Redhead duck, rubber duck เป็นต้น เป็ดเหล่านี้ส่งเสียงร้อง (quack) ได้
2. ตัวโปรแกรม Duck Simulator จะมีเป็ดเหล่านี้หลายตัว และจะเรียกคำสั่งให้เป็ดเหล่านี้ส่งเสียงร้องได้
3. พิจารณาห่าน ซึ่งคล้ายเป็ด แต่ห่านไม่ส่งเสียงร้องแบบ quack แต่ส่งเสียงแบบ honk ดังนั้น ห่านมีเมธอดไม่เหมือนเป็ด คือ เมธอด honk() แต่ไม่มีเมธอด quack()
4. ให้นิสิตเพิ่มโค้ดเพื่อให้ Duck Simulator ใช้กับห่านได้ด้วย โดยนำ adapter pattern มาใช้ wrap around ห่าน
5. นอกจากนั้น ให้นิสิตเพิ่มโค้ดเพื่อให้ Duck Simulator ใช้กับนกพิราบ (pigeon) ได้ด้วย แต่นกพิราบส่งเสียงร้องแบบ coo (มีเมธอด coo() แต่ไม่มีเมธอด quack()) แต่การ coo สั้นเกินจะเป็น quack ต้องให้นกพิราบส่งเสียงร้องแบบ coo สองครั้ง ถึงจะเท่ากับการ quack 1 ครั้ง

ส่วนที่ 2: เพิ่ม Decorator Pattern

1. ให้นิสิตเพิ่ม feature โดยให้ duck simulator สามารถนับจำนวนการ quack ได้ (แต่ไม่นับการ honk หรือ coo) โดยจะนำ decorator pattern มาช่วย ตามโค้ดในหนังสือ
2. ให้นิสิตเพิ่ม feature การทำให้มีเสียงก้องได้ หมายความว่า เมื่อเป็ดร้อง quack 1 ครั้ง จะมีการ quack ซ้ำอีกครั้งหนึ่งตามมา เสมือนเป็นเสียงก้องที่สะท้อนกลับมา โดยให้ใช้ decorator pattern มาช่วย และให้เติมสตริง “Echo :” เข้าไปก่อนการ quack ซ้ำ เพื่อแสดงให้เห็นว่าเสียงนี้เป็นเสียงก้อง
3. ให้นำสอง decorator pattern มาใช้ร่วมกันเช่น new QuackCounter(new QuackEcho(new MallardDuck()) หรือ new QuackEcho(new QuackCounter(new MallardDuck()) แล้วดูผลการรันแต่ละแบบ
4. สังเกตความแตกต่างระหว่าง Adapter Pattern กับ Decorator Pattern ด้วย

ส่วนที่ 3: เพิ่ม Abstract Factory Pattern

1. เนื่องจากเราต้องการให้มีการนับจำนวนการ quack และอยากให้แน่ใจว่า โปรแกรมเมอร์ไม่ลืมที่จะใช้ decorator pattern จึงนำ abstract factory pattern มาช่วยให้การสร้าง object ง่ายขึ้นและผิดพลาดน้อยลง
2. ให้นิสิตเพิ่ม duck factory 3 แบบ ดังนี้ (1) แบบที่ไม่มี decorator ใดเลย (2) แบบที่มี counter decorator (3) แบบที่ใช้ทั้ง counter decorator และ echo decorator
3. ให้ลองใช้ factory แต่ละแบบในการรันโปรแกรม เพื่อดูว่าให้ผลเป็นอย่างไร

ส่วนที่ 4: เพิ่ม Composite Pattern

1. เนื่องจากตอนนี้มีเป็ดจำนวนมากและอยู่รวมกันเป็นฝูง (ในฝูงอาจมีห่านหรือนกพิราบแอบแฝงเข้ามาด้วย) จึงอยากมีวิธีจัดการเป็ดเหล่านี้พร้อมกันทั้งฝูง จึงนำ composite pattern มาช่วย โดยสร้าง composite class คือคลาส Flock ซึ่งรวมเป็ดหลายตัวไว้ด้วยกัน เมื่อมีการ quack ของทั้งฝูงเป็ด เป็ดแต่ละตัวจะ quack หนึ่งครั้ง
2. ให้นิสิตปรับโค้ด โดยในฝูงจะมีจ่าฝูง ซึ่งเป็นตัวแรกที่ถูกเพิ่มเข้ามาใน flock เมื่อมีการ quack จ่าฝูงจะ quack 3 ครั้ง ส่วนเป็ดตัวอื่น จะ quack หนึ่งครั้ง
3. ให้นิสิตลองรันโปรแกรมเพื่อ simulate การ quack ของทั้งฝูงเป็ด