Programming Fundamentals II

ปฏิบัติการ 4

ข้อปฏิบัติ

1. ให้ส่งงานในช่องทางที่ผู้สอนกำหนด
2. ไม่ต้องทำตามลำดับ ถ้าเห็นว่าทำข้อไหนได้ก่อนก็ทำข้อนั้นก่อนได้
3. ปรึกษาเพื่อนได้ แต่ต้องทำงานด้วยตนเองหรือภายในกลุ่มตามที่กำหนด

---

1. ปรับปรุงคลาส Account และ AccountRegistry

โจทย์ข้อนี้จะคล้ายกับครั้งที่แล้ว แต่ไฟล์ตั้งต้นจะต่างจากเดิม และมีการใช้ `enum` และ `ArrayList` ในโค้ดใหม่

ใช้ไฟล์ `Account.java` ที่โจทย์ให้ไว้เป็นจุดตั้งต้น ต้องการปรับปรุงคลาส `Account` ให้สามารถรองรับการโอนเงินระหว่างบัญชีและการระงับการใช้งานบัญชีได้ โดยบัญชีที่ถูกระงับจะไม่สามารถฝากหรือถอนได้จนกว่าจะเปิดใช้งานอีกครั้ง

ให้เพิ่มเมทอดต่อไปนี้ในคลาส Account

`public void suspend()` ระงับการใช้ กำหนดสถานะบัญชีให้เป็นถูกระงับหรือ suspended

`public void reactivate()` เปิดใช้งานอีกครั้ง กำหนดสถานะบัญชีให้กลับเป็นปกติหรือ active

`public boolean isActive()` ตรวจสอบสถานะบัญชีว่าปกติหรือไม่ ถ้าปกติจะคืนค่ากลับเป็น `true` แต่ถ้าถูกระงับอยู่หรือถูกปิดไปแล้วจะคืนค่ากลับเป็น `false`

`public Account transferTo(Account destAcc, double amount)` โอนเงินจากบัญชีไปยังบัญชีปลายทาง (`destAcc`) ตามยอด (`amount`) ที่ระบุ และคืนค่ากลับเป็นตัวบัญชีเอง

สถานะของบัญชีในครั้งนี้มีการแทนด้วย `enum` สำหรับกรณีการ active หรือ close ไปแล้ว ให้เพิ่มชนิดสถานะถูกระงับใน `enum` เลย

ปรับปรุงเมทอดต่อไปนี้ในคลาส Account

`public Account deposit(double amount)` กำหนดให้คืนค่ากลับเป็นตัวบัญชีเอง

`public Account withdraw(double amount)` กำหนดให้คืนค่ากลับเป็นตัวบัญชีเอง

การปรับปรุงเมทอดให้คืนค่ากลับเป็นตัวอ็อบเจกต์เองเป็นการออกแบบเมทอดตามรูปแบบ fluent API ซึ่งจะทำให้สามารถเรียกใช้เมทอดแบบต่อโยงกันได้ เช่น `acc.deposit(100.00).withdraw(50.00).deposit(250.00)` เป็นต้น

และเพิ่มการตรวจสอบดังนี้

ถ้าบัญชีถูกระงับอยู่หรือปิดไปแล้ว จะไม่อนุญาตให้ฝาก ถอน หรือโอน ถ้ามีการฝาก ถอน หรือโอนเกิดขึ้นในกรณีนี้ ให้สั่งให้เกิด exception ชนิด `IllegalStateException` โดยให้มีข้อความแจ้งไปในตัว exception ว่า `Account is unavailable.`

หมายเหตุ: เราจะใช้ `IllegalStateException` เมื่อเกิดการทำงานในสถานะที่ไม่ได้รับอนุญาตให้ทำงานนั้น

ใช้ไฟล์ `AccountRegistry.java` ที่โจทย์ให้ไว้เป็นจุดตั้งต้น ต้องการปรับปรุงคลาส `AccountRegistry` ให้สามารถเลือกรายการบัญชีที่มีสถานะตรงกับที่กำหนดได้ เพิ่มเติมจากเดิมที่ได้เฉพาะบัญชีที่ active อยู่

ให้เพิ่มเมทอดต่อไปนี้ในคลาส AccountRegistry

`public Account[] listAccounts(Account.Status status)` เป็น overloaded method ที่ทำหน้าที่คัดกรองเอาเฉพาะบัญชีที่มีสถานะตรงตามที่กำหนด คืนค่ากลับในรูปของอาเรย์

ปรับปรุงเมทอดต่อไปนี้ในคลาส AccountRegistry

`public Account[] listAccounts()` ปรับปรุงให้เรียกใช้เมทอด `listAccounts(status)` อีกที ไม่ต้องคัดกรองเอง โดยเลือกเอาเฉพาะบัญชีที่มีสถานะ active

โจทย์กำหนดคลาส `MainApp` ซึ่งมีเมทอด `main()` ที่จะเรียกใช้คลาส `Account` เอาไว้ให้ในไฟล์ `MainApp.java` ผลลัพธ์เมื่อรันควรจะเป็นดังนี้

```
Newly created accounts:
Account Number: 1234-0001, Balance: 100.00 Baht
Account Number: 1234-0002, Balance: 200.00 Baht
Account Number: 1234-0003, Balance: 300.00 Baht
Account Number: 1234-0004, Balance: 400.00 Baht
Account Number: 1234-0005, Balance: 500.00 Baht
Exception caught: Account is unavailable.
Active accounts:
Account Number: 1234-0001, Balance: 0.00 Baht
Account Number: 1234-0003, Balance: 150.00 Baht
Account Number: 1234-0005, Balance: 700.00 Baht
Suspended accounts:
Account Number: 1234-0004, Balance: 400.00 Baht
Closed accounts:
Account Number: 1234-0002, Balance: 200.00 Baht
```

2. วิเคราะห์ปัญหาต่อไปนี้

ผู้ว่าจ้างต้องการให้พัฒนาแอปสำหรับบันทึกรายจ่าย (expense tracker) โดยผู้ใช้สามารถบันทึกรายจ่ายซึ่งประกอบด้วยยอดเงินและวันที่ลงในแอปได้ แอปสามารถแสดงรายการรายจ่ายทั้งหมดและสรุปยอดรายจ่ายรวมได้

2.1 เรียบเรียงปัญหาใหม่ให้เป็นรายการความต้องการ (requirements) โดยตัดส่วนที่ไม่เกี่ยวข้องออก ขยายความส่วนที่ไม่ชัดเจน และเพิ่มเติมรายละเอียด เงื่อนไข หรือข้อกำหนดที่อาจแฝงอยู่ (ถ้ามี)

2.2 วิเคราะห์คำนามจากรายการความต้องการ ทั้งนี้เนื่องจากโจทย์ปัญหานี้สั้น อาจวิเคราะห์ความต้องการทั้งหมดพร้อมกันในรอบเดียวเลย บันทึกคำนามทั้งหมดที่พบ และพิจารณาว่าคำนามนั้นควรเป็น conceptual class หรือ attribute หรือไม่ใช่ทั้งสองอย่าง

2.3 จากคำนามที่เป็น conceptual class นำมาวิเคราะห์ความรับผิดชอบ (responsibility) ของคลาสนั้น และพิจารณาแต่ละความรับผิดชอบเพื่อค้นหา conceptual class ที่เป็นผู้ร่วมทำงาน (collaborator) บันทึกลงใน CRC card โดยในขั้นตอนนี้อาจค้นพบคลาสเพิ่มเติมจากที่พบในขั้นตอนก่อนหน้านี้ได้

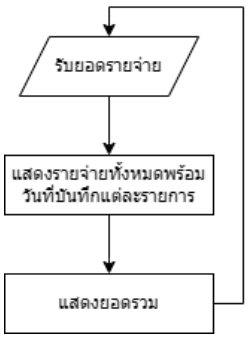
บันทึกงานลงในไฟล์ Analysis.docx ที่จัดเตรียมไว้ให้ งานนี้ไม่มีคำตอบที่ถูกต้องเพียงรูปแบบเดียว แต่ละคนอาจวิเคราะห์ออกมาได้แตกต่างกัน นอกจากนี้ ในระหว่างการทำแต่ละขั้นตอน อาจพบประเด็นที่ทำให้ต้องย้อนกลับไปปรับปรุงการวิเคราะห์ในขั้นตอนก่อนหน้าได้ ทั้งนี้สามารถดูตัวอย่างได้จากสไลด์บรรยาย

การบ้าน

1. เขียนโค้ดสำหรับ Expense Tracker

สืบเนื่องจากปฏิบัติการ 4 ข้อ 2 ให้นำผลวิเคราะห์ไปเขียนเป็นโค้ด โดยมีข้อกำหนดดังนี้

- สร้างคลาสโดยใช้ผลวิเคราะห์ conceptual class ที่บันทึกใน CRC card เป็นแนวทางกำหนด attribute และ method
- สร้างคลาส ExpenseTrackerApp เป็นตัวแทนตัวระบบของแอป โดยมี main อยู่ในคลาสนี้ ทำหน้าที่วนรับรายจ่ายจากผู้ใช้แล้วแสดงรายการทั้งหมดพร้อมยอดรวมก่อนจะกลับไปรับค่าใหม่ โดยจะมีการทำงานเบื้องต้นดังนี้



  - การเก็บข้อมูลวันที่ให้ใช้คลาส LocalDate โดยศึกษาการใช้งานเพิ่มเติมจากอินเทอร์เน็ต
  - รายละเอียดการแสดงผลให้กำหนดเองตามความเหมาะสม
  - ส่งไฟล์วิเคราะห์อีกรอบ (สามารถแก้ไขเพิ่มเติมจากตอนที่ส่งในห้องปฏิบัติการได้) พร้อมไฟล์โค้ดทั้งหมด