งานกลุ่มวิชา IS512

วัตถุประสงค์

1. เพื่อฝึกการวิเคราะห์และออกแบบโปรแกรมโดยนำหลักการของการเขียนโปรแกรมเชิงอ๊อบเจกต์มาใช้
2. ฝึกการทำงานเป็นทีม ให้ช่วยกันคิด ช่วยกันค้นคว้าหาคำตอบและช่วยกันแก้ปัญหาโปรแกรม

คำสั่ง ให้นักศึกษาจับกลุ่ม โดยแต่ละกลุ่มมีสมาชิกได้ไม่เกิน 4 คน และไม่น้อยกว่า 3 คน (กลุ่มที่มีสมาชิกน้อยกว่า 4 คน จะมีคะแนนเพิ่มให้) ให้แต่ละกลุ่มช่วยกันเขียนโปรแกรม และตอบคำถามตามโจทย์ต่อไปนี้ โดยให้ส่ง source code และตอบคำถามโดยพิมพ์ด้วย Microsoft Word ส่งทางอีเมลไปที่ wanchai@tbs.tu.ac.th ภายในวันที่ 24 สิงหาคม 2558

จากคลาส Account และ SavingAccount ที่อยู่ใน source code TestInheritance.java (สามารถดาวน์โหลดได้จากเว็บ http://www.bus.tu.ac.th/usr/wanchai/is512/) ที่ได้ทดลองในห้องเรียนดังรูปต่อไปนี้นั้น

```java
class Account {
  int no;
  double balance;
  void dump( ) {
    System.out.println("Balance of account is " + balance);
  }
  void showAccountInfo( ) {
    System.out.println("Account number " + no);
    dump();
  }

  double deposit(double amt) {
    balance += amt; return balance;
  }

  double withdraw(double amt) {
    balance -= amt; return balance;
  }
}

class SavingAccount extends Account {
    double withdraw(double amt) {
        if (amt <= balance)
            balance -= amt;
        else
            System.out.println("Not enough money");
        return balance;
    }
}

class TestInheritance {
  public static void main(String [] args) {
    SavingAccount sa = new SavingAccount();
    sa.deposit (5000.0); sa.dump();
    sa.withdraw(3000.0); sa.dump();
    sa.withdraw(4000.0); sa.dump();
  }
}
```

ให้นักศึกษาในกลุ่มช่วยกันปรับปรุงและขยายต่อเติมโปรแกรมนี้เพื่อให้โปรแกรมเมอร์คนอื่นนำคลาส account และ subclass ของมันไปใช้ โดยให้มีคลาสเพิ่ม อีกจำนวน 2 คลาส ดังนี้

ก. **คลาส `CheckingAccount`** เป็นคลาสสำหรับบัญชีกระแสรายวันซึ่งบัญชีประเภทนี้มีเงื่อนไขว่าจะถอนเงินจดหมดบัญชีไม่ได้ ธนาคารบังคับว่าจะต้องมีเงินคงเหลือค้างอยู่ในบัญชีจำนวนหนึ่ง เช่นไม่น้อยกว่า 2000 บาท ซึ่งจะใช้เงื่อนไขเดียวกันนี้กับทุกบัญชีของบัญชีประเภทนี้ แต่เนื่องจากจำนวนเงินที่บังคับว่าต้องค้างอยู่ในบัญชีอาจมีการเปลี่ยนแปลงได้ในอนาคตแต่ไม่บ่อยนัก ให้ตั้งตัวแปรสมาชิกชื่อว่า maintain สำหรับเก็บจำนวนเงินที่แต่ละบัญชีจะต้องรักษาไว้ไม่ให้ต่ำกว่าค่าของตัวแปรนี้ ให้กำหนดค่า 2000 ตายตัวให้กับตัวแปรนี้ สำหรับเมท็อด withdraw จะต้องนำตัวแปรนี้ไปใช้ในการตรวจสอบว่าจะอนุญาตให้ถอนได้หรือไม่ ถ้าถอนแล้วเงินคงเหลือในบัญชีมีค่าต่ำกว่าค่านี้จะไม่ยอมให้ถอน

ข. **คลาส `OverdraftAccount`** เป็นคลาสสำหรับบัญชีโอดีซึ่งสามารถถอนได้มากว่าเงินคงเหลือที่มีอยู่ในบัญชี ถ้าถอนแล้วเงินคงเหลือในบัญชีติดลบถือว่าเป็นการกู้เงินจากธนาคาร ธนาคารจะคิดดอกเบี้ยเงินกู้ (แต่เพื่อความง่ายในที่นี้จะไม่สนใจเรื่องดอกเบี้ย) เจ้าของบัญชีสามารถถอนเงินได้มากกว่าเงินที่คงเหลืออยู่ในบัญชีแต่จะถอนเกินเงินที่มีอยู่ในบัญชีได้เท่าไหร่นั้น ขึ้นอยู่กับวงเงินสินเชื่อที่ได้รับอนุมัติของแต่ละบัญชีซึ่งไม่จำเป็นต้องเท่ากัน ดังนั้นในคลาสจึงควรมีตัวแปรสำหรับเก็บวงเงินเบิกเกินบัญชีที่ได้รับอนุมัตินี้ไว้ด้วย โดยให้ชื่อตัวแปรว่า creditLimit เจ้าของบัญชีจะถอนในลักษณะที่เป็นการกู้ได้ไม่เกินวงเงินจำนวนนี้ ดังนั้นคลาสนี้จึงควรมีเมท็อด withdraw เป็นของตนเองเพื่อควบคุมให้เป็นไปตามเงื่อนไข ซึ่งเมท็อด withdraw จะต้องนำตัวแปร creditLimit ไปใช้ในการตรวจสอบก่อนให้ถอนเสมอ นอกจากนี้ คลาสนี้ยังจะต้องมีเมท็อดสำหรับกำหนดวงเงินสินเชื่ออีกด้วย โดยให้ตั้งชื่อเมท็อดว่า setCreditLimit ให้เมท็อดนี้นำวงเงินสินเชื่อที่ได้รับอนุมัติเข้ามาเป็นพารามิเตอร์แล้วนำไปใส่ไว้ในตัวแปร creditLimit

ให้นักศึกษากำหนดชนิด และตัวดัดแปร (modifier) ของตัวแปรสมาชิกให้เหมาะ และให้มีปรับแก้คลาส TestInheritance สำหรับทดสอบคลาสเหล่านี้

สมมติว่าทุก subclass ของคลาส Account มี constructor ที่รับพารามิเตอร์เป็นเลขที่บัญชีอยู่แล้ว ซึ่ง constructor จะรับเลขที่บัญชีไปเก็บไว้ในตัวแปร no ต่อจากนั้นจะไปค้นข้อมูลที่เกี่ยวข้องของบัญชีนี้จากฐานข้อมูลหรือจากไฟล์แล้วนำใส่ในตัวแปรที่เกี่ยวข้อง เช่นหาค่าเงินคงเหลือในบัญชีมาใส่ไว้ในตัวแปร balance ดังนั้น ให้ถือว่า เมื่อมีการสร้างอ๊อบเจ็กต์ขึ้นมาจะมีค่าเงินคงเหลือของบัญชีนั้น ๆ เก็บอยู่ในตัวแปร balance เสมอ และถ้ามีตัวแปรอื่นเพิ่มเติมในบางคลาสก็จะมีค่าข้อมูลของตัวแปรนั้นเรียบร้อยหลังจากสร้างอ๊อบเจกต์เสร็จ

ให้นักศึกษาอภิปรายกันในกลุ่มเพื่อช่วยกันทำให้โปรแกรมออกมาให้ดีที่สุดโดยทำตามเงื่อนไขและตอบคำถามต่อไปนี้

1. ตัวแปร no และ balance (บรรทัด 2 และ 3 ใน source code file) ที่ประกาศไว้ในคลาส Account เป็นตัวแปรที่ให้ subclass รับทอดไปใช้ได้ แต่ให้มีการปกป้องมิให้คลาสอื่นเข้าถึงได้ ยกเว้น subclass ของคลาส Account เท่านั้น จะใช้ตัวดัดแปร (modifier) ใดระหว่าง private กับ protected การใช้ตัวดัดแปรสองตัวนี้จะให้ผลต่างกันอย่างไร

2. เมท็อด dump() และ showAccountInfo() เป็นเมท็อดที่เขียนขึ้นมาเพื่อสาธิตการทำงานของโปรแกรมให้เห็นค่าของ no และ balance โปรแกรมที่ใช้งานจริงคลาส Account ไม่ควรมีการแสดงข้อมูลดังเช่นในตัวอย่าง ควรปล่อยให้เป็นหน้าที่ของคลาสอื่นที่จะแสดงข้อความตามที่ต้องการเอง แต่ทั้งนี้คลาสอื่นต้องสามารถสอบถามค่าของ no และ balance ได้ แต่ในเมื่อมีการปกป้องข้อมูลตามข้อ 1 ไปแล้วคลาสอื่นที่นำคลาส Account หรือ subclass ของ Account ไปใช้จะไม่สามาถเข้าถึงตัวแปร no และ balance ได้ ตามหลัก encapsulation แล้ว ควรมีเมท็อดที่ทำหน้าที่เข้าถึงข้อมูลเหล่านี้แทนการเข้าถึงตัวแปรจากนอกคลาสโดยตรง ดังนั้น จึงควรมีเมท็อดสำหรับให้สอบถามค่าของ no และ balance โดยให้ชื่อเมท็อดว่า getNo() และ getBalance() สำหรับสอบถามค่าของ no และ balance ตามลำดับ ให้นักศึกษาออกแบบและเขียนเมท็อดทั้ง 2 นี้ขึ้นมาให้คลาสใด ๆ ก็เรียกใช้ได้ ส่วนเมท็อด dump() และ showAccountInfo() ไม่จำเป็นต้องใช้อีกต่อไปก็ได้ เพราะมีเมท็อดให้สอบถามแล้ว ส่วนจะนำไปแสดงค่าข้อมูลอย่างไรก็สุดแล้วแต่ผู้ที่จะนำไปใช้ แต่ในบางครั้งผู้พัฒนาคลาสอาจยังต้องการใช้เมท็อด dump() และ showAccontInfo() อยู่เพื่อความสะดวกในการทดสอบโปรแกรม แต่ไม่ต้องการให้ผู้อื่นนำไปใช้ จะทำอย่างไรให้ยังคงมีสองเมท็อดนี้อยู่ แต่ไม่อนุญาตให้คลาสอื่นนอกจาก subclass เรียกใช้ ทำอย่างไรที่จะป้องกันไม่ให้มีการนำคลาส SavingAccount CheckingAccount และ OverdraftAccount ไปสร้าง subclass
3. สมมติว่าบัญชีประเภทต่าง ๆ มีการฝากเหมือนกันทุกประการ ดังนั้น subclass จึงไม่จำเป็นต้องเขียนเมท็อดนี้ขึ้นใช้เอง ให้รับทอดเมท็อด deposit() จากคลาส Account ไปใช้ได้เลย แต่ถ้าดูเมท็อด deposit() ของคลาส Account ตรงบรรทัด ๑๒ ยังมีข้อบกพร่องอยู่คือถ้ามีการเรียกเมท็อดนี้โดยส่งค่าติดลบมาให้ผลจะเป็นอย่างไร จะป้องกันไม่ให้มีการฝากค่าที่ติดลบได้อย่างไร
4. เมท็อด withdraw() ของคลาส Account ตรงบรรทัด ๑๖ เป็นเมท็อดสำหรับใช้ในการถอนเงินเป็นการทั่วไป ไม่มีการตรวจสอบเงื่อนไขว่าถอนได้หรือไม่ได้ ซึ่งบัญชีแต่ละประเภทจะมีเงื่อนไขในการถอนแตกต่างกัน ดังนั้น subclass ของคลาส Account จึงควรมีเมท็อด withdraw() เป็นของตนเอง (เท่ากับเป็นการ override เมท็อดของ superclass) เพื่อจะได้ตรวจสอบเงื่อนไขต่าง ๆ เสียก่อนที่จะมีการฝากหรือถอนจริง แต่เมท็อด withdraw() ของคลาส account ทำหน้าที่ที่สำคัญคือเมื่อมีการฝากหรือถอนแล้วจะมีการปรับปรุงข้อมูลในฐานข้อมูลของธนาคารให้ทันสมัย (แต่ตัวอย่างที่ให้ไม่มีการปรับปรุงข้อมูลในฐานข้อมูล เพื่อความง่าย ละไว้ในฐานที่เข้าใจของจริงจะมีการปรับปรุงข้อมูลในฐานข้อมูลหรือในไฟล์ของธนาคารทุกครั้งที่มีการเรียกเมท็อดเหล่านี้ ซึ่งเมท็อดจะมีความซับซ้อนยาวหลายบรรทัด) หน้าที่นี้ไม่ควรให้เมท็อด withdraw ของ subclass ของคลาส account ทำ เพราะจะซ้ำซ้อนกัน ควรนำหลัก inheritance มาใช้จะเหมาะกว่า ดังนั้น เมท็อด withdraw() ของ subclass จึงควรทำหน้าที่ในการตรวจสอบให้เป็นไปตามเงื่อนไขของบัญชีแต่ละประเภทว่าจะถอนได้หรือไม่ได้อย่างไร เมื่อผ่านการตรวจสอบเงื่อนไขแล้วเห็นว่าสามารถถอนได้จึงจะเรียกใช้เมท็อด withdraw() ของ superclass (ต้องใช้ super เป็นตัวช่วย เนื่องจากมีการ override เมท็อดเหล่านี้ของ superclass ไปแล้ว แต่ก็ยังสามารถยังขึ้นไปใช้เมท็อดใน superclass ได้อยู่ด้วยการใช้ super ช่วย) ในการปรับยอดเงินคงเหลือในบัญชีและทำการปรับปรุงข้อมูลในฐานข้อมูลหรือในไฟล์ให้ทันสมัยต่อไป โจทย์ข้อนี้ถึงแม้ subclass จะทำการ override เมท็อดใน superclass แต่ไม่ถึงกับใช้เมท็อดที่เขียนขึ้นใหม่ใน subclass ทำให้จนครบถ้วนเองทั้งหมด ยังคงอาศัยเมท็อดใน superclass ช่วยทำส่วนที่เหลือให้ด้วย ให้นักศึกษาอธิบายว่าการทำแบบนี้ดีหรือไม่ประการใด ให้อธิบายเหตุผลประกอบให้ชัดเจน
5. เมท็อด withdraw() ของคลาส SavingAccount ควรปรับปรุงให้เป็นไปตามโจทย์ในข้อ 4 คือตรงบรรทัด 24 ไม่ควรทำเอง แต่ให้ไปใช้เมท็อด withdraw() ของ superclass จะทำอย่างไร? และตรงบรรทัด 26 การแสดงข้อความ “Not enough money” แบบนี้เป็นวิธีการที่ไม่ดีนัก ควรปล่อยให้เป็น

หน้าที่ของคนเขียนโปรแกรมที่เรียกใช้เมท็อดนี้แสดงผลเองตามรูปแบบหรือวิธีการและข้อความตามที่ต้องการเองจะดีกว่า ทำไมอาจารย์จึงบอกว่าไม่ดี? วิธีที่ดีกว่าควรจะเป็นอย่างไร?

6. สำหรับคลาส CheckingAccount ที่ต้องเขียนเพิ่มควรมีเมท็อด withdraw() เป็นของตนเองเพื่อควบคุมให้เป็นไปตามเงื่อนไขการถอนเงินของบัญชีประเภทนี้คือห้ามถอนจนหมดเงินในบัญชี ต้องมีเงินคงค้างอยู่ในบัญชีระดับหนึ่ง โจทย์กำหนดให้มีตัวแปรสมาชิกชื่อ maintain เก็บจำนวนเงินขั้นต่ำที่ทุกบัญชีประเภทนี้จะต้องคงไว้ ถ้ามีการถอนต้องตรวจก่อนว่าถอนแล้วเงินจะเหลือในบัญชีน้อยกว่าค่าของตัวแปร maintain หรือไม่ ถ้าถอนแล้วเงินเหลือน้อยกว่าค่า maintain จะไม่ยอมให้ถอน
ตัวแปร maintain ควรเป็น instance variable หรือ class variable เพราะเหตุใด นักศึกษาจะประกาศตัวแปรตัวนี้ให้มีตัวดัดแปร (modifier) ใดบ้าง ให้เขียนประโยคข้อความสั่งสำหรับประกาศตัวแปรนี้แล้วอธิบายเหตุผลว่าทำไมจึงใช้ตัวดัดแปรแบบนั้น
7. คลาส OverdraftAccount ควรต้องมีเมท็อด withdraw() เป็นของตนเองเช่นกัน แต่ยอมให้ถอนมากกว่าเงินที่มีอยู่ในบัญชีได้ โดยยอมให้ติดลบได้ไม่เกินวงเงินที่ได้รับอนุมัติซึ่งเก็บไว้ในตัวแปรสมาชิก creditLimit เนื่องจากบัญชีประเภทนี้แต่ละบัญชีจะมีวงเงินที่ได้รับอนุมัติไม่เท่ากัน ดังนั้นจึงควรให้ตัวแปร creditLimit เป็น instance variable หรือ class variable เพราะเหตุใด
8. เนื่องจากคลาส Account มีความเป็นทั่วไปสูง ถ้าไม่ต้องการให้นำคลาส Account ไปสร้างอ๊อบเจกต์ จะป้องกันได้อย่างไร
9. ถ้าไม่ต้องการให้นักเขียนโปรแกรมนำ subclass ทั้ง 3 คือ SavingAccount CheckingAccount และ OverdraftAccount ไปสร้าง subclass ลึกลงไปกว่านี้อีกแล้ว จะป้องกันได้อย่างไร
10. โปรแกรมที่เขียนถ้าเขียนไม่ถูกต้องตามหลักภาษาจะคอมไพล์ไม่ผ่าน เมื่อแก้ไขโปรแกรมจนคอมไพล์ผ่านไปได้แล้ว เราจึงนำโปรแกรมไปรัน แต่โปรแกรมที่รันได้ไม่ได้หมายความว่าจะทำงานได้ถูกต้องเสมอไป อาจมีข้อบกพร่อง ผิดพลาดในโปรแกรมที่ทำให้ทำงานได้อย่างไม่หยุดชงักแต่ผลลัพธ์จากการทำงานไม่ถูกต้องก็เป็นได้ ข้อบกพร่องแบบนี้เรียกว่าข้อผิดพลาดทางตรรกะ (logic error) ให้นักศึกษาคุยกันแล้วคาดการณ์ว่ามีโอกาสที่โปรแกรมจะทำงานผิดพลาดที่จุดใดด้วยสาเหตุใดได้บ้าง ให้เสนอจุดที่มีความเป็นไปได้ว่าจะเกิดข้อผิดพลาดขึ้นอย่างน้อย 2 จุด หรือ 2 กรณี เราจะทำอย่างไรเพื่อหลีกเลี่ยงไม่ให้เกิดข้อผิดพลาดเลยหรือให้เกิดข้อผิดพลาดให้น้อยที่สุด
11. เป็นไปได้ว่าโปรแกรมเมอร์ที่เขียนโปรแกรมบางคนอาจคิดทุจริต เขียนโปรแกรมแบบไม่ตรงไปตรงมา เช่นยอมให้บางบัญชีถอนเงินได้มากกว่าเงินที่มีอยู่ในบัญชี คิดดอกเบี้ยเงินกู้กรณีเบิกเกินบัญชีสำหรับบางบัญชีให้ต่ำกว่าที่ควรจะเป็น ถ้านักศึกษาพบโดยบังเอิญว่าโปรแกรมเมอร์คนหนึ่งซึ่งเป็นเพื่อนรักของเราเขียนโปรแกรมที่ไม่ตรงไปตรงมาแบบนี้ นักศึกษาจะทำอย่างไร
12. เป็นไปได้หรือไม่ที่โปรแกรมเมอร์บางคนอาจไม่พอใจอะไรบางอย่างหรือเป็นโรคจิตแล้วแกล้งเขียนโปรแกรมให้ทำงานผิดพลาดบ้างในบางโอกาส หรือวางยาให้เกิดความผิดพลาดเสียหายให้เกิดขึ้นในอนาคต การกระทำเช่นนี้คิดว่าผิดจริยธรรมและผิดกฎหมายหรือไม่
13. จากข้อ 11 และ 12 เพื่อป้องกันปัญหาเรื่องการทุจริตยักยอกหาผลประโยชน์ใส่ตัวหรือพวกพ้อง หรือจงใจทำให้เกิดความเสียหาย นักศึกษาจะมีข้อเสนอแนะอย่างไรที่จะป้องกันปัญหาเหล่านี้
