ก่อนที่จะทำโจทย์ นักเรียนต้องรู้สิ่งต่อไปนี้ และปฏิบัติตามอย่างเคร่งครัด

1. **โจทย์ข้อนี้จะคล้ายกับที่ทำอาทิตย์ที่แล้ว แต่ไม่เหมือนกันบางส่วน หลายๆกรณีจะระบุชัดเจนขึ้น**
2. **จุดมุ่งหมายของการทำแล็บอาทิตย์นี้คือ ให้เราใช้ประสบการณ์ที่ได้เรียนรู้ในการทำแล็บครั้งที่ผ่านมา และที่เรียนรู้จากห้องเรียน ในการเขียนโค้ดของเราเองให้ดีขึ้น ด้วยตัวเราเอง**
3. **เพื่อให้เกิดการเรียนรู้สูงสุดและติดหัวนานที่สุด นักเรียนต้องเริ่มทำแล็บตั้งแต่ต้นใหม่เลย เขียนเองใหม่ทั้งหมด อาจดูเหมือนทำซ้ำ แต่สิ่งนี้จะเป็นตัวช่วยเสริมให้เราได้ทำการ ทบทวนแนวคิดในการเขียนโปรแกรม เรามีประสบการณ์จากสัปดาห์ที่แล้วอยู่ ดังนั้นจะต้องสามารถเอาประสบการณ์มาช่วยได้โดยไม่ต้องลอกโค้ด**
4. **กำหนดส่งคือ วันศุกร์ เวลา 1:00 AM (ตี 1 คืนวันพฤหัส)**
5. **ห้ามลอก ถ้าพบจะให้ 0 คะแนนในครั้งนั้นๆ**

จงสร้างโปรแกรมที่ประกอบด้วย class ต่างๆ ดังต่อไปนี้ และทำการเขียน method ต่างๆ ของแต่ละ class เพื่อจำลองการทำงานอย่างง่ายของธนาคาร ตู้เอทีเอ็ม และตู้ฝากเงิน

- รายชื่อ class ที่จำเป็นต้องมี

- **BankAccount : บัญชีของธนาคาร**
- **ATMMachine : ตู้เอทีเอ็ม**
- **DepositMachine : ตู้ฝากเงิน**
- **Bank : ธนาคาร**
- **Main : ใช้ในการทดสอบการทำงานของ class ต่างๆ**

รายละเอียดการทำงานของแต่ละ class

1. BankAccount

- Instance variable : จะต้องมีตัวแปรสำหรับเก็บสถานะของตัว account เองว่าขณะนี้มีการ access เพื่อใช้งานอยู่หรือไม่ โดยค่า default คือ ไม่มีการ access (เหมือนตอนนี้ห้องน้ำว่างอยู่ และถ้ามีการ access ก็คือมีคนเข้าห้องน้ำห้องนั้นๆ แล้วล็อคห้อง)
- Constructor : มีกี่ตัวก็ได้ แต่ตัวที่ละเอียดที่สุดจะต้องทำงานได้ดังนี้

- รับ "เลขบัญชี" (รหัส account) ซึ่งเป็นสิ่งกำหนดตัวตนของ account นั้นๆ เหมือนกับรหัสของนิสิต
- รับ "เงินต้น" เมื่อทำการสร้าง account ใหม่
- รับ "ตัวเลข 4 หลัก" สำหรับเป็นรหัสในการเข้าในงานผ่าน ATM (แนะนำว่าให้รับเป็น string)
- ถ้ารูปแบบของข้อมูลที่ได้รับไม่เป็นไปตามนี้ให้แสดงข้อผิดพลาดออกทาง System.out และหยุดการทำงานของโปรแกรม

· Method : ต้องมี method ต่างๆ ดังนี้

- deposit : สำหรับเรียกเมื่อทำการฝากเงิน
  - รับ "จำนวนเงิน" เป็น double
- withdraw : สำหรับเรียกเมื่อต้องการถอนเงินจากบัญชีนี้
  - รับ "จำนวนเงิน" เป็น double
- transfer : สำหรับเรียกเมื่อทำการโอนเงินระหว่างบัญชีนี้ไปยังบัญชีเป้าหมาย
  - รับ "จำนวนเงิน" และ "เลขบัญชี" เป้าหมายที่จะทำการโอนเงินไปให้
- getBalance : คืนจำนวนเงินที่มีอยู่ในบัญชีนั้นออกมาเป็น string
- toString : คืนรายละเอียดบัญชี ได้แก่ เลขบัญชี, จำนวนเงิน ออกมาในรูปแบบของ string

2. ATMMachine

· มีความเกี่ยวข้องในการทำงานกับ BankAccount

· Constructor :

- รับ "คลังบัญชี" ที่อยู่ในรูปของ ArrayList ของ BankAccount โดย"คลังบัญชี" จะได้รับมาจาก Bank อีกทีเมื่อทำการสร้าง ATM และเมื่อมีการเปลี่ยนแปลงใดๆ เกิดขึ้นกับบัญชีในคลังบัญชีนี้ การเปลี่ยนแปลงนั้นๆ จะต้องเกิดขึ้นกับตัวบัญชีของธนาคารด้วย

· Method :

- logIn : เพื่อทำการเปลี่ยนสถานะของบัญชี เพื่อให้บัญชีที่ต้องการใช้งานอยู่ในสถานะที่กำลังมีการ access เกิดขึ้น เหมือนกับการรับบัตร atm เข้าเครื่องแล้วทำการป้อนรหัสผ่าน เพื่อทำการตรวจสอบว่าสามารถทำการเข้าถึงบัญชีนั้นๆ ได้หรือไม่ และถ้าบัญชีนั้น

อยู่ในสถานะที่กำลังมีการ access อยู่แล้ว(เช่น จาก ATM เครื่องอื่น) ก็จะไม่สามารถทำการ access ซ้อนได้

- รับ "เลขบัญชี"
- รับ "ตัวเลข 4 หลัก" ที่เป็นรหัสผ่าน

- logOut : เพื่อทำการเปลี่ยนสถานะของบัญชี เพื่อให้บัญชีเป้าหมายอยู่ในสถานะที่ไม่สามารถดำเนินการด้วยได้ เหมือนกับการจบการทำงานแล้วนำบัตร ATM ออกจากตู้ บัญชีนั้นๆ จะต้องอยู่ในสถานที่กำลังมีการ access อยู่และการ access นั้นจะต้องเกิดขึ้นจากตู้ ATM ที่เรียกคำสั่งนี้เท่านั้นจึงจะสามารถทำการเปลี่ยนสถานะของบัญชีได้
- withdraw : สำหรับเรียกเมื่อทำการถอนเงิน
  - รับ "จำนวนเงิน" ที่จะถอน ซึ่งต้องไม่เกินจำนวนเงินที่มีในบัญชี และต้องหารด้วย 100 ลงตัวเท่านั้น
  - เงื่อนไข : บัญชีที่ใช้งานจะต้องมีสถานะที่ถูก access โดย atm ที่เรียกคำสั่งนี้
  - ให้แสดงข้อความแสดงผลการดำเนินการออกมาทาง System.out ด้วย ถ้าหากไม่สามารถดำเนินการได้ด้วยสาเหตุใดๆ ให้แจ้งสาเหตุนั้นด้วย เช่น บัญชีเป้าหมายไม่อยู่ในสถานะที่สามารถดำเนินการได้ เป็นต้น
- transfer : ทำการโอนเงินระหว่างบัญชี จากบัญชีที่ใช้งานอยู่ไปยังบัญชีเป้าหมาย
  - รับ "เลขบัญชี" ของบัญชีที่จะรับเงินที่โอน
  - รับ "จำนวนเงิน" ที่จะทำการโอน ซึ่งต้องไม่เกินจำนวนเงินที่มีในบัญชี
  - เงื่อนไข : บัญชีที่ใช้งานจะต้องมีสถานะที่ถูก access โดย atm ที่เรียกคำสั่งนี้ บัญชีที่จะโอนให้จะต้องมีอยู่ในคลังบัญชีของธนาคาร
  - ให้แสดงข้อความแสดงผลการดำเนินการออกมาทาง System.out ด้วย ถ้าหากไม่สามารถดำเนินการได้ด้วยสาเหตุใดๆ ให้แจ้งสาเหตุนั้นด้วย เช่น บัญชีเป้าหมายไม่มีอยู่ในคลังบัญชีของธนาคาร เป็นต้น
- toString : คืนรายละเอียดสถานะของ ATM (มีรายละเอียดของการทำงานของ ATM เช่า บัญชีที่ใช้งานอยู่)ณ เวลานั้นออกมาในรูป String
  - เงื่อนไข : บัญชีที่ใช้งานจะต้องมีสถานะที่ถูก access โดย atm ที่เรียกคำสั่งนี้

3. DepositMachine

- มีความเกี่ยวข้องในการทำงานกับ BankAccount

- Constructor :
    - รับ "คลังบัญชี" ที่อยู่ในรูปของ ArrayList ของ BankAccount โดย"คลังบัญชี" จะได้รับมาจาก Bank อีกทีเมื่อทำการสร้าง DepositMachine และเมื่อมีการเปลี่ยนแปลงใดๆ เกิดขึ้นกับบัญชีในคลังบัญชีนี้ การเปลี่ยนแปลงนั้นๆ จะต้องเกิดขึ้นกับตัวบัญชีของธนาคารด้วย
- Method :
    - logIn : เหมือนกับของ ATMMachine
    - logOut : เหมือนกับของ ATMMachine
    - deposit : สำหรับเรียกเพื่อทำการฝากเงิน
        - รับ "จำนวนเงิน" ที่จะทำการฝาก ต้องหารด้วย 100 ลงตัวเท่านั้น
        - เงื่อนไข : บัญชีที่ใช้งานจะต้องมีสถานะทถูก access โดย atm ที่เรียกคำสั่งนี้
        - ให้แสดงข้อความแสดงผลการดำเนินการออกมาทาง System.out ด้วย ถ้าหากไม่สามารถดำเนินการได้ด้วยสาเหตุใดๆ ให้แจ้งสาเหตุนั้นด้วย เช่น บัญชีเป้าหมายไม่อยู่ในสถานะที่สามารถดำเนินการได้

4. Bank

- คลังบัญชี เก็บรายการของ BankAccount ในรูปของ ArrayList
- เก็บรายการของ ATMMachine และ DepositMachine ในรูปของ ArrayList
- ต้องบันทึก "เลขบัญชี" ล่าสุดที่เคยใช้ในการสร้างบัญชีใหม่เอาไว้
- แต่ละ object ของ Bank ไม่ได้ใช้ข้อมูลร่วมกัน
- Method :
    - createAccount : เพื่อใช้ในการสร้างบัญชีใหม่ โดยให้บัญชีที่สร้างขึ้นใหม่ มีเลขบัญชีต่อจากเลขบัญชีล่าสุดที่เคยใช้ในการสร้าง
        - รับข้อมูลทุกอย่างที่จำเป็นในการสร้าง BankAccount ยกเว้นเลขบัญชีซึ่งจะถูกกำหนดโดยตัว Bank เอง
        - ตรวจสอบข้อมูลที่ได้รับว่าตรงตามรูปแบบที่ต้องการหรือไม่ เช่น ความยาวของรหัสผ่าน ถ้าข้อมูลไม่อยู่ในรูปแบบที่กำหนด ให้แจ้งเตือนทาง System.out และจบการทำงานของ method นี้โดยไม่ต้องทำการสร้าง Account ใหม่
    - deleteAccount : เพื่อทำการลบบัญชีออกจากรายการบัญชีของธนาคาร

        - รับ "เลขบัญชี" ของบัญชีที่ต้องการทำการลบ
    - importAccount : รับรายการของบัญชีที่มีอยู่แล้วเข้ามาเพิ่มในคลังบัญชีของธนาคาร เช่น นำรายการของบัญชีจากธนาคารอื่นเข้ามาเพิ่มในธนาคาร
        - รับ "ชุดรายการของ BankAccount" ที่อยู่ในรูปของ ArrayList
    - createNewATM : สร้าง ATM ใหม่แล้วนำไปเพิ่มในรายการของ ATMMachine ของธนาคาร
    - deleteATM : ลบ ATM ออกจากรายการ ATMMachine ของธนาคาร
        - รับ "index" ของ ATM ที่ต้องการทำการลบ
    - createNewDeposit : สร้าง DepositMachine ใหม่แล้วนำไปเพิ่มในรายการของ DepositMachine ของธนาคาร
    - deleteDeposit : ลบ DepositMachine ออกจากรายการ DepositMachine ของธนาคาร
        - รับ "index" ของ DepositMachine ที่ต้องการทำการลบ

5. Main

- ให้นำ class ต่างๆ ที่สร้างไว้ตามเงื่อนไขด้านบน มาทดสอบการทำงานร่วมกันใน method main ของ class นี้ โดยสิ่งที่ต้องทำการทดสอบได้แก่
    - ทำการสร้าง bank ขึ้นมา 2 ตัว
    - ทำการสร้าง account ขึ้นมาใหม่โดยเป็น account ของ Bank แต่ละ Bank อย่างน้อย Bank ละ 3 account
    - นำรายการ account ของ Bank หนึ่งไปเพิ่มให้กับรายการ account ของ Bank อีก Bank หนึ่ง
    - สร้างและใช้งานตู้ ATM เพื่อทำการถอนเงินและโอนเงินระหว่าง account
    - สร้างและใช้งานตู้ Deposit เพื่อทำการฝากเงิน
    - ลบตู้ ATM และ Deposit ออกจากรายการตู้ประเภทนั้นๆ ของ Bank
    - ลบ account พร้อมแสดงให้เห็นว่าได้ทำการลบแล้วจริง

ในการเขียนแต่ละ class อนุญาตให้เขียน method และตัวแปรเพิ่มได้ เพื่อให้แต่ละ class สามารถทำงานได้ตามที่กำหนด

การตั้งชื่อ project ให้ตั้งชื่อตามรูปแบบต่อไปนี้

LAB##_<รหัสนิสิต> เช่น LAB03_5270271821

สำหรับแต่ละไฟล์ที่เขียน ให้ขึ้นต้นไฟล์ด้วยข้อมูลเกี่ยวกับตัวเอง ดังนี้

```
/*
*       LAB##
*       รหัสนิสิต
*       ชื่อ-นามสกุล
*       หมายเลขห้อง หมายเลขเครื่องที่ใช้
*/
```

เช่น

```
/*
*       LAB03
*       5270271821
*       นายเพื่อ สกุลไทย
*       ห้อง 111 เครื่อง 101
*/
```

ไฟล์ที่ส่ง ให้ export ออกมาในรูปแบบ jar โดยให้มี source code (.java) อยู่ภายใน jar ด้วย และให้ตั้งชื่อตามรูปแบบดังต่อไปนี้

LAB03_<รหัสนิสิต>.jar เช่น รหัสของผม(TA) คือ 5270271821 ก็จะมีชื่อไฟล์เป็น LAB03_5270271821.jar

*** ห้ามลอก ถ้าพบจะให้ 0 คะแนนในครั้งนั้นๆ ***