204498 ปัญหาพิเศษ เทคโนโลยีสกุลเงินเข้ารหัส

การบ้านที่ 3 (กำหนดส่งก่อนเที่ยงคืนของวันที่ 16 กุมภาพันธ์ 2561)

กติการการทำการบ้าน

- *ส่งการบ้านทาง Google Classroom เป็นไฟล์ pdf รหัสเข้าร่วมคือ tpf5p2e*
- *ปรึกษากันได้เต็มที่ แต่ต้องเขียนคำตอบด้วยตนเอง*
- *ลอกกันหนึ่งครั้ง = ผู้ที่เกี่ยวข้องทั้งหมดได้ศูนย์คะแนนในงานชิ้นนั้น*
- *ถ้ามีการลอกกันในครั้งที่สอง = ได้ F และ/หรือ ต้อง drop ตัวเองออกจากวิชา*

การบ้านนี้นิสิตจะได้เขียนโปรแกรมเพื่อประมวลผล transaction และให้กำเนิดบัญชี ScroogeCoin โดยลักษณะสำคัญของคอยน์นี้คือมีระบบส่วนกลางที่บริหาร จัดาร โดย Scrooge และใน epoch หนึ่ง (ในช่วงเวลาหนึ่ง) Scrooge จะได้รับ transaction มาจากทุกทิศทาง ทำการ validate และโพสท์ transaction ที่ได้ validate แล้วลงในบัญชี ScroogeCoin

ใน epoch หนึ่ง transaction ที่ได้รับเข้ามาอาจจะมีการอ้างถึงกันไปกันมาได้ และในขณะเดียวกันอาจมี transaction ที่พยายามทำ double หรือ triple spend (หรือทำการ spend ใดที่เสมือนการใช้คอยน์ที่มีอยู่มากกว่าหนึ่งครั้ง) ดังนั้นการเลือก valid transaction จึงเป็นเรื่องที่ทำได้ไม่ง่ายนัก

โปรแกรมตั้งต้นที่ให้มาจะมี Transaction class ที่แทน ScroogeCoin transaction ซึ่งใน class นี้ จะมี inner class คือ Transaction.Output และ Transaction.Input

โดย transaction output จะมีข้อมูลระบุค่าของคอยน์และ public key ซึ่งใช้แทนกระเป๋าเงินอิเลกทรอนิกส์

Transaction input จะประกอบไปด้วย hash ของ transaction นั้นที่มี

- transaction output ที่เกี่ยวข้อง
- index ของ output นี้
- signature

สำหรับ valid input transaction ตัว signature จะต้องเป็น hash digest ของ transaction นี้ (getRawDataToSign(int index) method) ที่ถูกเข้ารหัสโดย private key ที่จับคู่กับ public key ทางฝั่ง output ที่จะถูกนำมาใช้จ่ายที่ transaction นี้ผ่านทาง input นี้

Transaction อันหนึ่งประกอบไปด้วยลิสต์ของ input output และ hash ของ transaction ที่สมบูณณ์ (getRawTx() method) และมี method ที่ใช้

- เพิ่มและลบ input หรือ output
- เพิ่ม output
- คำนวณ digest เพื่อ sign/hash

เพิ่ม signature ที่ input (การประมวลผล signature จะทำภายนอก class Transaction โดย entity ที่รู้ private key ที่เหมาะสม)
คำนวณและเก็บ hash สำหรับ transaction เมื่อทุกอย่าง input/output/signature ถูกเพิ่มเข้าไปเรียบร้อยแล้ว

ในโปรแกรมจะมี class UTXO ซึ่งแทน Unspent Transaction Output โดย UTXO อันหนึ่งจะประกอบไปด้วย hash ของ transaction ที่เป็นจุดเริ่มต้นของ UTXO พร้อมกับ index ของ output ใน transaction นั้น method equals() hashCode() และ compareTo ได้ถูก override เพื่อใช้ในการเปรียบเทียบ UTXO สองอันจาก index และข้อมูลที่อยู่ใน txHash array

นอกจากนั้นยังมี class UTXOPool ที่แทน UTXO ทั้งหมดและมี map ที่จับคู่ UTXO แต่ละอันไปที่ transaction output ที่เป็นต้นกำเนิดของ UTXO นี้ class นี้ยังมี

- constructor สำหรับสร้าง UTXOPool ที่ว่างเปล่าขึ้นมาใหม่ หรือสร้างก๊อปปี้ของ UTXOPool อันหนึ่ง
- method สำหรับเพิ่มและลบ UTXO ออกจาก pool
- method สำหรับหา output ที่เป็นที่มาของ UTXO ใดๆ
- method สำหรับตรวจสอบว่า UTXO อันหนึ่งอยู่ใน pool หรือไม่
- method สำหรับหา UTXO ทั้งหมดที่อยู่ใน pool (Hashmap จะค้นหา key โดยใช้ hashCode และ equals method)

ไฟล์ rsa.jar มีไว้สำหรับใช้งาน class RSAKey โดย public key ของ transaction output และ private key สำหรับใช้สร้าง signature สำหรับ transaction input จะถูกแทนด้วย RSAKey และ RSAKeyPair คือคู่ public/private key โดย API สำหรับ RSAKey และ RSAKeyPair อยู่ในไฟล์ที่แนบมาควบคู่กับการบ้านนี้

ให้สร้างไฟล์ TxHandler.java ตาม API ด้านล่างนี้

```
public class TxHandler {

    /* Creates a public ledger whose current UTXOPool (collection of unspent
     * transaction outputs) is utxoPool. This should make a defensive copy of
     * utxoPool by using the UTXOPool(UTXOPool uPool) constructor.
     */
    public TxHandler(UTXOPool utxoPool);

    /* Returns true if
     * (1) all outputs claimed by tx are in the current UTXO pool,
     * (2) the signatures on each input of tx are valid,
     * (3) no UTXO is claimed multiple times by tx,
     * (4) all of tx's output values are non-negative, and
     * (5) the sum of tx's input values is greater than or equal to the sum of
            its output values;
       and false otherwise.
     */
    public boolean isValidTx(Transaction tx);

    /* Handles each epoch by receiving an unordered array of proposed
     * transactions, checking each transaction for correctness,
     * returning a mutually valid array of accepted transactions,
     * and updating the current UTXO pool as appropriate.
     */
    public Transaction[] handleTxs(Transaction[] possibleTxs);

}
```

สิ่งที่ method handleTxs จะต้องกระทำมีดังต่อไปนี้

- ให้เซ็ทของ valid transaction ที่ไม่สามารถเพิ่มเติมได้อีก
- Update โครงสร้างข้อมูลใน UTXOPool เพื่อให้สะท้อนถึงสภาวะล่าสุดของ UTXO เพื่อให้การเรียกใช้งาน handleTxs/isValidTx สามารถจะประมวลผลและตรวจสอบ transaction ได้