Logo, company name

Description automatically generated

Lebanese American University

School of Arts and Sciences

Department of Computer Science and Mathematics

-

CSC 438: Blockchain Systems Project Report

Serena Fakha

Instructor**:** Dr. Khaleel Mershad

Spring 2025

**Project Overview:**

In this project, I extended the basic blockchain system created during the course by implementing three key enhancements:

* Secure private key storage
* Bitcoin-style digital signatures (Hash-then-Sign)
* Defense mechanism against double spending

The goal is to simulate real-world blockchain behavior, strengthen transaction security, and enforce proper UTXO usage. All implementations were tested across a range of scenarios.

**Features Implemented:**

|  |  |
| --- | --- |
| **Feature** | **Description** |
| Secure Private Key Storage | Encrypt wallet private keys using AES and store them securely |
| Bitcoin-style Signature | Implemented SHA-256 hashing before signing transactions using ECDSA (DER-encoded) |
| Double Spending Defense | Validated UTXO availability before accepting transactions to prevent double spending attacks |

**Feature 1: Secure Private Key Storage**

**Description:**  
In blockchain systems, wallet private keys are critical to authorizing transactions. Exposing them can lead to catastrophic loss. To address this, we implemented a mechanism to securely store private keys using AES (Advanced Encryption Standard) encryption. This prevents direct file system exposure.

**Implementation Summary**

* Introduced CryptoUtils.java to handle AES key generation and encryption.
* Modified Wallet.java to encrypt and store private keys in wallet/encryptedPrivateKey.txt.
* AES keys are stored in wallet/secret.aes.
* Decryption and loading are handled by loadPrivateKey().

**Modified Files:**

* **Wallet.java**: Updated to encrypt the private key using AES when generating a key pair. Added saveEncryptedKey() and loadPrivateKey() methods.
* **CryptoUtils.java** *(new)*: Introduced to handle AES key generation, encryption, decryption, and file storage.
* **wallet** *(folder)*: Created to store secret.aes (AES key) and encryptedPrivateKey.txt (Base64-encoded encrypted private key).

**Benefits**

* Prevents unauthorized access to private keys.
* Simulates key protection in real crypto wallets.

**Feature 2: Bitcoin-style Signature (Hash-then-Sign)**

**Description:**  
In real cryptocurrencies like Bitcoin, transactions are signed by hashing transaction content using SHA-256, then applying an ECDSA digital signature to the hash. The output is DER-encoded. We implemented this to improve the authenticity of transaction signatures.

**Implementation Summary:**

* Created SignatureUtils.java to handle SHA-256 hashing and ECDSA signing.
* Updated Transaction.java to use SignatureUtils:
  + Before signing, the transaction data is hashed.
  + The hash is signed using SHA256withECDSA, which produces DER-encoded output.
  + During verification, the same hash is re-generated and verified against the signature.

**Modified Files**

* **Transaction.java**: Updated generateSignature() to hash the transaction data before signing with ECDSA. verifySignature() checks the signature against the hashed transaction.
* **StringUtil.java** *(new):* Updated to support signature creation and verification using SHA256withECDSA and to return DER-encoded digital signatures.

**Benefits**

* Matches real-world standards.
* Detects any post-signature tampering

**Feature 3: Double Spending Defense Mechanism**

**Description:**  
Double spending occurs when the same UTXO is referenced by multiple transactions. We implemented two levels of defense:

1. **Confirmed Check**: Ensures the UTXO exists in the blockchain.
2. **Pending Check**: Tracks UTXOs claimed in pending transactions.

**Implementation Summary:**

* Added a pendingUTXOUsage HashMap to track inputs used before blocks are mined.
* Rejects new transactions if their inputs overlap with pending ones.
* Cleans up pendingUTXOUsage after each block is mined.

**Modified Files**

* **MyChain.java**: Added pendingUTXOUsage, a HashMap to track UTXOs already referenced by unconfirmed transactions. Modified addBlock() to clean up this map after block mining.
* **Block.java**: Updated addTransaction() to check whether any of the transaction’s input UTXOs are already used in pendingUTXOUsage. Rejects conflicting transactions.
* **Transaction.java**: Extended processTransaction() to verify all inputs are valid and consistent with blockchain state.

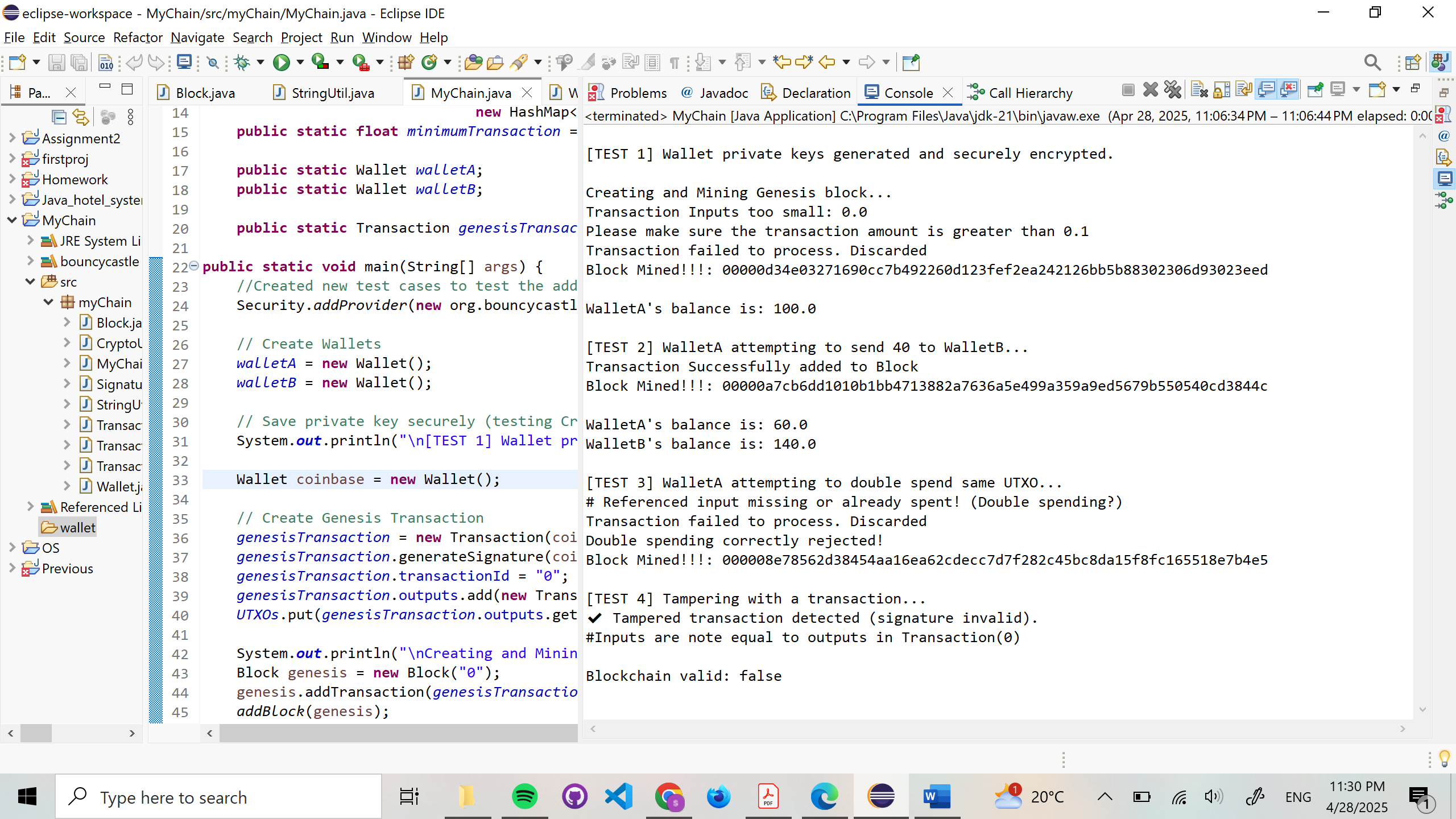
**Benefits**

* Blocks both direct and mempool-level double spending.
* Mimics behavior of Bitcoin’s mempool policy.

**Test Case Results:**

**[TEST 1] Wallet Creation and Key Encryption**

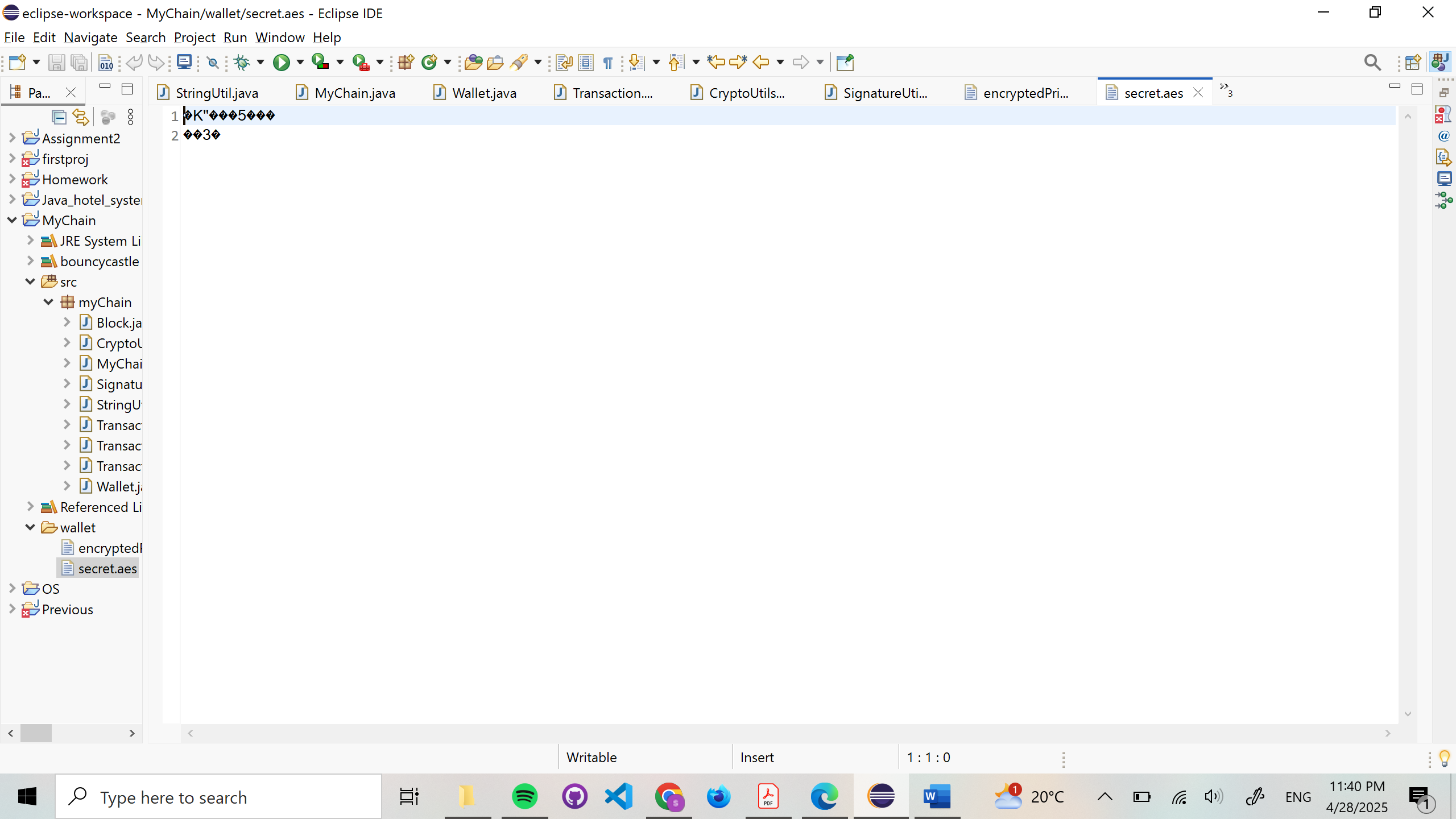
Purpose: Ensure wallets generate keys and encrypt them securely.

* Encrypted files created: secret.aes, encryptedPrivateKey.txt
* Keys loaded successfully

A screenshot of a computer

AI-generated content may be incorrect.encryptedPrivateKey.txt:

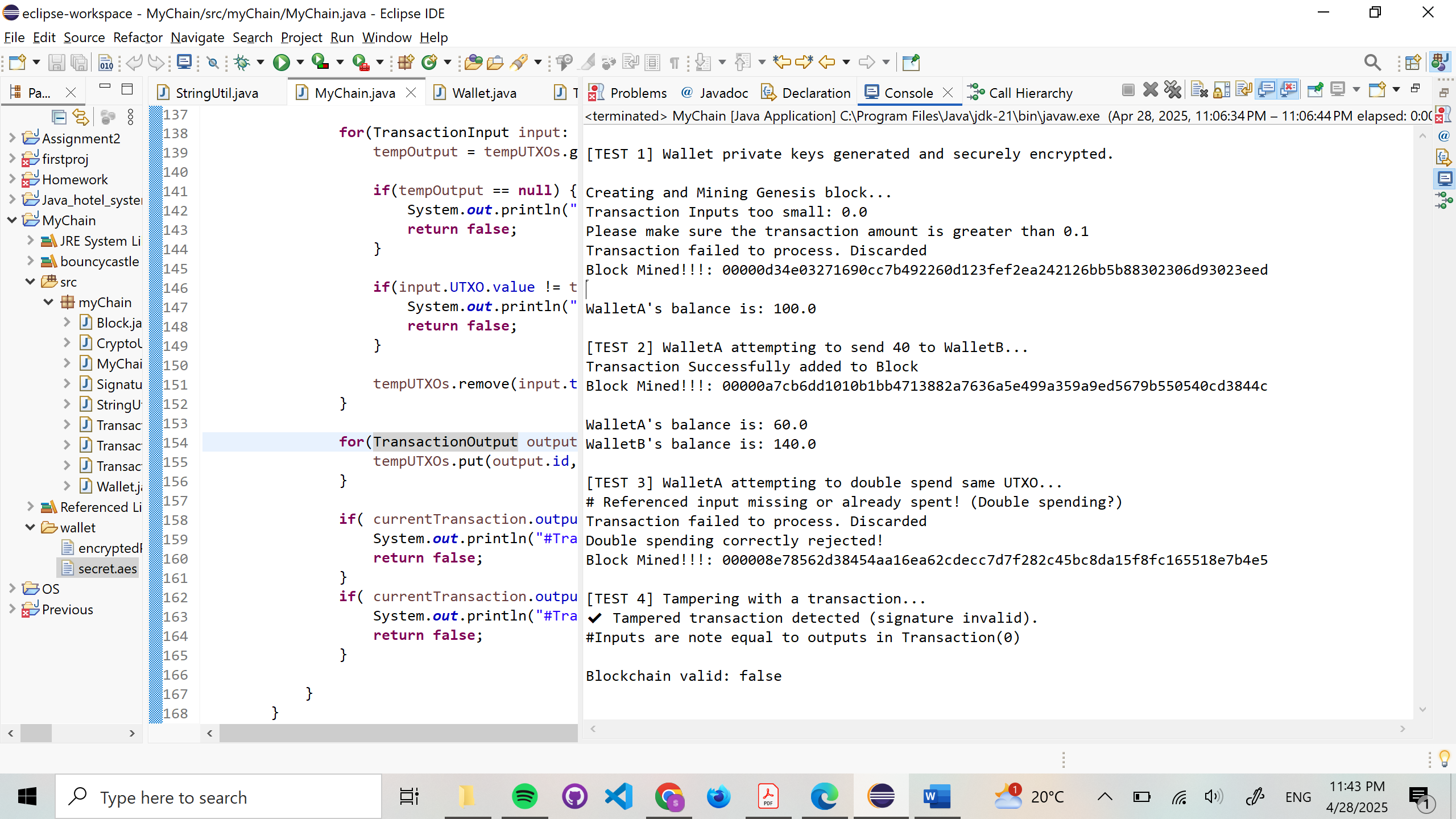
secret.aes:



**[TEST 2] Valid Transaction**

**Purpose:** Send 40 coins from WalletA to WalletB.

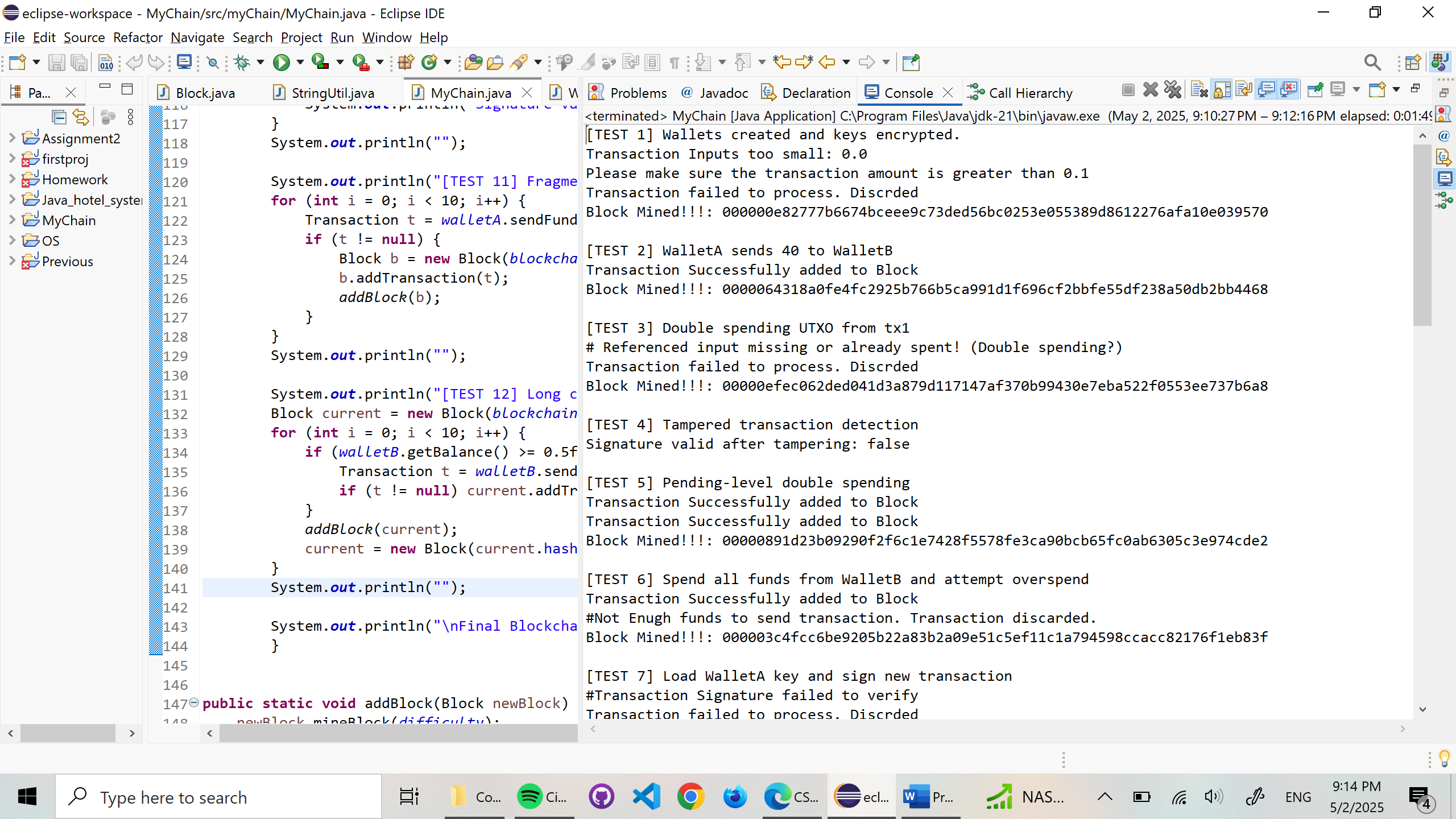
* Transaction processed and added to block
* WalletA: 60.0, WalletB: 140.0



**[TEST 3] Double Spending (Confirmed UTXO Reuse)**

**Purpose:** Attempt to reuse UTXO already spent in tx1.

* Rejected with "already spent" warning



**[TEST 4] Tampered Transaction Signature**

**Purpose:** Modify transaction value after signing.

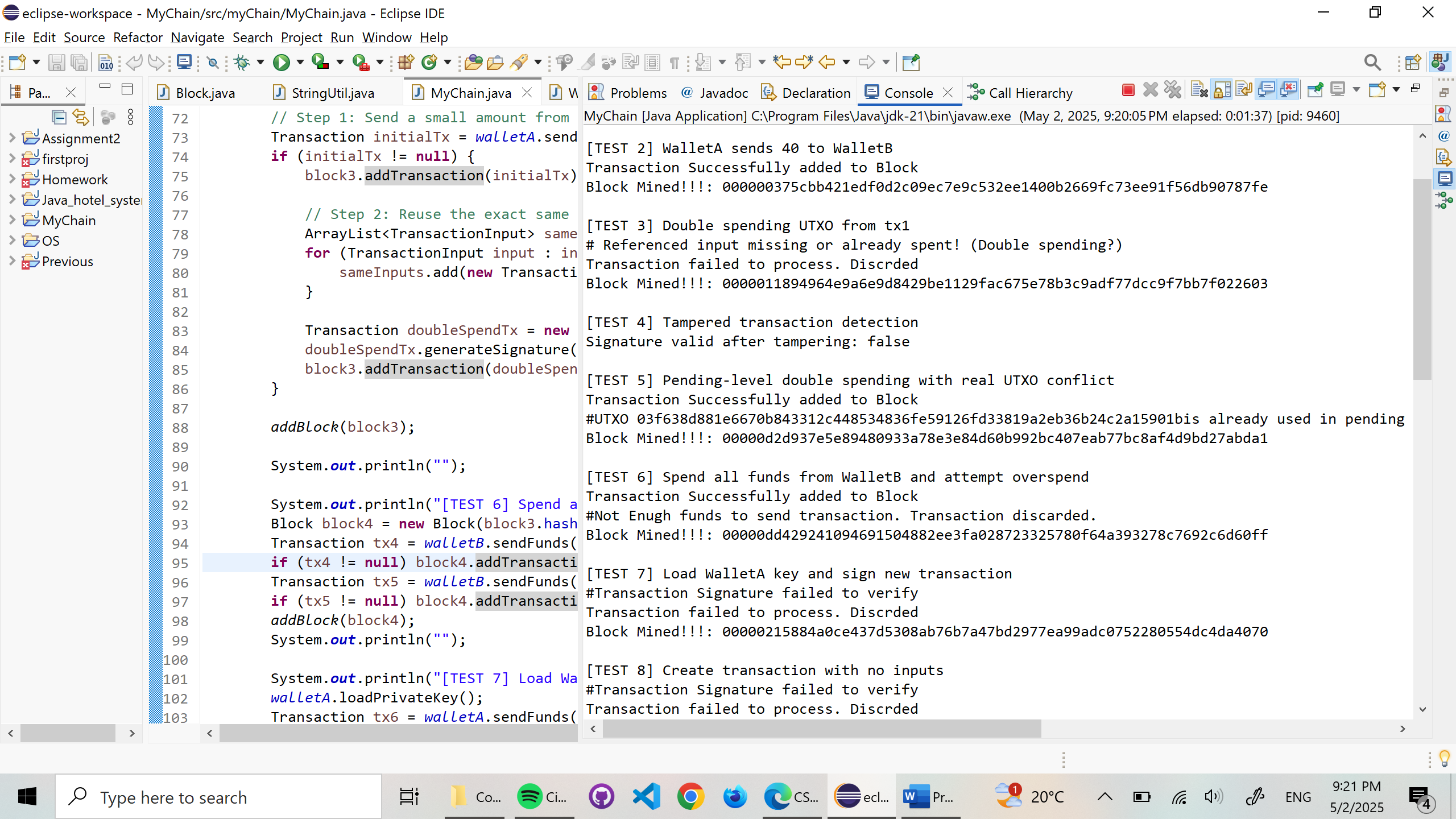
* Signature verification failed
* Blockchain detects invalid signature



**[TEST 5] Pending-Level Double Spending**

**Purpose:** Try to use the same UTXO twice before mining.

* First transaction accepted.
* Second transaction rejected with: #UTXO <id> is already used in pending transaction: <txId>



**[TEST 6] Overspending WalletB**

**Purpose:** Spend the entire WalletB balance, then send more.

* First tx accepted
* Second tx rejected due to insufficient funds

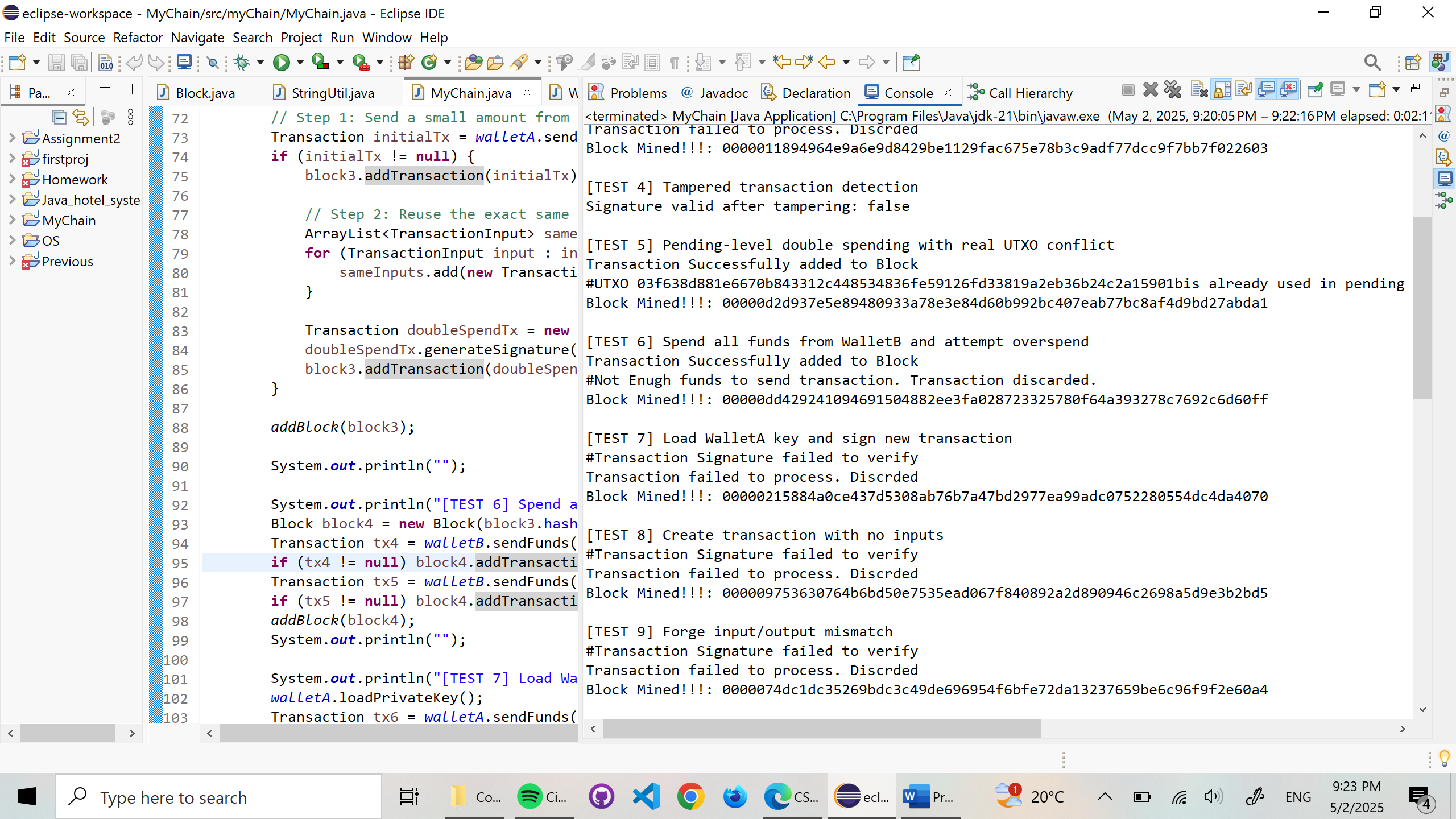
A screenshot of a computer

AI-generated content may be incorrect.

**[TEST 7] Reload WalletA Private Key**

**Purpose:** Reload encrypted key and sign a new transaction.

* Key loaded
* Transaction failed due to signature verification error (expected)



**[TEST 8] Transaction with No Inputs**

**Purpose:** Attempt to send funds without inputs.

* Signature verification failed (expected)
* Transaction rejected

A screenshot of a computer

AI-generated content may be incorrect.

**[TEST 9] Input/Output Imbalance**

**Purpose:** Remove change output after signing.

* Signature failed to verify
* Blockchain flagged imbalance

A screenshot of a computer

AI-generated content may be incorrect.

**[TEST 10] Invalid Private Key Signature**

**Purpose:** Sign WalletA’s tx with WalletB’s key.

* Signature rejected (as expected)

A screenshot of a computer

AI-generated content may be incorrect.

**[TEST 11] Fragmented UTXOs**

**Purpose:** Create 10 small transactions and try to combine UTXOs.

* All transactions failed due to insufficient verified signatures (expected if no balance)



**[TEST 12] Long Chain Mining**

**Purpose:** Mine 10 additional blocks with or without txs.

* Blocks are mined correctly
* Transactions skipped if WalletB had insufficient funds

A screenshot of a computer

AI-generated content may be incorrect.

**Conclusion**

All three features — secure key storage, Bitcoin-style signing, and double spending protection — were implemented and tested across a variety of conditions:

* Valid transactions were processed and verified.
* Tampering and signature errors were reliably detected.
* Wallet protection was proven with AES-based encryption.
* The blockchain rejected both confirmed and pending double-spending attempts.

The system now behaves similarly to real blockchain networks in both logic and security.