Project 3

Louis Chang (hungyic)

Task 0

Block.java

```java
package org.example;

import java.math.BigInteger;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.sql.Timestamp;

/**
 * Author: Louis Chang (hungyic)
 * Last Modified: 03/17/2024
 */
public class Block {
    private int index;
    private Timestamp timestamp;
    private String data;
    private String previousHash;
    private BigInteger nonce;
    private int difficulty;
    private String hash;

    public Block(int index, Timestamp timestamp, String data, int
difficulty) {
        this.index = index;
        this.timestamp = timestamp;
        this.data = data;
        this.difficulty = difficulty;
        this.previousHash = "0"; // Genesis block
        this.nonce = new BigInteger("0");
        this.hash = calculateHash();
    }

    // The calculateHash method is used to calculate the hash of the
block. It takes no parameters and returns a string.
    public String calculateHash() {
        /*
         * The calculateHash method is used to calculate the hash of the
block. It takes no parameters and returns a string.
         */
        try {
            // Create a new SHA-256 digest
            MessageDigest digest = MessageDigest.getInstance("SHA-256");
// Create a new SHA-256 digest
            String input = index + timestamp.toString() + data +
previousHash + nonce + difficulty; // Create a string to hash
```

```
            byte[] hash = digest.digest(input.getBytes()); // Hash the
input string
            StringBuilder hexString = new StringBuilder(); // Create a new
string builder

            // Convert the hash to a hex string
            for (byte b : hash) {
                String hex = Integer.toHexString(0xff & b); // Convert the
byte to a hex string
                if (hex.length() == 1) hexString.append('0'); // Add a
leading 0 if the hex string is only one character
                hexString.append(hex); // Add the hex string to the string
builder
            }
            return hexString.toString(); // Return the hex string

            // Catch the NoSuchAlgorithmException
        } catch (NoSuchAlgorithmException e) {
            throw new RuntimeException(e); // Throw a runtime exception
        }
    }

    // The proof of work methods finds a good hash.
    public String proofOfWork() {
        String target = new String(new char[difficulty]).replace('\0',
'0');
        while(!calculateHash().substring(0, difficulty).equals(target)) {
            nonce = nonce.add(BigInteger.ONE);
            hash = calculateHash();
        }
        return hash;
    }

    // Getters and Setters
    public int getIndex() { return index; }
    public void setIndex(int index) { this.index = index; }
    public Timestamp getTimestamp() { return timestamp; }
    public void setTimestamp(Timestamp timestamp) { this.timestamp =
timestamp; }
    public String getData() { return data; }
    public void setData(String data) { this.data = data; }
    public String getPreviousHash() { return previousHash; }
    public void setPreviousHash(String previousHash) { this.previousHash =
previousHash; }
    public BigInteger getNonce() { return nonce; }
    public int getDifficulty() { return difficulty; }
    public void setDifficulty(int difficulty) { this.difficulty =
difficulty; }

    @Override
    public String toString() {
        // Simplified JSON-like representation
        return
String.format("{\"index\":%d,\"timestamp\":\"%s\",\"data\":\"%s\",\"previo
usHash\":\"%s\",\"nonce\":\"%s\",\"difficulty\":%d,\"hash\":\"%s\"}",
                    index, timestamp.toString(), data, previousHash,
nonce.toString(), difficulty, hash);
    }

}
```

BlockChain.java

```
package org.example;

import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.sql.Timestamp;
import java.util.ArrayList;
import java.util.Scanner;

/**
 * Author: Louis Chang (hungyic)
 * Last Modified: 03/17/2024
 */
public class BlockChain {
    private ArrayList<Block> blockchain;
    private String chainHash;
    private int hashesPerSecond = 0;
    public final long numberOfHashes = 2000000; // 2 million hashes

    public BlockChain() {
        this.blockchain = new ArrayList<>();
        this.chainHash = "";
    }

    /**
     * A new Block is being added to the BlockChain.
     * @param newBlock
     */
    public void addBlock(Block newBlock) {
        if (!blockchain.isEmpty()) {
            newBlock.setPreviousHash(blockchain.get(blockchain.size() -
1).calculateHash());
        }
        newBlock.proofOfWork(); // Assuming proofOfWork sets the hash
inside the Block
        blockchain.add(newBlock);
        this.chainHash = newBlock.calculateHash(); // Update the chain
hash to the latest block's hash
    }

    /**
     * This method computes exactly 2 million hashes and times how long
that process takes.
     */
    public void computeHashesPerSecond() {
        final String textToHash = "00000000";
        try {
            MessageDigest digest = MessageDigest.getInstance("SHA-256");
            long startTime = System.nanoTime();
            for (int i = 0; i < numberOfHashes; i++) {
                byte[] hash = digest.digest(textToHash.getBytes());
            }
            long endTime = System.nanoTime();
            double durationInSeconds = (endTime - startTime) /
1_000_000_000.0; // Convert nanoseconds to seconds
            this.hashesPerSecond = (int) (numberOfHashes /
durationInSeconds);
            System.out.println("Hashes per second: " +
this.hashesPerSecond);
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        }
    }

    // Getter for hashesPerSecond
    public int getHashesPerSecond() {
        return hashesPerSecond;
    }

    /**
     * return block at position i
     * @param i
     * @return
     */
    public Block getBlock(int i) {
        return i >= 0 && i < blockchain.size() ? blockchain.get(i) : null;
    }

    public String getChainHash() {
        return this.chainHash;
    }

    public int getChainSize() {
        return blockchain.size();
    }

    public Block getLatestBlock() {
        return blockchain.isEmpty() ? null :
blockchain.get(blockchain.size() - 1);
    }

    /**
     * Compute and return the expected number of hashes required for the
entire chain.
     * @return
     */
    public double getTotalExpectedHashes() {
        double totalExpectedHashes = 0;
        for (Block block : blockchain) {
            int difficulty = block.getDifficulty();
            totalExpectedHashes += Math.pow(16, difficulty);
        }
        return totalExpectedHashes;
    }

    public int getTotalDifficulty() {
        int totalDifficulty = 0;
        for (Block block : blockchain) {
            totalDifficulty += block.getDifficulty();
        }
        return totalDifficulty;
    }

    // This method verifies the entire blockchain
    public String isChainValid() {
        if (blockchain.isEmpty()) {
            return "Blockchain is empty";
        }
        for (int i = 1; i < blockchain.size(); i++) {
            Block currentBlock = blockchain.get(i);
            Block previousBlock = blockchain.get(i - 1);

            if
(!currentBlock.getPreviousHash().equals(previousBlock.calculateHash())) {
                return "FALSE\nImproper hash on node " + (i-1) + " Does
not begin with 0000";
            }
        }
        return "TRUE";
    }

    // Additional methods like computeHashesPerSecond and repairChain
would go here

    @Override
    public String toString() {
        StringBuilder builder = new StringBuilder();
        for (Block block : blockchain) {
            builder.append(block.toString()).append("\n");
        }
        return builder.toString();
    }

    public static void main(String[] args) {
        BlockChain blockchain = new BlockChain();
        // Adding genesis block to the blockchain with a simple
transaction
        blockchain.addBlock(new Block(0, new
Timestamp(System.currentTimeMillis()), "Genesis", 2));

        while (true) {
            System.out.println("\n0. View basic blockchain status.");
            System.out.println("1. Add a transaction to the blockchain.");
```

```java
            System.out.println("2. Verify the blockchain.");
            System.out.println("3. View the blockchain.");
            System.out.println("4. Corrupt the chain.");
            System.out.println("5. Hide the corruption by repairing the
chain.");
            System.out.println("6. Exit");

            System.out.print("Enter your choice: ");
            Scanner scanner = new Scanner(System.in);
            int choice = scanner.nextInt();
            scanner.nextLine(); // Consume newline

            switch (choice) {
                case 0:
                    printBlockchainStatus(blockchain);
                    break;
                case 1:
                    addTransaction(blockchain);
                    break;
                case 2:
                    verifyBlockchain(blockchain);
                    break;
                case 3:
                    System.out.println(blockchain);
                    break;
                case 4:
                    corruptBlockchain(blockchain);
                    break;
                case 5:
                    System.out.println("Repairing the entire chain");
                    repairChain(blockchain);
                    break;
                case 6:
                    System.out.println("Exiting...");
                    System.exit(0);
                    break;
                default:
                    System.out.println("Invalid choice.");
            }
        }
    }

    private static void printBlockchainStatus(BlockChain blockchain) {
        System.out.println("Current size of chain: " +
blockchain.getChainSize());
        System.out.println("Difficulty of most recent block: " +
blockchain.getLatestBlock().getDifficulty());
        System.out.println("Total difficulty for all blocks: " +
blockchain.getTotalDifficulty());
        System.out.println("Experimented with: " +
blockchain.numberOfHashes + " hashes");
        System.out.println("Approximate hashes per second on this machine:
" + blockchain.getHashesPerSecond());
        System.out.println("Expected total hashes required for the whole
chain: " + blockchain.getTotalExpectedHashes());
        System.out.println("Nonce for most recent block: " +
blockchain.getLatestBlock().getNonce());
        System.out.println("Chain hash: " + blockchain.getChainHash());
    }

    // This method adds a new block to the blockchain
    private static void addTransaction(BlockChain blockchain) {
        System.out.print("Enter difficulty > ");
        Scanner scanner = new Scanner(System.in);
        int difficulty = scanner.nextInt();
        scanner.nextLine(); // Clean up newline
        System.out.print("Enter transaction: ");
        String transaction = scanner.nextLine();

        long startTime = System.currentTimeMillis();
        Block newBlock = new Block(blockchain.getChainSize(), new
Timestamp(System.currentTimeMillis()), transaction, difficulty);
        blockchain.addBlock(newBlock);
        long endTime = System.currentTimeMillis();
        System.out.println("Total execution time to add this block was " +
(endTime - startTime) + " milliseconds.");
    }

    // This method verifies the entire blockchain
    private static void verifyBlockchain(BlockChain blockchain) {
        String result = blockchain.isChainValid();
        System.out.println("Verifying entire chain");
        System.out.println("Chain verification: " + result);
    }

    // This method corrupts the blockchain
    private static void corruptBlockchain(BlockChain blockchain) {
        System.out.println("Currupt the Blockchain");
        System.out.print("Enter block ID of block to corrupt: ");
        Scanner scanner = new Scanner(System.in);
        int id = scanner.nextInt();
        scanner.nextLine(); // Clean up newline
        if (id >= 0 && id < blockchain.getChainSize()) {
            System.out.print("Enter new data for block " + id + ": ");
            String newData = scanner.nextLine();
            Block blockToCorrupt = blockchain.getBlock(id);
            blockToCorrupt.setData(newData);
            System.out.println("Block " + id + " now holds: " + newData);
        } else {
            System.out.println("Invalid block ID.");
        }
    }

    // This method repairs the entire blockchain
    public static void repairChain(BlockChain blockchain) {
        for (int i = 1; i < blockchain.blockchain.size(); i++) {
            Block currentBlock = blockchain.blockchain.get(i);
            Block previousBlock = blockchain.blockchain.get(i - 1);
```

```
              if
(!currentBlock.getPreviousHash().equals(previousBlock.calculateHash())) {

currentBlock.setPreviousHash(previousBlock.calculateHash());
                    currentBlock.proofOfWork();
              }
         }
     }
}
```

Execution Console

```
/Users/louischang/Library/Java/JavaVirtualMachines/openjdk-
20.0.1/Contents/Home/bin/java -javaagent:/Applications/IntelliJ
IDEA.app/Contents/lib/idea_rt.jar=57448:/Applications/IntelliJ IDEA.app/Contents/bin -
Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8 -
classpath /Users/louischang/Library/CloudStorage/OneDrive-
andrew.cmu.edu/DS/Project3/Project3Task0/target/classes:/Users/louischang/.m2/reposi
tory/com/google/code/gson/gson/2.9.0/gson-2.9.0.jar org.example.BlockChain


0. View basic blockchain status.

1. Add a transaction to the blockchain.

2. Verify the blockchain.

3. View the blockchain.

4. Corrupt the chain.

5. Hide the corruption by repairing the chain.

6. Exit

Enter your choice: 0

Current size of chain: 1

Difficulty of most recent block: 2

Total difficulty for all blocks: 2

Experimented with: 2000000 hashes

Approximate hashes per second on this machine: 0

Expected total hashes required for the whole chain: 256.0
```

```
Nonce for most recent block: 39

Chain hash:
0051b77008617946ce69ca9ce226e6852f77dabcbd003deda5b529d1b2e444a7


0. View basic blockchain status.

1. Add a transaction to the blockchain.

2. Verify the blockchain.

3. View the blockchain.

4. Corrupt the chain.

5. Hide the corruption by repairing the chain.

6. Exit

Enter your choice: 1

Enter difficulty > 4

Enter transaction: Alice pays Bob 100 DSCoin

Total execution time to add this block was 137 milliseconds.


0. View basic blockchain status.

1. Add a transaction to the blockchain.

2. Verify the blockchain.

3. View the blockchain.

4. Corrupt the chain.

5. Hide the corruption by repairing the chain.

6. Exit

Enter your choice: 1

Enter difficulty > 4

Enter transaction: Bob pays Carol 20 DSCoin
```

```
Total execution time to add this block was 530 milliseconds.


0. View basic blockchain status.

1. Add a transaction to the blockchain.

2. Verify the blockchain.

3. View the blockchain.

4. Corrupt the chain.

5. Hide the corruption by repairing the chain.

6. Exit

Enter your choice: 1

Enter difficulty > 4

Enter transaction: Carol pays Donna 10 DSCoin

Total execution time to add this block was 48 milliseconds.


0. View basic blockchain status.

1. Add a transaction to the blockchain.

2. Verify the blockchain.

3. View the blockchain.

4. Corrupt the chain.

5. Hide the corruption by repairing the chain.

6. Exit

Enter your choice: 3

{"index":0,"timestamp":"2024-03-17
10:15:40.999","data":"Genesis","previousHash":"0","nonce":"39","difficulty":2,"hash":"0051b
77008617946ce69ca9ce226e6852f77dabcbd003deda5b529d1b2e444a7"}

{"index":1,"timestamp":"2024-03-17 10:16:36.026","data":"Alice pays Bob 100
DSCoin","previousHash":"0051b77008617946ce69ca9ce226e6852f77dabcbd003deda5b5
29d1b2e444a7","nonce":"7641","difficulty":4,"hash":"0000255bc6484f623e162cfdf888ef17
45d9eb852d508d47a2548d1dc0562b95"}

{"index":2,"timestamp":"2024-03-17 10:16:46.247","data":"Bob pays Carol 20
DSCoin","previousHash":"0000255bc6484f623e162cfdf888ef1745d9eb852d508d47a2548
d1dc0562b95","nonce":"132766","difficulty":4,"hash":"00009e2aaa6f82e836742877053409
9ffe143b49cbfba66902c7a3cdbc332cd3"}

{"index":3,"timestamp":"2024-03-17 10:16:55.979","data":"Carol pays Donna 10
DSCoin","previousHash":"00009e2aaa6f82e8367428770534099ffe143b49cbfba66902c7a3
cdbc332cd3","nonce":"10013","difficulty":4,"hash":"0000340bf7f17a95765ca7b20137223bf
da0a10a35a5908493788196abeaf5d7"}



0. View basic blockchain status.

1. Add a transaction to the blockchain.

2. Verify the blockchain.

3. View the blockchain.

4. Corrupt the chain.

5. Hide the corruption by repairing the chain.

6. Exit

Enter your choice: 2

Verifying entire chain

Chain verification: TRUE


0. View basic blockchain status.

1. Add a transaction to the blockchain.

2. Verify the blockchain.

3. View the blockchain.

4. Corrupt the chain.
```

```
5. Hide the corruption by repairing the chain.

6. Exit

Enter your choice: 4

Currupt the Blockchain

Enter block ID of block to corrupt: 2

Enter new data for block 2: Bob pays Tony 30 DSCoin

Block 2 now holds: Bob pays Tony 30 DSCoin


0. View basic blockchain status.

1. Add a transaction to the blockchain.

2. Verify the blockchain.

3. View the blockchain.

4. Corrupt the chain.

5. Hide the corruption by repairing the chain.

6. Exit

Enter your choice: 3

{"index":0,"timestamp":"2024-03-17
10:15:40.999","data":"Genesis","previousHash":"0","nonce":"39","difficulty":2,"hash":"0051b
77008617946ce69ca9ce226e6852f77dabcbd003deda5b529d1b2e444a7"}

{"index":1,"timestamp":"2024-03-17 10:16:36.026","data":"Alice pays Bob 100
DSCoin","previousHash":"0051b77008617946ce69ca9ce226e6852f77dabcbd003deda5b5
29d1b2e444a7","nonce":"7641","difficulty":4,"hash":"0000255bc6484f623e162cfdf888ef17
45d9eb852d508d47a2548d1dc0562b95"}

{"index":2,"timestamp":"2024-03-17 10:16:46.247","data":"Bob pays Tony 30
DSCoin","previousHash":"0000255bc6484f623e162cfdf888ef1745d9eb852d508d47a2548
d1dc0562b95","nonce":"132766","difficulty":4,"hash":"00009e2aaa6f82e836742877053409
9ffe143b49cbfba66902c7a3cdbc332cd3"}

{"index":3,"timestamp":"2024-03-17 10:16:55.979","data":"Carol pays Donna 10
DSCoin","previousHash":"00009e2aaa6f82e8367428770534099ffe143b49cbfba66902c7a3
cdbc332cd3","nonce":"10013","difficulty":4,"hash":"0000340bf7f17a95765ca7b20137223bf
da0a10a35a5908493788196abeaf5d7"}


0. View basic blockchain status.

1. Add a transaction to the blockchain.

2. Verify the blockchain.

3. View the blockchain.

4. Corrupt the chain.

5. Hide the corruption by repairing the chain.

6. Exit

Enter your choice: 2

Verifying entire chain

Chain verification: FALSE

Improper hash on node 2 Does not begin with 0000


0. View basic blockchain status.

1. Add a transaction to the blockchain.

2. Verify the blockchain.

3. View the blockchain.

4. Corrupt the chain.

5. Hide the corruption by repairing the chain.

6. Exit

Enter your choice: 5

Repairing the entire chain
```

```
0. View basic blockchain status.

1. Add a transaction to the blockchain.

2. Verify the blockchain.

3. View the blockchain.

4. Corrupt the chain.

5. Hide the corruption by repairing the chain.

6. Exit

Enter your choice: 2

Verifying entire chain

Chain verification: TRUE


0. View basic blockchain status.

1. Add a transaction to the blockchain.

2. Verify the blockchain.

3. View the blockchain.

4. Corrupt the chain.

5. Hide the corruption by repairing the chain.

6. Exit

Enter your choice: 3

{"index":0,"timestamp":"2024-03-17
10:15:40.999","data":"Genesis","previousHash":"0","nonce":"39","difficulty":2,"hash":"0051b
77008617946ce69ca9ce226e6852f77dabcbd003deda5b529d1b2e444a7"}

{"index":1,"timestamp":"2024-03-17 10:16:36.026","data":"Alice pays Bob 100
DSCoin","previousHash":"0051b77008617946ce69ca9ce226e6852f77dabcbd003deda5b5
29d1b2e444a7","nonce":"7641","difficulty":4,"hash":"0000255bc6484f623e162cfdf888ef17
45d9eb852d508d47a2548d1dc0562b95"}

{"index":2,"timestamp":"2024-03-17 10:16:46.247","data":"Bob pays Tony 30
DSCoin","previousHash":"0000255bc6484f623e162cfdf888ef1745d9eb852d508d47a2548
d1dc0562b95","nonce":"132766","difficulty":4,"hash":"00009e2aaa6f82e836742877053409
9ffe143b49cbfba66902c7a3cdbc332cd3"}

{"index":3,"timestamp":"2024-03-17 10:16:55.979","data":"Carol pays Donna 10
DSCoin","previousHash":"bb21de141ec7dbd4c9864666b510d3d464559011dafead3fa164
6c33dcfc8517","nonce":"17422","difficulty":4,"hash":"0000c3a9fbd28760f0126a9135672ed
c48c376d4362eaca1bcd7be32782885d0"}



0. View basic blockchain status.

1. Add a transaction to the blockchain.

2. Verify the blockchain.

3. View the blockchain.

4. Corrupt the chain.

5. Hide the corruption by repairing the chain.

6. Exit

Enter your choice: 6

Exiting...


Process finished with exit code 0
```

Task 1

ServerTCP

```
package cmu.ds.project3;

import com.google.gson.Gson;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;
import java.sql.Timestamp;
import java.util.HashMap;
import java.util.Map;

/**
 * Author: Louis Chang (hungyic)
 * Last Modified: 03/17/2024
 */
public class ServerTCP {
    private static final BlockChain blockchain = new BlockChain();
    private static final Gson gson = new Gson();
    private static final int port = 7778;

    public static void main(String[] args) {
        ServerSocket serverSocket = null;
        try {
            serverSocket = new ServerSocket(port);
            System.out.println("Blockchain server running on port " +
port);

            Socket clientSocket = serverSocket.accept();
            BufferedReader in = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
            PrintWriter out = new
PrintWriter(clientSocket.getOutputStream(), true);

            // Adding genesis block to the blockchain with a simple
transaction
            blockchain.addBlock(new Block(0, new
Timestamp(System.currentTimeMillis()), "Genesis", 2));

            // Read input from client
            String inputLine;
            while ((inputLine = in.readLine()) != null) { // Keep reading
until client disconnects
                System.out.println("Received: " + inputLine);
                RequestMessage request = gson.fromJson(inputLine,
RequestMessage.class);
                ResponseMessage response = processRequest(request);
                out.println(gson.toJson(response));
                System.out.println("Sent: " + gson.toJson(response));
            }

            System.out.println("Client disconnected.");
            // Close resources for this client
            in.close();
            out.close();
            clientSocket.close();
        } catch (IOException e) {
            System.out.println("Server failed to start: " +
e.getMessage());
        } finally {
            if (serverSocket != null) {
                try {
                    serverSocket.close();
                } catch (IOException e) {
                    System.out.println("Could not close server socket: " +
e.getMessage());
                }
            }
        }
    }

    private static ResponseMessage processRequest(RequestMessage request)
{
        // Process the request and return a response
        Map<String, String> data = new HashMap<>();
        switch (request.getAction()) {
            case RequestMessage.VIEW_BLOCKCHAIN_STATUS:
                getBlockchainStatus(data);
                return new ResponseMessage(ResponseMessage.SUCCESS,
"Blockchain status printed.", data);

            case RequestMessage.ADD_TRANSACTION:
                addTransaction(request.getDifficulty(), request.getData(),
data);
                return new ResponseMessage(ResponseMessage.SUCCESS, "Block
added.", data);

            case RequestMessage.VERIFY_BLOCKCHAIN:
                verifyBlockchain(data);
                return new ResponseMessage(ResponseMessage.SUCCESS, "Chain
verification", data);

            case RequestMessage.VIEW_BLOCKCHAIN:
                data.put("blockchain", blockchain.toString());
                return new ResponseMessage(ResponseMessage.SUCCESS,
"Blockchain data.", data);

            case RequestMessage.CORRUPT_BLOCKCHAIN:
                try {
                    corruptBlockchain(request.getId(),
request.getNewData(), data);
                    return new ResponseMessage(ResponseMessage.SUCCESS,
"Currupt the Blockchain", data);
                } catch (IllegalArgumentException e) {
                    return new ResponseMessage(ResponseMessage.ERROR,
e.getMessage(), null);
                }
            case RequestMessage.REPAIR_CHAIN:
                repairChain(data);
                return new ResponseMessage(ResponseMessage.SUCCESS,
"Repairing the entire chain", data);
            default:
                return new ResponseMessage(ResponseMessage.ERROR,
"Unsupported action.", null);
        }
    }

    // Helper methods to process the request
    private static void getBlockchainStatus(Map<String, String> data) {
        data.put("Current size of chain",
String.valueOf(blockchain.getChainSize()));
        data.put("Difficulty of most recent block",
String.valueOf(blockchain.getLatestBlock().getDifficulty()));
        data.put("Experimented with hashes",
String.valueOf(blockchain.numberOfHashes));
        data.put("Total difficulty for all blocks",
String.valueOf(blockchain.getTotalDifficulty()));
        data.put("Approximate hashes per second on this machine",
String.valueOf(blockchain.getHashesPerSecond()));
        data.put("Expected total hashes required for the whole chain",
String.valueOf(blockchain.getTotalExpectedHashes()));
        data.put("Nonce for most recent block",
String.valueOf(blockchain.getLatestBlock().getNonce()));
        data.put("Chain hash", blockchain.getChainHash());
    }

    // This method adds a new block to the blockchain
    private static void addTransaction(int difficulty, String transaction,
Map<String, String> data) {
        long startTime = System.currentTimeMillis();
        Block newBlock = new Block(blockchain.getChainSize(), new
Timestamp(System.currentTimeMillis()), transaction, difficulty);
        blockchain.addBlock(newBlock);
        long endTime = System.currentTimeMillis();
        data.put("Transaction", transaction);
        data.put("Total execution time to add this block",
String.valueOf(endTime - startTime));
    }

    // This method verifies the entire blockchain
    private static void verifyBlockchain(Map<String, String> data) {
        String result = blockchain.isChainValid();
        data.put("Verification", result);
    }

    // This method corrupts the blockchain
```

```java
    private static void corruptBlockchain(int id, String newData,
Map<String, String> data) {
        if (id >= 0 && id < blockchain.getChainSize()) {
            System.out.print("Corrupting block " + id + " with new data: "
+ newData + "\n");
            Block blockToCorrupt = blockchain.getBlock(id);
            blockToCorrupt.setData(newData);
            data.put("CorruptedBlockID", String.valueOf(id));
            data.put("CorruptedBlockData", newData);
        } else {
            System.out.println("Invalid block ID.");
            throw new IllegalArgumentException("Invalid block ID.");
        }
    }

    // This method repairs the blockchain
    private static void repairChain(Map<String, String> data) {
        long startTime = System.currentTimeMillis();
        blockchain.repairChain();
        long endTime = System.currentTimeMillis();
        data.put("Total execution time to add this block",
String.valueOf(endTime - startTime));
    }
}
```

ClientTCP

```java
package cmu.ds.project3;

import com.google.gson.Gson;
import java.net.Socket;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.io.IOException;
import java.util.Map;
import java.util.Scanner;

/**
 * Author: Louis Chang (hungyic)
 * Last Modified: 03/17/2024
 */
public class ClientTCP {
    private static final Gson gson = new Gson();
    private static final int PORT = 7778;
    private static final String HOST = "localhost";

    public static void main(String[] args) {
        try {
            Socket socket = new Socket(HOST, PORT);
            PrintWriter out = new PrintWriter(socket.getOutputStream(),
true);
            BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
            Scanner scanner = new Scanner(System.in);

            String userInput;
            while (true) {
                System.out.println("\n0. View basic blockchain status.");
                System.out.println("1. Add a transaction to the
blockchain.");
                System.out.println("2. Verify the blockchain.");
                System.out.println("3. View the blockchain.");
                System.out.println("4. Corrupt the chain.");
                System.out.println("5. Hide the corruption by repairing
the chain.");
                System.out.println("6. Exit");

                System.out.print("Enter your choice: ");
                userInput = scanner.nextLine();

                if ("6".equals(userInput)) {
                    System.out.println("Exiting...");
                    break;
                }

                // Create request message based on user input
                RequestMessage request;
                switch (userInput) {
                    case "0":
                        request = new
RequestMessage(RequestMessage.VIEW_BLOCKCHAIN_STATUS);
                        break;
                    case "1":
                        System.out.print("Enter difficulty > ");
                        int difficulty =
Integer.parseInt(scanner.nextLine());
                        System.out.print("Enter transaction: ");
                        String transaction = scanner.nextLine();
                        request = new
RequestMessage(RequestMessage.ADD_TRANSACTION, transaction, difficulty);
                        break;
                    case "2":
                        request = new
RequestMessage(RequestMessage.VERIFY_BLOCKCHAIN);
                        break;
                    case "3":
                        request = new
RequestMessage(RequestMessage.VIEW_BLOCKCHAIN);
                        break;
                    case "4":
                        System.out.print("Enter block ID of block to
corrupt: ");
                        int id = Integer.parseInt(scanner.nextLine());
                        System.out.print("Enter new data for block " + id
+ ": ");
                        String newData = scanner.nextLine();
```

```
                            request = new
RequestMessage(RequestMessage.CORRUPT_BLOCKCHAIN, id, newData);
                            break;
                        case "5":
                            request = new
RequestMessage(RequestMessage.REPAIR_CHAIN);
                            break;
                        default:
                            System.out.println("Invalid choice.");
                            continue;
                    }

                    String jsonRequest = gson.toJson(request);
                    out.println(jsonRequest);

                    String jsonResponse = in.readLine();
                    ResponseMessage response = gson.fromJson(jsonResponse,
ResponseMessage.class);
                    System.out.println("Server response: " +
response.getMessage());
                    for (Map.Entry<String, String> entry :
response.getData().entrySet()) {
                        System.out.println(entry.getKey() + ": " +
entry.getValue());
                    }
                }
            } catch (IOException e) {
                System.out.println("Client error: " + e.getMessage());
            }
        }
}
```

RequestMessage

```
package cmu.ds.project3;

/**
 * Author: Louis Chang (hungyic)
 * Last Modified: 03/17/2024
 */
public class RequestMessage {
    public static final String VIEW_BLOCKCHAIN_STATUS =
"viewBlockchainStatus";
    public static final String ADD_TRANSACTION = "addTransaction";
    public static final String VERIFY_BLOCKCHAIN = "verifyBlockchain";
    public static final String VIEW_BLOCKCHAIN = "viewBlockchain";
    public static final String CORRUPT_BLOCKCHAIN = "corruptBlockchain";
    public static final String REPAIR_CHAIN = "repairChain";

    private String action;
    private String data;
    private int difficulty;
    private int id; // For corrupting a block
    private String newData; // For corrupting a block

    public RequestMessage(String action) {
        this.action = action;
    }

    public RequestMessage(String action, String data, int difficulty) {
        this.action = action;
        this.data = data;
        this.difficulty = difficulty;
    }

    public RequestMessage(String action, int id, String newData) {
        this.action = action;
        this.id = id;
        this.newData = newData;
    }

    // Additional getters for the new fields
    public int getId() {
        return id;
    }

    public String getNewData() {
        return newData;
    }

    public String getAction() {
        return action;
    }

    public void setAction(String action) {
        this.action = action;
    }

    public String getData() {
        return data;
    }

    public void setData(String data) {
        this.data = data;
    }

    public int getDifficulty() {
        return difficulty;
    }

    public void setDifficulty(int difficulty) {
        this.difficulty = difficulty;
    }

    public void setId(int id) {
        this.id = id;
    }
    public void setNewData(String newData) {
        this.newData = newData;
    }
}
```

ResponseMessage

```
package cmu.ds.project3;

import java.util.Map;

/**
 * Author: Louis Chang (hungyic)
 * Last Modified: 03/17/2024
 */
public class ResponseMessage {
    public static final String SUCCESS = "success";
    public static final String ERROR = "error";

    private String status;
    private String message;
    private Map<String, String> data;

    public ResponseMessage(String status, String message) {
        this.status = status;
        this.message = message;
    }

    public ResponseMessage(String status, String message, Map<String,
String> data) {
        this.status = status;
        this.message = message;
        this.data = data;
    }

    // Getters
    public String getStatus() {
        return status;
    }

    public String getMessage() {
        return message;
    }

    public Map<String, String> getData() {
        return data;
    }
}
```

Execution Console

Server

```
/Users/louischang/Library/Java/JavaVirtualMachines/openjdk-20.0.1/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA.app/Contents/lib/idea_rt.jar=58792:/Applications/IntelliJ IDEA.app/Contents/bin -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8 -classpath /Users/louischang/Library/CloudStorage/OneDrive-andrew.cmu.edu/DS/Project3/Project3Task1/target/classes:/Users/louischang/.m2/repository/com/google/code/gson/gson/2.9.0/gson-2.9.0.jar cmu.ds.project3.ServerTCP

Blockchain server running on port 7778

Received: {"action":"viewBlockchainStatus","difficulty":0,"id":0}

Sent: {"status":"success","message":"Blockchain status printed.","data":{"Approximate hashes per second on this machine":"0","Chain hash":"0010c3116b1a29228e058028606e94eb14b9535fbc219f8051283741bd98c6f5","Total difficulty for all blocks":"2","Current size of chain":"1","Difficulty of most recent block":"2","Experimented with hashes":"2000000","Expected total hashes required for the whole chain":"256.0","Nonce for most recent block":"15"}}

Received: {"action":"addTransaction","data":"Alice pays Bob 100 DSCoin","difficulty":4,"id":0}

Sent: {"status":"success","message":"Block added.","data":{"Total execution time to add this block":"495","Transaction":"Alice pays Bob 100 DSCoin"}}

Received: {"action":"addTransaction","data":"Bob pays Carol 20 DSCoin","difficulty":4,"id":0}

Sent: {"status":"success","message":"Block added.","data":{"Total execution time to add this block":"140","Transaction":"Bob pays Carol 20 DSCoin"}}

Received: {"action":"addTransaction","data":"Carol pays Donna 10 DSCoin","difficulty":4,"id":0}

Sent: {"status":"success","message":"Block added.","data":{"Total execution time to add this block":"71","Transaction":"Carol pays Donna 10 DSCoin"}}

Received: {"action":"viewBlockchain","difficulty":0,"id":0}

Sent: {"status":"success","message":"Blockchain data.","data":{"blockchain":"{\"index\":0,\"timestamp\":\"2024-03-17 10:48:17.259\",\"data\":\"Genesis\",\"previousHash\":\"0\",\"nonce\":\"15\",\"difficulty\":2,\"
```

```
hash\":\"0010c3116b1a29228e058028606e94eb14b9535fbc219f8051283741bd98c6f5\"}\
n{\"index\":1,\"timestamp\":\"2024-03-17 10:48:29.38\",\"data\":\"Alice pays Bob 100
DSCoin\",\"previousHash\":\"0010c3116b1a29228e058028606e94eb14b9535fbc219f8051
283741bd98c6f5\",\"nonce\":\"146698\",\"difficulty\":4,\"hash\":\"00006bd8f449480e8363
7c4631fa844dc516661e9014f7cd30d742b9a51c1c8e\"}\n{\"index\":2,\"timestamp\":\"202
4-03-17 10:48:37.136\",\"data\":\"Bob pays Carol 20
DSCoin\",\"previousHash\":\"00006bd8f449480e83637c4631fa844dc516661e9014f7cd30
d742b9a51c1c8e\",\"nonce\":\"68233\",\"difficulty\":4,\"hash\":\"0000f5e8e94c3d06ac36d
266c4f772280841e7ccb135d8811670cb6e41d8967b\"}\n{\"index\":3,\"timestamp\":\"2024
-03-17 10:48:45.687\",\"data\":\"Carol pays Donna 10
DSCoin\",\"previousHash\":\"0000f5e8e94c3d06ac36d266c4f772280841e7ccb135d88116
70cb6e41d8967b\",\"nonce\":\"31708\",\"difficulty\":4,\"hash\":\"0000715a36ec4b5da7dd
7e6917a09fc29261812b64aada4cb6952b5cae562ec6\"}\n"}}

Received: {"action":"verifyBlockchain","difficulty":0,"id":0}

Sent: {"status":"success","message":"Chain verification","data":{"Verification":"TRUE"}}

Received: {"action":"corruptBlockchain","difficulty":0,"id":2,"newData":"Bob pays Tony 30
DSCoin"}

Corrupting block 2 with new data: Bob pays Tony 30 DSCoin

Sent: {"status":"success","message":"Currupt the
Blockchain","data":{"CorruptedBlockID":"2","CorruptedBlockData":"Bob pays Tony 30
DSCoin"}}

Received: {"action":"viewBlockchain","difficulty":0,"id":0}

Sent: {"status":"success","message":"Blockchain
data.","data":{"blockchain":"{\"index\":0,\"timestamp\":\"2024-03-17
10:48:17.259\",\"data\":\"Genesis\",\"previousHash\":\"0\",\"nonce\":\"15\",\"difficulty\":2,\"
hash\":\"0010c3116b1a29228e058028606e94eb14b9535fbc219f8051283741bd98c6f5\"}\
n{\"index\":1,\"timestamp\":\"2024-03-17 10:48:29.38\",\"data\":\"Alice pays Bob 100
DSCoin\",\"previousHash\":\"0010c3116b1a29228e058028606e94eb14b9535fbc219f8051
283741bd98c6f5\",\"nonce\":\"146698\",\"difficulty\":4,\"hash\":\"00006bd8f449480e8363
7c4631fa844dc516661e9014f7cd30d742b9a51c1c8e\"}\n{\"index\":2,\"timestamp\":\"202
4-03-17 10:48:37.136\",\"data\":\"Bob pays Tony 30
DSCoin\",\"previousHash\":\"00006bd8f449480e83637c4631fa844dc516661e9014f7cd30
d742b9a51c1c8e\",\"nonce\":\"68233\",\"difficulty\":4,\"hash\":\"0000f5e8e94c3d06ac36d
266c4f772280841e7ccb135d8811670cb6e41d8967b\"}\n{\"index\":3,\"timestamp\":\"2024
-03-17 10:48:45.687\",\"data\":\"Carol pays Donna 10
```

```
DSCoin\",\"previousHash\":\"0000f5e8e94c3d06ac36d266c4f772280841e7ccb135d88116
70cb6e41d8967b\",\"nonce\":\"31708\",\"difficulty\":4,\"hash\":\"0000715a36ec4b5da7dd
7e6917a09fc29261812b64aada4cb6952b5cae562ec6\"}\n"}}

Received: {"action":"verifyBlockchain","difficulty":0,"id":0}

Sent: {"status":"success","message":"Chain
verification","data":{"Verification":"FALSE\nImproper hash on node 2 Does not begin with
0000"}}

Received: {"action":"repairChain","difficulty":0,"id":0}

Sent: {"status":"success","message":"Repairing the entire chain","data":{"Total execution
time to add this block":"1"}}

Received: {"action":"verifyBlockchain","difficulty":0,"id":0}

Sent: {"status":"success","message":"Chain verification","data":{"Verification":"TRUE"}}

Received: {"action":"viewBlockchain","difficulty":0,"id":0}

Sent: {"status":"success","message":"Blockchain
data.","data":{"blockchain":"{\"index\":0,\"timestamp\":\"2024-03-17
10:48:17.259\",\"data\":\"Genesis\",\"previousHash\":\"0\",\"nonce\":\"15\",\"difficulty\":2,\"
hash\":\"0010c3116b1a29228e058028606e94eb14b9535fbc219f8051283741bd98c6f5\"}\
n{\"index\":1,\"timestamp\":\"2024-03-17 10:48:29.38\",\"data\":\"Alice pays Bob 100
DSCoin\",\"previousHash\":\"0010c3116b1a29228e058028606e94eb14b9535fbc219f8051
283741bd98c6f5\",\"nonce\":\"146698\",\"difficulty\":4,\"hash\":\"00006bd8f449480e8363
7c4631fa844dc516661e9014f7cd30d742b9a51c1c8e\"}\n{\"index\":2,\"timestamp\":\"202
4-03-17 10:48:37.136\",\"data\":\"Bob pays Tony 30
DSCoin\",\"previousHash\":\"00006bd8f449480e83637c4631fa844dc516661e9014f7cd30
d742b9a51c1c8e\",\"nonce\":\"68233\",\"difficulty\":4,\"hash\":\"0000f5e8e94c3d06ac36d
266c4f772280841e7ccb135d8811670cb6e41d8967b\"}\n{\"index\":3,\"timestamp\":\"2024
-03-17 10:48:45.687\",\"data\":\"Carol pays Donna 10
DSCoin\",\"previousHash\":\"d0f8b8994f5ccae44bad9e555f2fe0d2bf59ddcc699dcaad4b6
3678ad03e7605\",\"nonce\":\"31708\",\"difficulty\":4,\"hash\":\"0000715a36ec4b5da7dd7
e6917a09fc29261812b64aada4cb6952b5cae562ec6\"}\n"}}

Client disconnected.


Process finished with exit code 0
```

Client

```
/Users/louischang/Library/Java/JavaVirtualMachines/openjdk-
20.0.1/Contents/Home/bin/java -javaagent:/Applications/IntelliJ
IDEA.app/Contents/lib/idea_rt.jar=58796:/Applications/IntelliJ IDEA.app/Contents/bin -
Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8 -
classpath /Users/louischang/Library/CloudStorage/OneDrive-
andrew.cmu.edu/DS/Project3/Project3Task1/target/classes:/Users/louischang/.m2/reposi
tory/com/google/code/gson/gson/2.9.0/gson-2.9.0.jar cmu.ds.project3.ClientTCP


0. View basic blockchain status.
1. Add a transaction to the blockchain.
2. Verify the blockchain.
3. View the blockchain.
4. Corrupt the chain.
5. Hide the corruption by repairing the chain.
6. Exit
Enter your choice: 0
Server response: Blockchain status printed.
Approximate hashes per second on this machine: 0
Chain hash: 0010c3116b1a29228e058028606e94eb14b9535fbc219f8051283741bd98c6f5
Total difficulty for all blocks: 2
Current size of chain: 1
Difficulty of most recent block: 2
Experimented with hashes: 2000000
Expected total hashes required for the whole chain: 256.0
Nonce for most recent block: 15
```

```
0. View basic blockchain status.

1. Add a transaction to the blockchain.

2. Verify the blockchain.

3. View the blockchain.

4. Corrupt the chain.

5. Hide the corruption by repairing the chain.

6. Exit

Enter your choice: 1

Enter difficulty > 4

Enter transaction: Alice pays Bob 100 DSCoin

Server response: Block added.

Total execution time to add this block: 495

Transaction: Alice pays Bob 100 DSCoin


0. View basic blockchain status.

1. Add a transaction to the blockchain.

2. Verify the blockchain.

3. View the blockchain.

4. Corrupt the chain.

5. Hide the corruption by repairing the chain.

6. Exit

Enter your choice: 1

Enter difficulty > 4

Enter transaction: Bob pays Carol 20 DSCoin

Server response: Block added.

Total execution time to add this block: 140
```

```
Transaction: Bob pays Carol 20 DSCoin


0. View basic blockchain status.

1. Add a transaction to the blockchain.

2. Verify the blockchain.

3. View the blockchain.

4. Corrupt the chain.

5. Hide the corruption by repairing the chain.

6. Exit

Enter your choice: 1

Enter difficulty > 4

Enter transaction: Carol pays Donna 10 DSCoin

Server response: Block added.

Total execution time to add this block: 71

Transaction: Carol pays Donna 10 DSCoin


0. View basic blockchain status.

1. Add a transaction to the blockchain.

2. Verify the blockchain.

3. View the blockchain.

4. Corrupt the chain.

5. Hide the corruption by repairing the chain.

6. Exit

Enter your choice: 3

Server response: Blockchain data.
```

```
blockchain: {"index":0,"timestamp":"2024-03-17
10:48:17.259","data":"Genesis","previousHash":"0","nonce":"15","difficulty":2,"hash":"0010c
3116b1a29228e058028606e94eb14b9535fbc219f8051283741bd98c6f5"}

{"index":1,"timestamp":"2024-03-17 10:48:29.38","data":"Alice pays Bob 100
DSCoin","previousHash":"0010c3116b1a29228e058028606e94eb14b9535fbc219f8051283
741bd98c6f5","nonce":"146698","difficulty":4,"hash":"00006bd8f449480e83637c4631fa844
dc516661e9014f7cd30d742b9a51c1c8e"}

{"index":2,"timestamp":"2024-03-17 10:48:37.136","data":"Bob pays Carol 20
DSCoin","previousHash":"00006bd8f449480e83637c4631fa844dc516661e9014f7cd30d74
2b9a51c1c8e","nonce":"68233","difficulty":4,"hash":"0000f5e8e94c3d06ac36d266c4f7722
80841e7ccb135d8811670cb6e41d8967b"}

{"index":3,"timestamp":"2024-03-17 10:48:45.687","data":"Carol pays Donna 10
DSCoin","previousHash":"0000f5e8e94c3d06ac36d266c4f772280841e7ccb135d8811670c
b6e41d8967b","nonce":"31708","difficulty":4,"hash":"0000715a36ec4b5da7dd7e6917a09fc
29261812b64aada4cb6952b5cae562ec6"}



0. View basic blockchain status.

1. Add a transaction to the blockchain.

2. Verify the blockchain.

3. View the blockchain.

4. Corrupt the chain.

5. Hide the corruption by repairing the chain.

6. Exit

Enter your choice: 2

Server response: Chain verification

Verification: TRUE


0. View basic blockchain status.
```

```
1. Add a transaction to the blockchain.

2. Verify the blockchain.

3. View the blockchain.

4. Corrupt the chain.

5. Hide the corruption by repairing the chain.

6. Exit

Enter your choice: 4

Enter block ID of block to corrupt: 2

Enter new data for block 2: Bob pays Tony 30 DSCoin

Server response: Currupt the Blockchain

CorruptedBlockID: 2

CorruptedBlockData: Bob pays Tony 30 DSCoin


0. View basic blockchain status.

1. Add a transaction to the blockchain.

2. Verify the blockchain.

3. View the blockchain.

4. Corrupt the chain.

5. Hide the corruption by repairing the chain.

6. Exit

Enter your choice: 3

Server response: Blockchain data.

blockchain: {"index":0,"timestamp":"2024-03-17
10:48:17.259","data":"Genesis","previousHash":"0","nonce":"15","difficulty":2,"hash":"0010c
3116b1a29228e058028606e94eb14b9535fbc219f8051283741bd98c6f5"}

{"index":1,"timestamp":"2024-03-17 10:48:29.38","data":"Alice pays Bob 100
DSCoin","previousHash":"0010c3116b1a29228e058028606e94eb14b9535fbc219f8051283
741bd98c6f5","nonce":"146698","difficulty":4,"hash":"00006bd8f449480e83637c4631fa844
dc516661e9014f7cd30d742b9a51c1c8e"}

{"index":2,"timestamp":"2024-03-17 10:48:37.136","data":"Bob pays Tony 30
DSCoin","previousHash":"00006bd8f449480e83637c4631fa844dc516661e9014f7cd30d74
2b9a51c1c8e","nonce":"68233","difficulty":4,"hash":"0000f5e8e94c3d06ac36d266c4f7722
80841e7ccb135d8811670cb6e41d8967b"}

{"index":3,"timestamp":"2024-03-17 10:48:45.687","data":"Carol pays Donna 10
DSCoin","previousHash":"0000f5e8e94c3d06ac36d266c4f772280841e7ccb135d8811670c
b6e41d8967b","nonce":"31708","difficulty":4,"hash":"0000715a36ec4b5da7dd7e6917a09fc
29261812b64aada4cb6952b5cae562ec6"}



0. View basic blockchain status.

1. Add a transaction to the blockchain.

2. Verify the blockchain.

3. View the blockchain.

4. Corrupt the chain.

5. Hide the corruption by repairing the chain.

6. Exit

Enter your choice: 2

Server response: Chain verification

Verification: FALSE

Improper hash on node 2 Does not begin with 0000


0. View basic blockchain status.

1. Add a transaction to the blockchain.

2. Verify the blockchain.

3. View the blockchain.
```

```
4. Corrupt the chain.

5. Hide the corruption by repairing the chain.

6. Exit

Enter your choice: 5

Server response: Repairing the entire chain

Total execution time to add this block: 1


0. View basic blockchain status.

1. Add a transaction to the blockchain.

2. Verify the blockchain.

3. View the blockchain.

4. Corrupt the chain.

5. Hide the corruption by repairing the chain.

6. Exit

Enter your choice: 2

Server response: Chain verification

Verification: TRUE


0. View basic blockchain status.

1. Add a transaction to the blockchain.

2. Verify the blockchain.

3. View the blockchain.

4. Corrupt the chain.

5. Hide the corruption by repairing the chain.

6. Exit

Enter your choice: 3
```

```
Server response: Blockchain data.

blockchain: {"index":0,"timestamp":"2024-03-17
10:48:17.259","data":"Genesis","previousHash":"0","nonce":"15","difficulty":2,"hash":"0010c
3116b1a29228e058028606e94eb14b9535fbc219f8051283741bd98c6f5"}

{"index":1,"timestamp":"2024-03-17 10:48:29.38","data":"Alice pays Bob 100
DSCoin","previousHash":"0010c3116b1a29228e058028606e94eb14b9535fbc219f8051283
741bd98c6f5","nonce":"146698","difficulty":4,"hash":"00006bd8f449480e83637c4631fa844
dc516661e9014f7cd30d742b9a51c1c8e"}

{"index":2,"timestamp":"2024-03-17 10:48:37.136","data":"Bob pays Tony 30
DSCoin","previousHash":"00006bd8f449480e83637c4631fa844dc516661e9014f7cd30d74
2b9a51c1c8e","nonce":"68233","difficulty":4,"hash":"0000f5e8e94c3d06ac36d266c4f7722
80841e7ccb135d8811670cb6e41d8967b"}

{"index":3,"timestamp":"2024-03-17 10:48:45.687","data":"Carol pays Donna 10
DSCoin","previousHash":"d0f8b8994f5ccae44bad9e555f2fe0d2bf59ddcc699dcaad4b6367
8ad03e7605","nonce":"31708","difficulty":4,"hash":"0000715a36ec4b5da7dd7e6917a09fc2
9261812b64aada4cb6952b5cae562ec6"}



0. View basic blockchain status.

1. Add a transaction to the blockchain.

2. Verify the blockchain.

3. View the blockchain.

4. Corrupt the chain.

5. Hide the corruption by repairing the chain.

6. Exit

Enter your choice: 6

Exiting...


Process finished with exit code 0
```

Task 2

VerifyingServerTCP

```
package cmu.ds.project3;

import java.io.*;
import java.math.BigInteger;
import java.net.ServerSocket;
import java.net.Socket;
import java.security.*;
import java.sql.Timestamp;
import java.util.Arrays;
import java.util.HashMap;
import java.util.Map;

import com.google.gson.Gson;

/**
 * Author: Louis Chang (hungyic)
 * Last Modified: 03/17/2024
 */
public class VerifyingServerTCP {
    private static final Gson gson = new Gson();
    private static final int PORT = 7778;
    private static final BlockChain blockchain = new BlockChain();

    public static void main(String[] args) {
        ServerSocket serverSocket = null;
        try {
            serverSocket = new ServerSocket(PORT);
            System.out.println("Blockchain server running on port " +
PORT);

            Socket clientSocket = serverSocket.accept();
            BufferedReader in = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
            PrintWriter out = new
PrintWriter(clientSocket.getOutputStream(), true);

            // Adding genesis block to the blockchain with a simple
transaction
            blockchain.addBlock(new Block(0, new
Timestamp(System.currentTimeMillis()), "Genesis", 2));

            // Read input from client
            String inputLine;
            while ((inputLine = in.readLine()) != null) { // Keep reading
until client disconnects
                System.out.println("Received: " + inputLine);
                SignedRequest signedRequest = gson.fromJson(inputLine,
SignedRequest.class);

                // Verify the signature and process the request
                if (verify(signedRequest)) {
                    ResponseMessage response =
processRequest(signedRequest.getRequest());
                    out.println(gson.toJson(response));
                    System.out.println("Sent: " + gson.toJson(response));
                } else {
                    out.println(gson.toJson(new ResponseMessage("Error",
"Invalid signature or request.")));
                }
            }

            System.out.println("Client disconnected.");
            // Close resources for this client
            in.close();
            out.close();
            clientSocket.close();
        } catch (Exception e) {
            System.out.println("Server failed to start: " +
e.getMessage());
        } finally {
            if (serverSocket != null) {
                try {
                    serverSocket.close();
                } catch (IOException e) {
                    System.out.println("Could not close server socket: " +
e.getMessage());
                }
            }
        }
    }

    /**
     * Verify the signature of the request
     * @param signedRequest
     * @return
     * @throws Exception
     */
    private static boolean verify(SignedRequest signedRequest) throws
Exception {
        BigInteger e = signedRequest.getE();
        BigInteger n = signedRequest.getN();
        BigInteger clientID = signedRequest.getClientID();
        String signature = signedRequest.getSignature();

        MessageDigest sha256 = MessageDigest.getInstance("SHA-256");
        String publicKeyConcat = e.toString() + n.toString(); // 串接 e 和 n
        byte[] publicKeyHash =
sha256.digest(publicKeyConcat.getBytes("UTF-8"));


        byte[] derivedClientIDBytes = Arrays.copyOfRange(publicKeyHash,
publicKeyHash.length - 20, publicKeyHash.length);
        BigInteger derivedClientID = new BigInteger(1,
```

```java
derivedClientIDBytes); // 將字節數組轉換為 BigInteger

        if (!clientID.equals(derivedClientID)) {
            return false;
        }

        String concatenatedValues =
signedRequest.getRequest().concatenateValues();

        byte[] hashOfConcatenatedValues =
sha256.digest(concatenatedValues.getBytes("UTF-8"));

        byte[] hashForVerification = new byte[3];
        hashForVerification[0] = 0;
        hashForVerification[1] = hashOfConcatenatedValues[0];
        hashForVerification[2] = hashOfConcatenatedValues[1];
        BigInteger hashBigInteger = new BigInteger(hashForVerification);

        BigInteger encryptedHash = new BigInteger(signature);
        BigInteger decryptedHash = encryptedHash.modPow(e, n);

        return hashBigInteger.compareTo(decryptedHash) == 0;
    }

    /**
     * Process the request and return a response
     * @param request
     * @return
     */
    private static ResponseMessage processRequest(RequestMessage request)
{
        Map<String, String> data = new HashMap<>();
        switch (request.getAction()) {
            case RequestMessage.VIEW_BLOCKCHAIN_STATUS:
                getBlockchainStatus(data);
                return new ResponseMessage(ResponseMessage.SUCCESS,
"Blockchain status printed.", data);
            case RequestMessage.ADD_TRANSACTION:
                addTransaction(request.getDifficulty(), request.getData(),
data);
                return new ResponseMessage(ResponseMessage.SUCCESS, "Block
added.", data);
            case RequestMessage.VERIFY_BLOCKCHAIN:
                verifyBlockchain(data);
                return new ResponseMessage(ResponseMessage.SUCCESS, "Chain
verification", data);
            case RequestMessage.VIEW_BLOCKCHAIN:
                data.put("blockchain", blockchain.toString());
                return new ResponseMessage(ResponseMessage.SUCCESS,
"Blockchain data.", data);
            case RequestMessage.CORRUPT_BLOCKCHAIN:
                try {
                    corruptBlockchain(request.getId(),
request.getNewData(), data);
                    return new ResponseMessage(ResponseMessage.SUCCESS,
"Currupt the Blockchain", data);
                } catch (IllegalArgumentException e) {
                    return new ResponseMessage(ResponseMessage.ERROR,
e.getMessage(), null);
                }
            case RequestMessage.REPAIR_CHAIN:
                repairChain(data);
                return new ResponseMessage(ResponseMessage.SUCCESS,
"Repairing the entire chain", data);
            default:
                return new ResponseMessage(ResponseMessage.ERROR,
"Unsupported action.", null);
        }
    }

    /**
     * Get the status of the blockchain
     * @param data
     */
    private static void getBlockchainStatus(Map<String, String> data) {
        data.put("Current size of chain",
String.valueOf(blockchain.getChainSize()));
        data.put("Difficulty of most recent block",
String.valueOf(blockchain.getLatestBlock().getDifficulty()));
        data.put("Experimented with hashes",
String.valueOf(blockchain.numberOfHashes));
        data.put("Total difficulty for all blocks",
String.valueOf(blockchain.getTotalDifficulty()));
        data.put("Approximate hashes per second on this machine",
String.valueOf(blockchain.getHashesPerSecond()));
        data.put("Expected total hashes required for the whole chain",
String.valueOf(blockchain.getTotalExpectedHashes()));
        data.put("Nonce for most recent block",
String.valueOf(blockchain.getLatestBlock().getNonce()));
        data.put("Chain hash", blockchain.getChainHash());
    }

    /**
     * Add a new block to the blockchain
     * @param difficulty
     * @param transaction
     * @param data
     */
    private static void addTransaction(int difficulty, String transaction,
Map<String, String> data) {
        long startTime = System.currentTimeMillis();
        Block newBlock = new Block(blockchain.getChainSize(), new
Timestamp(System.currentTimeMillis()), transaction, difficulty);
        blockchain.addBlock(newBlock);
        long endTime = System.currentTimeMillis();
        data.put("Transaction", transaction);
        data.put("Total execution time to add this block",
String.valueOf(endTime - startTime));
    }

    /**
     * Verify the entire blockchain
     * @param data
     */
    private static void verifyBlockchain(Map<String, String> data) {
        String result = blockchain.isChainValid();
        data.put("Verification", result);
    }

    /**
     * Corrupt the blockchain
     * @param id
     * @param newData
     * @param data
     */
    private static void corruptBlockchain(int id, String newData,
Map<String, String> data) {
        if (id >= 0 && id < blockchain.getChainSize()) {
            System.out.print("Corrupting block " + id + " with new data: "
+ newData + "\n");
            Block blockToCorrupt = blockchain.getBlock(id);
            blockToCorrupt.setData(newData);
            data.put("CorruptedBlockID", String.valueOf(id));
            data.put("CorruptedBlockData", newData);
        } else {
            System.out.println("Invalid block ID.");
            throw new IllegalArgumentException("Invalid block ID.");
        }
    }

    /**
     * Repair the blockchain
     * @param data
     */
    private static void repairChain(Map<String, String> data) {
        long startTime = System.currentTimeMillis();
        blockchain.repairChain();
        long endTime = System.currentTimeMillis();
        data.put("Total execution time to add this block",
String.valueOf(endTime - startTime));
    }
}
```

SigningClientTCP

```
package cmu.ds.project3;

import java.io.*;
import java.math.BigInteger;
import java.net.Socket;
```

```java
import java.security.*;
import java.security.spec.InvalidKeySpecException;
import java.util.Arrays;
import java.util.Random;
import java.util.Scanner;
import com.google.gson.Gson;

/**
 * Author: Louis Chang (hungyic)
 * Last Modified: 03/17/2024
 */
public class SigningClientTCP {
    private static final Gson gson = new Gson();
    private static final int PORT = 7778;
    private static final String HOST = "localhost";
    private static BigInteger e, d, n;
    private static BigInteger clientID;

    public static void main(String[] args) {
        try {
            generateRSAKeys();
            Socket socket = new Socket(HOST, PORT);
            PrintWriter out = new PrintWriter(socket.getOutputStream(),
true);
            BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
            Scanner scanner = new Scanner(System.in);

            String userInput;
            while (true) {
                System.out.println("\n0. View basic blockchain status.");
                System.out.println("1. Add a transaction to the
blockchain.");
                System.out.println("2. Verify the blockchain.");
                System.out.println("3. View the blockchain.");
                System.out.println("4. Corrupt the chain.");
                System.out.println("5. Hide the corruption by repairing
the chain.");
                System.out.println("6. Exit");

                System.out.print("Enter your choice: ");
                userInput = scanner.nextLine();

                if ("6".equals(userInput)) {
                    System.out.println("Exiting...");
                    break;
                }

                // Create request message based on user input
                clientID = getClientID();
                RequestMessage request = createRequestMessage(userInput,
scanner);
                if (request != null) {
                    SignedRequest signedRequest = new SignedRequest(e, n,
clientID, request, sign(request));
                    out.println(gson.toJson(signedRequest));

                    String jsonResponse = in.readLine();
                    ResponseMessage response = gson.fromJson(jsonResponse,
ResponseMessage.class);
                    printResponse(response);

                } else {
                    System.out.println("Invalid choice.");
                }
            }
        } catch (Exception e) {
            System.out.println("Client error: " + e.getMessage());
            e.printStackTrace();
        }
    }

    private static void generateRSAKeys() throws NoSuchAlgorithmException,
InvalidKeySpecException {
        Random rnd = new Random();

        // Step 1: Generate two large random primes.
        // We use 400 bits here, but best practice for security is 2048
bits.
        // Change 400 to 2048, recompile, and run the program again and
you will
        // notice it takes much longer to do the math with that many bits.
        BigInteger p = new BigInteger(400, 100, rnd);
        BigInteger q = new BigInteger(400, 100, rnd);

        // Step 2: Compute n by the equation n = p * q.
        n = p.multiply(q);

        // Step 3: Compute phi(n) = (p-1) * (q-1)
        BigInteger phi =
(p.subtract(BigInteger.ONE)).multiply(q.subtract(BigInteger.ONE));

        // Step 4: Select a small odd integer e that is relatively prime
to phi(n).
        // By convention the prime 65537 is used as the public exponent.
        e = new BigInteger("65537");

        // Step 5: Compute d as the multiplicative inverse of e modulo
phi(n).
        d = e.modInverse(phi);
        System.out.println("Public Key: (e=" + e + ", n=" + n + ")");
    }

    // This method takes a string and signs it using an RSA private key.
    private static BigInteger getClientID() throws
NoSuchAlgorithmException {
        MessageDigest sha = MessageDigest.getInstance("SHA-256");
        sha.update((e.toString() + n.toString()).getBytes());
        byte[] digest = sha.digest();
        byte[] last20 = Arrays.copyOfRange(digest, digest.length - 20,
digest.length);
        return new BigInteger(1, last20); // Ensure positive
    }

    /**
     * Sign the request message
     * @param request
     * @return
     * @throws Exception
     */
    private static String sign(RequestMessage request) throws Exception {
        // compute the digest with SHA-256
        String message = request.concatenateValues();
        byte[] bytesOfMessage = message.getBytes("UTF-8");
        MessageDigest md = MessageDigest.getInstance("SHA-256");
        byte[] bigDigest = md.digest(bytesOfMessage);

        // we only want two bytes of the hash for ShortMessageSign
        // we add a 0 byte as the most significant byte to keep
        // the value to be signed non-negative.
        byte[] messageDigest = new byte[3];
        messageDigest[0] = 0;    // most significant set to 0
        messageDigest[1] = bigDigest[0]; // take a byte from SHA-256
        messageDigest[2] = bigDigest[1]; // take a byte from SHA-256

        // The message digest now has three bytes. Two from SHA-256
        // and one is 0.

        // From the digest, create a BigInteger
        BigInteger m = new BigInteger(messageDigest);

        // encrypt the digest with the private key
        BigInteger c = m.modPow(d, n);

        // return this as a big integer string
        return c.toString();
    }

    /**
     * Create a request message based on user input
     * @param userInput
     * @param scanner
     * @return
     */
    private static RequestMessage createRequestMessage(String userInput,
Scanner scanner) {
        switch (userInput) {
            case "0":
                return new
RequestMessage(RequestMessage.VIEW_BLOCKCHAIN_STATUS);
            case "1":
                System.out.print("Enter difficulty > ");
                int difficulty = Integer.parseInt(scanner.nextLine());
                System.out.print("Enter transaction: ");
                String transaction = scanner.nextLine();
                return new RequestMessage(RequestMessage.ADD_TRANSACTION,
transaction, difficulty);
            case "2":
                return new
RequestMessage(RequestMessage.VERIFY_BLOCKCHAIN);
            case "3":
                return new RequestMessage(RequestMessage.VIEW_BLOCKCHAIN);
            case "4":
                System.out.print("Enter block ID of block to corrupt: ");
                int id = Integer.parseInt(scanner.nextLine());
                System.out.print("Enter new data for block " + id + ": ");
                String newData = scanner.nextLine();
                return new
RequestMessage(RequestMessage.CORRUPT_BLOCKCHAIN, id, newData);
            case "5":
                return new RequestMessage(RequestMessage.REPAIR_CHAIN);
            default:
                System.out.println("Invalid choice.");
                return null;
        }
    }

    private static void printResponse(ResponseMessage response) {
        System.out.println("Server response: " + response.getMessage());
        if (response.getData() != null) {
            response.getData().forEach((key, value) ->
System.out.println(key + ": " + value));
        }
    }
}
```

SignedRequest

```
package cmu.ds.project3;

import java.math.BigInteger;

/**
 * Author: Louis Chang (hungyic)
 * Last Modified: 03/17/2024
 */
public class SignedRequest {
    private BigInteger e;
    private BigInteger n;
    private BigInteger clientID;
    private RequestMessage request;
    private String signature;

    public SignedRequest(BigInteger e, BigInteger n, BigInteger clientID,
RequestMessage request, String signature) {
```

```java
        this.e = e;
        this.n = n;
        this.clientID = clientID;
        this.request = request;
        this.signature = signature;
    }

    // Getters
    public BigInteger getE() {
        return e;
    }

    public BigInteger getN() {
        return n;
    }

    public BigInteger getClientID() {
        return clientID;
    }

    public RequestMessage getRequest() {
        return request;
    }

    public String getSignature() {
        return signature;
    }
}
```

RequestMessage

```java
package cmu.ds.project3;

public class RequestMessage {
    public static final String VIEW_BLOCKCHAIN_STATUS =
"viewBlockchainStatus";
    public static final String ADD_TRANSACTION = "addTransaction";
    public static final String VERIFY_BLOCKCHAIN = "verifyBlockchain";
    public static final String VIEW_BLOCKCHAIN = "viewBlockchain";
    public static final String CORRUPT_BLOCKCHAIN = "corruptBlockchain";
    public static final String REPAIR_CHAIN = "repairChain";

    private String action;
    private String data;
    private int difficulty;
    private int id; // For corrupting a block
    private String newData; // For corrupting a block

    public RequestMessage(String action) {
        this.action = action;
    }

    public RequestMessage(String action, String data, int difficulty) {
        this.action = action;
        this.data = data;
        this.difficulty = difficulty;
    }

    public RequestMessage(String action, int id, String newData) {
        this.action = action;
        this.id = id;
        this.newData = newData;
    }

    /**
     * Concatenate the values of the fields
     * @return
     */
    public String concatenateValues() {
        StringBuilder sb = new StringBuilder();
        if (data != null) {
            sb.append(data);
        }
        if (difficulty != 0) { // suppose difficulty is 0 means not set or
not applicable
            sb.append(difficulty);
        }
        if (id != 0) { // suppose id is 0 means not set or not applicable
            sb.append(id);
        }
        if (newData != null) {
            sb.append(newData);
        }
        return sb.toString();
    }

    // Additional getters for the new fields
    public int getId() {
        return id;
    }

    public String getNewData() {
        return newData;
    }

    public String getAction() {
        return action;
    }

    public void setAction(String action) {
        this.action = action;
    }

    public String getData() {
        return data;
    }
    public void setData(String data) {
        this.data = data;
    }

    public int getDifficulty() {
        return difficulty;
    }

    public void setDifficulty(int difficulty) {
        this.difficulty = difficulty;
    }

    public void setId(int id) {
        this.id = id;
    }

    public void setNewData(String newData) {
        this.newData = newData;
    }
}
```

Execution Console

Server

```
/Users/louischang/Library/Java/JavaVirtualMachines/openjdk-
20.0.1/Contents/Home/bin/java -javaagent:/Applications/IntelliJ
IDEA.app/Contents/lib/idea_rt.jar=65464:/Applications/IntelliJ IDEA.app/Contents/bin -
Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8 -
classpath /Users/louischang/Library/CloudStorage/OneDrive-
andrew.cmu.edu/DS/Project3/Project3Task2/target/classes:/Users/louischang/.m2/reposi
tory/com/google/code/gson/gson/2.9.0/gson-2.9.0.jar cmu.ds.project3.VerifyingServerTCP

Blockchain server running on port 7778

Received:
{"e":65537,"n":31628221721510134529050659918083850580749740873725747082833342
9702056971286239913138888228728392367773050265984527519374976443368109218
4976189666531987755501290269984306006889953260419275133762765869779542467
1541450644504621336883557449461821,"clientID":100978176840623971660577204994
4593691522891574781,"request":{"action":"viewBlockchainStatus","difficulty":0,"id":0},"sig
nature":"17209934553773204983786267037752430072769217489045047098194761056
9637560300035526900331403014711960642701461520033036891715408845193139667
4836042739496831662039291794193324983812826057128256452329127383842714245
096654319088970384090760650374"}
```

```
Sent: {"status":"success","message":"Blockchain status printed.","data":{"Approximate
hashes per second on this machine":"0","Chain
hash":"00a9c58cc248dab03b9578553724f5c1bbf184b922ee3065dd82c60c8ed33566","To
tal difficulty for all blocks":"2","Current size of chain":"1","Difficulty of most recent
block":"2","Experimented with hashes":"2000000","Expected total hashes required for the
whole chain":"256.0","Nonce for most recent block":"206"}}

Received:
{"e":65537,"n":3162822172151013452905065991808385058074974087372574708283342
9702056971286239913138888228728392367773050265984527519374976443368109218
4976189666531987755501290269984306006889953260419275133762765869779542467
1541450644504621336883557449461821,"clientID":100978176840623971660577204994
4593691522891574781,"request":{"action":"addTransaction","data":"Alice pays Bob 100
DSCoin","difficulty":4,"id":0},"signature":"1114577076864499339889439188147169950654
3383829169855748002707625022604196394387036739428872452135091393660266587
2801361474763225030700328477608155099337483794838672727152884253560669593
7673098617030050648516375583997161587834318687112641313"}

Sent: {"status":"success","message":"Block added.","data":{"Total execution time to add this
block":"274","Transaction":"Alice pays Bob 100 DSCoin"}}

Received:
{"e":65537,"n":3162822172151013452905065991808385058074974087372574708283342
9702056971286239913138888228728392367773050265984527519374976443368109218
4976189666531987755501290269984306006889953260419275133762765869779542467
1541450644504621336883557449461821,"clientID":100978176840623971660577204994
4593691522891574781,"request":{"action":"addTransaction","data":"Bob pays Carol 20
DSCoin","difficulty":4,"id":0},"signature":"1688870854697233523223310445805242163532
1355174545429557424443889630928784998981542731315974044909548199096301180
6062317489385887921791049256359035092160886300868368756483061057907742651
7739348827895527485432089811799158541859440794240978078"}

Sent: {"status":"success","message":"Block added.","data":{"Total execution time to add this
block":"8","Transaction":"Bob pays Carol 20 DSCoin"}}

Received:
{"e":65537,"n":3162822172151013452905065991808385058074974087372574708283342
9702056971286239913138888228728392367773050265984527519374976443368109218
4976189666531987755501290269984306006889953260419275133762765869779542467
1541450644504621336883557449461821,"clientID":100978176840623971660577204994
```

```
4593691522891574781,"request":{"action":"addTransaction","data":"Carol pays Donna 10
DSCoin","difficulty":4,"id":0},"signature":"9620156288005753238911508043141570793476
5320210390702512188920228188528468904624930301003912491806689437349908469
4187657725066066483927899349222496562479843821175273395799797007862061068
450352860277438411123744195181522405737586911265184166"}

Sent: {"status":"success","message":"Block added.","data":{"Total execution time to add this
block":"77","Transaction":"Carol pays Donna 10 DSCoin"}}

Received:
{"e":65537,"n":3162822172151013452905065991808385058074974087372574708283342
9702056971286239913138888228728392367773050265984527519374976443368109218
4976189666531987755501290269984306006889953260419275133762765869779542467
1541450644504621336883557449461821,"clientID":100978176840623971660577204994
4593691522891574781,"request":{"action":"viewBlockchain","difficulty":0,"id":0},"signature
":"17209934553773204983786267037752430072769217489045047098194761056963756
0300035526900331403014711960642701461520033036891715408845193139667483604
2739496831662039291794193324983812826057128256452329127383842714245096654
319088970384090760650374"}

Sent: {"status":"success","message":"Blockchain
data.","data":{"blockchain":"{\"index\":0,\"timestamp\":\"2024-03-18
18:44:58.063\",\"data\":\"Genesis\",\"previousHash\":\"0\",\"nonce\":\"206\",\"difficulty\":2,
\"hash\":\"00a9c58cc248dab03b9578553724f5c1bbf184b922ee3065dd82c60c8ed33566\
"}\n{\"index\":1,\"timestamp\":\"2024-03-18 18:45:11.97\",\"data\":\"Alice pays Bob 100
DSCoin\",\"previousHash\":\"00a9c58cc248dab03b9578553724f5c1bbf184b922ee3065dd
82c60c8ed33566\",\"nonce\":\"70356\",\"difficulty\":4,\"hash\":\"00005659c591d2308274f
c673dedd2098bc652c40df88e27b32e1a5ccc902f6d\"}\n{\"index\":2,\"timestamp\":\"2024
-03-18 18:45:20.792\",\"data\":\"Bob pays Carol 20
DSCoin\",\"previousHash\":\"00005659c591d2308274fc673dedd2098bc652c40df88e27b3
2e1a5ccc902f6d\",\"nonce\":\"1735\",\"difficulty\":4,\"hash\":\"0000a0b1bb8105f77ea566
b3cacc77659f9deb083813b6076ef696d8a880d6e5\"}\n{\"index\":3,\"timestamp\":\"2024-
03-18 18:45:28.859\",\"data\":\"Carol pays Donna 10
DSCoin\",\"previousHash\":\"0000a0b1bb8105f77ea566b3cacc77659f9deb083813b6076e
f696d8a880d6e5\",\"nonce\":\"32612\",\"difficulty\":4,\"hash\":\"0000a0d116763f1e84084
16c312c9535d1d600fc2c91f6a8c7e1e513c47c508e\"}\n"}}

Received:
{"e":65537,"n":3162822172151013452905065991808385058074974087372574708283342
9702056971286239913138888228728392367773050265984527519374976443368109218
```

```
4976189666531987755501290269984306006889953260419275133762765869779542467
1541450644504621336883557449461821,"clientID":1009781768406239716605772049994
4593691522891574781,"request":{"action":"verifyBlockchain","difficulty":0,"id":0},"signatur
e":"1720993455377320498378626703775243007276921748904504709819476105696375
6030003552690033140301471196064270146152003303689171540884519313966748360
4273949683166203929179419332498381282605712825645232912738384271424509665
431908897038409076065O374"}

Sent: {"status":"success","message":"Chain verification","data":{"Verification":"TRUE"}}

Received:
{"e":65537,"n":3162822172151013452905065991808385058074974087372574708283342
9702056971286239913138888228728392367773050265984527519374976443368109218
4976189666531987755501290269984306006889953260419275133762765869779542467
1541450644504621336883557449461821,"clientID":1009781768406239716605772049994
4593691522891574781,"request":{"action":"corruptBlockchain","difficulty":0,"id":2,"newDa
ta":"Bob pays Tony 30
DSCoin"},"signature":"2753572578283173258491993091309068188251750969938786573
0277294801564873752152637505216214141570175307688101996115520109956771512
4712176845065913172488366653083394347611898871713540161703446547698004666
4679467814599791632488072898678981277168"}

Corrupting block 2 with new data: Bob pays Tony 30 DSCoin

Sent: {"status":"success","message":"Currupt the
Blockchain","data":{"CorruptedBlockID":"2","CorruptedBlockData":"Bob pays Tony 30
DSCoin"}}

Received:
{"e":65537,"n":3162822172151013452905065991808385058074974087372574708283342
9702056971286239913138888228728392367773050265984527519374976443368109218
4976189666531987755501290269984306006889953260419275133762765869779542467
1541450644504621336883557449461821,"clientID":1009781768406239716605772049994
4593691522891574781,"request":{"action":"viewBlockchain","difficulty":0,"id":0},"signature
":"17209934553773204983786267037752430072769217489045047098194761056963756
0300035526900331403014711960642701461520033036891715408845193139667483604
2739496831662039291794193324983812826057128256452329127383842714245096654
31908897038409076065O374"}

Sent: {"status":"success","message":"Blockchain
data.","data":{"blockchain":"{\"index\":0,\"timestamp\":\"2024-03-18
18:44:58.063\",\"data\":\"Genesis\",\"previousHash\":\"0\",\"nonce\":\"206\",\"difficulty\":2,
\"hash\":\"00a9c58cc248dab03b9578553724f5c1bbf184b922ee3065dd82c60c8ed33566\
"}\n{\"index\":1,\"timestamp\":\"2024-03-18 18:45:11.97\",\"data\":\"Alice pays Bob 100
DSCoin\",\"previousHash\":\"00a9c58cc248dab03b9578553724f5c1bbf184b922ee3065dd
82c60c8ed33566\",\"nonce\":\"70356\",\"difficulty\":4,\"hash\":\"00005659c591d2308274f
c673dedd2098bc652c40df88e27b32e1a5ccc902f6d\"}\n{\"index\":2,\"timestamp\":\"2024
-03-18 18:45:20.792\",\"data\":\"Bob pays Tony 30
DSCoin\",\"previousHash\":\"00005659c591d2308274fc673dedd2098bc652c40df88e27b3
2e1a5ccc902f6d\",\"nonce\":\"1735\",\"difficulty\":4,\"hash\":\"0000a0b1bb8105f77ea566
b3cacc77659f9deb083813b6076ef696d8a880d6e5\"}\n{\"index\":3,\"timestamp\":\"2024-
03-18 18:45:28.859\",\"data\":\"Carol pays Donna 10
DSCoin\",\"previousHash\":\"0000a0b1bb8105f77ea566b3cacc77659f9deb083813b6076e
f696d8a880d6e5\",\"nonce\":\"32612\",\"difficulty\":4,\"hash\":\"0000a0d116763f1e84084
16c312c9535d1d600fc2c91f6a8c7e1e513c47c508e\"}\n"}}

Received:
{"e":65537,"n":3162822172151013452905065991808385058074974087372574708283342
9702056971286239913138888228728392367773050265984527519374976443368109218
4976189666531987755501290269984306006889953260419275133762765869779542467
1541450644504621336883557449461821,"clientID":100978176840623971660577204994
4593691522891574781,"request":{"action":"verifyBlockchain","difficulty":0,"id":0},"signatur
e":"1720993455377320498378626703775243007276921748904504709819476105696375
6030003552690033140301471196064270146152003303689171540884519313966748360
4273949683166203929179419332498381282605712825645232912738384271424509665
4319088970384090760650374"}

Sent: {"status":"success","message":"Chain
verification","data":{"Verification":"FALSE\nImproper hash on node 2 Does not begin with
62b65da6c73066531f5ab5d0d33db1f1f2549fb4ca13b29d4546a18e1bd1ec3b"}}

Received:
{"e":65537,"n":3162822172151013452905065991808385058074974087372574708283342
9702056971286239913138888228728392367773050265984527519374976443368109218
4976189666531987755501290269984306006889953260419275133762765869779542467
1541450644504621336883557449461821,"clientID":100978176840623971660577204994
4593691522891574781,"request":{"action":"repairChain","difficulty":0,"id":0},"signature":"1
7209934553773204983786267037752430072769217489045047098194761056963756030
0035526900331403014711960642701461520033036891715408845193139667483604273
```

9496831662039291794193324983812826057128256452329127383842714245096654319
088970384090760650374"}

Sent: {"status":"success","message":"Repairing the entire chain","data":{"Total execution
time to add this block":"0"}}

Received:
{"e":65537,"n":3162822172151013452905065991808385058074974087372574708283342
9702056971286239913138888228728392367773050265984527519374976443368109218
4976189666531987755501290269984306006889953260419275133762765869779542467
1541450644504621336883557449461821,"clientID":10097817684062397166057720499
4459369152289157478 1,"request":{"action":"verifyBlockchain","difficulty":0,"id":0},"signatur
e":"17209934553773204983786267037752430072769217489045047098194761056963 75
6030003552690033140301471196064270146152003303689171540884519313966748360
4273949683166203929179419332498381282605712825645232912738384271424509665
4319088970384090760650374"}

Sent: {"status":"success","message":"Chain verification","data":{"Verification":"TRUE"}}

Received:
{"e":65537,"n":3162822172151013452905065991808385058074974087372574708283342
9702056971286239913138888228728392367773050265984527519374976443368109218
4976189666531987755501290269984306006889953260419275133762765869779542467
1541450644504621336883557449461821,"clientID":10097817684062397166057720499
4459369152289157478 1,"request":{"action":"viewBlockchain","difficulty":0,"id":0},"signature
":"172099345537732049837862670377524300727692174890450470981947610569637 56
0300035526900331403014711960642701461520033036891715408845193139667483604
2739496831662039291794193324983812826057128256452329127383842714245096654
3190889703840907606503 74"}

Sent: {"status":"success","message":"Blockchain
data.","data":{"blockchain":"{\"index\":0,\"timestamp\":\"2024-03-18
18:44:58.063\",\"data\":\"Genesis\",\"previousHash\":\"0\",\"nonce\":\"206\",\"difficulty\":2,
\"hash\":\"00a9c58cc248dab03b9578553724f5c1bbf184b922ee3065dd82c60c8ed33566\
"}\n{\"index\":1,\"timestamp\":\"2024-03-18 18:45:11.97\",\"data\":\"Alice pays Bob 100
DSCoin\",\"previousHash\":\"00a9c58cc248dab03b9578553724f5c1bbf184b922ee3065dd
82c60c8ed33566\",\"nonce\":\"70356\",\"difficulty\":4,\"hash\":\"00005659c591d2308274f
c673dedd2098bc652c40df88e27b32e1a5ccc902f6d\"}\n{\"index\":2,\"timestamp\":\"2024
-03-18 18:45:20.792\",\"data\":\"Bob pays Tony 30
DSCoin\",\"previousHash\":\"00005659c591d2308274fc673dedd2098bc652c40df88e27b3

```
2e1a5ccc902f6d\",\"nonce\":\"1735\",\"difficulty\":4,\"hash\":\"0000a0b1bb8105f77ea566
b3cacc77659f9deb083813b6076ef696d8a880d6e5\"}\n{\"index\":3,\"timestamp\":\"2024-
03-18 18:45:28.859\",\"data\":\"Carol pays Donna 10
DSCoin\",\"previousHash\":\"62b65da6c73066531f5ab5d0d33db1f1f2549fb4ca13b29d45
46a18e1bd1ec3b\",\"nonce\":\"32612\",\"difficulty\":4,\"hash\":\"0000a0d116763f1e84084
16c312c9535d1d600fc2c91f6a8c7e1e513c47c508e\"}\n"}}

Client disconnected.


Process finished with exit code 0
```

Client

```
/Users/louischang/Library/Java/JavaVirtualMachines/openjdk-
20.0.1/Contents/Home/bin/java -javaagent:/Applications/IntelliJ
IDEA.app/Contents/lib/idea_rt.jar=65470:/Applications/IntelliJ IDEA.app/Contents/bin -
Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8 -
classpath /Users/louischang/Library/CloudStorage/OneDrive-
andrew.cmu.edu/DS/Project3/Project3Task2/target/classes:/Users/louischang/.m2/reposi
tory/com/google/code/gson/gson/2.9.0/gson-2.9.0.jar cmu.ds.project3.SigningClientTCP

Public Key: (e=65537,
n=3162822172151013452905065991808385058074974087372574708283342970205697
1286239913138888228728392367773050265984527519374976443368109218497618966
6531987755501290269984306006889953260419275133762765869779542467154145064
4504621336883557449461821)


0. View basic blockchain status.

1. Add a transaction to the blockchain.

2. Verify the blockchain.

3. View the blockchain.

4. Corrupt the chain.

5. Hide the corruption by repairing the chain.

6. Exit
```

```
Enter your choice: 0

Server response: Blockchain status printed.

Approximate hashes per second on this machine: 0

Chain hash:
00a9c58cc248dab03b9578553724f5c1bbf184b922ee3065dd82c60c8ed33566

Total difficulty for all blocks: 2

Current size of chain: 1

Difficulty of most recent block: 2

Experimented with hashes: 2000000

Expected total hashes required for the whole chain: 256.0

Nonce for most recent block: 206


0. View basic blockchain status.

1. Add a transaction to the blockchain.

2. Verify the blockchain.

3. View the blockchain.

4. Corrupt the chain.

5. Hide the corruption by repairing the chain.

6. Exit

Enter your choice: 1

Enter difficulty > 4

Enter transaction: Alice pays Bob 100 DSCoin

Server response: Block added.

Total execution time to add this block: 274

Transaction: Alice pays Bob 100 DSCoin
```

```
0. View basic blockchain status.
1. Add a transaction to the blockchain.
2. Verify the blockchain.
3. View the blockchain.
4. Corrupt the chain.
5. Hide the corruption by repairing the chain.
6. Exit
Enter your choice: 1
Enter difficulty > 4
Enter transaction: Bob pays Carol 20 DSCoin
Server response: Block added.
Total execution time to add this block: 8
Transaction: Bob pays Carol 20 DSCoin

0. View basic blockchain status.
1. Add a transaction to the blockchain.
2. Verify the blockchain.
3. View the blockchain.
4. Corrupt the chain.
5. Hide the corruption by repairing the chain.
6. Exit
Enter your choice: 1
Enter difficulty > 4
Enter transaction: Carol pays Donna 10 DSCoin
Server response: Block added.
Total execution time to add this block: 77
```

```
Transaction: Carol pays Donna 10 DSCoin


0. View basic blockchain status.

1. Add a transaction to the blockchain.

2. Verify the blockchain.

3. View the blockchain.

4. Corrupt the chain.

5. Hide the corruption by repairing the chain.

6. Exit

Enter your choice: 3

Server response: Blockchain data.

blockchain: {"index":0,"timestamp":"2024-03-18
18:44:58.063","data":"Genesis","previousHash":"0","nonce":"206","difficulty":2,"hash":"00a9
c58cc248dab03b9578553724f5c1bbf184b922ee3065dd82c60c8ed33566"}

{"index":1,"timestamp":"2024-03-18 18:45:11.97","data":"Alice pays Bob 100
DSCoin","previousHash":"00a9c58cc248dab03b9578553724f5c1bbf184b922ee3065dd82
c60c8ed33566","nonce":"70356","difficulty":4,"hash":"00005659c591d2308274fc673dedd2
098bc652c40df88e27b32e1a5ccc902f6d"}

{"index":2,"timestamp":"2024-03-18 18:45:20.792","data":"Bob pays Carol 20
DSCoin","previousHash":"00005659c591d2308274fc673dedd2098bc652c40df88e27b32e
1a5ccc902f6d","nonce":"1735","difficulty":4,"hash":"0000a0b1bb8105f77ea566b3cacc776
59f9deb083813b6076ef696d8a880d6e5"}

{"index":3,"timestamp":"2024-03-18 18:45:28.859","data":"Carol pays Donna 10
DSCoin","previousHash":"0000a0b1bb8105f77ea566b3cacc77659f9deb083813b6076ef69
6d8a880d6e5","nonce":"32612","difficulty":4,"hash":"0000a0d116763f1e8408416c312c953
5d1d600fc2c91f6a8c7e1e513c47c508e"}



0. View basic blockchain status.
```

```
1. Add a transaction to the blockchain.

2. Verify the blockchain.

3. View the blockchain.

4. Corrupt the chain.

5. Hide the corruption by repairing the chain.

6. Exit

Enter your choice: 2

Server response: Chain verification

Verification: TRUE


0. View basic blockchain status.

1. Add a transaction to the blockchain.

2. Verify the blockchain.

3. View the blockchain.

4. Corrupt the chain.

5. Hide the corruption by repairing the chain.

6. Exit

Enter your choice: 4

Enter block ID of block to corrupt: 2

Enter new data for block 2: Bob pays Tony 30 DSCoin

Server response: Currupt the Blockchain

CorruptedBlockID: 2

CorruptedBlockData: Bob pays Tony 30 DSCoin


0. View basic blockchain status.

1. Add a transaction to the blockchain.
```

```
2. Verify the blockchain.

3. View the blockchain.

4. Corrupt the chain.

5. Hide the corruption by repairing the chain.

6. Exit

Enter your choice: 3

Server response: Blockchain data.

blockchain: {"index":0,"timestamp":"2024-03-18
18:44:58.063","data":"Genesis","previousHash":"0","nonce":"206","difficulty":2,"hash":"00a9
c58cc248dab03b9578553724f5c1bbf184b922ee3065dd82c60c8ed33566"}

{"index":1,"timestamp":"2024-03-18 18:45:11.97","data":"Alice pays Bob 100
DSCoin","previousHash":"00a9c58cc248dab03b9578553724f5c1bbf184b922ee3065dd82
c60c8ed33566","nonce":"70356","difficulty":4,"hash":"00005659c591d2308274fc673dedd2
098bc652c40df88e27b32e1a5ccc902f6d"}

{"index":2,"timestamp":"2024-03-18 18:45:20.792","data":"Bob pays Tony 30
DSCoin","previousHash":"00005659c591d2308274fc673dedd2098bc652c40df88e27b32e
1a5ccc902f6d","nonce":"1735","difficulty":4,"hash":"0000a0b1bb8105f77ea566b3cacc776
59f9deb083813b6076ef696d8a880d6e5"}

{"index":3,"timestamp":"2024-03-18 18:45:28.859","data":"Carol pays Donna 10
DSCoin","previousHash":"0000a0b1bb8105f77ea566b3cacc77659f9deb083813b6076ef69
6d8a880d6e5","nonce":"32612","difficulty":4,"hash":"0000a0d116763f1e8408416c312c953
5d1d600fc2c91f6a8c7e1e513c47c508e"}


0. View basic blockchain status.

1. Add a transaction to the blockchain.

2. Verify the blockchain.

3. View the blockchain.

4. Corrupt the chain.

5. Hide the corruption by repairing the chain.
```

```
6. Exit
Enter your choice: 2
Server response: Chain verification
Verification: FALSE
Improper hash on node 2 Does not begin with
62b65da6c73066531f5ab5d0d33db1f1f2549fb4ca13b29d4546a18e1bd1ec3b

0. View basic blockchain status.
1. Add a transaction to the blockchain.
2. Verify the blockchain.
3. View the blockchain.
4. Corrupt the chain.
5. Hide the corruption by repairing the chain.
6. Exit
Enter your choice: 5
Server response: Repairing the entire chain
Total execution time to add this block: 0

0. View basic blockchain status.
1. Add a transaction to the blockchain.
2. Verify the blockchain.
3. View the blockchain.
4. Corrupt the chain.
5. Hide the corruption by repairing the chain.
6. Exit
Enter your choice: 2
```

```
Server response: Chain verification

Verification: TRUE


0. View basic blockchain status.

1. Add a transaction to the blockchain.

2. Verify the blockchain.

3. View the blockchain.

4. Corrupt the chain.

5. Hide the corruption by repairing the chain.

6. Exit

Enter your choice: 3

Server response: Blockchain data.

blockchain: {"index":0,"timestamp":"2024-03-18
18:44:58.063","data":"Genesis","previousHash":"0","nonce":"206","difficulty":2,"hash":"00a9
c58cc248dab03b9578553724f5c1bbf184b922ee3065dd82c60c8ed33566"}

{"index":1,"timestamp":"2024-03-18 18:45:11.97","data":"Alice pays Bob 100
DSCoin","previousHash":"00a9c58cc248dab03b9578553724f5c1bbf184b922ee3065dd82
c60c8ed33566","nonce":"70356","difficulty":4,"hash":"00005659c591d2308274fc673dedd2
098bc652c40df88e27b32e1a5ccc902f6d"}

{"index":2,"timestamp":"2024-03-18 18:45:20.792","data":"Bob pays Tony 30
DSCoin","previousHash":"00005659c591d2308274fc673dedd2098bc652c40df88e27b32e
1a5ccc902f6d","nonce":"1735","difficulty":4,"hash":"0000a0b1bb8105f77ea566b3cacc776
59f9deb083813b6076ef696d8a880d6e5"}

{"index":3,"timestamp":"2024-03-18 18:45:28.859","data":"Carol pays Donna 10
DSCoin","previousHash":"62b65da6c73066531f5ab5d0d33db1f1f2549fb4ca13b29d4546a
18e1bd1ec3b","nonce":"32612","difficulty":4,"hash":"0000a0d116763f1e8408416c312c953
5d1d600fc2c91f6a8c7e1e513c47c508e"}


0. View basic blockchain status.
```

```
1. Add a transaction to the blockchain.

2. Verify the blockchain.

3. View the blockchain.

4. Corrupt the chain.

5. Hide the corruption by repairing the chain.

6. Exit

Enter your choice: 6

Exiting...


Process finished with exit code 0
```