Project3Task0: Code for Block.java

```java
import com.google.gson.Gson;
import java.math.BigInteger;
import java.nio.charset.StandardCharsets;
import java.sql.Timestamp;
import java.security.NoSuchAlgorithmException;
import java.security.MessageDigest;

// Name: Leo Lin
// AndrewID: hungfanl

public class Block extends java.lang.Object {
    private static final char[] HEX_ARRAY = "0123456789ABCDEF".toCharArray();
    int index;
    Timestamp timestamp;
    String data;
    String previousHash;
    BigInteger nonce;
    int difficulty;
    Block (int index, Timestamp timestamp, String data, int difficulty) {
        this.index = index;
        this.timestamp = timestamp;
        this.data = data;
        this.difficulty = difficulty;
        nonce = new BigInteger("0");
    }
    // Calculate the hash for the block
    public String calculateHash() {
        String information = index + timestamp.toString() + data + previousHash + nonce +
difficulty;
        String hash_value = null;
        try {
            MessageDigest md;
            // Compute SHA-265 code for the input
            md = MessageDigest.getInstance("SHA-256");
            md.update(information.getBytes(StandardCharsets.UTF_8));
            hash_value = bytesToHex(md.digest());
        }
        catch(NoSuchAlgorithmException e) {
            System.out.println("No Hash available" + e);
        }
        return String.valueOf(hash_value);
    }

    public String getData() {
        return data;
    }

    public int getDifficulty() {
        return difficulty;
    }

    public int getIndex() {
        return index;
    }

    public BigInteger getNonce() {
        return nonce;
    }

    public String getPreviousHash() {
        return previousHash;
    }

    public Timestamp getTimestamp() {
        return timestamp;
    }
    // Make sure that the has for the block meets the requirement, otherwise add the nonce by one
and recalculate.
    public String proofOfWork() {
        String hash_value = null;
        boolean leading_zero = false;
        while(!leading_zero) {
            leading_zero = true;
            hash_value = calculateHash();
            for(int i = 0; i < difficulty; i++){
                if(hash_value.charAt(i) != '0') {
                    leading_zero = false;
                    nonce = nonce.add(BigInteger.valueOf(1));
                    break;
                }
            }
        }
        return hash_value;
    }

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

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

    public void setIndex(int index) {
        this.index = index;
    }

    public void setPreviousHash(String previousHash) {
        this.previousHash = previousHash;
    }

    public void setTimestamp(Timestamp timestamp) {
        this.timestamp = timestamp;
    }

    // Transfer the block in the form of json.
    public String toString() {
        Block b = new Block (index, timestamp, data, difficulty);
        b.nonce = nonce;
        b.setPreviousHash(previousHash);
        Gson gson = new Gson();
        String messageToSend = gson.toJson(b);
        return messageToSend;
    }
    // Reference from Lab1 submission
    public static String bytesToHex(byte[] bytes) {
        char[] hexChars = new char[bytes.length * 2];
        for (int j = 0; j < bytes.length; j++) {
            int v = bytes[j] & 0xFF;
            hexChars[j * 2] = HEX_ARRAY[v >>> 4];
            hexChars[j * 2 + 1] = HEX_ARRAY[v & 0x0F];
        }
        return new String(hexChars);
    }
}
```

Project3Task0: Code for BlockChain.java

```java
import com.google.gson.Gson;

import java.nio.charset.StandardCharsets;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.sql.Timestamp;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

// Name: Leo Lin
// AndrewID: hungfanl

public class BlockChain extends java.lang.Object{
    public List<Block> blockchain;
    public String chain_hash;
    public int hashes_per_second;
    public static final int HASH_ROUND = 2000000;
    private static final char[] HEX_ARRAY = "0123456789ABCDEF".toCharArray();

    private static final String LEADING_ZERO =
"000000000000000000000000000000000000000000000000000";

    public BlockChain(){
        blockchain = new ArrayList<>();
        chain_hash = "";
        hashes_per_second = 0;
    }

    public static void main(String[] args) {
        // This list contains all the selection that the user is allowed to choose.
        String[] message = {"View basic blockchain status.", "Add a transaction to the
blockchain.",
        "Verify the blockchain.", "View the blockchain.", "Corrupt the chain.",
        "Hide the corruption by repairing the chain.", "Exit."};
        BlockChain bc = new BlockChain();
        // Create a new genesis block with the difficulty of 2
        Block b = new Block(bc.getChainSize(), bc.getTime(), "Genesis", 2);
        bc.addBlock(b);
        bc.computeHashesPerSecond();
        Scanner readInput = new Scanner(System.in);
        Timestamp start;
        Timestamp end;
        int time;
        // Set a boolean finish that only if the user insert 6 will be true.
        boolean finish = false;
        while(!finish) {
            for(int i = 0; i < message.length; i++) {
                System.out.println(i + ". " + message[i]);
            }
            int option = Integer.parseInt(readInput.nextLine());
            switch (option) {
                    // Check the information of the chain
                case 0:
                    System.out.println("Current size of chain: " + bc.getChainSize());
                    System.out.println("Difficulty of most recent block: " +
bc.getBlock(bc.getChainSize()-1).getDifficulty());
                    System.out.println("Total difficulty for all blocks: " +
bc.getTotalDifficulty());
                    System.out.println("Approximate hashes per second on this machine: " +
bc.getHashesPerSecond());
                    System.out.println("Expected total hashes required for the whole chain: " +
bc.getTotalExpectedHashes());
                    System.out.println("Nonce for most recent block: " +
bc.getBlock(bc.getChainSize()-1).getNonce());
                    System.out.println("Chain hash: " + bc.getChainHash());
                    break;
                 // Allows the user to add a block to the chain.
                case 1:
                    System.out.println("Enter difficulty > 0");
                    int difficulty = Integer.parseInt(readInput.nextLine());
                    System.out.println("Enter transaction");
                    String data = readInput.nextLine();
                    start = bc.getTime();
                    // Generate a new block with the information get from the user.
                    bc.addBlock(new Block(bc.getChainSize(), bc.getTime(), data, difficulty));
                    // record the time for adding the block
                    end = bc.getTime();
                    time = (int) (end.getTime() - start.getTime());
                    System.out.println("Total execution time to add this block was " + time + "
milliseconds");
                    break;
                // Check if the chain is valid.
                case 2:
                    start = bc.getTime();
                    String validation = bc.isChainValid();
                    end = bc.getTime();
                    time = (int) (end.getTime() - start.getTime());
                    System.out.print("Chain verification: ");
                    if (!validation.equals("True")) System.out.println("False");
                    System.out.println(validation);
                    System.out.println("Total execution time to verify the chain was " + time + "
milliseconds");
                    break;
                // View the block (json style)
                case 3:
                    System.out.println("View the Blockchain");
                    System.out.println(bc.toString());
                    break;
                // Corrupt the chain by changing the information in the block.
                case 4:
                    System.out.println("corrupt the Blockchain");
                    System.out.println("Enter block ID of block to corrupt");
                    int index = Integer.parseInt(readInput.nextLine());
                    System.out.println("Enter new data for block " + index);
                    String corrupt_message = readInput.nextLine();
                    bc.getBlock(index).setData(corrupt_message);
                    System.out.println("Block " + index + " now holds " + corrupt_message);
                    break;
                // Fix the block by recomputing the nonce to meet the difficulty requirement.
                case 5:
                    start = bc.getTime();
                    if(!bc.isChainValid().equals("True")) bc.repairChain();
                    end = bc.getTime();
                    time = (int) (end.getTime() - start.getTime());
                    System.out.println("Total execution time required to repair the chain was " +
time +" milliseconds");
                    break;
                case 6:
                    finish = true;
                    break;
                default:
                    break;
            }

        }
        readInput.close();
    }

    public String getChainHash() {
        return chain_hash;
    }

    public Timestamp getTime(){
        return new Timestamp(System.currentTimeMillis());
    }

    public Block getLatestBlock(){
        return blockchain.get(this.getChainSize()-1);
    }

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

    // Get the information of a particular block in the chain.
    public Block getBlock(int i) {
        if(i >= getChainSize()) {
            System.out.println("Insert number exceed block size");
            return null;
        }
        return blockchain.get(i);
    }
    // Compute the expected time of calculating hash by doing the calculation for 20000 times and
get the average time.
    public void computeHashesPerSecond() {
        String s = "00000000";
        Timestamp start = getTime();
        for(int i = 0; i < HASH_ROUND; i++) {
            calculateHash(s);
        }
        Timestamp end = getTime();
        hashes_per_second = (int) ( (double)HASH_ROUND / (end.getTime() - start.getTime()) *
1000);
    }

    public int getHashesPerSecond() {
        return  hashes_per_second;
    }
    // Add a block in the chain and revise the chain information.
    public void addBlock(Block block){
        if(getChainSize() == 0) {
            block.setPreviousHash("");
        }
        else block.setPreviousHash(chain_hash);
        blockchain.add(block);
        chain_hash = block.proofOfWork();
    }
    // Transfer the chain in the form of json.
    public String toString() {
        BlockChain bc = new BlockChain();
        for(int i = 0; i < getChainSize(); i++) {
            bc.blockchain.add(getBlock(i));
        }
        bc.hashes_per_second = getHashesPerSecond();
        bc.chain_hash = getChainHash();
        Gson gson = new Gson();
        String messageToSend = gson.toJson(bc);
        return messageToSend;
    }
    // Adding the difficulty of all blocks and come up with a total difficulty of the chain.
    public int getTotalDifficulty() {
        int totalDifficulty = 0;
        for(Block b: blockchain) {
            totalDifficulty += b.getDifficulty();
        }
        return totalDifficulty;
    }

    // Get the expected number of hashes the chain requires to compute.
    public double getTotalExpectedHashes() {
        double totalExpectedHashes = 0;
        for(Block b: blockchain) {
            totalExpectedHashes += Math.pow(16, b.getDifficulty());
        }
        return  totalExpectedHashes;
    }
    // Return true if the function founds no error, the type of error if the function finds any.
    public String isChainValid() {
        for(int i = 0; i < getChainSize(); i++) {
            Block b = getBlock(i);
            String s = b.calculateHash();
            for(int j = 0; j < b.getDifficulty(); j++) {
                //Improper hash on node 1 Does not begin with 00
                if(s.charAt(j) != '0')
                    return "Improper hash on node " + i + " does not begin with " +
                            LEADING_ZERO.substring(0, b.getDifficulty());
            }
            if(i != 0 && !getBlock(i-1).calculateHash().equals(b.getPreviousHash()))
                return "Block " + i + " does not have a matching hash.";
        }
        if(!getBlock(getChainSize()-1).calculateHash().equals(chain_hash))
            return "The chain hash is different from the hash of the last block.";
        return "True";
    }
    // Fix the chain by changing the nonce and get the right hash number.
    public void repairChain() {
        for(int i = 0; i < getChainSize(); i++) {
            Block b = getBlock(i);
            if(i != getChainSize()-1) getBlock(i+1).previousHash = b.proofOfWork();
            else chain_hash = b.proofOfWork();
        }
    }
    // Calculate the hash number.
    public String calculateHash(String s) {
        String hash_value = null;
        try {
            MessageDigest md;
            // Compute SHA-265 code for the input
            md = MessageDigest.getInstance("SHA-256");
            md.update(s.getBytes(StandardCharsets.UTF_8));
            hash_value = bytesToHex(md.digest());
        }
        catch(NoSuchAlgorithmException e) {
            System.out.println("No Hash available" + e);
        }
        return String.valueOf(hash_value);
    }
    // Transfer the byte representation of a string to a hex value.
    public static String bytesToHex(byte[] bytes) {
        char[] hexChars = new char[bytes.length * 2];
        for (int j = 0; j < bytes.length; j++) {
            int v = bytes[j] & 0xFF;
            hexChars[j * 2] = HEX_ARRAY[v >>> 4];
            hexChars[j * 2 + 1] = HEX_ARRAY[v & 0x0F];
        }
        return new String(hexChars);
    }
}
```

Project3Task0: Output

```
/Library/Java/JavaVirtualMachines/jdk-17.0.3.1.jdk/Contents/Home/bin/java -
javaagent:/Applications/IntelliJ IDEA.app/Contents/lib/idea_rt.jar=59730:/Applications/IntelliJ
IDEA.app/Contents/bin -Dfile.encoding=UTF-8 -classpath
/Users/linhungfan/Desktop/CMU/Semester 2/Distributed
System/Project3/Project3Task0/target/classes:/Users/linhungfan/.m2/repository/com/google/
code/gson/gson/2.9.0/gson-2.9.0.jar 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.
0
Current size of chain: 1
Difficulty of most recent block: 2
Total difficulty for all blocks: 2
Approximate hashes per second on this machine: 1718213
Expected total hashes required for the whole chain: 256.0
```

```
Nonce for most recent block: 958
Chain hash: 000344DA0511274D76DF011EFA9B278098C0EF8A2266768633C93BB85D7526D8
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.
1
Enter difficulty > 0
2
Enter transaction
Alice pays Bob 100 DS Coin
Total execution time to add this block was 24 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.
1
Enter difficulty > 0
2
Enter transaction
Bob pays Carol 50 DS Coin
Total execution time to add this block was 6 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.
1
Enter difficulty > 0
2
Enter transaction
Carol pays Donna 10 DS Coin
Total execution time to add this block was 3 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.
2
Chain verification: True
Total execution time to verify the chain was 2 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.
3
View the Blockchain
{"blockchain":[{"index":0,"timestamp":"Oct 24, 2022, 12:03:08
AM","data":"Genesis","previousHash":"","nonce":958,"difficulty":2},{"index":1,"timestamp":"O
ct 24, 2022, 12:03:39 AM","data":"Alice pays Bob 100 DS
Coin","previousHash":"000344DA0511274D76DF011EFA9B278098C0EF8A2266768633C93BB85
D7526D8","nonce":778,"difficulty":2},{"index":2,"timestamp":"Oct 24, 2022, 12:04:01
AM","data":"Bob pays Carol 50 DS
Coin","previousHash":"00257E9EDF9F671274F0D3022698DC34AE15725C8C8ABD63D33AA0DA
CF7CC4E2","nonce":9,"difficulty":2},{"index":3,"timestamp":"Oct 24, 2022, 12:04:16
AM","data":"Carol pays Donna 10 DS
Coin","previousHash":"004EEE64A034303E133DD02DECF255703E7F8DE20E34EA0A438D274C5
912E1FF","nonce":521,"difficulty":2}],"chain_hash":"0036B62F581246605FBFFD65ADBFEE1262
51B18941453D89D3DB3B88F3BCF0EE","hashes_per_second":1718213}
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.
4
corrupt the Blockchain
Enter block ID of block to corrupt
1
Enter new data for block 1
Alice pays Bob 76 DS Coin
Block 1 now holds Alice pays Bob 76 DS Coin
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.
3
View the Blockchain
{"blockchain":[{"index":0,"timestamp":"Oct 24, 2022, 12:03:08
AM","data":"Genesis","previousHash":"","nonce":958,"difficulty":2},{"index":1,"timestamp":"O
ct 24, 2022, 12:03:39 AM","data":"Alice pays Bob 76 DS
Coin","previousHash":"000344DA0511274D76DF011EFA9B278098C0EF8A2266768633C93BB85
D7526D8","nonce":778,"difficulty":2},{"index":2,"timestamp":"Oct 24, 2022, 12:04:01
AM","data":"Bob pays Carol 50 DS
Coin","previousHash":"00257E9EDF9F671274F0D3022698DC34AE15725C8C8ABD63D33AA0DA
CF7CC4E2","nonce":9,"difficulty":2},{"index":3,"timestamp":"Oct 24, 2022, 12:04:16
AM","data":"Carol pays Donna 10 DS
Coin","previousHash":"004EEE64A034303E133DD02DECF255703E7F8DE20E34EA0A438D274C5
912E1FF","nonce":521,"difficulty":2}],"chain_hash":"0036B62F581246605FBFFD65ADBFEE1262
51B18941453D89D3DB3B88F3BCF0EE","hashes_per_second":1718213}
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.
2
Chain verification: False
Improper hash on node 1 does not begin with 00
Total execution time to verify the chain was 1 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.
5
Total execution time required to repair the chain was 12 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.
2
Chain verification: True
Total execution time to verify the chain was 1 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.
1
Enter difficulty > 0
4
Enter transaction
Donna pays Sean 25 DS Coin
Total execution time to add this block was 100 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.
0
Current size of chain: 5
Difficulty of most recent block: 4
Total difficulty for all blocks: 12
Approximate hashes per second on this machine: 1718213
Expected total hashes required for the whole chain: 66560.0
Nonce for most recent block: 30538
Chain hash: 00007012A44D8DA0C0527978E4A64053FBFBDAD90BDA38FDC9C9239A6C70E6A6
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.
6

Process finished with exit code 0
```

Project3Task1: Code for RequestMessage.java

```java
// Refer to https://www.javatpoint.com/java-json-example
import org.json.simple.JSONObject;
import org.json.simple.JSONValue;

import java.net.*;
import java.io.*;
import java.util.Scanner;
// Name: Leo Lin
// AndrewID: hungfanl

public class RequestMessage {
    static Socket clientSocket;
    static int serverPort = 7777;
    static BufferedReader in;
    static PrintWriter out;
    static Scanner readInput = new Scanner(System.in);
    static boolean finish = false;
    static JSONObject json = new JSONObject();
    public static void main(String args[]) {
        try{
            clientSocket = new Socket("localhost", serverPort);
            BufferedReader typed = new BufferedReader(new InputStreamReader(System.in));
            while(true) {
                // settle the json string
                int option = getSelection();
                if(finish) break;
                // pass the json to the server
                pass(option);
            }

        } catch (IOException e) {
            System.out.println("IO Exception: " + e.getMessage());
        } finally {
            try {
                if(clientSocket != null) clientSocket.close();
            } catch (IOException e) {

            }
        }
    }

    // Allows the user to choose from different operations and collect necessary information.
    public static int getSelection() {
        String[] message = {"View basic blockchain status.", "Add a transaction to the
blockchain.",
                "Verify the blockchain.", "View the blockchain.", "Corrupt the chain.",
                "Hide the corruption by repairing the chain.", "Exit."};
        json.clear();
        for(int i = 0; i < message.length; i++) {
            System.out.println(i + ". " + message[i]);
        }
        int option = Integer.parseInt(readInput.nextLine());
        switch (option) {
            case 0:
                json.put("selection", 0);
                break;
            case 1:
                System.out.println("Enter difficulty > 0");
                int difficulty = Integer.parseInt(readInput.nextLine());
                System.out.println("Enter transaction");
                String data = readInput.nextLine();
                json.put("selection", 1);
                json.put("difficulty", difficulty);
                json.put("data", data);
                break;
            case 2:
                json.put("selection", 2);
                break;
            case 3:
                json.put("selection", 3);
                break;
            case 4:
                System.out.println("Enter block ID of block to corrupt");
                int index = Integer.parseInt(readInput.nextLine());
                System.out.println("Enter new data for block " + index);
                String corrupt_message = readInput.nextLine();
                json.put("selection", 4);
                json.put("index", index);
                json.put("data", corrupt_message);
                break;
            case 5:
                json.put("selection", 5);
                break;
            case 6:
                finish = true;
                break;
            default:
                break;
        }
        return option;
    }

    // Pass the information to the server
    public static void pass(int option){
        try {
            in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));
            out = new PrintWriter(new BufferedWriter(new
OutputStreamWriter(clientSocket.getOutputStream())));
            out.println(json.toJSONString());
            out.flush();
            json = (JSONObject) JSONValue.parse(in.readLine()); // read a line of data from the
stream
        } catch (IOException e) {
            System.out.println("IO Exception:" + e.getMessage());
        }
        switch (option) {
            case 0:
                System.out.println("Current size of chain: " + ((Long)
json.get("size")).intValue());
                System.out.println("Difficulty of most recent block: " + ((Long)
json.get("diff")).intValue());
                System.out.println("Total difficulty for all blocks: " + ((Long)
json.get("totalDiff")).intValue());
                System.out.println("Approximate hashes per second on this machine: " + ((Long)
json.get("hps")).intValue());
                System.out.println("Expected total hashes required for the whole chain: " +
(double) json.get("totalHashes"));
                System.out.println("Nonce for most recent block: " + ((Long)
json.get("recentNonce")).intValue());
                System.out.println("Chain hash: " + json.get("chainHash"));
                break;
            case 1:
                System.out.println(json.get("response"));
                break;
            case 2:
                System.out.println("Chain verification: " + json.get("verification"));
                if(json.get("verification").equals("False"))
System.out.println(json.get("errorMessage"));
                System.out.println(json.get("response"));
                break;
            case 3:
                // Intentionally not break;
                System.out.println("View the Blockchain");
                System.out.println(json.get("response"));
                break;
            case 4:
                System.out.println("corrupt the Blockchain");
            case 5:
                System.out.println(json.get("response"));
                break;
        }
    }
}
```

Project3Task1: Code for ResponseMessage.java

```
import com.google.gson.Gson;
import java.net.*;
import java.io.*;
import java.nio.charset.StandardCharsets;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.sql.Timestamp;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
// Refer to https://www.javatpoint.com/java-json-example
// Name: Leo Lin
// AndrewID: hungfanl
import org.json.simple.JSONObject;
import org.json.simple.JSONValue;

public class ResponseMessage {
    static Socket clientSocket = null;
    static int serverPort = 7777;
    static JSONObject listenJson = new JSONObject();
    static JSONObject returnJson = new JSONObject();
    static BlockChain bc;

    public static void main(String[] args) {
        bc = new BlockChain();
        // Create a new genesis block with the difficulty of 2
        Block b = new Block(bc.getChainSize(), bc.getTime(), "Genesis", 2);
        bc.addBlock(b);
        bc.computeHashesPerSecond();
        System.out.println("Blockchain server running");
        try{
            ServerSocket listenSocket = new ServerSocket(serverPort);
            clientSocket = listenSocket.accept();
            Scanner in = new Scanner(clientSocket.getInputStream());
            PrintWriter out;
            out = new PrintWriter(new BufferedWriter(new
OutputStreamWriter(clientSocket.getOutputStream())));
            System.out.println("We have a visitor");
            // The server will run permanently
            while (true) {
                // If the user remains connected. Listen to the user.
                if (in.hasNextLine()) {
                    String info = in.nextLine();
                    listenJson = (JSONObject) JSONValue.parse(info);
                    Long l = (Long) listenJson.get("selection");
                    int option = l.intValue();
                    process(option);
                    out.println(returnJson.toJSONString());
                    out.flush();
                }
                // If there is no user, wait for another user.
                else {
                    clientSocket = listenSocket.accept();
                    in = new Scanner(clientSocket.getInputStream());
                    out = new PrintWriter(new BufferedWriter(new
OutputStreamWriter(clientSocket.getOutputStream())));
                }
            }
        } catch (IOException e) {
            System.out.println("IO Exception:" + e.getMessage());
        } finally {
            try {
                if(clientSocket != null) clientSocket.close();
            } catch (IOException e) {
            }
        }
```

```java
    }

    // According to different option from the client, perform different operations.
    public static void process(int option) {
        Timestamp start;
        returnJson.clear();
        Timestamp end;
        String response;
        int time;
        switch (option) {
            // Check the information of the chain
            case 0:
                returnJson.put("selection", 0);
                returnJson.put("size", bc.getChainSize());
                returnJson.put("chainHash", bc.getChainHash());
                returnJson.put("totalHashes", bc.getTotalExpectedHashes());
                returnJson.put("totalDiff", bc.getTotalDifficulty());
                returnJson.put("recentNonce", bc.getBlock(bc.getChainSize()-1).getNonce());
                returnJson.put("diff", bc.getBlock(bc.getChainSize()-1).getDifficulty());
                returnJson.put("hps", bc.getHashesPerSecond());
                System.out.println("Response : " + returnJson.toJSONString());
                break;
            // Allows the user to add a block to the chain.
            case 1:
                System.out.println("Adding a block");
                start = bc.getTime();
                bc.addBlock(new Block(bc.getChainSize(), bc.getTime(), (String)
listenJson.get("data"), ((Long) listenJson.get("difficulty")).intValue()));
                end = bc.getTime();
                time = (int) (end.getTime() - start.getTime());
                response = "Total execution time to add this block was " + time +" milliseconds";
                System.out.println("Setting response to " + response);
                returnJson.put("selection", 1);
                returnJson.put("response", response);
                System.out.println("..." + returnJson.toJSONString());
                break;
            // Check if the chain is valid.
            case 2:
                System.out.println("Verifying entire chain");
                start = bc.getTime();
                String validation = bc.isChainValid();
                end = bc.getTime();
                time = (int) (end.getTime() - start.getTime());
                System.out.print("Chain verification: ");

                if (!validation.equals("True")) {
                    returnJson.put("verification", "False");
                    returnJson.put("errorMessage", validation);
                    System.out.println("False");
                }
                else returnJson.put("verification", "True");
                System.out.println(validation);
                response = "Total execution time required to verify the chain was " + time +"
milliseconds";
                System.out.println(response);
                System.out.println("Setting response to " + response);
                returnJson.put("response", response);
                break;
            // View the block (json style)
            case 3:
                System.out.println("View the Blockchain");
                System.out.println("Setting response to " + bc.toString());
                returnJson.put("response", bc.toString());
                break;
            // Corrupt the chain by changing the information in the block.
            case 4:
                System.out.println("Corrupt the Blockchain");
                int index = ((Long) listenJson.get("index")).intValue();
                String corrupt_message = (String) listenJson.get("data");
                bc.getBlock(index).setData(corrupt_message);
                response = "Block " + index + " now holds " + corrupt_message;
                System.out.println(response);
                returnJson.put("response",response);
                break;
            // Fix the block by recomputing the nonce to meet the difficulty requirement.
            case 5:
                System.out.println("Repairing the entire chain");
                start = bc.getTime();
                if(!bc.isChainValid().equals("True")) bc.repairChain();
                end = bc.getTime();
                time = (int) (end.getTime() - start.getTime());
                response = "Total execution time required to repair the chain was " + time +"
milliseconds";
                System.out.println("Setting response to " + response);
                returnJson.put("response",response);
                break;

        }

    }
    // An inner class same as task 0.
    public static class BlockChain extends java.lang.Object{
        public List<Block> blockchain;
        public String chain_hash;
        public int hashes_per_second;
        public static final int HASH_ROUND = 2000000;
        private static final char[] HEX_ARRAY = "0123456789ABCDEF".toCharArray();

        private static final String LEADING_ZERO =
"0000000000000000000000000000000000000000000000000";

        public BlockChain(){
            blockchain = new ArrayList<>();
            chain_hash = "";
            hashes_per_second = 0;
        }
        public String getChainHash() {
            return chain_hash;
        }
        public Timestamp getTime(){
            return new Timestamp(System.currentTimeMillis());
        }
        public Block getLatestBlock(){
            return blockchain.get(this.getChainSize()-1);
        }
        public int getChainSize(){
            return blockchain.size();
        }

        // Get the information of a particular block in the chain.
        public Block getBlock(int i) {
            if(i >= getChainSize()) {
                System.out.println("Insert number exceed block size");
                return null;
            }
            return blockchain.get(i);
        }

        // Compute the expected time of calculating hash by doing the calculation for 20000 times
and get the average time.
        public void computeHashesPerSecond() {
            String s = "00000000";
            Timestamp start = getTime();
            for(int i = 0; i < HASH_ROUND; i++) {
                calculateHash(s);
            }
            Timestamp end = getTime();
            hashes_per_second = (int) ( (double)HASH_ROUND / (end.getTime() - start.getTime()) *
1000);
        }
        public int getHashesPerSecond() {
            return  hashes_per_second;
        }

        // Add a block in the chain and revise the chain information.
        public void addBlock(Block block){
            if(getChainSize() == 0) {
                block.setPreviousHash("");
            }
            else block.setPreviousHash(chain_hash);
            blockchain.add(block);
            chain_hash = block.proofOfWork();
        }

        // Transfer the chain in the form of json.
        public String toString() {
            BlockChain bc = new BlockChain();
            for(int i = 0; i < getChainSize(); i++) {
                bc.blockchain.add(getBlock(i));
            }
            bc.hashes_per_second = getHashesPerSecond();
            bc.chain_hash = getChainHash();
            Gson gson = new Gson();
            String messageToSend = gson.toJson(bc);
            return messageToSend;
        }

        // Adding the difficulty of all blocks and come up with a total difficulty of the chain.
        public int getTotalDifficulty() {
            int totalDifficulty = 0;
            for(Block b: blockchain) {
                totalDifficulty += b.getDifficulty();
            }
            return totalDifficulty;
        }

        // Get the expected number of hashes the chain requires to compute.
        public double getTotalExpectedHashes() {
            double totalExpectedHashes = 0;
            for(Block b: blockchain) {
                totalExpectedHashes += Math.pow(16, b.getDifficulty());
            }
            return  totalExpectedHashes;
        }

        // Return true if the function founds no error, the type of error if the function finds
any.
        public String isChainValid() {
            for(int i = 0; i < getChainSize(); i++) {
                Block b = getBlock(i);
                String s = b.calculateHash();
                for(int j = 0; j < b.getDifficulty(); j++) {
                    //Improper hash on node 1 Does not begin with 00
                    if(s.charAt(j) != '0')
                        return "Improper hash on node " + i + " does not begin with " +
                                LEADING_ZERO.substring(0, b.getDifficulty());
                }
                if(i != 0 && !getBlock(i-1).calculateHash().equals(b.getPreviousHash()))
                    return "Block " + i + " does not have a matching hash.";
            }
            if(!getBlock(getChainSize()-1).calculateHash().equals(chain_hash))
                return "The chain hash is different from the hash of the last block.";
            return "True";
        }

        // Fix the chain by changing the nonce and get the right hash number.
        public void repairChain() {
            for(int i = 0; i < getChainSize(); i++) {
                Block b = getBlock(i);
                if(i != getChainSize()-1) getBlock(i+1).previousHash = b.proofOfWork();
                else chain_hash = b.proofOfWork();
            }
        }

        // Calculate the hash number.
        public String calculateHash(String s) {
            String hash_value = null;
            try {
                MessageDigest md;
                // Compute SHA-265 code for the input
                md = MessageDigest.getInstance("SHA-256");
                md.update(s.getBytes(StandardCharsets.UTF_8));
                hash_value = bytesToHex(md.digest());
            }
            catch(NoSuchAlgorithmException e) {
                System.out.println("No Hash available" + e);
            }
            return String.valueOf(hash_value);
        }

        // Transfer the byte representation of a string to a hex value.
        public static String bytesToHex(byte[] bytes) {
            char[] hexChars = new char[bytes.length * 2];
            for (int j = 0; j < bytes.length; j++) {
                int v = bytes[j] & 0xFF;
                hexChars[j * 2] = HEX_ARRAY[v >>> 4];
                hexChars[j * 2 + 1] = HEX_ARRAY[v & 0x0F];
            }
            return new String(hexChars);
        }
    }
}
```

Project3Task1: Code for Block.java(Same as task0)

```
import com.google.gson.Gson;
import java.math.BigInteger;
import java.nio.charset.StandardCharsets;
import java.sql.Timestamp;
import java.security.NoSuchAlgorithmException;
import java.security.MessageDigest;

// Name: Leo Lin
// AndrewID: hungfanl

public class Block extends java.lang.Object {
    private static final char[] HEX_ARRAY = "0123456789ABCDEF".toCharArray();
    int index;
    Timestamp timestamp;
    String data;
    String previousHash;
    BigInteger nonce;
    int difficulty;
    Block (int index, Timestamp timestamp, String data, int difficulty) {
        this.index = index;
        this.timestamp = timestamp;
        this.data = data;
        this.difficulty = difficulty;
        nonce = new BigInteger("0");
    }
    // Calculate the hash for the block
    public String calculateHash() {
        String information = index + timestamp.toString() + data + previousHash + nonce +
difficulty;
        String hash_value = null;
        try {
            MessageDigest md;
            // Compute SHA-265 code for the input
            md = MessageDigest.getInstance("SHA-256");
            md.update(information.getBytes(StandardCharsets.UTF_8));
            hash_value = bytesToHex(md.digest());
        }
        catch(NoSuchAlgorithmException e) {
            System.out.println("No Hash available" + e);
        }
        return String.valueOf(hash_value);
    }

    public String getData() {
        return data;
    }

    public int getDifficulty() {
        return difficulty;
    }

    public int getIndex() {
        return index;
    }

    public BigInteger getNonce() {
        return nonce;
    }

    public String getPreviousHash() {
        return previousHash;
    }

    public Timestamp getTimestamp() {
        return timestamp;
    }
    // Make sure that the has for the block meets the requirement, otherwise add the nonce by one
and recalculate.
    public String proofOfWork() {
        String hash_value = null;
        boolean leading_zero = false;
        while(!leading_zero) {
            leading_zero = true;
            hash_value = calculateHash();
            for(int i = 0; i < difficulty; i++){
                if(hash_value.charAt(i) != '0') {
                    leading_zero = false;
                    nonce = nonce.add(BigInteger.valueOf(1));
                    break;
                }
            }
        }
        return hash_value;
    }

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

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

    public void setIndex(int index) {
        this.index = index;
    }

    public void setPreviousHash(String previousHash) {
        this.previousHash = previousHash;
    }

    public void setTimestamp(Timestamp timestamp) {
        this.timestamp = timestamp;
    }

    // Transfer the block in the form of json.
```

```java
    public String toString() {
        Block b = new Block (index, timestamp, data, difficulty);
        b.nonce = nonce;
        b.setPreviousHash(previousHash);
        Gson gson = new Gson();
        String messageToSend = gson.toJson(b);
        return messageToSend;
    }
    // Reference from Lab1 submission
    public static String bytesToHex(byte[] bytes) {
        char[] hexChars = new char[bytes.length * 2];
        for (int j = 0; j < bytes.length; j++) {
            int v = bytes[j] & 0xFF;
            hexChars[j * 2] = HEX_ARRAY[v >>> 4];
            hexChars[j * 2 + 1] = HEX_ARRAY[v & 0x0F];
        }
        return new String(hexChars);
    }
}
```

Project3Task1 Server-Side Execution

```
/Library/Java/JavaVirtualMachines/jdk-17.0.3.1.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA.app/Contents/lib/idea_rt.jar=60050:/Applications/IntelliJ IDEA.app/Contents/bin -Dfile.encoding=UTF-8 -classpath /Users/linhungfan/Desktop/CMU/Semester 2/Distributed System/Project3/Project3Task1/target/classes:/Users/linhungfan/.m2/repository/com/google/code/gson/gson/2.9.0/gson-2.9.0.jar:/Users/linhungfan/.m2/repository/com/googlecode/json-simple/json-simple/1.1/json-simple-1.1.jar ResponseMessage
Blockchain server running
We have a visitor
Response :
{"selection":0,"size":1,"chainHash":"00D4B2140050B3BD92A023589DD0A1B95FE255413613FF19D92F88FBF278678D","totalHashes":256.0,"totalDiff":2,"recentNonce":157,"diff":2,"hps":1694915}
Adding a block
Setting response to Total execution time to add this block was 10 milliseconds
...{"selection":1,"response":"Total execution time to add this block was 10 milliseconds"}
Adding a block
Setting response to Total execution time to add this block was 7 milliseconds
...{"selection":1,"response":"Total execution time to add this block was 7 milliseconds"}
Adding a block
Setting response to Total execution time to add this block was 4 milliseconds
...{"selection":1,"response":"Total execution time to add this block was 4 milliseconds"}
Verifying entire chain
Chain verification: True
Total execution time required to verify the chain was 0 milliseconds
Setting response to Total execution time required to verify the chain was 0 milliseconds
View the Blockchain
Setting response to {"blockchain":[{"index":0,"timestamp":"Oct 24, 2022, 12:54:22 AM","data":"Genesis","previousHash":"","nonce":157,"difficulty":2},{"index":1,"timestamp":"Oct 24, 2022, 12:54:38 AM","data":"Alice pays Bob 100 DS Coin","previousHash":"00D4B2140050B3BD92A023589DD0A1B95FE255413613FF19D92F88FBF278678D","nonce":156,"difficulty":2},{"index":2,"timestamp":"Oct 24, 2022, 12:54:47 AM","data":"Bob pays Carol 50 DS Coin","previousHash":"00CFD38EE2A7CB814D365C9150732DDDEDD5A54DEFC6D184E19B935D834C47DF","nonce":150,"difficulty":2},{"index":3,"timestamp":"Oct 24, 2022, 12:54:55 AM","data":"Carol pays Donna 10 DS Coin","previousHash":"009AE63A3E76F2C8571D1AEB5CB0BE67F4179BCE04248AA2B66B5716F72F8B13","no
nce":32,"difficulty":2}],"chain_hash":"009E912084008B151B8D784A8D00916F37C23E8C0E265CDD069C4D7
DF3D70FD6","hashes_per_second":1694915}
Corrupt the Blockchain
Block 1 now holds Alice pays Bob 76 DS Coin
View the Blockchain
Setting response to {"blockchain":[{"index":0,"timestamp":"Oct 24, 2022, 12:54:22
AM","data":"Genesis","previousHash":"","nonce":157,"difficulty":2},{"index":1,"timestamp":"Oct 24, 2022,
12:54:38 AM","data":"Alice pays Bob 76 DS
Coin","previousHash":"00D4B2140050B3BD92A023589DD0A1B95FE255413613FF19D92F88FBF278678D","n
once":156,"difficulty":2},{"index":2,"timestamp":"Oct 24, 2022, 12:54:47 AM","data":"Bob pays Carol 50 DS
Coin","previousHash":"00CFD38EE2A7CB814D365C9150732DDDEDD5A54DEFC6D184E19B935D834C47DF","
nonce":150,"difficulty":2},{"index":3,"timestamp":"Oct 24, 2022, 12:54:55 AM","data":"Carol pays Donna 10
DS
Coin","previousHash":"009AE63A3E76F2C8571D1AEB5CB0BE67F4179BCE04248AA2B66B5716F72F8B13","no
nce":32,"difficulty":2}],"chain_hash":"009E912084008B151B8D784A8D00916F37C23E8C0E265CDD069C4D7
DF3D70FD6","hashes_per_second":1694915}
Verifying entire chain
Chain verification: False
Improper hash on node 1 does not begin with 00
Total execution time required to verify the chain was 2 milliseconds
Setting response to Total execution time required to verify the chain was 2 milliseconds
Repairing the entire chain
Setting response to Total execution time required to repair the chain was 16 milliseconds
Verifying entire chain
Chain verification: True
Total execution time required to verify the chain was 4 milliseconds
Setting response to Total execution time required to verify the chain was 4 milliseconds
Adding a block
Setting response to Total execution time to add this block was 66 milliseconds
...{"selection":1,"response":"Total execution time to add this block was 66 milliseconds"}
Response :
{"selection":0,"size":5,"chainHash":"00000CDA125693D33FD8F1F54AAF5DD3E7212EEABBF6C8859F799F496
7EA28BC","totalHashes":66560.0,"totalDiff":12,"recentNonce":13434,"diff":4,"hps":1694915}
```

Project3Task1 Client-Side Execution

```
/Library/Java/JavaVirtualMachines/jdk-17.0.3.1.jdk/Contents/Home/bin/java -javaagent:/Applications/IntelliJ
IDEA.app/Contents/lib/idea_rt.jar=60054:/Applications/IntelliJ IDEA.app/Contents/bin -Dfile.encoding=UTF-8
-classpath /Users/linhungfan/Desktop/CMU/Semester 2/Distributed
System/Project3/Project3Task1/target/classes:/Users/linhungfan/.m2/repository/com/google/code/gson/gs
on/2.9.0/gson-2.9.0.jar:/Users/linhungfan/.m2/repository/com/googlecode/json-simple/json-
simple/1.1/json-simple-1.1.jar RequestMessage
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.
```

```
0
Current size of chain: 1
Difficulty of most recent block: 2
Total difficulty for all blocks: 2
Approximate hashes per second on this machine: 1694915
Expected total hashes required for the whole chain: 256.0
Nonce for most recent block: 157
Chain hash: 00D4B2140050B3BD92A023589DD0A1B95FE255413613FF19D92F88FBF278678D
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.
1
Enter difficulty > 0
2
Enter transaction
Alice pays Bob 100 DS Coin
Total execution time to add this block was 10 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.
1
Enter difficulty > 0
2
Enter transaction
Bob pays Carol 50 DS Coin
Total execution time to add this block was 7 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.
1
Enter difficulty > 0
2
Enter transaction
Carol pays Donna 10 DS Coin
Total execution time to add this block was 4 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.
2
Chain verification: True
Total execution time required to verify the chain was 0 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.
3
View the Blockchain
{"blockchain":[{"index":0,"timestamp":"Oct 24, 2022, 12:54:22
AM","data":"Genesis","previousHash":"","nonce":157,"difficulty":2},{"index":1,"timestamp":"Oct 24, 2022,
12:54:38 AM","data":"Alice pays Bob 100 DS
Coin","previousHash":"00D4B2140050B3BD92A023589DD0A1B95FE255413613FF19D92F88FBF278678D","n
once":156,"difficulty":2},{"index":2,"timestamp":"Oct 24, 2022, 12:54:47 AM","data":"Bob pays Carol 50 DS
Coin","previousHash":"00CFD38EE2A7CB814D365C9150732DDDEDD5A54DEFC6D184E19B935D834C47DF","
nonce":150,"difficulty":2},{"index":3,"timestamp":"Oct 24, 2022, 12:54:55 AM","data":"Carol pays Donna 10
DS
Coin","previousHash":"009AE63A3E76F2C8571D1AEB5CB0BE67F4179BCE04248AA2B66B5716F72F8B13","no
nce":32,"difficulty":2}],"chain_hash":"009E912084008B151B8D784A8D00916F37C23E8C0E265CDD069C4D7
DF3D70FD6","hashes_per_second":1694915}
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.
4
Enter block ID of block to corrupt
1
Enter new data for block 1
Alice pays Bob 76 DS Coin
corrupt the Blockchain
Block 1 now holds Alice pays Bob 76 DS Coin
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.
3
View the Blockchain
```

```
{"blockchain":[{"index":0,"timestamp":"Oct 24, 2022, 12:54:22
AM","data":"Genesis","previousHash":"","nonce":157,"difficulty":2},{"index":1,"timestamp":"Oct 24, 2022,
12:54:38 AM","data":"Alice pays Bob 76 DS
Coin","previousHash":"00D4B2140050B3BD92A023589DD0A1B95FE255413613FF19D92F88FBF278678D","n
once":156,"difficulty":2},{"index":2,"timestamp":"Oct 24, 2022, 12:54:47 AM","data":"Bob pays Carol 50 DS
Coin","previousHash":"00CFD38EE2A7CB814D365C9150732DDDEDD5A54DEFC6D184E19B935D834C47DF","
nonce":150,"difficulty":2},{"index":3,"timestamp":"Oct 24, 2022, 12:54:55 AM","data":"Carol pays Donna 10
DS
Coin","previousHash":"009AE63A3E76F2C8571D1AEB5CB0BE67F4179BCE04248AA2B66B5716F72F8B13","no
nce":32,"difficulty":2}],"chain_hash":"009E912084008B151B8D784A8D00916F37C23E8C0E265CDD069C4D7
DF3D70FD6","hashes_per_second":1694915}
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.
2
Chain verification: False
Improper hash on node 1 does not begin with 00
Total execution time required to verify the chain was 2 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.
5
Total execution time required to repair the chain was 16 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.
2
Chain verification: True
Total execution time required to verify the chain was 4 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.
1
Enter difficulty > 0
```

```
4
Enter transaction
Donna pays Sean 25 DS Coin
Total execution time to add this block was 66 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.
0
Current size of chain: 5
Difficulty of most recent block: 4
Total difficulty for all blocks: 12
Approximate hashes per second on this machine: 1694915
Expected total hashes required for the whole chain: 66560.0
Nonce for most recent block: 13434
Chain hash: 00000CDA125693D33FD8F1F54AAF5DD3E7212EEABBF6C8859F799F4967EA28BC
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.
6

Process finished with exit code 0
```