Project3Task0

1.Task 0 Execution

```
"C:\Program Files\Eclipse Adoptium\jdk-17.0.8.101-hotspot\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2023.2.1\lib\idea_rt.jar=59282:C:\Program Files\JetBrains\IntelliJ IDEA 2023.2.1\bin" -Dfile.encoding=UTF-8 -classpath C:\Users\Administrator\IdeaProjects\Project3Task0\target\classes;C:\Users\Administrator\.m2\repository\com\google\code\gson\gson\2.9.0\gson-2.9.0.jar ds.Project3Task0.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
Total difficulty for all blocks: 2
Total difficulty for all blocks: 2
Approximate hashes per second on this machine: 1648000
Expected total hashes required for the whole chain: 256.0
Nonce for most recent block: 1
Chain
hash:00DE520F324D259F946114921287D82A526161F1ECD80397E0F6E3DD85EDD9EA
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
Mike pays Marty 100 DSCoin
Total execution time to add this block 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.

1
Enter difficulty > 0
2
Enter transaction
Marty pays Joe 50 DSCoin
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
Joe pays Andy 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 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.
```

```
3
View the Blockchain
{"block_chain":[{"index":0,"timestamp":"Oct 27, 2023, 2:08:07
PM","data":"Genesis","previousHash":"","nonce":1,"difficulty":2},{"in
dex":1,"timestamp":"Oct 27, 2023, 2:08:14 PM","data":"Mike pays Marty 100
DSCoin","previousHash":"00DE520F324D259F946114921287D82A526161F1ECD80
397E0F6E3DD85EDD9EA","nonce":37,"difficulty":2},{"index":2,"timestamp
":"Oct 27, 2023, 2:08:22 PM","data":"Marty pays Joe 50
DSCoin","previousHash":"00BBC1D543C122D987FDD9B46ED0D3CEE132172579B2D
6A7C9D11AE0478034F7","nonce":223,"difficulty":2},{"index":3,"timestam
p":"Oct 27, 2023, 2:08:31 PM","data":"Joe pays Andy 10 DS
Coin","previousHash":"00F9349591EC0B352BE366A3D3AF37304523EF2D18585B1
91E54A433831D05B4","nonce":262,"difficulty":2}],"chainHash":"0005C062
E87BF67854E17E9AE9354E01B46B8F0862971FAD6AE6D6C2E8D830CC","hashes_per
_second":1648000}
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
Mike pays Marty 76 DSCoin
Block 1 now holds Mike pays Marty 76 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.

3
View the Blockchain
{"block_chain":[{"index":0,"timestamp":"Oct 27, 2023, 2:08:07
PM","data":"Genesis","previousHash":"","nonce":1,"difficulty":2},{"in
dex":1,"timestamp":"Oct 27, 2023, 2:08:14 PM","data":"Mike pays Marty 76
DSCoin","previousHash":"00DE520F324D259F946114921287D82A526161F1ECD80
```

397E0F6E3DD85EDD9EA","nonce":37,"difficulty":2},{"index":2,"timestamp":"Oct 27, 2023, 2:08:22 PM","data":"Marty pays Joe 50 DSCoin","previousHash":"00BBC1D543C122D987FDD9B46ED0D3CEE132172579B2D6A7C9D11AE0478034F7","nonce":223,"difficulty":2},{"index":3,"timestamp":"Oct 27, 2023, 2:08:31 PM","data":"Joe pays Andy 10 DS Coin","previousHash":"00F9349591EC0B352BE366A3D3AF37304523EF2D18585B191E54A433831D05B4","nonce":262,"difficulty":2}],"chainHash":"0005C062E87BF67854E17E9AE9354E01B46B8F0862971FAD6AE6D6C2E8D830CC","hashes_per_second":1648000}

```
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 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.

5
Total execution time required to repair 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.

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.

1
Enter difficulty > 0
4
Enter transaction
Andy pays Sean 25 DSCoin
Total execution time to add this block was 222 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
Total difficulty for all blocks: 4
Total difficulty for all blocks: 12
Approximate hashes per second on this machine: 1648000
Expected total hashes required for the whole chain: 66560.0
Nonce for most recent block: 108233
Chain
hash:0000A2EB7CCC64F2E15F2FD6520AFBD6FB41EECE63238695E46027B94937F193
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
5
Enter transaction
Xinyuan pays Marty 100 DSCoin
Total execution time to add this block was 777 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.

6

Process finished with exit code 0
```

2.Task 0 Block.java

```
package ds.Project3Task0;

//Work Cited:
//https://www.andrew.cmu.edu/course/95-702/examples/javadoc/blockchaintask0/Block.html
//Self Learning:
//The Block Class
//This class represents a simple Block.
//Each Block object has an index - the position of the block on the chain. The first block (the so called Genesis block) has an index of 0.
//Each block has a timestamp - a Java Timestamp object, it holds the time of the block's creation.
//Each block has a field named data - a String holding the block's single transaction details.
//Each block has a String field named previousHash - the SHA256 hash of a block's parent. This is also called a hash pointer.
//Each block holds a nonce - a BigInteger value determined by a proof of work routine. This has to be found by the proof of work logic.
// It has to be found so that this block has a hash of the proper difficulty.
// The difficulty is specified by a small integer representing the minimum number of leading hex zeroes the hash must have.
//
//Each block has a field named difficulty -
// it is an int that specifies the minimum number of left most hex digits needed by a proper hash.
// The hash is represented in hexadecimal. If,
// for example, the difficulty is 3, the hash must have at least three leading hex 0's (or, 1 and 1/2 bytes). Each hex digit represents 4 bits. (Some parts of the comments also cited from the upper website.)
// I have also taken a look at Piazza and discussed some of the purposes of functions with my classmates KW.
```

```java
/**
 * This Java code defines a Block class representing an individual block in a blockchain.
 * Each block has an index, timestamp, data, previous hash, nonce, and difficulty level
for mining.
 * The class provides methods for calculating the hash, performing proof-of-work, and
managing block attributes.
 * */


import com.google.gson.Gson;
import java.math.BigInteger;
import java.io.UnsupportedEncodingException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.sql.Timestamp;

public class Block {
    private int index;
    private Timestamp timestamp;
    private String data;
    private String previousHash;
    private BigInteger nonce;
    private int difficulty;


    /**
     * Constructor for the Block class.
     *
     * @param index Block index.
     * @param timestamp Timestamp of the block.
     * @param data Data to be stored in the block.
     * @param difficulty Difficulty level for mining.
     * @returns Initializes a new Block object with the provided parameters.
     */
    public Block(int index, Timestamp timestamp,String data, int difficulty) {
        this.index = index;
        this.timestamp = timestamp;
        this.data = data;
        this.difficulty = difficulty;
    }

    /**
     * Calculates the hash of the block.
     *
```

```
     * @returns SHA-256 hash of the block's content as a hexadecimal string.
     */
    public String calculateHash() {
        String block_to_string = "";
        block_to_string = index + timestamp.toString() + data + previousHash + nonce
+ difficulty;
        try {
            // Work Cited:
https://github.com/CMU-Heinz-95702/Lab1-InstallationAndRaft
            MessageDigest md = MessageDigest.getInstance("SHA-256");
            md.update(block_to_string.getBytes("UTF-8"),0,block_to_string.length());
            byte[] bytes_info;
            bytes_info = md.digest();
            return bytesToHex(bytes_info);
        }
        catch(NoSuchAlgorithmException e) {
            System.out.println("NoSuchAlgorithmException" + e);
        } catch (UnsupportedEncodingException e) {
            throw new RuntimeException(e);
        }
        return null;
    }
    private static final char[] HEX_ARRAY = "0123456789ABCDEF".toCharArray();
    private String bytesToHex(byte[] bytesInfo) {
        char[] hexChars = new char[bytesInfo.length * 2];
        for (int j = 0; j < bytesInfo.length; j++) {
            int v = bytesInfo[j] & 0xFF;
            hexChars[j * 2] = HEX_ARRAY[v >>> 4];
            hexChars[j * 2 + 1] = HEX_ARRAY[v & 0x0F];
        }
        return new String(hexChars);
    }

    /**
     * Retrieves the data of the block.
     *
     * @returns Data stored in the block.
     */
    public String getData() {

        return data;
    }

    /**
```

```
     * Retrieves the difficulty level of the block.
     *
     * @returns Difficulty level of the block.
     */
    public int getDifficulty() {

        return difficulty;
    }

    /**
     * Retrieves the index of the block.
     *
     * @returns Index of the block.
     */
    public int getIndex() {

        return index;
    }

    /**
     * Retrieves the nonce value of the block.
     *
     * @returns Nonce value of the block.
     */
    public BigInteger getNonce() {

        return this.nonce;
    }


    /**
     * Retrieves the previous hash of the block.
     *
     * @returns Previous hash of the block.
     */
    public String getPreviousHash() {

        return previousHash;
    }


    /**
     * Retrieves the timestamp of the block.
     *
```

```
     * @returns Timestamp of the block.
     */
    public Timestamp getTimestamp() {

        return timestamp;
    }

    public static void main(String[] args) {
    }

    /**
     * Performs a proof-of-work to find a valid hash for the block.
     *
     * @returns Valid hash meeting the required difficulty level.
     */
    public String proofOfWork() {
        //Work
Cited:https://docs.oracle.com/javase%2F7%2Fdocs%2Fapi%2F%2F/java/math/BigInteger.ht
ml
        nonce = BigInteger.ZERO;
        String zeros = "0".repeat(difficulty);

        while (true) {
            String hash = calculateHash();
            if (hash.startsWith(zeros)) {
                return hash;
            } else {
                nonce =  this.nonce.add(BigInteger.ONE);
            }
        }
    }

    /**
     * Sets the data of the block.
     *
     * @param data New data to be set in the block.
     * @returns No explicit return value.
     */
    public void setData(String data) {

        this.data = data;
    }

    /**
```

```
     * Sets the difficulty level of the block.
     *
     * @param difficulty New difficulty level to be set.
     * @returns No explicit return value.
     */
    public void setDifficulty(int difficulty) {

        this.difficulty = difficulty;
    }

    /**
     * Sets the index of the block.
     *
     * @param index New index to be set.
     * @returns No explicit return value.
     */
    public void setIndex(int index) {

        this.index = index;
    }

    /**
     * Sets the previous hash of the block.
     *
     * @param previousHash New previous hash to be set.
     * @returns No explicit return value.
     */
    public void setPreviousHash(String previousHash) {

        this.previousHash = previousHash;
    }

    /**
     * Sets the timestamp of the block.
     *
     * @param timestamp New timestamp to be set.
     * @returns No explicit return value.
     */
    public void setTimestamp(Timestamp timestamp) {

        this.timestamp = timestamp;
    }

    /**
```

```
     * Creates a JSON representation of the block.
     *
     * @returns JSON representation of the block.
     */
    public String toString() {
        Block block_helper = new Block (index, timestamp, data, difficulty);
        block_helper.nonce = nonce;
        block_helper.setPreviousHash(previousHash);
        Gson gson = new Gson();
        return gson.toJson(block_helper);
    }

}
```

3.Task 0 BlockChain.java

```
package ds.Project3Task0;
import com.google.gson.Gson;
import java.io.UnsupportedEncodingException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.sql.Timestamp;
import java.util.ArrayList;
import java.util.Scanner;

/**
 * This Java code defines a BlockChain class for a simple blockchain. It offers methods
for adding blocks, verifying integrity, and repair.
 * The main method serves as a command-line interface for user interaction, enabling
actions like adding transactions and monitoring blockchain statistics.
 * */

//Work
Cited:https://www.andrew.cmu.edu/course/95-702/examples/javadoc/blockchaintask0/Blo
ckChain.html
//Self Learning:
// This class represents a simple BlockChain.(Some parts of the comments also cited from
the upper website.)
// I have also taken a look at Piazza and discussed some of the purposes of functions
with my classmates KW.

public class BlockChain {
    //This BlockChain has exactly three instance members - an ArrayList to hold Blocks
and a chain hash to hold a SHA256 hash of the most recently added Block. It also maintains
an instance variable holding the approximate number of hashes per second on this computer.
```

```
This constructor creates an empty ArrayList for Block storage.
    // This constructor sets the chain hash to the empty string and sets hashes per second
to 0.
    private final ArrayList<Block> block_chain;
    private String chainHash;
    private int hashes_per_second;


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


    /**
     * Get the chain hash.
     *
     * @return The chain hash as a string.
     */
    public String getChainHash() {

        return chainHash;
    }

    /**
     * Get the current timestamp.
     *
     * @return The current timestamp as a Timestamp object.
     */
    public Timestamp getTime() {
        //Work Cited:
https://www.tutorialspoint.com/java/lang/system_currenttimemillis.htm
        return new Timestamp(System.currentTimeMillis());
    }

    /**
     * Get the last block in the block chain.
     *
     * @return The last block as a Block object.
     */
    public Block getLastBlock() {

        return block_chain.get(block_chain.size()-1);
    }
```

```
    /**
     * Get the size of the block chain.
     *
     * @return The size of the block chain as an integer.
     */
    public int getChainSize() {

        return block_chain.size();
    }

    /**
     * Computes the number of hashes per second and sets the instance variable
hashesPerSecond.
     */
    public void computeHashesPerSecond() {
        //This method computes exactly 2 million hashes and times how long that process
takes. So, hashes per second is approximated as (2 million / number of seconds). It is
run on start up and sets the instance variable hashesPerSecond.
        // It uses a simple string - "00000000" to hash.
        Timestamp launch_time = getTime();
        int num = 0;
        for (int i = 0; i< 2000000; i++){
            // Work Cited:
https://github.com/CMU-Heinz-95702/Lab1-InstallationAndRaft
            try {
                MessageDigest md = MessageDigest.getInstance("SHA-256");
                byte[] bytes_info;
                md.update("00000000".getBytes("UTF-8"), 0, "00000000".length());
                bytes_info = md.digest();
                char[] HEX_ARRAY = "0123456789ABCDEF".toCharArray();
                char[] hexChars = new char[bytes_info.length * 2];
                for (int j = 0; j < bytes_info.length; j++) {
                    int v = bytes_info[j] & 0xFF;
                    hexChars[j * 2] = HEX_ARRAY[v >>> 4];
                    hexChars[j * 2 + 1] = HEX_ARRAY[v & 0x0F];
                }
            } catch (UnsupportedEncodingException | NoSuchAlgorithmException e) {
                throw new RuntimeException(e);
            }
            }
        Timestamp end_time = getTime();
        //Work
Cited:https://docs.oracle.com/javase/8/docs/api/java/sql/Timestamp.html
```

```
        hashes_per_second = (int) (2000000 / (end_time.getTime() -
launch_time.getTime())*1000.0);
    }

    /**
     * Get the number of hashes per second.
     *
     * @return The number of hashes per second as an integer.
     */
    public int getHashesPerSecond() {

        return hashes_per_second;
    }

    /**
     * Add a new block to the block chain.
     *
     * @param newBlock The new block to be added.
     */
    public void addBlock(Block newBlock) {
        //Within the block x, there is a previous hash field.
        newBlock.setPreviousHash(chainHash);
        block_chain.add(newBlock);
        //It is important to also maintain a hash of the most recently added block in
a chain hash.
        chainHash = newBlock.proofOfWork();
    }



    /**
     * Generate a JSON representation of the entire block chain.
     *
     * @return A JSON representation of the block chain as a string.
     */
    public String toString() {
        BlockChain blockchain_to_json = new BlockChain();
        for(int i = 0; i < getChainSize(); i++) {
            blockchain_to_json.block_chain.add(getBlock(i));
        }
        blockchain_to_json.hashes_per_second = getHashesPerSecond();
        blockchain_to_json.chainHash = getChainHash();
        Gson gson = new Gson();
        return gson.toJson(blockchain_to_json);
    }
```

```
    /**
     * Get a specific block from the block chain.
     *
     * @param i The index of the block to retrieve.
     * @return The block at the specified index as a Block object.
     */
    public Block getBlock(int i ) {

        return block_chain.get(i);
    }

    /**
     * Calculate the total difficulty of all blocks in the chain.
     *
     * @return The total difficulty as an integer.
     */
    public int getTotalDifficulty() {
        int sum_of_difficulty = 0;
        for (int i = 0; i < block_chain.size(); i++) {
            sum_of_difficulty += block_chain.get(i).getDifficulty();
        }
        return sum_of_difficulty;
    }

    /**
     * Calculate the total expected hashes required for the entire chain.
     *
     * @return The total expected hashes as a double.
     */
    public double getTotalExpectedHashes() {
        double sum_of_TotalExpectedHashes = 0;
        for(Block helper: block_chain) {
            sum_of_TotalExpectedHashes += Math.pow("0123456789ABCDEF".length(),
helper.getDifficulty());
        }
        return sum_of_TotalExpectedHashes;
    }

    /**
     * Check if the chain is valid and has not been tampered with.
     *
     * @return "True" if the chain is valid, "False" if not.
     */
```

```
    public String isChainValid() {
        Block current_block;
        String hash;
        String previous_hash = "";
        for (int i = 0; i < this.block_chain.size(); i++){
            current_block = this.block_chain.get(i);
            //In the case of the first block,
            // the initial value of preHash is an empty string "",
            // not an actual hash value.
            // Therefore, even if there is no previousHash in the first block,
            // the loop does not throw an error
            // because the function sets an empty string as the initial value,
            // which matches the Genesis block.
            if (!current_block.getPreviousHash().equals(previous_hash)){
//                System.out.printf("Improper hash on node %d Does not match
with %s\n",i, previous_hash);
                String error_message = "Improper hash on node " + i + " Does not match
with " + previous_hash;
                return "FALSE" +";" + error_message;
            }
            String begin_of_zero_amount = "0".repeat(current_block.getDifficulty());
            hash = current_block.calculateHash();
            if (!hash.startsWith(begin_of_zero_amount)){
//                System.out.printf("Improper hash on node %d Does not begin
with %s\n",i,begin_of_zero_amount);
                String error_message = "Improper hash on node " + i +  " Does not begin
with " + begin_of_zero_amount;
                return "FALSE" +";" + error_message;
            }
            previous_hash = hash;
        }
        if (!chainHash.equals(previous_hash)){
            return "FALSE" +";" + "Improper hash";
        }
        return "TRUE";

    }

    /**
     * Repair the chain by updating previous hashes and chain hash.
     */
    public void repairChain() {
        Block current_block;
        String previous_hash = "";
```

```
        for (int i = 0; i < this.block_chain.size(); i++){
            current_block = this.block_chain.get(i);
            current_block.setPreviousHash(previous_hash);
            previous_hash = current_block.proofOfWork();
        }
        this.chainHash = previous_hash;
    }


    public static void main(java.lang.String[] args) {
        //This routine acts as a test driver for your Blockchain.
        // It will begin by creating a BlockChain object and then adding the Genesis
block to the chain.
        // The Genesis block will be created with an empty string as the pervious hash
and a difficulty of 2.
        BlockChain main_block_chain = new BlockChain();
        main_block_chain.computeHashesPerSecond();
        Block genesisBlock = new Block(0, main_block_chain.getTime(), "Genesis", 2);
        main_block_chain.addBlock(genesisBlock);
        //main_block_chain.computeHashesPerSecond();
        Scanner scanner = new Scanner(System.in);
        Timestamp launch_time;
        Timestamp finished_time;
        while(true){
            System.out.println("0. View basic blockchain status.\n" +
                    "1. Add a transaction to the blockchain.\n" +
                    "2. Verify the blockchain.\n" +
                    "3. View the blockchain.\n" +
                    "4. Corrupt the chain.\n" +
                    "5. Hide the corruption by repairing the chain.\n" +
                    "6. Exit.\n");
            int user_choice = scanner.nextInt();

            if (user_choice == 0) {
                //If the user selects option 0, the program will display:
                // The number of blocks on the chain Difficulty of most recent block
                // The total difficulty for all blocks
                // Approximate hashes per second on this machine.
                // Expected total hashes required for the whole chain.
                // The computed nonce for most recent block. The chain hash (hash of
the most recent block).
                System.out.println("Current size of chain: " +
main_block_chain.getChainSize());
                System.out.println("Total difficulty for all blocks: " +
```

```
main_block_chain.getLastBlock().getDifficulty());
                System.out.println("Total difficulty for all blocks: " +
main_block_chain.getTotalDifficulty());
                System.out.println("Approximate hashes per second on this machine: "
+ main_block_chain.getHashesPerSecond());
                System.out.println("Expected total hashes required for the whole chain:
" + main_block_chain.getTotalExpectedHashes());
                System.out.println("Nonce for most recent block: " +
main_block_chain.getLastBlock().getNonce());
                System.out.println("Chain hash:" + main_block_chain.getChainHash());
            } else if (user_choice == 1) {
                //If the user selects option 1, the program will prompt for and then
read the difficulty level for this block.
                // It will then prompt for and then read a line of data from the user
(representing a transaction).
                // The program will then add a block containing that transaction to the
block chain.
                // The program will display the time it took to add this block.
                // Note: The first block added after Genesis has index 1. The second
has 2 and so on. The Genesis block is at position 0.
                System.out.println("Enter difficulty > 0");
                int difficulty = scanner.nextInt();
                System.out.println("Enter transaction");
                scanner.nextLine();
                String data_for_new_block = scanner.nextLine();
                Block helper_block = new Block(main_block_chain.getChainSize(),
main_block_chain.getTime(), data_for_new_block, difficulty);
                launch_time = main_block_chain.getTime();
                main_block_chain.addBlock(helper_block);
                finished_time = main_block_chain.getTime();
                System.out.println("Total execution time to add this block was " + (int)
(finished_time.getTime() - launch_time.getTime()) + " milliseconds");
            } else if (user_choice == 2) {
                //If the user selects option 2, then call the isChainValid method and
display the results.
                // It is important to note that this method will execute fast.
                // Blockchains are easy to validate but time consuming to modify.
                // Your program needs to display the number of milliseconds it took for
validate to run.
                launch_time = main_block_chain.getTime();
                String answer = main_block_chain.isChainValid();
                if(answer.equals("TRUE")){
                    System.out.println("Chain verification: " + answer);
                }
```

```
                else{
                    String[] result = answer.split(";");
                    System.out.println("Chain verification: " + result[0]);
                    System.out.println(result[1]);}
                finished_time = main_block_chain.getTime();
                System.out.println("Total execution time required to verify the chain
was " + (int) (finished_time.getTime() - launch_time.getTime()) + " milliseconds");
            } else if (user_choice == 3) {
                //If the user selects option 3, display the entire Blockchain contents
as a correctly formed JSON document. See www.json.org.
                System.out.println("View the Blockchain");
                System.out.println(main_block_chain.toString());
            } else if (user_choice == 4) {
                //If the user selects option 4, she wants to corrupt the chain.
                // Ask her for the block index (0..size-1) and ask her for the new data
that will be placed in the block.
                System.out.println("corrupt the Blockchain");
                System.out.println("Enter block ID of block to corrupt: ");
                int blockId = scanner.nextInt();
                System.out.println("Enter new data for block " + blockId);
                scanner.nextLine();
                String new_data = scanner.nextLine();
                Block corrupt_block = main_block_chain.block_chain.get(blockId);
                corrupt_block.setData(new_data);
                System.out.printf("Block %d now holds %s\n", blockId, new_data);
            } else if (user_choice == 5) {
                //If the user selects 5, she wants to repair the chain.
                // That is, she wants to recompute the proof of work for each node that
has become invalid - due perhaps, to an earlier selection of option 4.
                // The program begins at the Genesis block and checks each block in turn.
                // If any block is found to be invalid, it executes repair logic.
                launch_time = main_block_chain.getTime();
                main_block_chain.repairChain();
                finished_time = main_block_chain.getTime();
                System.out.println("Total execution time required to repair the chain
was " + (int) (finished_time.getTime() - launch_time.getTime()) + " milliseconds");
            } else if (user_choice == 6) {
                break;
            } else {
                System.out.println("Invalid option.");
            }

        }
    }
```

```
}
```

Requirement3: Within your comments in the main routine, you must describe how this system behaves as the difficulty level increases. Run some experiments by adding new blocks with increasing difficulties. Describe what you find. Be specific and quote some times. You need not employ a system clock.

To answer this question, I specifically tested with the following different difficulty-leveled transactions as well as corruptions happened at blocks with different difficult levels, similarly from the sample commands offered on the github, we can see that with the difficulty level increases, the time it takes to carry out the commands such as transaction as well as repair the chains becomes longer.

```
"C:\Program Files\Eclipse Adoptium\jdk-17.0.8.101-hotspot\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2023.2.1\lib\idea_rt.jar=59744:C:\Program Files\JetBrains\IntelliJ IDEA 2023.2.1\bin" -Dfile.encoding=UTF-8 -classpath C:\Users\Administrator\IdeaProjects\Project3Task0\target\classes;C:\Users\Administrator\.m2\repository\com\google\code\gson\gson\2.9.0\gson-2.9.0.jar ds.Project3Task0.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
Total difficulty for all blocks: 2
Total difficulty for all blocks: 2
Approximate hashes per second on this machine: 2317000
Expected total hashes required for the whole chain: 256.0
Nonce for most recent block: 399
Chain hash:008842E4D3DF15FEFA0B15C1A8C4F2E28B2AEF377EA78C2DD29A470F145090B2
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
1
Enter transaction
Chanel pays Dior 20 DSCoin
Total execution time to add this block 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.

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.

1
Enter difficulty > 0
2
Enter transaction
Chanel pays Louis 20 DSCoin
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 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.

4
corrupt the Blockchain
Enter block ID of block to corrupt:
1
Enter new data for block 1
Lauren pays Chanel 2 DSCoin
Block 1 now holds Lauren pays Chanel 2 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.

2
Chain verification: FALSE
Improper hash on node 1 Does not begin with 0
Total execution time required to verify the chain 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.

5
Total execution time required to repair 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
Chanel pays Parma 20 DSCoin
Total execution time to add this block was 160 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.

4
corrupt the Blockchain
Enter block ID of block to corrupt:
3
Enter new data for block 3
Mac pays Chanel 20 DSCoin
Block 3 now holds Mac pays Chanel 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.

2
Chain verification: FALSE
Improper hash on node 3 Does not begin with 0000
```

```
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.

5
Total execution time required to repair the chain was 59 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.

6

Process finished with exit code 0
```

Project3Task1

1. Task 1 Client Side Execution

```
"C:\Program Files\Eclipse Adoptium\jdk-17.0.8.101-hotspot\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2023.2.1\lib\idea_rt.jar=59633:C:\Program Files\JetBrains\IntelliJ IDEA 2023.2.1\bin" -Dfile.encoding=UTF-8 -classpath C:\Users\Administrator\IdeaProjects\Project3Task1\target\classes;C:\Users\Administrator\.m2\repository\com\google\code\gson\gson\2.9.0\gson-2.9.0.jar ds.Project3Task1.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.

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: 2139000
Expected total hashes required for the whole chain: 256.0
Nonce for most recent block: 75
Chain                                                                    hash:
00DBCC04F0B9AD3AF6AF8D311960FA6963DFC5700123B420724056AC3A7D6702
null
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
Mike pays Marty 100 DSCoin
Total execution time to add this block 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.

1
Enter difficulty > 0
2
Enter transaction
```

```
Marty pays Joe 50 DSCoin
Total execution time to add this block 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.

1
Enter difficulty > 0
2
Enter transaction
Joe pays Andy 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 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
{"block_chain":[{"index":0,"timestamp":"Oct 27, 2023, 11:34:02
PM","data":"Genesis","previousHash":"","nonce":75,"difficulty":2},{"i
ndex":1,"timestamp":"Oct 27, 2023, 11:34:29 PM","data":"Mike pays Marty
100
DSCoin","previousHash":"00DBCC04F0B9AD3AF6AF8D311960FA6963DFC5700123B
420724056AC3A7D6702","nonce":97,"difficulty":2},{"index":2,"timestamp
":"Oct 27, 2023, 11:34:35 PM","data":"Marty pays Joe 50
```

DSCoin","previousHash":"0051F5362834C31A6C1F425F0E3CBA53785FED0F47598
2647923E580A36DAEEE","nonce":117,"difficulty":2},{"index":3,"timestam
p":"Oct 27, 2023, 11:34:42 PM","data":"Joe pays Andy 10 DS
Coin","previousHash":"00773448E6F85BDBB82EB3FBF86FCCDB0CFFA18FC645C32
DF49BEEF316978221","nonce":138,"difficulty":2}],"chainHash":"0096ED5A
89160D1169D1A5366B7ECC0B168D092E2F8F6F9614A0971B4BCEEF40","hashes_per
_second":2139000}

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
Mike pays Marty 76 DSCoin
Block 1 now holds Mike pays Marty 76 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.

3
View the Blockchain

{"block_chain":[{"index":0,"timestamp":"Oct 27, 2023, 11:34:02
PM","data":"Genesis","previousHash":"","nonce":75,"difficulty":2},{"i
ndex":1,"timestamp":"Oct 27, 2023, 11:34:29 PM","data":"Mike pays Marty
76
DSCoin","previousHash":"00DBCC04F0B9AD3AF6AF8D311960FA6963DFC5700123B
420724056AC3A7D6702","nonce":97,"difficulty":2},{"index":2,"timestamp
":"Oct 27, 2023, 11:34:35 PM","data":"Marty pays Joe 50
DSCoin","previousHash":"0051F5362834C31A6C1F425F0E3CBA53785FED0F47598
2647923E580A36DAEEE","nonce":117,"difficulty":2},{"index":3,"timestam
p":"Oct 27, 2023, 11:34:42 PM","data":"Joe pays Andy 10 DS
Coin","previousHash":"00773448E6F85BDBB82EB3FBF86FCCDB0CFFA18FC645C32
DF49BEEF316978221","nonce":138,"difficulty":2}],"chainHash":"0096ED5A

89160D1169D1A5366B7ECC0B168D092E2F8F6F9614A0971B4BCEEF40","hashes_per
_second":2139000}

```
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 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.

5
Total execution time required to repair the chain was 9 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.

1
```

```
Enter difficulty > 0
4
Enter transaction
Andy pays Sean 25 DSCoin
Total execution time to add this block was 120 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: 2139000
Expected total hashes required for the whole chain: 66560.0
Nonce for most recent block: 84829
Chain                                                                hash:
000044B11299F8C78B19A0BA78F509AB1ECA0E62288BA9D38C69C56D765DF0D3
Total execution time to add this block was 120 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
5
Enter transaction
Xinyuan pays Marty 100 DSCoin
Total execution time to add this block was 659 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.
```

```
6

Process finished with exit code 0
```

2. Task 1 Server Side Execution

```
"C:\Program Files\Eclipse Adoptium\jdk-17.0.8.101-hotspot\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2023.2.1\lib\idea_rt.jar=59629:C:\Program Files\JetBrains\IntelliJ IDEA 2023.2.1\bin" -Dfile.encoding=UTF-8 -classpath C:\Users\Administrator\IdeaProjects\Project3Task1\target\classes;C:\Users\Administrator\.m2\repository\com\google\code\gson\gson\2.9.0\gson-2.9.0.jar ds.Project3Task1.ServerTCP
Blockchain server running
------------------------------------------------------------
We have a visitor
THE JSON REQUEST MESSAGE IS SHOWN HERE:
{"selection":0,"difficulty":0,"index":0}
User's choice: View current blockchain status
THE JSON RESPONSE MESSAGE IS SHOWN HERE :
{"size":1,"difficulty":2,"sum_of_difficulty":2,"totalHashes":256.0,"nonce":75,"chainHash":"00DBCC04F0B9AD3AF6AF8D311960FA6963DFC5700123B420724056AC3A7D6702","hash_per_second":2139000}
------------------------------------------------------------
We have a visitor
THE JSON REQUEST MESSAGE IS SHOWN HERE:
{"selection":1,"difficulty":2,"data":"Mike pays Marty 100 DSCoin","index":0}
User's choice: New Transaction(Adding a new block)
THE JSON RESPONSE MESSAGE IS SHOWN HERE :
{"size":1,"difficulty":2,"sum_of_difficulty":2,"totalHashes":256.0,"nonce":75,"chainHash":"00DBCC04F0B9AD3AF6AF8D311960FA6963DFC5700123B420724056AC3A7D6702","response":"Total execution time to add this block was 2 milliseconds","hash_per_second":2139000}
------------------------------------------------------------
We have a visitor
THE JSON REQUEST MESSAGE IS SHOWN HERE:
{"selection":1,"difficulty":2,"data":"Marty pays Joe 50 DSCoin","index":0}
User's choice: New Transaction(Adding a new block)
THE JSON RESPONSE MESSAGE IS SHOWN HERE :
{"size":1,"difficulty":2,"sum_of_difficulty":2,"totalHashes":256.0,"nonce":75,"chainHash":"00DBCC04F0B9AD3AF6AF8D311960FA6963DFC5700123B42
```

0724056AC3A7D6702","response":"Total execution time to add this block was 2 milliseconds","hash_per_second":2139000}

We have a visitor

THE JSON REQUEST MESSAGE IS SHOWN HERE:

{"selection":1,"difficulty":2,"data":"Joe pays Andy 10 DS Coin","index":0}

User's choice: New Transaction(Adding a new block)

THE JSON RESPONSE MESSAGE IS SHOWN HERE :

{"size":1,"difficulty":2,"sum_of_difficulty":2,"totalHashes":256.0,"nonce":75,"chainHash":"00DBCC04F0B9AD3AF6AF8D311960FA6963DFC5700123B420724056AC3A7D6702","response":"Total execution time to add this block was 3 milliseconds","hash_per_second":2139000}

We have a visitor

THE JSON REQUEST MESSAGE IS SHOWN HERE:

{"selection":2,"difficulty":2,"data":"Joe pays Andy 10 DS Coin","index":0}

User's choice: Blockchain Verification

Chain verification: TRUE

Total execution time required to verify the chain was 0 milliseconds

THE JSON RESPONSE MESSAGE IS SHOWN HERE :

{"size":1,"difficulty":2,"sum_of_difficulty":2,"totalHashes":256.0,"verify_result":"TRUE","nonce":75,"chainHash":"00DBCC04F0B9AD3AF6AF8D311960FA6963DFC5700123B420724056AC3A7D6702","response":"Total execution time required to verify the chain was 0 milliseconds","hash_per_second":2139000}

We have a visitor

THE JSON REQUEST MESSAGE IS SHOWN HERE:

{"selection":3,"difficulty":2,"data":"Joe pays Andy 10 DS Coin","index":0}

User's choice: Blockchain Details

THE JSON RESPONSE MESSAGE IS SHOWN HERE :

{"size":1,"difficulty":2,"sum_of_difficulty":2,"totalHashes":256.0,"verify_result":"TRUE","nonce":75,"chainHash":"00DBCC04F0B9AD3AF6AF8D311960FA6963DFC5700123B420724056AC3A7D6702","response":"{\"block_chain\":[{\"index\":0,\"timestamp\":\"Oct 27, 2023, 11:34:02 PM\",\"data\":\"Genesis\",\"previousHash\":\"\",\"nonce\":75,\"difficulty\":2},{\"index\":1,\"timestamp\":\"Oct 27, 2023, 11:34:29 PM\",\"data\":\"Mike pays Marty 100 DSCoin\",\"previousHash\":\"00DBCC04F0B9AD3AF6AF8D311960FA6963DFC5700123B420724056AC3A7D6702\",\"nonce\":97,\"difficulty\":2},{\"index\":2,\"timestamp\":\"Oct 27, 2023, 11:34:35 PM\",\"data\":\"Marty pays Joe 50

DSCoin\",\"previousHash\":\"0051F5362834C31A6C1F425F0E3CBA53785FED0F4
75982647923E580A36DAEEE\",\"nonce\":117,\"difficulty\":2},{\"index\":
3,\"timestamp\":\"Oct 27, 2023, 11:34:42 PM\",\"data\":\"Joe pays Andy
10 DS
Coin\",\"previousHash\":\"00773448E6F85BDBB82EB3FBF86FCCDB0CFFA18FC64
5C32DF49BEEF316978221\",\"nonce\":138,\"difficulty\":2}],\"chainHash\
":\"0096ED5A89160D1169D1A5366B7ECC0B168D092E2F8F6F9614A0971B4BCEEF40\
",\"hashes_per_second\":2139000}","hash_per_second":2139000}

We have a visitor

THE JSON REQUEST MESSAGE IS SHOWN HERE:

{"selection":4,"difficulty":2,"data":"Mike pays Marty 76
DSCoin","index":1}

User's choice: Corrupt the Blockchain

THE JSON RESPONSE MESSAGE IS SHOWN HERE :

{"size":1,"difficulty":2,"sum_of_difficulty":2,"totalHashes":256.0,"v
erify_result":"TRUE","nonce":75,"chainHash":"00DBCC04F0B9AD3AF6AF8D31
1960FA6963DFC5700123B420724056AC3A7D6702","response":"Block 1 now
holds Mike pays Marty 76 DSCoin","hash_per_second":2139000}

We have a visitor

THE JSON REQUEST MESSAGE IS SHOWN HERE:

{"selection":3,"difficulty":2,"data":"Mike pays Marty 76
DSCoin","index":1}

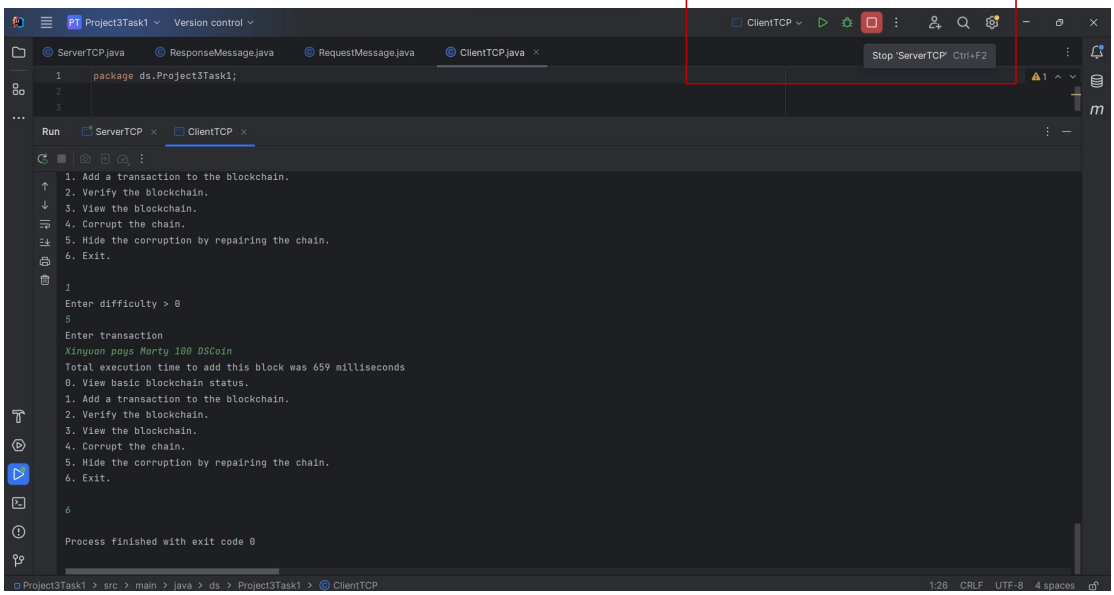
User's choice: Blockchain Details

THE JSON RESPONSE MESSAGE IS SHOWN HERE :

{"size":1,"difficulty":2,"sum_of_difficulty":2,"totalHashes":256.0,"v
erify_result":"TRUE","nonce":75,"chainHash":"00DBCC04F0B9AD3AF6AF8D31
1960FA6963DFC5700123B420724056AC3A7D6702","response":"{\"block_chain\
":[{\"index\":0,\"timestamp\":\"Oct 27, 2023, 11:34:02
PM\",\"data\":\"Genesis\",\"previousHash\":\"\",\"nonce\":75,\"diffic
ulty\":2},{\"index\":1,\"timestamp\":\"Oct 27, 2023, 11:34:29
PM\",\"data\":\"Mike pays Marty 76
DSCoin\",\"previousHash\":\"00DBCC04F0B9AD3AF6AF8D311960FA6963DFC5700
123B420724056AC3A7D6702\",\"nonce\":97,\"difficulty\":2},{\"index\":2,
\"timestamp\":\"Oct 27, 2023, 11:34:35 PM\",\"data\":\"Marty pays Joe 50
DSCoin\",\"previousHash\":\"0051F5362834C31A6C1F425F0E3CBA53785FED0F4
75982647923E580A36DAEEE\",\"nonce\":117,\"difficulty\":2},{\"index\":
3,\"timestamp\":\"Oct 27, 2023, 11:34:42 PM\",\"data\":\"Joe pays Andy
10 DS
Coin\",\"previousHash\":\"00773448E6F85BDBB82EB3FBF86FCCDB0CFFA18FC64
5C32DF49BEEF316978221\",\"nonce\":138,\"difficulty\":2}],\"chainHash\
":\"0096ED5A89160D1169D1A5366B7ECC0B168D092E2F8F6F9614A0971B4BCEEF40\
",\"hashes_per_second\":2139000}","hash_per_second":2139000}

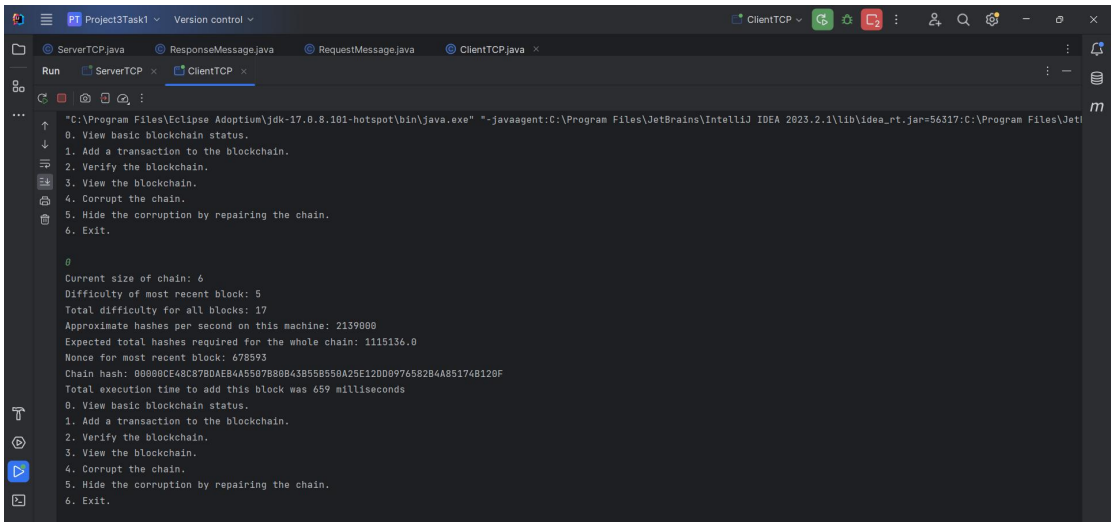
```
------------------------------------------------------------
We have a visitor
THE JSON REQUEST MESSAGE IS SHOWN HERE:
{"selection":2,"difficulty":2,"data":"Mike pays Marty 76
DSCoin","index":1}
User's choice: Blockchain Verification
Chain verification: FALSE
Improper hash on node 1 Does not begin with 00
Total execution time required to verify the chain was 0 milliseconds
THE JSON RESPONSE MESSAGE IS SHOWN HERE :
{"size":1,"difficulty":2,"sum_of_difficulty":2,"totalHashes":256.0,"v
erify_result":"FALSE","errorMessage":"Improper hash on node 1 Does not
begin with
00","nonce":75,"chainHash":"00DBCC04F0B9AD3AF6AF8D311960FA6963DFC5700
123B420724056AC3A7D6702","response":"Total execution time required to
verify the chain was 0 milliseconds","hash_per_second":2139000}
------------------------------------------------------------
We have a visitor
THE JSON REQUEST MESSAGE IS SHOWN HERE:
{"selection":5,"difficulty":2,"data":"Mike pays Marty 76
DSCoin","index":1}
User's choice: Blockchain Repair
THE JSON RESPONSE MESSAGE IS SHOWN HERE :
{"size":1,"difficulty":2,"sum_of_difficulty":2,"totalHashes":256.0,"v
erify_result":"FALSE","errorMessage":"Improper hash on node 1 Does not
begin with
00","nonce":75,"chainHash":"00DBCC04F0B9AD3AF6AF8D311960FA6963DFC5700
123B420724056AC3A7D6702","response":"Total execution time required to
repair the chain was 9 milliseconds","hash_per_second":2139000}
------------------------------------------------------------
We have a visitor
THE JSON REQUEST MESSAGE IS SHOWN HERE:
{"selection":2,"difficulty":2,"data":"Mike pays Marty 76
DSCoin","index":1}
User's choice: Blockchain Verification
Chain verification: TRUE
Total execution time required to verify the chain was 0 milliseconds
THE JSON RESPONSE MESSAGE IS SHOWN HERE :
{"size":1,"difficulty":2,"sum_of_difficulty":2,"totalHashes":256.0,"v
erify_result":"TRUE","errorMessage":"Improper hash on node 1 Does not
begin with
00","nonce":75,"chainHash":"00DBCC04F0B9AD3AF6AF8D311960FA6963DFC5700
123B420724056AC3A7D6702","response":"Total execution time required to
verify the chain was 0 milliseconds","hash_per_second":2139000}
```

```
------------------------------------------------------------
We have a visitor
THE JSON REQUEST MESSAGE IS SHOWN HERE:
{"selection":1,"difficulty":4,"data":"Andy pays Sean 25
DSCoin","index":1}
User's choice: New Transaction(Adding a new block)
THE JSON RESPONSE MESSAGE IS SHOWN HERE :
{"size":1,"difficulty":2,"sum_of_difficulty":2,"totalHashes":256.0,"v
erify_result":"TRUE","errorMessage":"Improper hash on node 1 Does not
begin with
00","nonce":75,"chainHash":"00DBCC04F0B9AD3AF6AF8D311960FA6963DFC5700
123B420724056AC3A7D6702","response":"Total execution time to add this
block was 120 milliseconds","hash_per_second":2139000}
------------------------------------------------------------
We have a visitor
THE JSON REQUEST MESSAGE IS SHOWN HERE:
{"selection":0,"difficulty":4,"data":"Andy pays Sean 25
DSCoin","index":1}
User's choice: View current blockchain status
THE JSON RESPONSE MESSAGE IS SHOWN HERE :
{"size":5,"difficulty":4,"sum_of_difficulty":12,"totalHashes":66560.0,
"verify_result":"TRUE","errorMessage":"Improper hash on node 1 Does not
begin with
00","nonce":84829,"chainHash":"000044B11299F8C78B19A0BA78F509AB1ECA0E
62288BA9D38C69C56D765DF0D3","response":"Total execution time to add this
block was 120 milliseconds","hash_per_second":2139000}
------------------------------------------------------------
We have a visitor
THE JSON REQUEST MESSAGE IS SHOWN HERE:
{"selection":1,"difficulty":5,"data":"Xinyuan pays Marty 100
DSCoin","index":1}
User's choice: New Transaction(Adding a new block)
THE JSON RESPONSE MESSAGE IS SHOWN HERE :
{"size":5,"difficulty":4,"sum_of_difficulty":12,"totalHashes":66560.0,
"verify_result":"TRUE","errorMessage":"Improper hash on node 1 Does not
begin with
00","nonce":84829,"chainHash":"000044B11299F8C78B19A0BA78F509AB1ECA0E
62288BA9D38C69C56D765DF0D3","response":"Total execution time to add this
block was 659 milliseconds","hash_per_second":2139000}
------------------------------------------------------------
```



We can see that when the client side exits, the server is still running, as is required by our github prompt:"If the client exits, the server will still handle new requests with the existing blockchain intact".

I therefore re-launch the client side to test, and select the 0 option to view the updates:



We can see from the outputs such as current size of chain and total difficulty, that the latest transaction we made in last launch of the client("Xinyuan pays Marty 100 DSCoin") is added. Therefore, we prove that "If the client exits, the server will still handle new requests with the existing blockchain intact".

As is mentioned in the github prompt, the clientTCP and serverTCP share the use of the RequestMessage and ResponseMessage, Therefore, I will also attached these two classes here:

3. RequestMessage

```
package ds.Project3Task1;


/**
 * A class to encapsulate the request message
```

```
 * */

//For this question, I have discussed with PW, XY for som function clarifications.
// Work Cited: Piazza

public class RequestMessage {
    private int selection;
    private int difficulty;
    private String data;
    private int index;

    /**
     * Default constructor for RequestMessage.
     */
    public RequestMessage() {
    }


    /**
     * Sets the selection value.
     *
     * @param selection The selection value to set.
     */
    public void setSelection(int selection) {

        this.selection = selection;
    }

    /**
     * Sets the difficulty value.
     *
     * @param difficulty The difficulty value to set.
     */
    public void setDifficulty(int difficulty) {

        this.difficulty = difficulty;
    }

    /**
     * Sets the data string.
     *
     * @param data The data string to set.
     */
    public void setData(String data) {
```

```
        this.data = data;
    }

    /**
     * Sets the index value.
     *
     * @param index The index value to set.
     */
    public void setIndex(int index) {

        this.index = index;
    }

    /**
     * Gets the selection value.
     *
     * @return The selection value.
     */
    public int getSelection() {

        return selection;
    }
    /**
     * Gets the difficulty value.
     *
     * @return The difficulty value.
     */
    public int getDifficulty() {

        return difficulty;
    }
    /**
     * Gets the data string.
     *
     * @return The data string.
     */
    public String getData() {

        return data;
    }
    /**
     * Gets the index value.
     *
```

```
     * @return The index value.
     */
    public int getIndex() {

        return index;
    }

}
```

4. ResponseMessage

```
package ds.Project3Task1;

import java.math.BigInteger;

/**
 * This is a response message class used to deal with the production of response message
 * */

//For this question, I have discussed with PW, XY for som function clarifications.
// Work Cited: Piazza

public class ResponseMessage {
    private int size;
    private int difficulty;
    private int sum_of_difficulty;
    private Double totalHashes;
    private String verify_result;
    private String errorMessage;
    private BigInteger nonce;
    private String chainHash;
    private String response;
    private int hash_per_second;

    /**
     * Get the hash per second.
     *
     */
    public int get_hash_per_second() {
        return hash_per_second;
    }

    /**
     * Set the hash per second.
```

```
     *
     */
    public void set_hash_per_second(int second) {
        this.hash_per_second = second;
    }

    public ResponseMessage() {
    }

    /**
     * Set the selection value.
     *
     * @param selection The selection value to set.
     */
    public void setSelection(int selection) {
    }
    /**
     * Get the size value.
     *
     * @return The size value.
     */
    public int get_chain_size() {

        return size;
    }
    /**
     * Set the size value.
     *
     * @param size The size value to set.
     */
    public void set_chain_size(int size) {

        this.size = size;
    }
    /**
     * Get the difficulty value.
     *
     * @return The difficulty value.
     */
    public int get_difficulty() {

        return difficulty;
    }
    /**
```

```
     * Set the difficulty value.
     *
     * @param level The diff value to set.
     */
    public void set_difficulty(int level) {
        this.difficulty = level;
    }
    /**
     * Get the totalDiff value.
     *
     * @return The totalDiff value.
     */
    public int get_sum_of_difficulty() {

        return sum_of_difficulty;
    }
    /**
     * Set the totalDiff value.
     *
     * @param total The totalDiff value to set.
     */
    public void set_sum_of_difficulty(int total) {
        this.sum_of_difficulty = total;
    }

    /**
     * Get the totalHashes value.
     *
     * @return The totalHashes value.
     */
    public Double getTotalHashes() {

        return totalHashes;
    }
    /**
     * Set the totalHashes value.
     *
     * @param totalHashes The totalHashes value to set.
     */
    public void setTotalHashes(Double totalHashes) {

        this.totalHashes = totalHashes;
    }
    /**
```

```
     * Get the latest Nonce value.
     *
     * @return The latest Nonce value.
     */
    public BigInteger getLatestNonce() {

        return nonce;
    }
    /**
     * Set the latest Nonce value.
     *
     * @param Nonce The latest Nonce value to set.
     */
    public void setLatestNonce(BigInteger Nonce) {

        this.nonce = Nonce;
    }
    /**
     * Get the chainHash value.
     *
     * @return The chainHash value.
     */
    public String getChainHash() {

        return chainHash;
    }
    /**
     * Set the chainHash value.
     *
     * @param chainHash The chainHash value to set.
     */
    public void setChainHash(String chainHash) {

        this.chainHash = chainHash;
    }
    /**
     * Get the response string.
     *
     * @return The response string.
     */
    public String getResponse() {

        return response;
    }
```

```
/**
 * Set the response string.
 *
 * @param response The response string to set.
 */
public void setResponse(String response) {

    this.response = response;
}
/**
 * Get the verification string.
 *
 * @return The verification string.
 */
public String getVerification() {

    return verify_result;
}
/**
 * Set the verification string.
 *
 * @param verification The verification string to set.
 */
public void setVerification(String verification) {

    this.verify_result = verification;
}

/**
 * Get the error message for the second option: blockchain verification.
 *
 */
public String getErrorMessage() {

    return errorMessage;
}
/**
 * Set the error message for the second option: blockchain verification.
 *
 */
public void setErrorMessage(String errorMessage) {

    this.errorMessage = errorMessage;
}
```

```
}
```

5. Task 1 Client Source Code

```
package ds.Project3Task1;


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

/**
 * This Java code file represents a TCP client for interacting with a blockchain server.
 * Users are presented with a menu of options to select from.
 * Based on the user's selection, the program constructs a request message and
communicates with the server using Gson for JSON serialization.
 * The server responds with a message, which is then processed and displayed to the user.
 **/
public class ClientTCP {
    private static Socket clientSocket;
    private static final Scanner readInput = new Scanner(System.in);
    private static final RequestMessage request_message = new RequestMessage();
    private static ResponseMessage response_message = new ResponseMessage();
    private static final Gson gson = new Gson();

    public static void main(String args[]) {
        try {
            int serverPort = 7777;
            clientSocket = new Socket("localhost", serverPort);
            while(true){
                System.out.println("0. View basic blockchain status.\n" +
                        "1. Add a transaction to the blockchain.\n" +
                        "2. Verify the blockchain.\n" +
                        "3. View the blockchain.\n" +
                        "4. Corrupt the chain.\n" +
                        "5. Hide the corruption by repairing the chain.\n" +
                        "6. Exit.\n");
                int option = Integer.parseInt(readInput.nextLine());
```

```
                if (option < 0 || option > 6) {
                    System.out.println("Invalid option.");
                }else{
                request_message.setSelection(option);
                if (option == 1) {
                    System.out.println("Enter difficulty > 0");
                    int difficulty = readInput.nextInt();
                    request_message.setDifficulty(difficulty);
                    System.out.println("Enter transaction");
                    readInput.nextLine();
                    String data = readInput.nextLine();
                    request_message.setData(data);
                } else if (option == 4) {
                    System.out.println("corrupt the Blockchain");
                    System.out.println("Enter block ID of block to corrupt");
                    int index = readInput.nextInt();
                    request_message.setIndex(index);
                    System.out.println("Enter new data for block " + index);
                    readInput.nextLine();
                    String corrupt_message = readInput.nextLine();
                    request_message.setData(corrupt_message);
                } else if (option == 6) {
                    break;
                }
                try {
                    //Work
Cited:https://github.com/CMU-Heinz-95702/Project-2-Client-Server
                    BufferedReader in = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
                    PrintWriter out = new PrintWriter(new BufferedWriter(new
OutputStreamWriter(clientSocket.getOutputStream())));
                    String data = gson.toJson(request_message);
                    out.println(data);
                    out.flush();
                    response_message = gson.fromJson(in.readLine(),
ResponseMessage.class);
                } catch (IOException e) {
                    System.out.println("IO Exception:" + e.getMessage());
                }
                if (option == 0) {
                    System.out.println("Current size of chain: " +
response_message.get_chain_size());
                    System.out.println("Difficulty of most recent block: " +
response_message.get_difficulty());
```

```
                    System.out.println("Total difficulty for all blocks: " +
response_message.get_sum_of_difficulty());
                    System.out.println("Approximate hashes per second on this machine:
" + response_message.get_hash_per_second());
                    System.out.println("Expected total hashes required for the whole
chain: " + response_message.getTotalHashes());
                    System.out.println("Nonce for most recent block: " +
response_message.getLatestNonce());
                    System.out.println("Chain hash: " +
response_message.getChainHash());
                } else if (option == 2) {
                    System.out.println("Chain verification: " +
response_message.getVerification());
                    if(!(response_message.getVerification().equals("TRUE"))){
                    System.out.println(response_message.getErrorMessage());}
                } else if (option == 3) {
                    System.out.println("View the Blockchain");
                }
                    System.out.println(response_message.getResponse());
                }
            }
        } catch (IOException e) {
            System.out.println("IO Exception:" + e.getMessage());
        } finally {
            try {
                if (clientSocket != null) {
                    clientSocket.close();
                }
            } catch (IOException ignored) {
            }
        }
    }


}
```

6. Task 1 Server Source Code

```
package ds.Project3Task1;

/**
 * This code represents a Java server that communicates with a blockchain client using
JSON messages.
```

```
 * The server maintains a blockchain and responds to client requests.
 * It listens on a specific port, receives JSON requests from the client, processes them,
and sends back JSON responses.
 *  */
import com.google.gson.Gson;

import java.io.*;
import java.net.ServerSocket;
import java.net.Socket;
import java.sql.Timestamp;
import java.util.Scanner;


public class ServerTCP {
    private static Socket clientSocket = null;
    private static int serverPort = 7777;
    private static ResponseMessage reply_message = new ResponseMessage();
    private static RequestMessage request_message = new RequestMessage();
    private static BlockChain block_chain;
    private static Gson gson = new Gson();

    public static void main(String[] args) {
        block_chain = new BlockChain();
        Block init = new Block(block_chain.getChainSize(), block_chain.getTime(),
"Genesis", 2);
        block_chain.addBlock(init);
        block_chain.computeHashesPerSecond();
        try{
            ServerSocket listenSocket = new ServerSocket(serverPort);
            System.out.println("Blockchain server running");
            //Work Cited:https://github.com/CMU-Heinz-95702/Project-2-Client-Server
            clientSocket = listenSocket.accept();
            Scanner in = new Scanner(clientSocket.getInputStream());
            PrintWriter out;
            out = new PrintWriter(new BufferedWriter(new
OutputStreamWriter(clientSocket.getOutputStream())));
            while (true) {

System.out.println("------------------------------------------------------------");
                if (in.hasNextLine()) {
                    String info = in.nextLine();
                    request_message = gson.fromJson(info, RequestMessage.class);
                    if(request_message.getSelection() != 6){
                        System.out.println("We have a visitor");
```

```
                    }
                    System.out.println("THE JSON REQUEST MESSAGE IS SHOWN HERE: " +
info);
                    option_activity(request_message.getSelection());
                    out.println(gson.toJson(reply_message));
                    out.flush();
                } else {
                    //Work
Cited:https://github.com/CMU-Heinz-95702/Project-2-Client-Server
                    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) {
            }
        }
    }


    /**
     * Performs specific actions based on the given user option, updates the reply
message, and prints details.
     *
     * @param option The user's selected option for blockchain operation.
     */
    public static void option_activity(int option) {
        Timestamp launch_time;
        Timestamp finished_time;
        String response;
        if (option == 0) {
            System.out.println("User's choice: View current blockchain status");
            reply_message.setSelection(0);
            reply_message.set_chain_size(block_chain.getChainSize());
            reply_message.setChainHash(block_chain.getChainHash());
            reply_message.setTotalHashes(block_chain.getTotalExpectedHashes());
            reply_message.set_sum_of_difficulty(block_chain.getTotalDifficulty());
```

```
reply_message.setLatestNonce(block_chain.getBlock(block_chain.getChainSize()-1).get
Nonce());

reply_message.set_difficulty(block_chain.getBlock(block_chain.getChainSize()-1).get
Difficulty());
            reply_message.set_hash_per_second(block_chain.getHashesPerSecond());
        } else if (option == 1) {
            System.out.println("User's choice: New Transaction(Adding a new block)");
            launch_time = block_chain.getTime();
            block_chain.addBlock(new Block(block_chain.getChainSize(),
block_chain.getTime(), request_message.getData(), request_message.getDifficulty()));
            finished_time = block_chain.getTime();
            response = "Total execution time to add this block was " + (int)
(finished_time.getTime() - launch_time.getTime()) +" milliseconds";
            reply_message.setSelection(1);
            reply_message.setResponse(response);
        } else if (option == 2) {
            System.out.println("User's choice: Blockchain Verification");
            launch_time = block_chain.getTime();
            String validation = block_chain.isChainValid();
            if (validation.equals("TRUE")) {
                reply_message.setVerification("TRUE");
                System.out.println("Chain verification: " + validation);
            } else {
                reply_message.setVerification(validation.split(";")[0]);
                reply_message.setErrorMessage(validation.split(";")[1]);
                System.out.println("Chain verification: " + validation.split(";")[0]);
                System.out.println(validation.split(";")[1]);
            }
            finished_time = block_chain.getTime();
            response = "Total execution time required to verify the chain was " + (int)
(finished_time.getTime() - launch_time.getTime()) + " milliseconds";
            System.out.println("Total execution time required to verify the chain was
" + (int) (finished_time.getTime() - launch_time.getTime()) + " milliseconds");
            reply_message.setResponse(response);
        } else if (option == 3) {
            System.out.println("User's choice: Blockchain Details");
            reply_message.setResponse(block_chain.toString());
        } else if (option == 4) {
            System.out.println("User's choice: Corrupt the Blockchain");
            int index = request_message.getIndex();
            String corrupt_message = request_message.getData();
            block_chain.getBlock(index).setData(corrupt_message);
            response = "Block " + index + " now holds " + corrupt_message;
```

```
            reply_message.setResponse(response);
        } else if (option == 5) {
            System.out.println("User's choice: Blockchain Repair");
            launch_time = block_chain.getTime();
            if (!block_chain.isChainValid().equals("TRUE")) {
                block_chain.repairChain();
            }
            finished_time = block_chain.getTime();
            response = "Total execution time required to repair the chain was " + (int)
(finished_time.getTime() - launch_time.getTime()) + " milliseconds";
            reply_message.setResponse(response);
        }
        System.out.println("THE JSON RESPONSE MESSAGE IS SHOWN HERE : " +
gson.toJson(reply_message));
    }
}
```