

AndrewID: muyut

Name: Muyu Tong

Task 0 Execution

```
1 0. View basic blockchain status.
2 1. Add a transaction to the blockchain.
3 2. Verify the blockchain.
4 3. View the blockchain.
5 4. corrupt the chain.
6 5. Hide the corruption by repairing the chain.
7 6. Exit
8 0
9 Current size of chain: 1
10 Difficulty of most recent block: 2
11 Total difficulty for all blocks: 2
12 Approximate hashes per second on this machine: 2234
13 Expected total hashes required for the whole chain: 256.000000
14 Nonce for most recent block: 500
15 Chain hash: 00C5FEA351D4678902F296E17675AA9D355C845F1F07A7DA0E4087728E70919C
16 0. View basic blockchain status.
17 1. Add a transaction to the blockchain.
18 2. Verify the blockchain.
19 3. View the blockchain.
20 4. corrupt the chain.
21 5. Hide the corruption by repairing the chain.
22 6. Exit
23 1
24 Enter difficulty > 0
25 2
26 Enter transaction
27 Alice pays Bob 100 DScoin
28 Total execution time to add this block was 28 milliseconds
29 0. View basic blockchain status.
30 1. Add a transaction to the blockchain.
31 2. Verify the blockchain.
32 3. View the blockchain.
33 4. corrupt the chain.
34 5. Hide the corruption by repairing the chain.
35 6. Exit
36 1
37 Enter difficulty > 0
38 2
39 Enter transaction
40 Bob pays Carol 50 DScoin
41 Total execution time to add this block was 22 milliseconds
42 0. View basic blockchain status.
```

```

43 1. Add a transaction to the blockchain.
44 2. Verify the blockchain.
45 3. View the blockchain.
46 4. corrupt the chain.
47 5. Hide the corruption by repairing the chain.
48 6. Exit
49 1
50 Enter difficulty > 0
51 2
52 Enter transaction
53 Carol pays Andy 10 DScoin
54 Total execution time to add this block was 26 milliseconds
55 0. View basic blockchain status.
56 1. Add a transaction to the blockchain.
57 2. Verify the blockchain.
58 3. View the blockchain.
59 4. corrupt the chain.
60 5. Hide the corruption by repairing the chain.
61 6. Exit
62 2
63 Total execution time to add this block was 1 milliseconds
64 Chain verification: TRUE
65 0. View basic blockchain status.
66 1. Add a transaction to the blockchain.
67 2. Verify the blockchain.
68 3. View the blockchain.
69 4. corrupt the chain.
70 5. Hide the corruption by repairing the chain.
71 6. Exit
72 3
73 View the block chain
74 Blockchain: {chain: [{index: 0, timestamp: 2022-03-19 16:29:34.895, data: '', difficulty: 2,
75   previousHash: '', nonce: 500}
76   , {index: 1, timestamp: 2022-03-19 16:29:45.629, data: 'Alice pays Bob 100 DScoin',
   difficulty: 2, previousHash:
   '00C5FEA351D4678902F296E17675AA9D355C845F1F07A7DA0E4087728E70919C', nonce: 19}
77   , {index: 2, timestamp: 2022-03-19 16:30:23.917, data: 'Bob pays Carol 50 DScoin',
   difficulty: 2, previousHash:
   '00746F2541D68122AD760E6917A85DBB5B4BB7B3B12E49107A94E93AF6563B24', nonce: 116}
78   , {index: 3, timestamp: 2022-03-19 16:30:45.4, data: 'Carol pays Andy 10 DScoin', difficulty:
79   2, previousHash: '00F3CED4D03CD9B66B6E39C04A8DBB5E8B24807DC03F492897101A84EC293728', nonce:
80   85}
81   ], chainHash: '006427688CFDE049B2C1AFAB910284BD99BA83547D8BD0329ED4A1A144BB9551'}
82 0. View basic blockchain status.
83 1. Add a transaction to the blockchain.
84 2. Verify the blockchain.
85 3. View the blockchain.
86 4. corrupt the chain.
87 5. Hide the corruption by repairing the chain.
88 6. Exit
89 4

```

```

87 Corrupt the Blockchain
88 Enter block ID of block to corrupt
89 1
90 Enter new data for block 1
91 Alice pays Bob 76 DScoin
92 Block 1 now holds Alice pays Bob 76 DScoin
93 0. View basic blockchain status.
94 1. Add a transaction to the blockchain.
95 2. Verify the blockchain.
96 3. View the blockchain.
97 4. corrupt the chain.
98 5. Hide the corruption by repairing the chain.
99 6. Exit
100 3
101 View the block chain
102 Blockchain: {chain: [{index: 0, timestamp: 2022-03-19 16:29:34.895, data: '', difficulty: 2,
103 previousHash: '', nonce: 500}
, {index: 1, timestamp: 2022-03-19 16:29:45.629, data: 'Alice pays Bob 76 DScoin',
difficulty: 2, previousHash:
'00C5FEA351D4678902F296E17675AA9D355C845F1F07A7DA0E4087728E70919C', nonce: 19}
104 , {index: 2, timestamp: 2022-03-19 16:30:23.917, data: 'Bob pays Carol 50 DScoin',
difficulty: 2, previousHash:
'00746F2541D68122AD760E6917A85DBB5B4BB7B3B12E49107A94E93AF6563B24', nonce: 116}
105 , {index: 3, timestamp: 2022-03-19 16:30:45.4, data: 'Carol pays Andy 10 DScoin', difficulty:
2, previousHash: '00F3CED4D03CD9B66B6E39C04A8DBB5E8B24807DC03F492897101A84EC293728', nonce:
85}
106 ], chainHash: '006427688CFDE049B2C1AFAB910284BD99BA83547D8BD0329ED4A1A144BB9551'}
107 0. View basic blockchain status.
108 1. Add a transaction to the blockchain.
109 2. Verify the blockchain.
110 3. View the blockchain.
111 4. corrupt the chain.
112 5. Hide the corruption by repairing the chain.
113 6. Exit
114 2
115 Improper hash on node 1 Does not begin with 00
116 Total execution time to verify the chain was 7 milliseconds
117 Improper hash on node 2 Previous Hash is incorrect
118 Total execution time to verify the chain was 8 milliseconds
119 Chain verification: FALSE
120 0. View basic blockchain status.
121 1. Add a transaction to the blockchain.
122 2. Verify the blockchain.
123 3. View the blockchain.
124 4. corrupt the chain.
125 5. Hide the corruption by repairing the chain.
126 6. Exit
127 5
128 Total execution time required to repair the chain was 5 milliseconds
129 0. View basic blockchain status.
130 1. Add a transaction to the blockchain.

```

```

131 2. Verify the blockchain.
132 3. View the blockchain.
133 4. corrupt the chain.
134 5. Hide the corruption by repairing the chain.
135 6. Exit
136 2
137 Total execution time to add this block was 9 milliseconds
138 Chain verification: TRUE
139 0. View basic blockchain status.
140 1. Add a transaction to the blockchain.
141 2. Verify the blockchain.
142 3. View the blockchain.
143 4. corrupt the chain.
144 5. Hide the corruption by repairing the chain.
145 6. Exit
146 1
147 Enter difficulty > 0
148 4
149 Enter transaction
150 Andy pays Sean 25 DScoin
151 Total execution time to add this block was 187 milliseconds
152 0. View basic blockchain status.
153 1. Add a transaction to the blockchain.
154 2. Verify the blockchain.
155 3. View the blockchain.
156 4. corrupt the chain.
157 5. Hide the corruption by repairing the chain.
158 6. Exit
159 0
160 Current size of chain: 5
161 Difficulty of most recent block: 4
162 Total difficulty for all blocks: 12
163 Approximate hashes per second on this machine: 2234
164 Expected total hashes required for the whole chain: 66560.000000
165 Nonce for most recent block: 31761
166 Chain hash: 00008E64732C7E00F814F2A0EB9ED25CE3CA5439699F69BD17448F8C60B38807
167 0. View basic blockchain status.
168 1. Add a transaction to the blockchain.
169 2. Verify the blockchain.
170 3. View the blockchain.
171 4. corrupt the chain.
172 5. Hide the corruption by repairing the chain.
173 6. Exit
174 6
175
176 Process finished with exit code 0

```

Task 0 Block.java

```

1  import java.math.BigInteger;

```

```

2  import java.sql.Timestamp;
3
4  public class Block {
5      int index;
6      Timestamp timestamp;
7      String data;
8      int difficulty;
9      String previousHash;
10     BigInteger nonce;
11
12     public Block(int index, Timestamp timestamp, String data, int difficulty) {
13         this.index = index;
14         this.timestamp = timestamp;
15         this.data = data;
16         this.difficulty = difficulty;
17         this.previousHash = "";
18         this.nonce = new BigInteger("0");
19     }
20
21     public String calculateHash() {
22         String hash = "";
23         hash =
Utils.getHashString(index+timestamp.toString()+data+previousHash+nonce+difficulty);
24         return hash;
25     }
26
27     public BigInteger getNonce() {
28         return nonce;
29     }
30
31     public String proofOfWork() {
32         String hash = calculateHash();
33         while (!hash.matches("0".repeat(difficulty)+".*")) {
34             nonce = nonce.add(new BigInteger("1"));
35             hash = calculateHash();
36         }
37         return hash;
38     }
39
40     public int getDifficulty() {
41         return difficulty;
42     }
43
44     public void setDifficulty(int difficulty) {
45         this.difficulty = difficulty;
46     }
47
48     @Override
49     public String toString() {
50         return "{" +
51             "index: " + index +

```

```

52         ", timestamp: " + timestamp +
53         ", data: '" + data + '\'' +
54         ", difficulty: " + difficulty +
55         ", previousHash: '" + previousHash + '\'' +
56         ", nonce: " + nonce +
57         "}\n";
58     }
59
60     public String getPreviousHash() {
61         return previousHash;
62     }
63
64     public void setPreviousHash(String previousHash) {
65         this.previousHash = previousHash;
66     }
67
68     public int getIndex() {
69         return index;
70     }
71
72     public void setIndex(int index) {
73         this.index = index;
74     }
75
76     public Timestamp getTimestamp() {
77         return timestamp;
78     }
79
80     public void setTimestamp(Timestamp timestamp) {
81         this.timestamp = timestamp;
82     }
83
84     public String getData() {
85         return data;
86     }
87
88     public void setData(String data) {
89         this.data = data;
90     }
91 }
92

```

Task 0 Blockchain.java

```

1  import java.nio.charset.StandardCharsets;
2  import java.security.MessageDigest;
3  import java.security.NoSuchAlgorithmException;
4  import java.sql.Timestamp;
5  import java.util.ArrayList;
6  import java.util.List;

```

```

7  import java.util.Scanner;
8
9  public class Blockchain {
10     List<Block> chain;
11     String chainHash;
12     int hashPerSec;
13
14     public Blockchain() {
15         chain = new ArrayList<>();
16         chainHash = "";
17     }
18
19     public void addBlock(Block newBlock) {
20         String testHash = newBlock.calculateHash();
21         if (newBlock.previousHash.equals(chainHash) &&
testHash.matches("0".repeat(newBlock.difficulty)+".*")) {
22             chain.add(newBlock);
23             chainHash = testHash;
24         } else {
25             System.out.println("Illegal block!");
26         }
27     }
28
29     public String getChainHash() {
30         return chainHash;
31     }
32
33     public Timestamp getTime() {
34         return new Timestamp(System.currentTimeMillis());
35     }
36
37     public Block getLatestBlock() {
38         return chain.get(chain.size()-1);
39     }
40
41     public int getChainSize() {
42         return chain.size();
43     }
44
45     public void computeHashesPerSecond() {
46         String s = "00000000";
47         byte[] bytes = s.getBytes(StandardCharsets.UTF_8);
48         byte[] hs;
49         long start = System.currentTimeMillis();
50         MessageDigest md = null;
51         try {
52             md = MessageDigest.getInstance("SHA-256");
53         } catch (NoSuchAlgorithmException e) {
54             e.printStackTrace();
55         }
56         for (int i = 0; i < 2000000; i++) {

```

```

57         hs = md.digest(bytes);
58     }
59     long end = System.currentTimeMillis();
60     hashPerSec = (int) (2000000/(end-start));
61 }
62
63 public int getHashesPerSecond() {
64     return hashPerSec;
65 }
66
67 @Override
68 public String toString() {
69     return "Blockchain: {" +
70         "chain: " + chain +
71         ", chainHash: '" + chainHash + '\'' +
72         '}';
73 }
74
75 public Block getBlock(int i) {
76     return chain.get(i);
77 }
78
79 public int getTotalDifficulty() {
80     int total = 0;
81     for (Block block: chain) {
82         total += block.getDifficulty();
83     }
84     return total;
85 }
86
87 public double getTotalExpectedHashes() {
88     double total = 0;
89     for (Block block: chain) {
90         total += Math.pow(16, block.getDifficulty());
91     }
92
93     return total;
94 }
95
96 public String isChainValid() {
97     long start = System.currentTimeMillis();
98     long end;
99     String block1Hash = ""; // previous block hash
100    String block2Hash = ""; // current block hash
101    for (Block block : chain) { // each round test if current block hash match previous
one
102        block2Hash = block.calculateHash();
103        if (!block2Hash.matches("0".repeat(block.getDifficulty()) + ".*")){
104            System.out.printf("Improper hash on node %d Does not begin with " +
"0".repeat(block.getDifficulty()) + "\n", block.getIndex());
105            end = System.currentTimeMillis();

```



```

106         System.out.printf("Total execution time to verify the chain was %d
milliseconds\n", end-start);
107     } else if (!block.getPreviousHash().equals(block1Hash)) {
108         System.out.printf("Improper hash on node %d Previous Hash is incorrect\n",
block.getIndex());
109         end = System.currentTimeMillis();
110         System.out.printf("Total execution time to verify the chain was %d
milliseconds\n", end-start);
111         return "FALSE";
112     }
113     block1Hash = block2Hash;
114 }
115
116 //check if the last block hash equals chain hash
117 if (!block1Hash.equals(chainHash)) {
118     System.out.println("The chain hash is incorrect");
119     return "FALSE";
120 }
121 end = System.currentTimeMillis();
122 System.out.printf("Total execution time to add this block was %d milliseconds\n",
end-start);
123 return "TRUE";
124 }
125
126 public void repairChain() {
127     long start = System.currentTimeMillis();
128     String block1Hash = ""; // previous block hash
129     String block2Hash = ""; // current block hash
130     boolean repair = false;
131     int i;
132     for (i = 0; i < chain.size(); i++) { // each round test if current block hash match
previous one
133         Block block = chain.get(i);
134         block2Hash = block.calculateHash();
135
136         if (!block2Hash.matches("0".repeat(block.getDifficulty()) + ".*")
|| !block.getPreviousHash().equals(block1Hash)) {
137             break;
138         }
139     }
140     block1Hash = block2Hash; //set previous hash
141 }
142     if (i == chain.size() && !block1Hash.equals(chainHash)) i = chain.size()-1; // the
last block hash not equals to chain hash
143     else if (i == chain.size()) return; // the chain is correct
144
145     for (; i < chain.size(); i++) { //repair the chain from the corrupt block
146         Block block = chain.get(i);
147         block.setPreviousHash(i==0?"":chain.get(i-1).calculateHash()); // previous hash
is empty for the first block
148         block.proofOfWork();
149         chain.set(i, block);

```

```

150     }
151
152     chainHash = chain.get(chain.size()-1).calculateHash(); // update the chain hash in
the end
153     long end = System.currentTimeMillis();
154     System.out.printf("Total execution time required to repair the chain was %d
milliseconds\n", end-start);
155 }
156
157 public static void main(String[] args) {
158     Blockchain chain = new Blockchain();
159     Block gen = new Block(0, new Timestamp(System.currentTimeMillis()), "", 2);
160     gen.proofOfWork();
161     chain.addBlock(gen);
162     chain.computeHashesPerSecond();
163
164     Scanner scanner = new Scanner(System.in);
165     int option;
166     do {
167         System.out.println("0. View basic blockchain status.\n" +
168             "1. Add a transaction to the blockchain.\n" +
169             "2. Verify the blockchain.\n" +
170             "3. View the blockchain.\n" +
171             "4. corrupt the chain.\n" +
172             "5. Hide the corruption by repairing the chain.\n" +
173             "6. Exit");
174         option = Integer.parseInt(scanner.nextLine());
175         switch (option) {
176             case 0:
177                 System.out.printf("Current size of chain: %d\n", chain.getChainSize());
178                 System.out.printf("Difficulty of most recent block: %d\n",
chain.getLatestBlock().getDifficulty());
179                 System.out.printf("Total difficulty for all blocks: %d\n",
chain.getTotalDifficulty());
180                 System.out.printf("Approximate hashes per second on this machine: %d\n",
chain.getHashesPerSecond());
181                 System.out.printf("Expected total hashes required for the whole chain:
%f\n", chain.getTotalExpectedHashes());
182                 System.out.printf("Nonce for most recent block: %s\n",
chain.getLatestBlock().getNonce());
183                 System.out.printf("Chain hash: %s\n", chain.getChainHash());
184                 break;
185             case 1:
186                 System.out.println("Enter difficulty > 0");
187                 int diff = Integer.parseInt(scanner.nextLine());
188                 System.out.println("Enter transaction");
189                 String transaction = scanner.nextLine();
190                 long start = System.currentTimeMillis();
191                 Block newBlock = new Block(chain.getChainSize(), new
Timestamp(System.currentTimeMillis()), transaction, diff);
192                 newBlock.setPreviousHash(chain.chainHash);

```

```

193         newBlock.proofOfWork();
194         chain.addBlock(newBlock);
195         long end = System.currentTimeMillis();
196         System.out.printf("Total execution time to add this block was %d
milliseconds\n", end-start);
197         break;
198     case 2:
199         String res = chain.isChainValid();
200         System.out.println("Chain verification: " + res);
201         break;
202     case 3:
203         System.out.println("View the block chain");
204         System.out.println(chain);
205         break;
206     case 4:
207         System.out.println("Corrupt the Blockchain");
208         System.out.println("Enter block ID of block to corrupt");
209         int id = Integer.parseInt(scanner.nextLine());
210         System.out.printf("Enter new data for block %d\n", id);
211         String d = scanner.nextLine();
212         chain.getBlock(id).setData(d);
213         System.out.printf("Block %d now holds %s\n", id, d);
214         break;
215     case 5:
216         chain.repairChain();
217         break;
218     default:
219         break;
220     }
221     } while (option != 6);
222 }
223 }
224

```

Task 1 Client Side Execution

```

1  Client running
2  0. View basic blockchain status.
3  1. Add a transaction to the blockchain.
4  2. Verify the blockchain.
5  3. View the blockchain.
6  4. corrupt the chain.
7  5. Hide the corruption by repairing the chain.
8  6. Exit
9  0
10 {selection:0, size:1,
    chainHash:'000F82ED0C896150FF2FC2D19F8E110EC440C008FBDA19B88E5767545F1BC27F',
    totalHashes:256.0, totalDiff:2, recentNonce:61, hps:2188,
11  response:''}
12 0. View basic blockchain status.

```

```
13 1. Add a transaction to the blockchain.
14 2. Verify the blockchain.
15 3. View the blockchain.
16 4. corrupt the chain.
17 5. Hide the corruption by repairing the chain.
18 6. Exit
19 1
20 Enter difficulty > 0
21 2
22 Enter transaction
23 Alice pays Bob 100 DScoin
24 {selection:1, size:0, chainHash:'null', totalHashes:0.0, totalDiff:0, recentNonce:null,
    hps:0,
25 response:'Total execution time to add this block was 32 milliseconds'}
26 0. View basic blockchain status.
27 1. Add a transaction to the blockchain.
28 2. Verify the blockchain.
29 3. View the blockchain.
30 4. corrupt the chain.
31 5. Hide the corruption by repairing the chain.
32 6. Exit
33 1
34 Enter difficulty > 0
35 2
36 Enter transaction
37 Bob pays Carol 50 DScoin
38 {selection:1, size:0, chainHash:'null', totalHashes:0.0, totalDiff:0, recentNonce:null,
    hps:0,
39 response:'Total execution time to add this block was 21 milliseconds'}
40 0. View basic blockchain status.
41 1. Add a transaction to the blockchain.
42 2. Verify the blockchain.
43 3. View the blockchain.
44 4. corrupt the chain.
45 5. Hide the corruption by repairing the chain.
46 6. Exit
47 1
48 Enter difficulty > 0
49 2
50 Enter transaction
51 Carol pays Andy 10DScoin
52 {selection:1, size:0, chainHash:'null', totalHashes:0.0, totalDiff:0, recentNonce:null,
    hps:0,
53 response:'Total execution time to add this block was 29 milliseconds'}
54 0. View basic blockchain status.
55 1. Add a transaction to the blockchain.
56 2. Verify the blockchain.
57 3. View the blockchain.
58 4. corrupt the chain.
59 5. Hide the corruption by repairing the chain.
60 6. Exit
```

```

61 2
62 {selection:2, size:0, chainHash:'null', totalHashes:0.0, totalDiff:0, recentNonce:null,
hps:0,
63 response:'Chain verification: TRUE
64 Total execution time to verify the chain was 3 milliseconds'}
65 0. View basic blockchain status.
66 1. Add a transaction to the blockchain.
67 2. Verify the blockchain.
68 3. View the blockchain.
69 4. corrupt the chain.
70 5. Hide the corruption by repairing the chain.
71 6. Exit
72 3
73 {selection:3, size:0, chainHash:'null', totalHashes:0.0, totalDiff:0, recentNonce:null,
hps:0,
74 response:'BlockChain: {chain: [{index: 0, timestamp: 2022-03-19 16:53:27.798, data: '',
difficulty: 2, previousHash: '', nonce: 61}
75 , {index: 1, timestamp: 2022-03-19 16:53:47.658, data: 'Alice pays Bob 100 DScoin',
difficulty: 2, previousHash:
'000F82ED0C896150FF2FC2D19F8E110EC440C008FBDA19B88E5767545F1BC27F', nonce: 104}
76 , {index: 2, timestamp: 2022-03-19 16:53:56.887, data: 'Bob pays Carol 50 DScoin',
difficulty: 2, previousHash:
'00072139DA6872E12DDE1D160BD8FD674C12A53D1D5752E1318D289C4A147E27', nonce: 135}
77 , {index: 3, timestamp: 2022-03-19 16:54:07.05, data: 'Carol pays Andy 10DScoin', difficulty:
2, previousHash: '00ECA036511A5B77D2192D0673E2F02A04C1C41F8C3EB82399D66F7E5A73FD90', nonce:
234}
78 ], chainHash: '00614422F0012102AAEC04501BB7448C6CAD1B79CA4C54A83870126D81CEAD9C'}}}
79 0. View basic blockchain status.
80 1. Add a transaction to the blockchain.
81 2. Verify the blockchain.
82 3. View the blockchain.
83 4. corrupt the chain.
84 5. Hide the corruption by repairing the chain.
85 6. Exit
86 4
87 Corrupt the Blockchain
88 Enter block ID of block to corrupt
89 1
90 Enter new data for block 1
91 Alice pays Bob 76 DScoin
92 {selection:4, size:0, chainHash:'null', totalHashes:0.0, totalDiff:0, recentNonce:null,
hps:0,
93 response:'Block 1 now holds Alice pays Bob 76 DScoin'}
94 0. View basic blockchain status.
95 1. Add a transaction to the blockchain.
96 2. Verify the blockchain.
97 3. View the blockchain.
98 4. corrupt the chain.
99 5. Hide the corruption by repairing the chain.
100 6. Exit
101 3

```

```

102 {selection:3, size:0, chainHash:'null', totalHashes:0.0, totalDiff:0, recentNonce:null,
103 hps:0,
104 response:'BlockChain: {chain: [{index: 0, timestamp: 2022-03-19 16:53:27.798, data: '',
105 difficulty: 2, previousHash: '', nonce: 61}
106 , {index: 1, timestamp: 2022-03-19 16:53:47.658, data: 'Alice pays Bob 76 DScoin',
107 difficulty: 2, previousHash:
108 '000F82ED0C896150FF2FC2D19F8E110EC440C008FBDA19B88E5767545F1BC27F', nonce: 104}
109 , {index: 2, timestamp: 2022-03-19 16:53:56.887, data: 'Bob pays Carol 50 DScoin',
110 difficulty: 2, previousHash:
111 '00072139DA6872E12DDE1D160BD8FD674C12A53D1D5752E1318D289C4A147E27', nonce: 135}
112 , {index: 3, timestamp: 2022-03-19 16:54:07.05, data: 'Carol pays Andy 10DScoin', difficulty:
113 2, previousHash: '00ECA036511A5B77D2192D0673E2F02A04C1C41F8C3EB82399D66F7E5A73FD90', nonce:
114 234}
115 ], chainHash: '00614422F0012102AACE04501BB7448C6CAD1B79CA4C54A83870126D81CEAD9C'}}}
116 0. View basic blockchain status.
117 1. Add a transaction to the blockchain.
118 2. Verify the blockchain.
119 3. View the blockchain.
120 4. corrupt the chain.
121 5. Hide the corruption by repairing the chain.
122 6. Exit
123 2
124 {selection:2, size:0, chainHash:'null', totalHashes:0.0, totalDiff:0, recentNonce:null,
125 hps:0,
126 response:'Chain verification: FALSE
127 Total execution time to verify the chain was 11 milliseconds'}
128 0. View basic blockchain status.
129 1. Add a transaction to the blockchain.
130 2. Verify the blockchain.
131 3. View the blockchain.
132 4. corrupt the chain.
133 5. Hide the corruption by repairing the chain.
134 6. Exit
135 5
136 {selection:5, size:0, chainHash:'null', totalHashes:0.0, totalDiff:0, recentNonce:null,
137 hps:0,
138 response:'Total execution time required to repair the chain was 45 milliseconds'}
139 0. View basic blockchain status.
140 1. Add a transaction to the blockchain.
141 2. Verify the blockchain.
142 3. View the blockchain.
143 4. corrupt the chain.
144 5. Hide the corruption by repairing the chain.
145 6. Exit
146 2
147 {selection:2, size:0, chainHash:'null', totalHashes:0.0, totalDiff:0, recentNonce:null,
148 hps:0,
149 response:'Chain verification: TRUE
150 Total execution time to verify the chain was 9 milliseconds'}
151 0. View basic blockchain status.
152 1. Add a transaction to the blockchain.

```

```

142 2. Verify the blockchain.
143 3. View the blockchain.
144 4. corrupt the chain.
145 5. Hide the corruption by repairing the chain.
146 6. Exit
147 1
148 Enter difficulty > 0
149 4
150 Enter transaction
151 Andy pays Sean 25 DSCoin
152 {selection:1, size:0, chainHash:'null', totalHashes:0.0, totalDiff:0, recentNonce:null,
    hps:0,
153 response:'Total execution time to add this block was 192 milliseconds'}
154 0. View basic blockchain status.
155 1. Add a transaction to the blockchain.
156 2. Verify the blockchain.
157 3. View the blockchain.
158 4. corrupt the chain.
159 5. Hide the corruption by repairing the chain.
160 6. Exit
161 6
162 {selection:6, size:0, chainHash:'null', totalHashes:0.0, totalDiff:0, recentNonce:null,
    hps:0,
163 response:''}
164
165 Process finished with exit code 0
166

```

Task 1 Server Side Execution

```

1 Blockchain server running
2 We have a visitor
3 Setting response to
4 {"selection":0,"size":1,"chainHash":"000F82ED0C896150FF2FC2D19F8E110EC440C008FBDA19B88E5767545
  F1BC27F","totalHashes":256.0,"totalDiff":2,"recentNonce":61,"hps":2188,"response":""}
5 Adding a block
6 Setting response to Total execution time to add this block was 32 milliseconds
7 {"selection":1,"size":0,"totalHashes":0.0,"totalDiff":0,"hps":0,"response":"Total execution
  time to add this block was 32 milliseconds"}
8 Adding a block
9 Setting response to Total execution time to add this block was 21 milliseconds
10 {"selection":1,"size":0,"totalHashes":0.0,"totalDiff":0,"hps":0,"response":"Total execution
  time to add this block was 21 milliseconds"}
11 Adding a block
12 Setting response to Total execution time to add this block was 29 milliseconds
13 {"selection":1,"size":0,"totalHashes":0.0,"totalDiff":0,"hps":0,"response":"Total execution
  time to add this block was 29 milliseconds"}
14 Chain verification: TRUE
15 Setting response to Chain verification: TRUE
16 Total execution time to verify the chain was 3 milliseconds

```

```
17 {"selection":2,"size":0,"totalHashes":0.0,"totalDiff":0,"hps":0,"response":"Chain
18 verification: TRUE\nTotal execution time to verify the chain was 3 milliseconds"}
19 View the Blockchain
19 Setting response to BlockChain: {chain: [{index: 0, timestamp: 2022-03-19 16:53:27.798, data:
20 '', difficulty: 2, previousHash: '', nonce: 61}
20 , {index: 1, timestamp: 2022-03-19 16:53:47.658, data: 'Alice pays Bob 100 DScoin',
20 difficulty: 2, previousHash:
20 '000F82ED0C896150FF2FC2D19F8E110EC440C008FBDA19B88E5767545F1BC27F', nonce: 104}
21 , {index: 2, timestamp: 2022-03-19 16:53:56.887, data: 'Bob pays Carol 50 DScoin', difficulty:
21 2, previousHash: '00072139DA6872E12DDE1D160BD8FD674C12A53D1D5752E1318D289C4A147E27', nonce:
21 135}
22 , {index: 3, timestamp: 2022-03-19 16:54:07.05, data: 'Carol pays Andy 10DScoin', difficulty:
22 2, previousHash: '00ECA036511A5B77D2192D0673E2F02A04C1C41F8C3EB82399D66F7E5A73FD90', nonce:
22 234}
23 ], chainHash: '00614422F0012102AAE04501BB7448C6CAD1B79CA4C54A83870126D81CEAD9C'}
24 {"selection":3,"size":0,"totalHashes":0.0,"totalDiff":0,"hps":0,"response":"BlockChain:
24 {chain: [{index: 0, timestamp: 2022-03-19 16:53:27.798, data: \u0027\u0027, difficulty: 2,
24 previousHash: \u0027\u0027, nonce: 61}\n, {index: 1, timestamp: 2022-03-19 16:53:47.658, data:
24 \u0027Alice pays Bob 100 DScoin\u0027, difficulty: 2, previousHash:
24 \u0027000F82ED0C896150FF2FC2D19F8E110EC440C008FBDA19B88E5767545F1BC27F\u0027, nonce: 104}\n,
24 {index: 2, timestamp: 2022-03-19 16:53:56.887, data: \u0027Bob pays Carol 50 DScoin\u0027,
24 difficulty: 2, previousHash:
24 \u002700072139DA6872E12DDE1D160BD8FD674C12A53D1D5752E1318D289C4A147E27\u0027, nonce: 135}\n,
24 {index: 3, timestamp: 2022-03-19 16:54:07.05, data: \u0027Carol pays Andy 10DScoin\u0027,
24 difficulty: 2, previousHash:
24 \u002700ECA036511A5B77D2192D0673E2F02A04C1C41F8C3EB82399D66F7E5A73FD90\u0027, nonce: 234}\n],
24 chainHash: \u002700614422F0012102AAE04501BB7448C6CAD1B79CA4C54A83870126D81CEAD9C\u0027"}
25 Corrupt the Blockchain
26 Setting response to Block 1 now holds Alice pays Bob 76 DScoin
27 {"selection":4,"size":0,"totalHashes":0.0,"totalDiff":0,"hps":0,"response":"Block 1 now holds
27 Alice pays Bob 76 DScoin"}
28 View the Blockchain
29 Setting response to BlockChain: {chain: [{index: 0, timestamp: 2022-03-19 16:53:27.798, data:
29 '', difficulty: 2, previousHash: '', nonce: 61}
30 , {index: 1, timestamp: 2022-03-19 16:53:47.658, data: 'Alice pays Bob 76 DScoin', difficulty:
30 2, previousHash: '000F82ED0C896150FF2FC2D19F8E110EC440C008FBDA19B88E5767545F1BC27F', nonce:
30 104}
31 , {index: 2, timestamp: 2022-03-19 16:53:56.887, data: 'Bob pays Carol 50 DScoin', difficulty:
31 2, previousHash: '00072139DA6872E12DDE1D160BD8FD674C12A53D1D5752E1318D289C4A147E27', nonce:
31 135}
32 , {index: 3, timestamp: 2022-03-19 16:54:07.05, data: 'Carol pays Andy 10DScoin', difficulty:
32 2, previousHash: '00ECA036511A5B77D2192D0673E2F02A04C1C41F8C3EB82399D66F7E5A73FD90', nonce:
32 234}
33 ], chainHash: '00614422F0012102AAE04501BB7448C6CAD1B79CA4C54A83870126D81CEAD9C'}
```



```

34  {"selection":3,"size":0,"totalHashes":0.0,"totalDiff":0,"hps":0,"response":"BlockChain:
    {chain: [{index: 0, timestamp: 2022-03-19 16:53:27.798, data: \u0027\u0027, difficulty: 2,
    previousHash: \u0027\u0027, nonce: 61}\n, {index: 1, timestamp: 2022-03-19 16:53:47.658, data:
    \u0027Alice pays Bob 76 DScoin\u0027, difficulty: 2, previousHash:
    \u0027000F82ED0C896150FF2FC2D19F8E110EC440C008FBDA19B88E5767545F1BC27F\u0027, nonce: 104}\n,
    {index: 2, timestamp: 2022-03-19 16:53:56.887, data: \u0027Bob pays Carol 50 DScoin\u0027,
    difficulty: 2, previousHash:
    \u002700072139DA6872E12DDE1D160BD8FD674C12A53D1D5752E1318D289C4A147E27\u0027, nonce: 135}\n,
    {index: 3, timestamp: 2022-03-19 16:54:07.05, data: \u0027Carol pays Andy 10DScoin\u0027,
    difficulty: 2, previousHash:
    \u002700ECA036511A5B77D2192D0673E2F02A04C1C41F8C3EB82399D66F7E5A73FD90\u0027, nonce: 234}\n],
    chainHash: \u002700614422F0012102ACE04501BB7448C6CAD1B79CA4C54A83870126D81CEAD9C\u0027"}
35  Improper hash on node 1 Does not begin with 00
36  Improper hash on node 2 Previous Hash is incorrect
37  Chain verification: FALSE
38  Setting response to Chain verification: FALSE
39  Total execution time to verify the chain was 11 milliseconds
40  {"selection":2,"size":0,"totalHashes":0.0,"totalDiff":0,"hps":0,"response":"Chain
    verification: FALSE\nTotal execution time to verify the chain was 11 milliseconds"}
41  Repairing the entire chain
42  Setting response to Total execution time required to repair the chain was 45 milliseconds
43  {"selection":5,"size":0,"totalHashes":0.0,"totalDiff":0,"hps":0,"response":"Total execution
    time required to repair the chain was 45 milliseconds"}
44  Chain verification: TRUE
45  Setting response to Chain verification: TRUE
46  Total execution time to verify the chain was 9 milliseconds
47  {"selection":2,"size":0,"totalHashes":0.0,"totalDiff":0,"hps":0,"response":"Chain
    verification: TRUE\nTotal execution time to verify the chain was 9 milliseconds"}
48  Adding a block
49  Setting response to Total execution time to add this block was 192 milliseconds
50  {"selection":1,"size":0,"totalHashes":0.0,"totalDiff":0,"hps":0,"response":"Total execution
    time to add this block was 192 milliseconds"}
51  Setting response to
52  {"selection":6,"size":0,"totalHashes":0.0,"totalDiff":0,"hps":0,"response":""}

```

Task 1 Client Source Code

```

1  import com.google.gson.Gson;
2
3  import java.io.*;
4  import java.net.Socket;
5  import java.util.Scanner;
6
7  //https://github.com/CMU-Heinz-95702/lab4-http-server/tree/master/Example-Socket-Code
8  public class BlockchainClient {
9      public void init() throws IOException {
10         System.out.println("Client running");
11         Gson gson = new Gson();
12         Socket socket = new Socket("localhost", 7777);

```

```

13     BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
14     PrintWriter out = new PrintWriter(new BufferedWriter(new
OutputStreamWriter(socket.getOutputStream())));
15     Scanner scanner = new Scanner(System.in);
16     int option;
17     do {
18         RequestMessage req = new RequestMessage();
19         System.out.println("0. View basic blockchain status.\n" +
20             "1. Add a transaction to the blockchain.\n" +
21             "2. Verify the blockchain.\n" +
22             "3. View the blockchain.\n" +
23             "4. corrupt the chain.\n" +
24             "5. Hide the corruption by repairing the chain.\n" +
25             "6. Exit");
26         option = Integer.parseInt(scanner.nextLine());
27         switch (option) {
28             case 0:
29                 req.setSelection(0);
30                 break;
31             case 1:
32                 System.out.println("Enter difficulty > 0");
33                 int diff = Integer.parseInt(scanner.nextLine());
34                 System.out.println("Enter transaction");
35                 String transaction = scanner.nextLine();
36                 req = new RequestMessage();
37                 req.setSelection(1);
38                 req.setDifficulty(diff);
39                 req.setMessage(transaction);
40                 break;
41             case 2:
42                 req.setSelection(2);
43                 break;
44             case 3:
45                 req.setSelection(3);
46                 break;
47             case 4:
48                 System.out.println("Corrupt the Blockchain");
49                 System.out.println("Enter block ID of block to corrupt");
50                 int id = Integer.parseInt(scanner.nextLine());
51                 System.out.printf("Enter new data for block %d\n", id);
52                 String d = scanner.nextLine();
53                 req.setSelection(4);
54                 req.setId(id);
55                 req.setMessage(d);
56                 break;
57             case 5:
58                 req.setSelection(5);
59                 break;
60             case 6:
61                 req.setSelection(6);

```

```

62         break;
63     default:
64         break;
65     }
66     out.println(gson.toJson(req));
67     out.flush();
68     ResponseMessage res = gson.fromJson(in.readLine(), ResponseMessage.class);
69     System.out.println(res);
70     } while (option != 6);
71 }
72 public static void main(String[] args) {
73     BlockchainClient blockchainClient = new BlockchainClient();
74     try {
75         blockchainClient.init();
76     } catch (IOException e) {
77         e.printStackTrace();
78     }
79 }
80 }

```

Task 1 Server Source Code

```

1  import com.google.gson.Gson;
2
3  import java.io.BufferedWriter;
4  import java.io.IOException;
5  import java.io.OutputStreamWriter;
6  import java.io.PrintWriter;
7  import java.net.ServerSocket;
8  import java.net.Socket;
9  import java.sql.Timestamp;
10 import java.util.Scanner;
11
12 //https://github.com/CMU-Heinz-95702/lab4-http-server/tree/master/Example-Socket-Code
13 public class BlockchainServer {
14     Blockchain chain;
15     Gson gson;
16     public BlockchainServer() {
17         gson = new Gson();
18     }
19
20     public ResponseMessage exec(RequestMessage req) {
21         int selection = req.selection;
22         ResponseMessage res = new ResponseMessage();
23         res.setSelection(req.getSelection());
24         String resMes = "";
25         long start;
26         long end;
27         switch (selection) {
28             case 0:

```

```

29         res.setChainHash(chain.getChainHash());
30         res.setHps(chain.getHashesPerSecond());
31         res.setRecentNonce(chain.getLatestBlock().getNonce());
32         res.setSize(chain.getChainSize());
33         res.setTotalDiff(chain.getTotalDifficulty());
34         res.setTotalHashes(chain.getTotalExpectedHashes());
35         break;
36     case 1:
37         System.out.println("Adding a block");
38         start = System.currentTimeMillis();
39         Block newBlock = new Block(chain.getChainSize(), new
Timestamp(System.currentTimeMillis()),
40             req.getMessage(), req.getDifficulty());
41         newBlock.setPreviousHash(chain.chainHash);
42         newBlock.proofOfWork();
43         chain.addBlock(newBlock);
44         end = System.currentTimeMillis();
45         resMes = "Total execution time to add this block was " + (end-start)+"
milliseconds";
46         break;
47     case 2:
48         start = System.currentTimeMillis();
49         resMes = chain.isChainValid();
50         resMes = "Chain verification: " + resMes;
51         System.out.println(resMes);
52         end = System.currentTimeMillis();
53         resMes += "\n" + "Total execution time to verify the chain was " + (end-start)+"
milliseconds";
54         break;
55     case 3:
56         System.out.println("View the Blockchain");
57         resMes = chain.toString();
58         break;
59     case 4:
60         System.out.println("Corrupt the Blockchain");
61         int id = req.getId();
62         String d = req.getMessage();
63         chain.getBlock(id).setData(d);
64         resMes = "Block "+id+" now holds "+d;
65         break;
66     case 5:
67         start = System.currentTimeMillis();
68         System.out.println("Repairing the entire chain");
69         chain.repairChain();
70         end = System.currentTimeMillis();
71         resMes = "Total execution time required to repair the chain was " + (end-
start)+" milliseconds";
72         break;
73     default:
74         break;
75 }

```

```

76     System.out.println("Setting response to " + resMes);
77     res.setResponse(resMes);
78     System.out.println(gson.toJson(res));
79     return res;
80 }
81
82 public void init() {
83     Socket clientSocket = null;
84     chain = new Blockchain();
85     try {
86         int serverPort = 7777;
87         ServerSocket listenSocket = new ServerSocket(serverPort);
88         System.out.println("Blockchain server running");
89         while (true) {
90             String request;
91             clientSocket = listenSocket.accept();
92             Scanner scanner = new Scanner(clientSocket.getInputStream());
93             PrintWriter outToSocket = new PrintWriter(new BufferedWriter(new
OutputStreamWriter(clientSocket.getOutputStream())));
94             System.out.println("We have a visitor");
95             do {
96                 request = scanner.nextLine();
97                 RequestMessage req = gson.fromJson(request, RequestMessage.class);
98                 ResponseMessage res = exec(req);
99                 outToSocket.println(gson.toJson(res));
100                outToSocket.flush();
101                if (req.getSelection() == 6) break;
102            } while (scanner.hasNextLine());
103        }
104    } catch (IOException e) {
105        e.printStackTrace();
106    }
107 }
108 public static void main(String[] args) {
109     BlockchainServer blockchainServer = new BlockchainServer();
110     blockchainServer.init();
111 }
112 }
113

```