Practical No. 4

Name: Bhairavi Narendra Rewatkar

Subject: Blockchain Technology Laboratory

Roll No.: DMET1221006

Date: 02/12/2024

Title: Proof of Work (PoW) Implementation

Aim: Implement proof of work to simulate mining in java.

Source Code:

HashUtil.java

```
import java.security.MessageDigest;

public class HashUtil {
    public static String applySha256(String input) {
        try {
            MessageDigest digest = MessageDigest.getInstance("SHA-256");
            byte[] hash = digest.digest(input.getBytes("UTF-8"));
            StringBuilder hexString = new StringBuilder();
            for (byte b : hash) {
                String hex = Integer.toHexString(0xff & b);
                if (hex.length() == 1) hexString.append('0');
                hexString.append(hex);
            }
            return hexString.toString();
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }
}
```

Block.java

```
public class Block {
    private int index;
    private long timestamp;
    private String data;
    private String previousHash;
    private String currentHash;
    private int nonce;

    // Constructor
    public Block(int index, String data, String previousHash) {
        this.index = index;
        this.data = data;
        this.previousHash = previousHash;
        this.timestamp = System.currentTimeMillis();
        this.nonce = 0;
        this.currentHash = calculateHash();
    }

    // Method to calculate hash
    public String calculateHash() {
        String content = index + Long.toString(timestamp) + data + previousHash + nonce;
        return HashUtil.applySha256(content);
    }

    // Method to mine the block
    public void mineBlock(int difficulty) {
        String target = new String(new char[difficulty]).replace('\0', '0'); // Target hash
        while (!currentHash.substring(0, difficulty).equals(target)) {
            nonce++;
            currentHash = calculateHash();
        }
        System.out.println("Block mined! Hash: " + currentHash);
```

```
}

// Getters
public String getHash() {
    return currentHash;
}

public String getPreviousHash() {
    return previousHash;
}

public int getIndex() {
    return index;
}

public String getData() {
    return data;
}

public int getNonce() {
    return nonce;
}

// toString method for block representation
@Override
public String toString() {
    return "Block{" +
            "index=" + index +
            ", timestamp=" + timestamp +
            ", data='" + data + '\'' +
            ", previousHash='" + previousHash + '\'' +
            ", currentHash='" + currentHash + '\'' +
            ", nonce=" + nonce +
            '}';
    }
}
```

Blockchain.java

```
import java.util.ArrayList;
import java.util.List;

public class Blockchain {
    private List<Block> chain; // List to store blocks
    private int difficulty;    // Difficulty for mining

    // Constructor
    public Blockchain(int difficulty) {
        this.chain = new ArrayList<>();
        this.difficulty = difficulty;
        chain.add(createGenesisBlock()); // Add the genesis block
    }

    // Create the genesis block
    public Block createGenesisBlock() {
        return new Block(0, "Genesis Block", "0");
    }

    // Get the last block in the chain
    public Block getLastBlock() {
        return chain.get(chain.size() - 1);
    }

    // Add a new block to the chain
    public void addBlock(String data) {
        Block previousBlock = getLastBlock();
        Block newBlock = new Block(previousBlock.getIndex() + 1, data, previousBlock.getHash());
        System.out.println("Mining block " + newBlock.getIndex() + "...");
        newBlock.mineBlock(difficulty);
        chain.add(newBlock);
    }

    // Print the entire blockchain
    public void printBlockchain() {
        for (Block block : chain) {
            System.out.println(block);
        }
    }
}
```

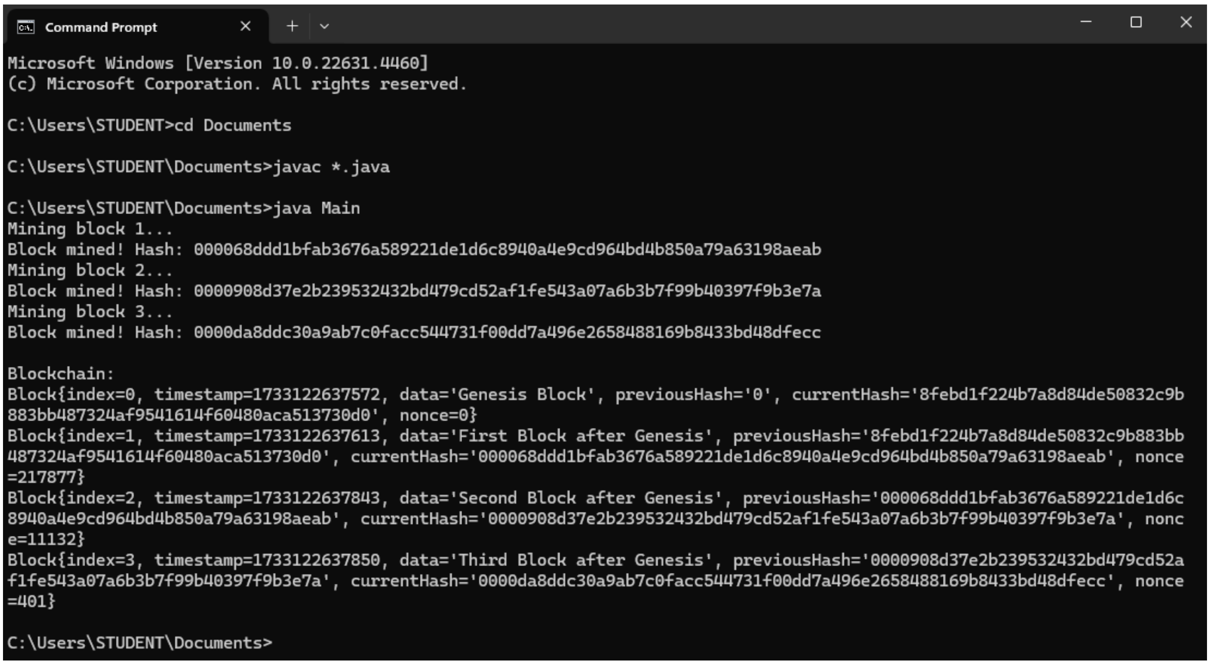
Main.java

```
public class Main {
    public static void main(String[] args) {
        int difficulty = 4; // Number of leading zeros required in the hash
        Blockchain blockchain = new Blockchain(difficulty);

        blockchain.addBlock("First Block after Genesis");
        blockchain.addBlock("Second Block after Genesis");
        blockchain.addBlock("Third Block after Genesis");

        System.out.println("\nBlockchain:");
        blockchain.printBlockchain();
    }
}
```

Output:



```
Microsoft Windows [Version 10.0.22631.4460]
(c) Microsoft Corporation. All rights reserved.

C:\Users\STUDENT>cd Documents

C:\Users\STUDENT\Documents>javac *.java

C:\Users\STUDENT\Documents>java Main
Mining block 1...
Block mined! Hash: 000068ddd1bfab3676a589221de1d6c8940a4e9cd964bd4b850a79a63198aeab
Mining block 2...
Block mined! Hash: 0000908d37e2b239532432bd479cd52af1fe543a07a6b3b7f99b40397f9b3e7a
Mining block 3...
Block mined! Hash: 0000da8ddc30a9ab7c0facc544731f00dd7a496e2658488169b8433bd48dfecc

Blockchain:
Block{index=0, timestamp=1733122637572, data='Genesis Block', previousHash='0', currentHash='8febd1f224b7a8d84de50832c9b
883bb487324af9541614f60480aca513730d0', nonce=0}
Block{index=1, timestamp=1733122637613, data='First Block after Genesis', previousHash='8febd1f224b7a8d84de50832c9b883bb
487324af9541614f60480aca513730d0', currentHash='000068ddd1bfab3676a589221de1d6c8940a4e9cd964bd4b850a79a63198aeab', nonce
=217877}
Block{index=2, timestamp=1733122637843, data='Second Block after Genesis', previousHash='000068ddd1bfab3676a589221de1d6c
8940a4e9cd964bd4b850a79a63198aeab', currentHash='0000908d37e2b239532432bd479cd52af1fe543a07a6b3b7f99b40397f9b3e7a', nonc
e=11132}
Block{index=3, timestamp=1733122637850, data='Third Block after Genesis', previousHash='0000908d37e2b239532432bd479cd52a
f1fe543a07a6b3b7f99b40397f9b3e7a', currentHash='0000da8ddc30a9ab7c0facc544731f00dd7a496e2658488169b8433bd48dfecc', nonce
=401}

C:\Users\STUDENT\Documents>
```