Practical No.11

Name: Bhairavi Narendra Rewatkar

Roll No.: DMET1221006

Subject: Blockchain Technology Laboratory

Date: 20/01/2025

Title: Blockchain Consensus Algorithm (Proof-of-Stake)

Aim: Write a program to implement a Proof-of-Stake (PoS) consensus algorithm.

Source Code:

Validator.java

```
public class Validator {
    private String id;
    private double stake;
    public Validator(String id, double stake) {
        this.id = id;
        this.stake = stake;}
    public String getId() {
        return id;}
    public double getStake() {
        return stake;  }}
```

Block.java

```
public class Block {
    private String previousHash;
    private String hash;
    private String data;
    public Block(String data, String previousHash) {
        this.data = data;
        this.previousHash = previousHash;
        this.hash = HashUtil.applySHA256(previousHash + data);    }
    public String getHash() {
        return this.hash; }
    @Override
    public String toString() {
```

```
        return "Block{" +
                "previousHash='" + previousHash + '\'' +
                ", hash='" + hash + '\'' +
                ", data='" + data + '\'' +
                '}'; }}
```

HashUtil.java

```
import java.security.MessageDigest;
public class HashUtil {
    public static String applySHA256(String input) {
        try {
            MessageDigest digest = MessageDigest.getInstance("SHA-256");
            byte[] hashBytes = digest.digest(input.getBytes("UTF-8"));
            StringBuilder hexString = new StringBuilder();
            for (byte b : hashBytes) {
                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);        }    }}
```

Blockchain.java

```
import java.util.ArrayList;
import java.util.List;
import java.util.Random;
public class Blockchain {
    private List<Block> chain;
    private List<Validator> validators;
    private double totalStake;
    public Blockchain(List<Validator> validators) {
        this.chain = new ArrayList<>();
        this.validators = validators;
        this.totalStake = 0;
```

```
        // Calculate total stake
        for (Validator validator : validators) {
            totalStake += validator.getStake();  }
        // Add the genesis block to the blockchain
        chain.add(createGenesisBlock());  }
    private Block createGenesisBlock() {
        return new Block("Genesis Block", "0"); }
    public void addBlock() {
        Validator selectedValidator = selectValidatorBasedOnStake();
        System.out.println("Validator " + selectedValidator.getId() + " is selected to create a block.");
        Block newBlock = new Block("Block data", chain.get(chain.size() - 1).getHash());
        chain.add(newBlock);    }
    private Validator selectValidatorBasedOnStake() {
        Random random = new Random();
        double randomStake = random.nextDouble() * totalStake;
        double cumulativeStake = 0;
        for (Validator validator : validators) {
            cumulativeStake += validator.getStake();
            if (cumulativeStake >= randomStake) {
                return validator;}     }
        return null; // This case won't happen if the totalStake is calculated correctly    }
    public void printChain() {
        for (Block block : chain) {
            System.out.println(block);  }   }}
```

Main.java

```
import java.util.ArrayList;
import java.util.List;
public class Main {
    public static void main(String[] args) {
        // Creating a list of validators with their respective stakes
        List<Validator> validators = new ArrayList<>();
        validators.add(new Validator("Validator1", 10));
```

```
        validators.add(new Validator("Validator2", 20));
        validators.add(new Validator("Validator3", 30));
        // Initializing the blockchain with validators
        Blockchain blockchain = new Blockchain(validators);
        // Adding blocks to the blockchain
        for (int i = 0; i < 5; i++) {
    blockchain.addBlock();
        }
        // Printing the blockchain
        blockchain.printChain();
    }
}
```

Output: