

## **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:

```

Microsoft Windows [Version 10.0.22631.4751]
(c) Microsoft Corporation. All rights reserved.

C:\Users\STUDENT>cd Desktop

C:\Users\STUDENT\Desktop>javac Validator.java Block.java HashUtil.java Blockchain.java Main.java

C:\Users\STUDENT\Desktop>java Main.java
Validator Validator2 is selected to create a block.
Validator Validator2 is selected to create a block.
Validator Validator3 is selected to create a block.
Validator Validator2 is selected to create a block.
Validator Validator2 is selected to create a block.
Block{previousHash='0', hash='075c27741a3506846368fa6e5b3477f85b31ceee71a5716e2f12b40fa21d23aa', data='Genesis Block'}
Block{previousHash='075c27741a3506846368fa6e5b3477f85b31ceee71a5716e2f12b40fa21d23aa', hash='a3b01de5cc1931c0582cca9f7da7143fc7fc13c6f4e56e4bc821185d4acbe1a1', data='Block data'}
Block{previousHash='a3b01de5cc1931c0582cca9f7da7143fc7fc13c6f4e56e4bc821185d4acbe1a1', hash='29323e376edee44496124c0e06d30f61630f34d4e778dda0b726a173b4e278d8', hash='b82c5c8e951a56579291aa88e7a07d522670f26793ba367cf9c096ad60f6170d', data='Block data'}
Block{previousHash='29323e376edee44496124c0e06d30f61630f34d4e778dda0b726a173b4e278d8', hash='b82c5c8e951a56579291aa88e7a07d522670f26793ba367cf9c096ad60f6170d', data='Block data'}
Block{previousHash='b82c5c8e951a56579291aa88e7a07d522670f26793ba367cf9c096ad60f6170d', hash='4bf41c9a816397c7c5cc37f5bb3d9fc79aae5acfd2c590bf4099ebbb4335c737', data='Block data'}
Block{previousHash='4bf41c9a816397c7c5cc37f5bb3d9fc79aae5acfd2c590bf4099ebbb4335c737', hash='e4a4fffc61ff573e73d113b77506c69561b806954ecad80472daf7a98644e6b9', data='Block data'}

C:\Users\STUDENT\Desktop>

```