A Mini Project Report

On

Blockchain Implementation using Java

Submitted in partial fulfillment of requirements for the Course
CSE18R272 - JAVA PROGRAMMING

Bachelor's of Technology

In

Computer Science and Engineering

Submitted By

Vikrant V Joliya
9919004368

Kunal Vasudeven
9919004153

Jayachandra Pal
9919004082

Under the guidance of

Dr. R. RAMALAKSHMI

(Associate Professor)

Department of Computer Science and Engineering
Kalasalingam Academy of Research and Education
Anand Nagar, Krishnankoil-626126
NOVEMBER 2020

ABSTRACT

A blockchain is a public ledger to which everyone has access but without a central authority having control. It is an enabling technology for individuals and companies to collaborate with trust and transparency. One of the best know applications of blockchains are the cryptographic currencies such as Bitcoin and others, but many other applications are possible. Blockchain technology is considered to be the driving force of the next fundamental revolution in information technology. Many implementations of blockchain technology are widely available today, each having its particular strength for a specific application domain. The tutorial provides the participants with insights and practical experience on Blockchain technology and applications in practice, as well as theory based exploration of possible business cases.

...

DECLARATION

I hereby declare that the work presented in this report entitled "**Blockchain Implementation using Java**", in partial fulfilment of the requirements for the course CSE18R272- Java Programming and submitted in **Department of Computer Science and Engineering, Kalasalingam Academy of Research and Education** (**Deemed to be University**) is an authentic record of our own work carried out during the period from **Nov 2020** under the guidance of **Dr. R. Ramalakshmi** (Associate Professor).

The work reported in this has not been submitted by me for the award of any other degree of this or any other institute.

Vikrant V Joliya
9919004368

Kunal Vasudeven
9919004153

Jayachandra Pal
9919004082

ACKNOWLEDGEMENT

I would like to express my special appreciation and profound thanks to my enthusiastic Project Supervisor **Dr.R.Ramalakshmi Ph.D, Associate Professor at Kalasalingam Academy of Research and Education [KARE]** for her inspiring guidance, constant encouragement with my work during all stages. We are extremely glad that we had a chance to do the Project under her guidance, who truly practices and appreciates deep thinking.

And during the most difficult times of the Century when writing this report,I respect All the Professors Teaching Staff making all efforts and hard work they put behind us. It will be forever my gratitude...

And last but not the least, I want to thank my teammate to be my back, all the way through this project.

Vikrant V Joliya
9919004368

Kunal Vasudeven
9919004153

Jayachandra Pal
9919004082

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

INTRODUCTION

Given an introduction about your project...

1.0.1 Objectives

List the objectives of the project work...

1. To develop a code using java.
2. To implement a project on Blockchain Implementation.

Blockchain as a Solution

A Blockchain is a list of records called blocks that are linked together using linked lists and use the cryptographic technique. Each block contains its own digital fingerprint called Hash, the hash of the previous block, a timestamp and the data of the transaction made, making it more secure towards any kind of data breach.

Blockchain is a distributed network of systems. Therefore, data breaches are very difficult to be carried out. Since, Blockchain generated hashes of each block, therefore, it is very difficult to carry out malicious attacks. Data Tampering will change the hash of each block which will make the blockchain invalid

Chapter 2

PROJECT DESCRIPTION

Here we have described about JAVA PACKAGES

1. java.security.NoSuchAlgorithmException;
2. java.security.MessageDigest;
3. java.nio.charset.StandardCharsets;
4. java.util.Base64;
5. java.util.Date;
6. java.util.ArrayList;
7. java.util.List;

>NoSuchAlgorithmException extends GeneralSecurityException
This exception is thrown when a particular cryptographic algorithm is requested but is not available in the environment.

>MessageDigest extends MessageDigestSpi
This MessageDigest class provides applications the functionality of a message digest algorithm, such as SHA-1 or SHA-256. Message digests are secure one-way hash functions that take arbitrary-sized data and output a fixed-length hash value.

>StandardCharsets extends Object

Constant definitions for the standard Charsets. These charsets are guaranteed to be available on every implementation of the Java platform.

>Charset UTF8

Eight-bit UCS Transformation Format

>Base64 extends Object

This class consists exclusively of static methods for obtaining encoders and decoders for the Base64 encoding scheme. The implementation of this class supports the following types of Base64 as specified in RFC 4648 and RFC 2045.

>ArryList

ArrayList creates an array of objects where the array can grow dynamically.

>List

List interface creates a collection of elements that are stored in a sequence and they are identified and accessed using the index.

Here we have described about IMPLEMENTATION MODEL Figure 2.2

>Making the BlocK.

A blockchain is just a chain/list of blocks. Each block in the blockchain will have its own digital fingerprint ie Timestamp, Version, contain digital fingerprint of the previous block, and have some data (this is in Reference to BlocK class Figure 2.1).

>Hash = Digital Fingerprint. computeHash()

Each block doesn't just contain the hash of the block before it, but its own hash is in part, calculated from the previous hash. If the previous block's data is changed then the previous block's hash will change in turn affecting all the hashes of the blocks there after. Calculating and comparing the hashes allow us to see if a blockchain is invalid.Here SHA-256 Algorithm is used. [computeHash() is the method used.]

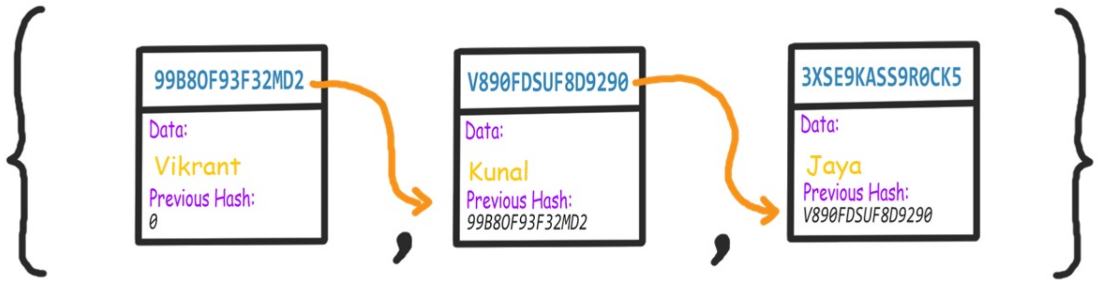


Figure 2.1: Input Example

`>isValid()`

isValid() method in the BlockchaiN class, that will loop through all blocks in the chain and compare the hashes. This method will need to check the hash variable is actually equal to the calculated hash, and the previous block's hash is equal to the previousHash variable.

`>generateGenesis() and addBlocK(BlocK blk) methods`

this method will store the data of 1st Block and addBlock will keep adding the next Block to continue the Chaining process.

`>displayChain() method`

this method will enumerate the data and print all the Block in the BlockchaiN with ...OUTPUT Figure 2.3

1. Block
2. Version: ie 20/11/2020 132e294
3. Timestamp: ie Execution Time
4. PreviousHash: ie Previous Block Data Hashed value
5. Hash: ie Current Block Data Hashed value

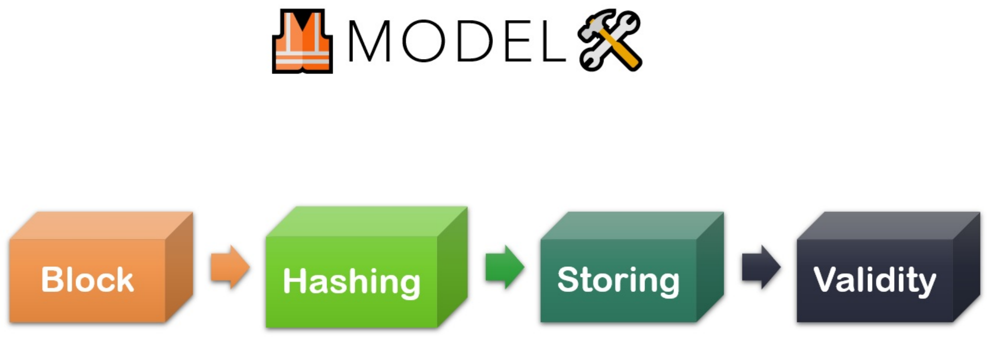
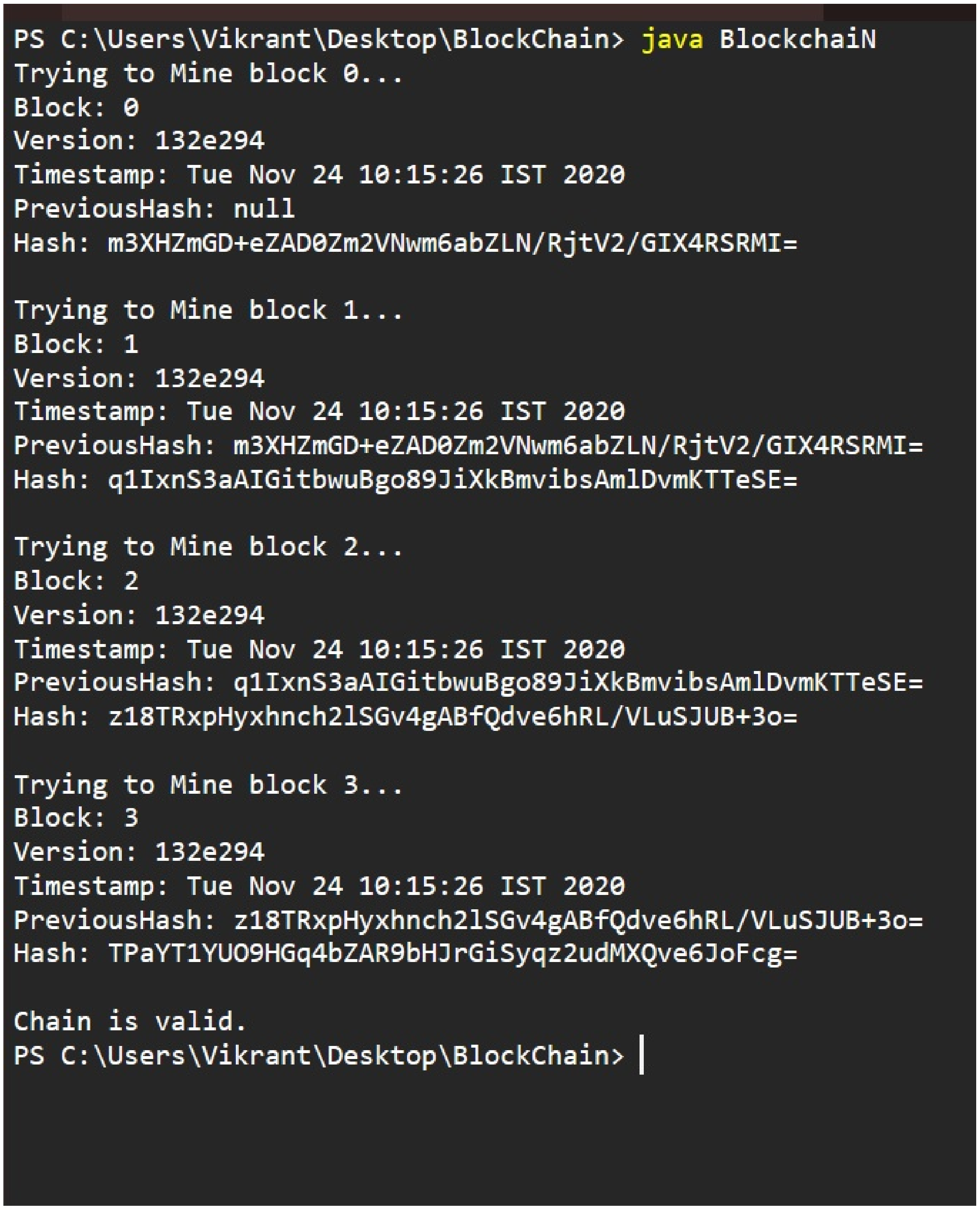


Figure 2.2: Project Model.



```
PS C:\Users\Vikrant\Desktop\BlockChain> java BlockchaiN
Trying to Mine block 0...
Block: 0
Version: 132e294
Timestamp: Tue Nov 24 10:15:26 IST 2020
PreviousHash: null
Hash: m3XHZmGD+eZAD0Zm2VNwm6abZLN/RjtV2/GIX4RSRMI=

Trying to Mine block 1...
Block: 1
Version: 132e294
Timestamp: Tue Nov 24 10:15:26 IST 2020
PreviousHash: m3XHZmGD+eZAD0Zm2VNwm6abZLN/RjtV2/GIX4RSRMI=
Hash: q1IxnS3aAIGitbwuBgo89JiXkBmvibsAmlDvmKTTeSE=

Trying to Mine block 2...
Block: 2
Version: 132e294
Timestamp: Tue Nov 24 10:15:26 IST 2020
PreviousHash: q1IxnS3aAIGitbwuBgo89JiXkBmvibsAmlDvmKTTeSE=
Hash: z18TRxpHyxhnch2lSGv4gABfQdve6hRL/VLuSJUB+3o=

Trying to Mine block 3...
Block: 3
Version: 132e294
Timestamp: Tue Nov 24 10:15:26 IST 2020
PreviousHash: z18TRxpHyxhnch2lSGv4gABfQdve6hRL/VLuSJUB+3o=
Hash: TPaYT1YUO9HGq4bZAR9bHJrGiSyqz2udMXQve6JoFcg=

Chain is valid.
PS C:\Users\Vikrant\Desktop\BlockChain>
```

Figure 2.3: Running this Project should look like

Chapter 3

CONCLUSION

So far **Implementing Blockchain using java** we can conclude the following contrast ie Blockchain has promise as an approach to developing systems for a number of applications within cybersecurity. In Blockchain-based systems, data and authority can be distributed, and transparent and reliable transaction ledgers created. Some of the key advantages of Blockchain for cybersecurity applications are in conflict with privacy properties, yet many of the potential applications have complex requirements for privacy. Privacy-enabling approaches for Blockchain have been introduced, such as private Blockchains, and methods for enabling parties to act pseudonymously, but it is as yet unclear which approaches are suitable in which applications. We explore a set of proposed uses of Blockchain within cybersecurity and consider their requirements for privacy. We compare these requirements with the privacy provision of Blockchain and explore the trade-off between security and privacy, reflecting on the effect of using privacy-enabling approaches on the security advantages that Blockchain can offer.

REFERENCES

- https://docs.oracle.com/javase/8/docs/api/java/
- MajorHackingLeague ICON LocalHost ,Azure Learning ,Geeksforgeeks and GitHub. . .
- https://en.wikipedia.org/wiki/Blockchain
- https://www.javatpoint.com/blockchain-tutorial
- https://medium.com/@essentia1/50-examples-of-how-blockchains-are-taking-over-the-world-4276bf488a4b

Appendices

SOURCE CODE-1

```
import java.nio.charset.StandardCharsets;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.Base64;
import java.util.Date;

public class BlocK {

    private String version;
    private Date Timestamp;
    private String hash;
    private String previousHash;
    private String data;
    private int nonce;

    public BlocK(String version, Date timestamp, String
       ↪ data) {
        this.version = version;
        this.Timestamp = timestamp;
        this.data = data;
        this.hash = computeHash();
    }

    public String computeHash() {

        String dataToHash = "" + this.version + this.
           ↪ Timestamp + this.previousHash + this.data
           ↪ ;

        MessageDigest digest;
        String encoded = null;

        try {
            digest = MessageDigest.getInstance("SHA-256
               ↪ ");
            byte[] hash = digest.digest(dataToHash.
               ↪ getBytes(StandardCharsets.UTF_8));
            encoded = Base64.getEncoder().
               ↪ encodeToString(hash);
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
        }

        this.hash = encoded;
        return encoded;

    }

    public String getVersion() {
        return version;
    }

    public void setVersion(String version) {
        this.version = version;
    }

    public Date getTimestamp() {
        return Timestamp;
    }

    public void setTimestamp(Date timestamp) {
        Timestamp = timestamp;
    }

    public String getHash() {
        return hash;
    }

    public void setHash(String hash) {
        this.hash = hash;
    }

    public String getPreviousHash() {
        return previousHash;
    }

    public void setPreviousHash(String previousHash) {
        this.previousHash = previousHash;
    }

    public String getData() {
        return data;
    }

    public void setData(String data) {
        this.data = data;
    }


}
```

SOURCE CODE-2

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

public class BlockchaiN {

    private List<BlocK> chain;

    public BlockchaiN() {
        chain = new ArrayList<BlocK>();
        chain.add(generateGenesis());
    }

    private BlocK generateGenesis() {
        BlocK genesis = new BlocK("132e294", new java.
            ↪ util.Date(), "GENESIS");
        genesis.setPreviousHash(null);
        genesis.computeHash();
        return genesis;
    }

    public void addBlocK(BlocK blk) {
        BlocK newBlock = blk;
    newBlock.setPreviousHash(chain.get(chain.size()
       ↪ -1).getHash());
    newBlock.computeHash();
    this.chain.add(newBlock);
}

public void displayChain() {

    for(int i=0; i<chain.size(); i++) {
        System.out.println("Trying to Mine block "+
           ↪ i+"... ");
        System.out.println("Block: " + i);
        System.out.println("Version: " + chain.get(
           ↪ i).getVersion());
        System.out.println("Timestamp: " + chain.
           ↪ get(i).getTimestamp());
        System.out.println("PreviousHash: " + chain
           ↪ .get(i).getPreviousHash());
        System.out.println("Hash: " + chain.get(i).
           ↪ getHash());
        System.out.println();
    }

}

public BlocK getLatestBlock() {
    return this.chain.get(chain.size()-1);
}

public void isValid() {

    for(int i=chain.size()-1; i>0; i--) {
        if(    !(chain.get(i).getHash().equals(chain
           ↪ .get(i).computeHash()))    ) {
            System.out.println("Chain is not valid"
               ↪ );
            return;
        }
```

```
            if(   !(chain.get(i).getPreviousHash().
                ↪ equals(chain.get(i-1).computeHash()))
                ↪    ) {
                 System.out.println("Chain␣is␣not␣valid"
                     ↪ );
                 return;
            }
        }

        System.out.println("Chain␣is␣valid.");

    }


    public static void main(String args[]) throws
        ↪ NullPointerException{

        BlockchaiN KLUCoin = new BlockchaiN();

        BlocK a = new BlocK("132e294", new java.util.
            ↪ Date(), "Vikrant");

        BlocK b = new BlocK("132e294", new java.util.
            ↪ Date(), "Kunal");

        BlocK c = new BlocK("132e294", new java.util.
            ↪ Date(), "Jaya");

        KLUCoin.addBlocK(a);

        KLUCoin.addBlocK(b);

        KLUCoin.addBlocK(c);

        //KLUCoin.getLatestBlock().setPreviousHash("
            ↪ ABCDEFG");

        KLUCoin.displayChain();
        //System.out.println("\nBlockchain is Valid: ")
            ↪ ;
        KLUCoin.isValid();


    }



}
```