

# Workshop on Blockchain - Overview

---

Aadharsh Aadhithya A<sup>\*1</sup>, Akshaya Jayaprakash B<sup>†1</sup>, Anirudh Edpuganti<sup>‡1</sup>,  
Haripranav <sup>§1</sup>, Nithesh B<sup>¶1</sup>, and Vishnu Radhakrishnan<sup>||1</sup>

<sup>1</sup>Center for Computational Engineering and Networking , Amrita Vishwa  
Vidhyapeetham

15th May , 2021

## 1 Overview of Classes

### 1.1 Hashing class

The Hashing class should contain helper encrypting functions to be used in other classes. one of the most commonly used cryptographic algorithm is SHA-256. We can use SHA-256 as well.

This class, at this phase is expected to contain one function createHash.

**String createHash(data)** This function should be able to take in variable number of arguments, and give out the SHA-256 encrypted string based on the incoming data. The encryption should be the same irrespective of the order of the data.

**String hex2Bin(sting data)**

Takes in a hexadecimal string and returns its binary equivalent.

**String bin2Hex(sting data)**

Takes in a binary string and returns its hexadecimal equivalent .

---

<sup>\*</sup>cb.en.u4aie20001@cb.students.amrita.edu

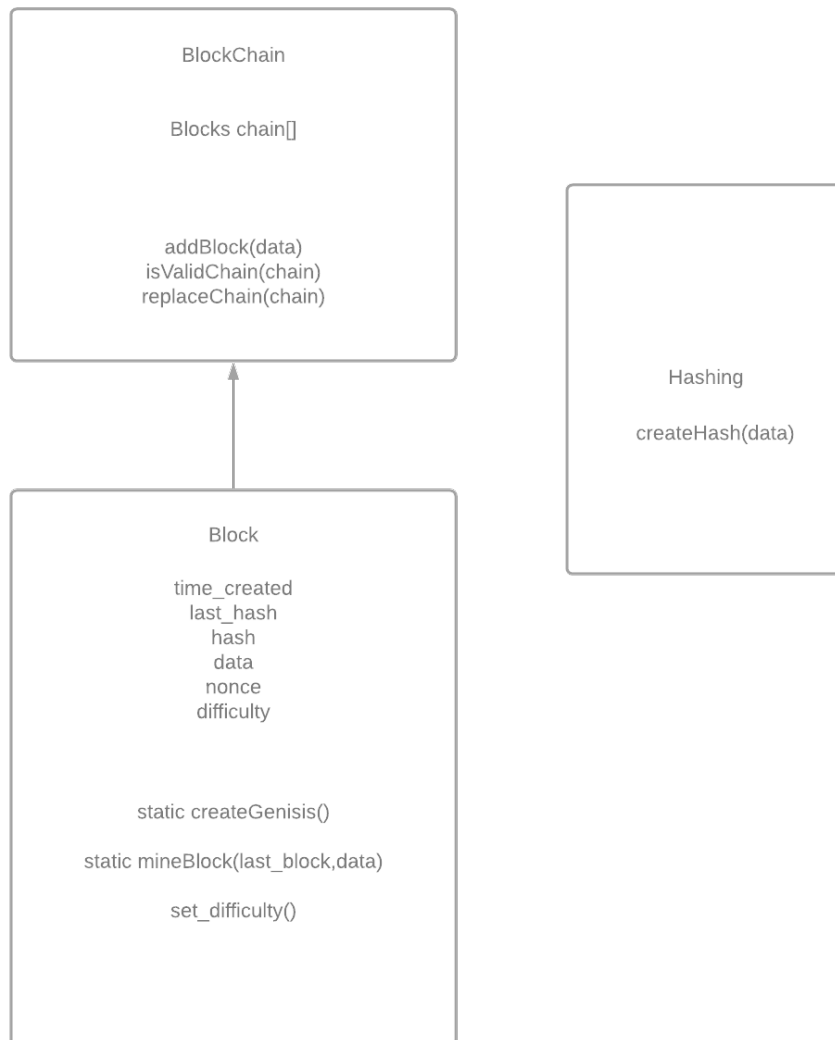
<sup>†</sup>cb.en.u4aie20003@cb.students.amrita.edu

<sup>‡</sup>cb.en.u4aie20005@cb.students.amrita.edu

<sup>§</sup>cb.en.u4aie20021@cb.students.amrita.edu

<sup>¶</sup>cb.en.u4aie20043@cb.students.amrita.edu

<sup>||</sup>cb.en.u4aie20074@cb.students.amrita.edu



## 1.2 Block Class

The Specification of Block class is given below

---

```
1  Class Block{
2
3  properties:
4      time_created
5      last_hash
6      hash
7      data
8      nonce
9      difficulty
10
11  methods:
12      Block(data,last_hash,hash,time_created,nonce ,difficulty){
13          timestamp = time of creating the block
14
15          last_hash = hash of the previous block.
16          The has of the genesis block can be arbitrary
17
18          hash = Hashing.createHash(time_created,last_hash,data,nonce,difficulty)
19
20          nonce and difficulty must be dynamically adjusted.
21      }
22
23
24      static createGenesis(){
25          return  new Block(GENISISDATA)
26      }
27
28
29      static mineBlock(last_block,data){
30          a block new block must be returned , if there is Proof of work.
31          One simple proof of work algorithm is HashCash.
32
33
34      }
35
36      static setDifficulty(Block,  time){
37          Difficulty must be adjusted based on the mining rate parameter that is to be set.
38      }
39
40  }
```

---

### 1.2.1 Block class checklist

- Block has all the properties specified

- `static genesis()`

Should return an instance of the block. And the Returned instance should contain a specific genesis data.

- `static mineBlock(last_block , data)`

- Should return an block instance

- Should set the

- `hash`

- property of returned block to the hash of the previous block in the chain.

- Should set the

- `data`

- property to the incoming data , and the *time<sub>created</sub> should be the current time Should create SHA-256 hash based on the inputs.*

- Hash should also be based on

- `difficulty`

- Difficulty should be adjusted.

- `setDifficulty(block,time)`

.

- If the time taken to mine the new block is less than the mining rate , then difficulty should be increased. If the time taken to mine the new block is high , then difficulty should be decreased. The change in difficulty should always be 1

- the difficulty should not go less than 1

click here to read more about Nonce and difficulty. Also see here for a more deeper dive. The HashCash algorithm is used in bitcoins network.

### 1.3 Blockchain Class

The class description is below.

---

```
1  ass Blockchain{
2
3  ock chain[]; // Dynamic Array of blocks.
4  Each Block should contain a reference to its previous block
5
6  ockchain(){
7  ain[0] = Block.genesis()
8  creates genesis block
9
10
11  dBlock(data){
12  is should mine a new block ,
13  sed on the previous block and data.
14  e mined block should be added in the end of
15  e chain
16
17
18  atic isValidChain(chain){
19  ecks if a chain is valid.
20  lidity criterion is given below
21
22
23
24  atic replaceChain(chain){
25
26  ould check if the incoming chain is valid and
27  place hte chain of hte blockchain with the incoming chain
28
29
30
31
32
33
```

---

#### Blockchain class checklist

- Should contain a
  - Block chain[]
- Should start with THE genesis block
- isValidChain()
  - Chain should start with THE genesis block

- Should return False if a lasthash reference is tampered
  - Should return false if the block has n invalid feild
  - Should return false if the chain contains a jumped difficulty. (Succesive difficulties should always vary by 1)
  - If all the blocks are valid and none of the above cases hold , then return true
- `replaceChain(chain[])`
    - When the incoming chain is shorter than the Block-chain.length , do not replace.
    - WHne the incoming chain is longer than the block chain .length , replace , iff the chain  
`isValid(chain)`