

# BLOCKCHAIN CRYPTOCURRENCY PROJECT

## MEMBERS:

NIKHIL KRISHNA 2018A7PS0496H

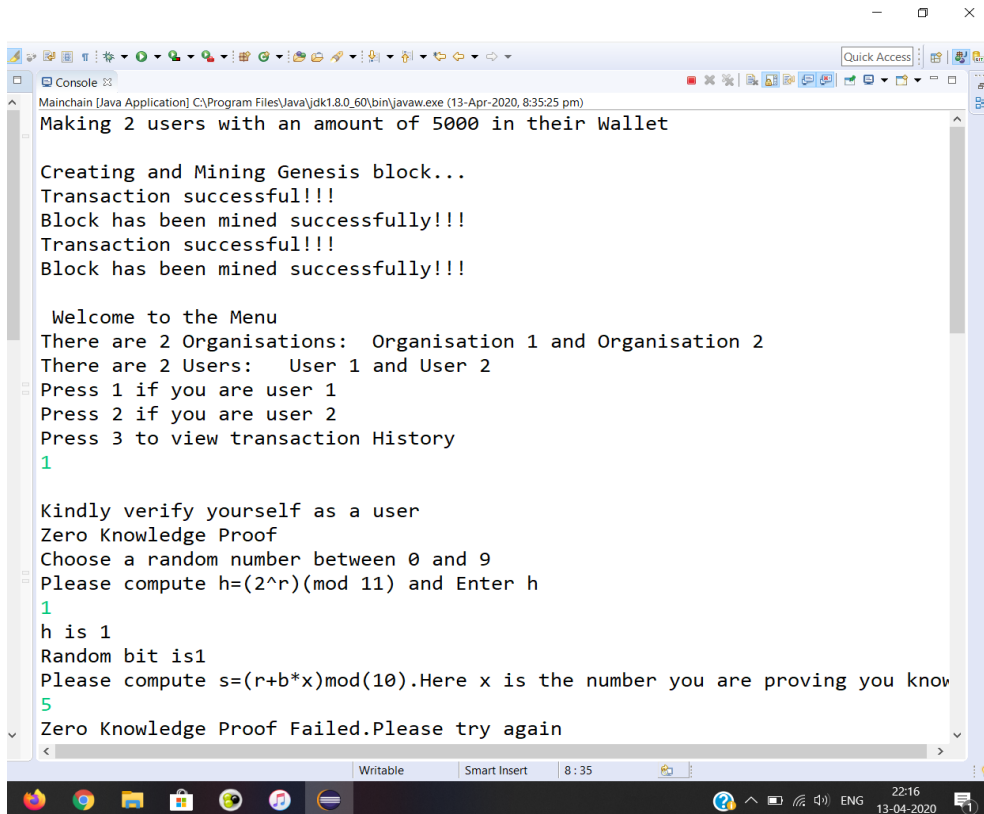
MIHIR BANSAL 2018A7PS0215H

BHAVYESH DESAI 2018A7PS0164H

KUSHAGRA LAVANIA 2018A7PS0216H

ADITYA JHAVERI 2018A7PS0209H

# STEP 1 : USER AUTHENTICATION



The screenshot shows a Java application window titled 'Mainchain [Java Application] C:\Program Files\Java\jdk1.8.0\_60\bin\javaw.exe (13-Apr-2020, 8:35:25 pm)'. The console output is as follows:

```
Making 2 users with an amount of 5000 in their Wallet

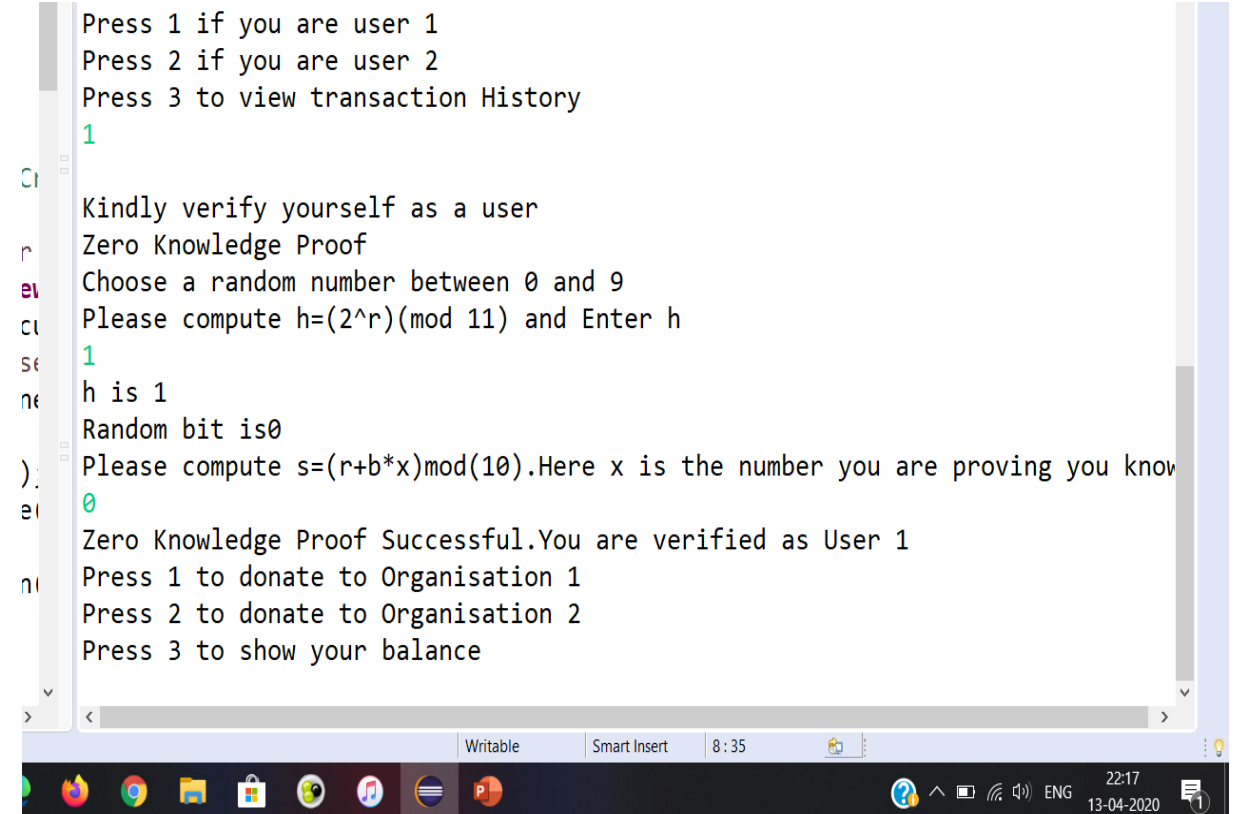
Creating and Mining Genesis block...
Transaction successful!!!
Block has been mined successfully!!!
Transaction successful!!!
Block has been mined successfully!!!

Welcome to the Menu
There are 2 Organisations: Organisation 1 and Organisation 2
There are 2 Users: User 1 and User 2
Press 1 if you are user 1
Press 2 if you are user 2
Press 3 to view transaction History
1

Kindly verify yourself as a user
Zero Knowledge Proof
Choose a random number between 0 and 9
Please compute  $h=(2^r)(\text{mod } 11)$  and Enter h
1
h is 1
Random bit is 1
Please compute  $s=(r+b*x)\text{mod}(10)$ . Here x is the number you are proving you know
5
Zero Knowledge Proof Failed.Please try again
```

The application is running on a Windows 10 desktop with a taskbar at the bottom showing various icons and the system clock at 22:16 on 13-04-2020.

FAILED AUTHENTICATION



The screenshot shows the same Java application window as the previous one, but with the following console output:

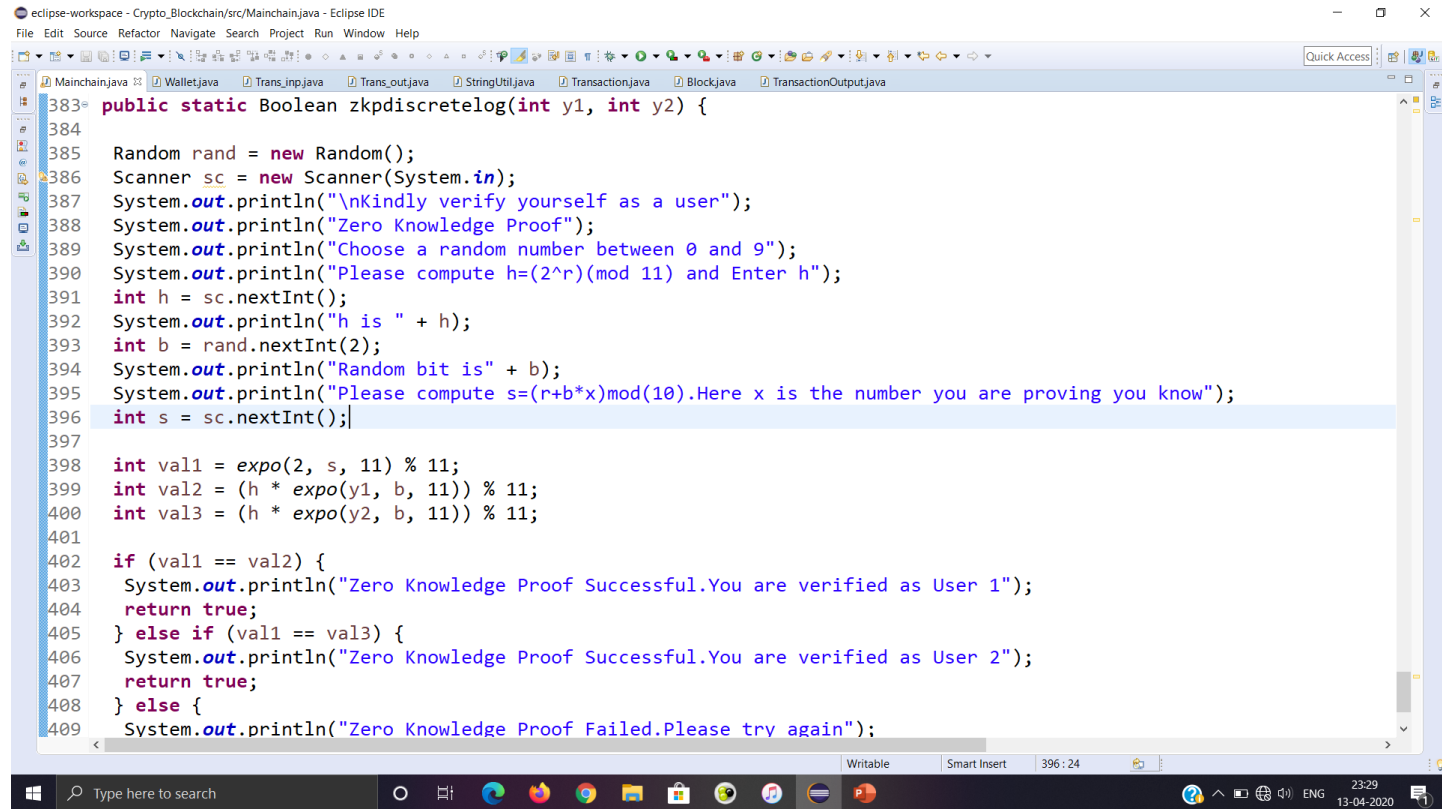
```
Press 1 if you are user 1
Press 2 if you are user 2
Press 3 to view transaction History
1

Kindly verify yourself as a user
Zero Knowledge Proof
Choose a random number between 0 and 9
Please compute  $h=(2^r)(\text{mod } 11)$  and Enter h
1
h is 1
Random bit is 0
Please compute  $s=(r+b*x)\text{mod}(10)$ . Here x is the number you are proving you know
0
Zero Knowledge Proof Successful.You are verified as User 1
Press 1 to donate to Organisation 1
Press 2 to donate to Organisation 2
Press 3 to show your balance
```

The application is running on a Windows 10 desktop with a taskbar at the bottom showing various icons and the system clock at 22:17 on 13-04-2020.

SUCCESSFUL AUTHENTICATION

# PROCESS OF USER VERIFICATION

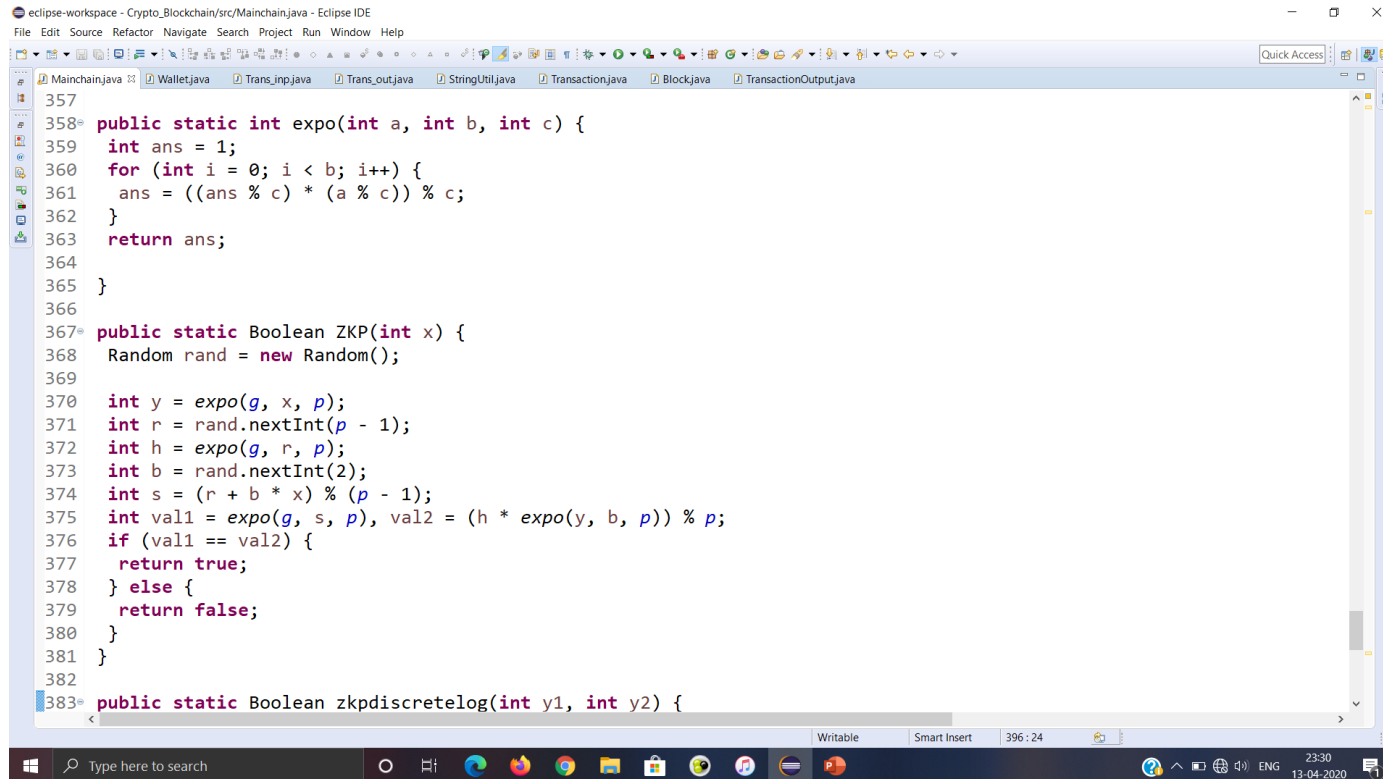
The image shows a screenshot of the Eclipse IDE interface. The title bar indicates the workspace is 'Crypto\_Blockchain/src/Mainchain.java'. The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The toolbar contains various icons for file operations and development tools. The project explorer on the left shows a package structure with files: Mainchain.java, Wallet.java, Trans\_inp.java, Trans\_out.java, StringUtil.java, Transaction.java, Block.java, and TransactionOutput.java. The editor window displays the code for the 'zkpdiscretelog' method in Mainchain.java. The code is as follows:

```
383 public static Boolean zkpdiscretelog(int y1, int y2) {
384
385     Random rand = new Random();
386     Scanner sc = new Scanner(System.in);
387     System.out.println("\nKindly verify yourself as a user");
388     System.out.println("Zero Knowledge Proof");
389     System.out.println("Choose a random number between 0 and 9");
390     System.out.println("Please compute h=(2^r)(mod 11) and Enter h");
391     int h = sc.nextInt();
392     System.out.println("h is " + h);
393     int b = rand.nextInt(2);
394     System.out.println("Random bit is" + b);
395     System.out.println("Please compute s=(r+b*x)mod(10).Here x is the number you are proving you know");
396     int s = sc.nextInt();
397
398     int val1 = expo(2, s, 11) % 11;
399     int val2 = (h * expo(y1, b, 11)) % 11;
400     int val3 = (h * expo(y2, b, 11)) % 11;
401
402     if (val1 == val2) {
403         System.out.println("Zero Knowledge Proof Successful.You are verified as User 1");
404         return true;
405     } else if (val1 == val3) {
406         System.out.println("Zero Knowledge Proof Successful.You are verified as User 2");
407         return true;
408     } else {
409         System.out.println("Zero Knowledge Proof Failed.Please try again");
```

The status bar at the bottom shows 'Writeable', 'Smart Insert', '396 : 24', and the system clock '23:29 13-04-2020'.

This involves solving the Discrete logarithm problem which is the toughest to compute by an attacker. Hence, it leads to a secure blockchain network.

# THE DISCRETE LOGARITHM PROBLEM

A screenshot of the Eclipse IDE interface. The title bar reads 'eclipse-workspace - Crypto\_Blockchain/src/Mainchain.java - Eclipse IDE'. The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The toolbar contains various icons for file operations and development tools. The editor window shows a Java file named Mainchain.java with the following code:

```
357
358 public static int expo(int a, int b, int c) {
359     int ans = 1;
360     for (int i = 0; i < b; i++) {
361         ans = ((ans % c) * (a % c)) % c;
362     }
363     return ans;
364 }
365
366
367 public static Boolean ZKP(int x) {
368     Random rand = new Random();
369
370     int y = expo(g, x, p);
371     int r = rand.nextInt(p - 1);
372     int h = expo(g, r, p);
373     int b = rand.nextInt(2);
374     int s = (r + b * x) % (p - 1);
375     int val1 = expo(g, s, p), val2 = (h * expo(y, b, p)) % p;
376     if (val1 == val2) {
377         return true;
378     } else {
379         return false;
380     }
381 }
382
383 public static Boolean zkpdiscritelog(int y1, int y2) {
```

The status bar at the bottom shows 'Type here to search', a taskbar with application icons, and system information including '23:30 13-04-2020'.

In the mathematics of the real numbers, the logarithm  $\log_b a$  is a number  $x$  such that  $b^x = a$ , for given numbers  $a$  and  $b$ . Analogously, in any group  $G$ , powers  $b^k$  can be defined for all integers  $k$ , and the discrete logarithm  $\log_b a$  is an integer  $k$  such that  $b^k = a$ . In number theory, the more commonly used term is index: we can write  $x = \text{ind}_r a \pmod{m}$  (read the index of  $a$  to the base  $r$  modulo  $m$ ) for  $r^x \equiv a \pmod{m}$  if  $r$  is a primitive root of  $m$  and  $\text{gcd}(a, m) = 1$

# BLOCK CREATION

- The user's data such as Aadhar Number and PAN Number are stored in a block in the form of a String.
- The variable 'epochTime' stores time elapsed since 1/1/1970.
- Nonce is a arbitrary number that is used once.
- Genesis block is the first block of the Blockchain. It does not have a previous block so we set "0" as the previous hash.
- A block is created on calling the constructor of Block.java class which takes previous hash value and user's userid as arguments.

# MINING A BLOCK

- Bitcoin mining is the process of adding transaction records to Bitcoin's public ledger of past transactions or blockchain.
- It is done using function `blockMine()` inside `Block.java` class by using a merkle tree structure concept.
- Any node in the merkle tree is the hash of all its children
- Hence the merkle root is the hash of the hashes of all transactions in the Block
- Adding or removing even one transaction changes the value of all parents of the transaction
- Hence adding or removing a transaction also changes the merkle root hash
- So we can use the merkle root as a value to verify the integrity of all transactions under it
- In doing so, we do not need the body of the transactions

# GENERATING KEY PAIRS

- Elliptic curve Cryptography is used to generate each user's private and public keys in a KeyPair.
- The Elliptic Curve Digital Signature Algorithm is implemented in the generateKeyPair() method inside Wallet.java
- The private key of each user is kept secret and is used for generating the digital signature of every user after each transaction.
- To view a transaction, the public key of the recipient is used to decrypt the hashed transaction block inside the merkle tree of every user.
- The public key of every user is made public to all whereas the private key is kept secret with himself.

# TRANSACTION PROCESSING

- The user is asked if he wants to make a donation to Organization 1 or Organization 2.

```
Press 1 to donate to Organisation 1
Press 2 to donate to Organisation 2
Press 3 to show your balance
1
Enter amount to donate
450
Transaction successful!!!
Block has been mined successfully!!!
Now Your balance is 3750.0
Press 1 if you are user 1
Press 2 if you are user 2
Press 3 to view transaction History
3
|
User1's balance is 3750.0
User2's balance is 5000.0
Organization 1's balance is 1250.0
Organization 2's balance is 0.0
Press 1 to view user 1 Transaction History
Press 2 to view user 2 Transaction History
Press 3 to exit
```



# TRANSACTION VERIFICATION

- The user is asked if he wants to make a donation to Organization 1 or Organization 2.
- On selecting the appropriate option, the transaction is hashed with the public key of the organization and the transaction block is added to the history of transactions under the user authenticated.
- This is done using `sendfunds()` function inside the `Wallet.java` class.
- The transaction is signed by the sender's digital signature which is hashed using private key of the sender.
- Inside `StringUtil.java`, `toApplyECDSASig(PrivateKey privateKey,String input)` applies ECDSA Signature to the input and the sender's private key and returns the result as bytes.
- In `StringUtil.java`, `toVerifyECDSASig(PublicKey publicKey, String data, byte[] signature)` will verify the signature using the public key and the string data. It returns `True` or `False` if the signature is valid
- Only the public key of the recipient will be able to successfully verify the transaction.

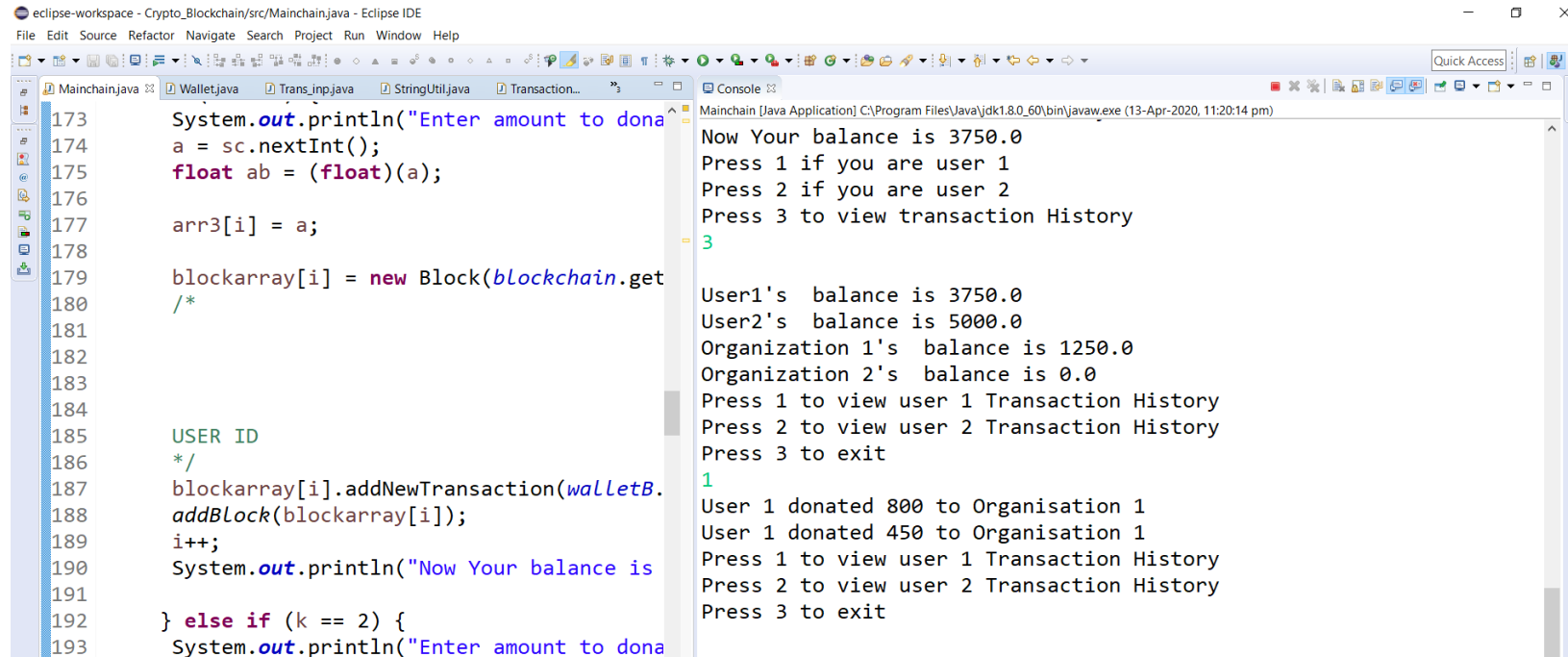
# SHA256 ALGORITHM

- Inside StringUtil.java, generate256Sign(String input) applies the SHA256 algorithm to a string and returns the result.
- It generates a 256 bit signature for a text.
- A hash cannot be decrypted back to the original text.
- **SHA-256** is one of the successor hash functions to SHA-1 (collectively referred to as SHA-2), and is one of the strongest hash functions available. SHA-256 is not much more complex to code than SHA-1, and has not yet been compromised in any way.
- The 256-bit key makes it a good partner-function for AES.

# VIEW USER'S TRANSACTION HISTORY

- The details of the user can be received by opting for menu no. 3, which displays all the user's transaction details.
- The user's wallet balance can be checked by decrypting the block which has the transaction details in the merkle tree after a transaction is made successfully.
- The sum of all the unspent transaction outputs addressed to the user is the wallet balance
- This is done using `getBalance()` function inside the `Wallet.java` class which decrypts the hashed block of the specified receiver of transaction using its public key using `isMine()` function.

# VIEWING USER'S TRANSACTION HISTORY



The screenshot shows the Eclipse IDE with a project named 'Crypto\_Blockchain'. The 'Mainchain.java' file is open, displaying Java code for a blockchain application. The code includes a loop for adding transactions and a section for viewing transaction history. The console output shows the application's execution, including the current balance and the transaction history for User 1.

```
173 System.out.println("Enter amount to dona
174 a = sc.nextInt();
175 float ab = (float)(a);
176
177 arr3[i] = a;
178
179 blockarray[i] = new Block(blockchain.get
180 /*
181
182
183
184
185 USER ID
186 */
187 blockarray[i].addNewTransaction(walletB.
188 addBlock(blockarray[i]);
189 i++;
190 System.out.println("Now Your balance is
191
192 } else if (k == 2) {
193 System.out.println("Enter amount to dona
```

Console Output:

```
Mainchain [Java Application] C:\Program Files\Java\jdk1.8.0_60\bin\javaw.exe (13-Apr-2020, 11:20:14 pm)
Now Your balance is 3750.0
Press 1 if you are user 1
Press 2 if you are user 2
Press 3 to view transaction History
3
User1's balance is 3750.0
User2's balance is 5000.0
Organization 1's balance is 1250.0
Organization 2's balance is 0.0
Press 1 to view user 1 Transaction History
Press 2 to view user 2 Transaction History
Press 3 to exit
1
User 1 donated 800 to Organisation 1
User 1 donated 450 to Organisation 1
Press 1 to view user 1 Transaction History
Press 2 to view user 2 Transaction History
Press 3 to exit
-
```

## USER 1'S TRANSACTION HISTORY