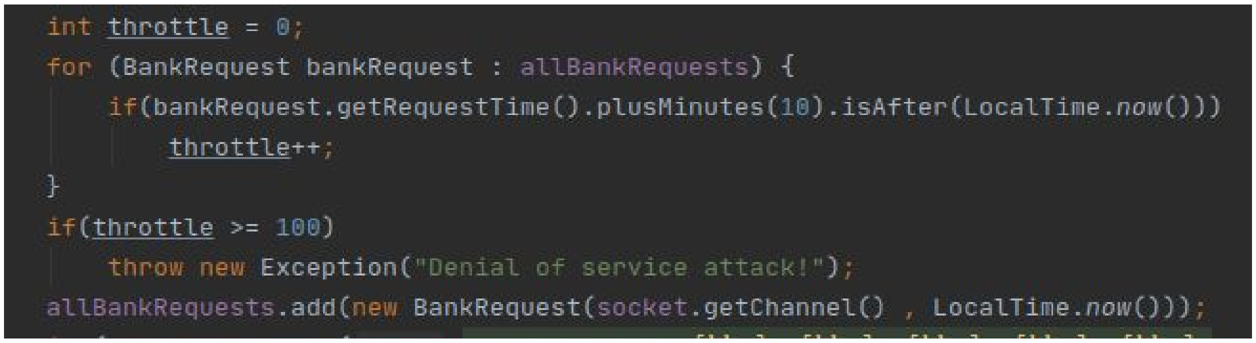
In the name of god

Advanced Programming Project
Security Documentation

Contributors:
MohammadMahdi Gheidi
Mohadese Ghaffari

Denial of Service

Denial of service attack is when a specific user tries to overwhelm our server with too many requests in a short period of time.
In our project, we have used an approach to prevent this from happening to some extent.
Below you will see the code which our bank server(the client handler class) runs to prevent DoS:

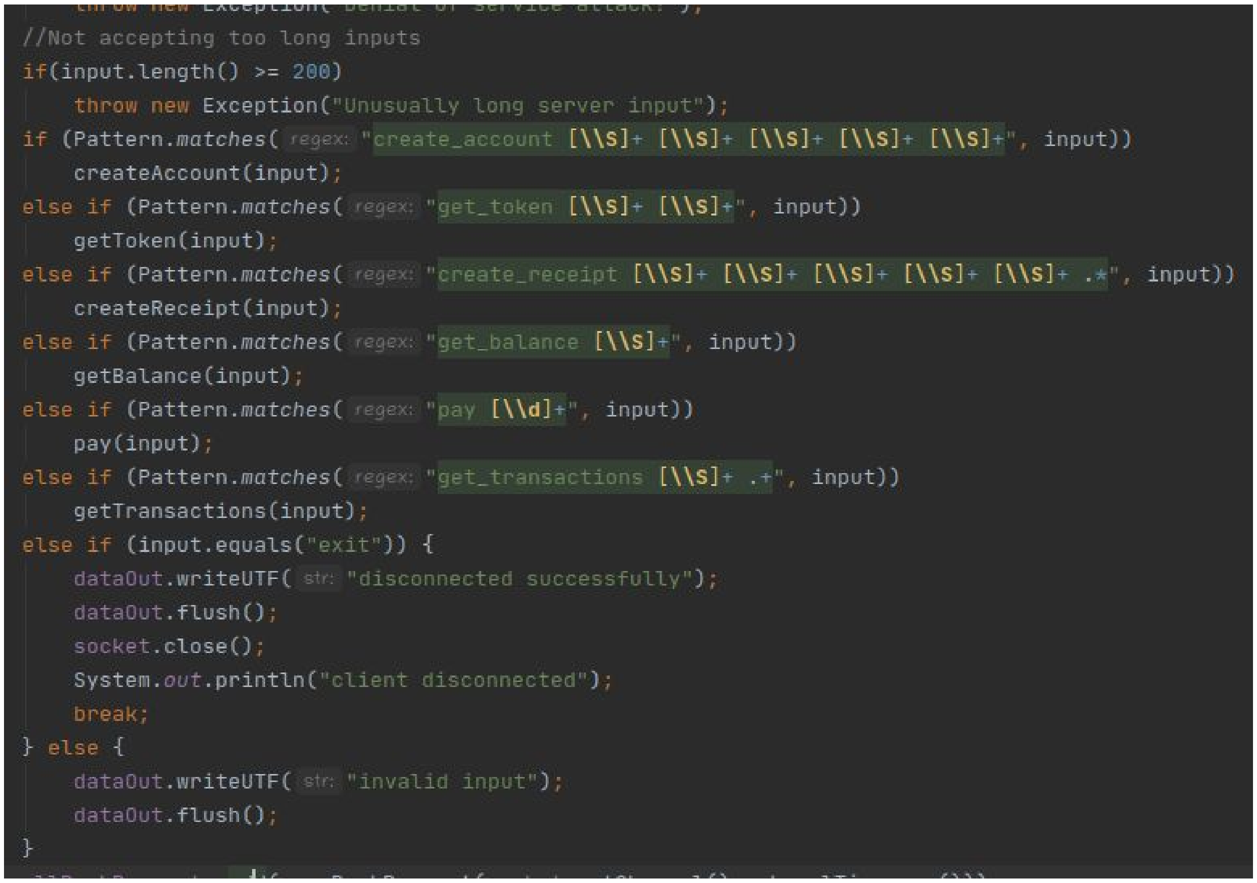


```
int throttle = 0;
for (BankRequest bankRequest : allBankRequests) {
    if(bankRequest.getRequestTime().plusMinutes(10).isAfter(LocalTime.now()))
        throttle++;
}
if(throttle >= 100)
    throw new Exception("Denial of service attack!");
allBankRequests.add(new BankRequest(socket.getChannel() , LocalTime.now()));
```

It searches on all requests we have received and counts how many requests in the last 10 minutes we have received of an specific channel(in other words same user) and if it is an unreasonable number we don’t accept the request from the user and send an DoS exception.

Improper inputs

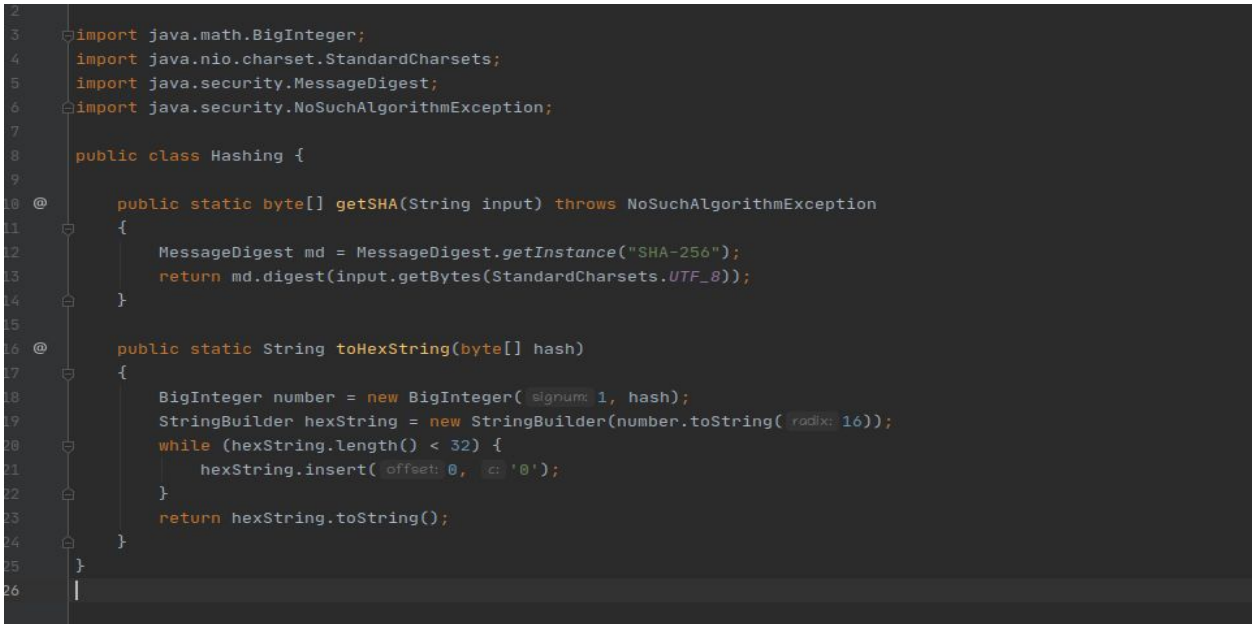
By the use of regular expressions and if/else statements when handling clients inputs we've been able to maintain our server under all circumstances and not allow malicious code enter the program.



```
    throw new Exception("Denial of service attack!");
//Not accepting too long inputs
if(input.length() >= 200)
    throw new Exception("Unusually long server input");
if (Pattern.matches( regex: "create_account [\\S]+ [\\S]+ [\\S]+ [\\S]+ [\\S]+", input))
    createAccount(input);
else if (Pattern.matches( regex: "get_token [\\S]+ [\\S]+", input))
    getToken(input);
else if (Pattern.matches( regex: "create_receipt [\\S]+ [\\S]+ [\\S]+ [\\S]+ [\\S]+ .*", input))
    createReceipt(input);
else if (Pattern.matches( regex: "get_balance [\\S]+", input))
    getBalance(input);
else if (Pattern.matches( regex: "pay [\\d]+", input))
    pay(input);
else if (Pattern.matches( regex: "get_transactions [\\S]+ .+", input))
    getTransactions(input);
else if (input.equals("exit")) {
    dataOut.writeUTF( str: "disconnected successfully");
    dataOut.flush();
    socket.close();
    System.out.println("client disconnected");
    break;
} else {
    dataOut.writeUTF( str: "invalid input");
    dataOut.flush();
}
```

Man in the middle

As we do know, in the MITM attack someone tries to sniff sensitive data from the server the attack is performing on.
We tried to protect the sensitive data(such as passwords, bank account numbers, emails, ...) in our app by communicating them in the hash(SHA-256 to be specific) form so the user sniffing the data could not use the sniffed data.



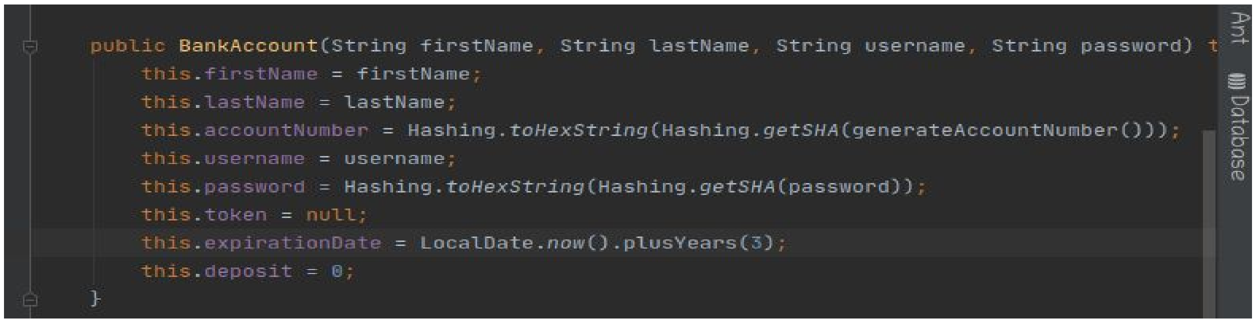
```java
import java.math.BigInteger;
import java.nio.charset.StandardCharsets;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

public class Hashing {

    public static byte[] getSHA(String input) throws NoSuchAlgorithmException
    {
        MessageDigest md = MessageDigest.getInstance("SHA-256");
        return md.digest(input.getBytes(StandardCharsets.UTF_8));
    }

    public static String toHexString(byte[] hash)
    {
        BigInteger number = new BigInteger( signum: 1, hash);
        StringBuilder hexString = new StringBuilder(number.toString( radix: 16));
        while (hexString.length() < 32) {
            hexString.insert( offset: 0,  c: '0');
        }
        return hexString.toString();
    }
}
```

This shows the algorithm we used to generate hashes based on string inputs,



```java
    public BankAccount(String firstName, String lastName, String username, String password) t
        this.firstName = firstName;
        this.lastName = lastName;
        this.accountNumber = Hashing.toHexString(Hashing.getSHA(generateAccountNumber()));
        this.username = username;
        this.password = Hashing.toHexString(Hashing.getSHA(password));
        this.token = null;
        this.expirationDate = LocalDate.now().plusYears(3);
        this.deposit = 0;
    }
```

And this is a usage of the hash algorithm in the code.