**Bullet Points** :

\* DAO pattern has been use.

\* StringUtils use for equality which is optimise and null safe

\* Added success and fail test cases for transfer service

\* Mthod level comment added

\* Making 'Money transfer' service as a trade safe

**Assumptions:**

1. you cann't make transacations on same account.

2. The amount to transfer should always be a positive number.

3. we do not support overdrafts

**Steps to run :**

Build the project using gradle.:

./gradlew clean build

Once the project is built, you can run the jar as follows:

java -jar build/libs/Java\ asset-management-service-0.0.1-SNAPSHOT.jar

Example request:

curl -i -X PUT \

-H "Content-Type:application/json" \

-d \

'{

"accountFromId":"Id-1",

"accountToId":"Id-2",

"amount": "40"

}' \

'http://localhost:18080/v1/accounts/transfer'

**Further Improvements :**

1. We can use JWT token for authentication, and client will pass that token in Authetication header , in this way we can authenticate the request.

2. Instead of having a concurrent hashmap, in production we could have a database for storing the accounts as well as for scalability.

3. We could setup LB(Load Balancer)/LTM(Load trafic manager) to handle a multiple request and provide fail over without impacting current ongoing transacations.

4. Place mutual authentication mechanism to provide secure connection.

5. Another advantage of using a database is that we can handle transactional atomic account updates.

Design :

