Contain:

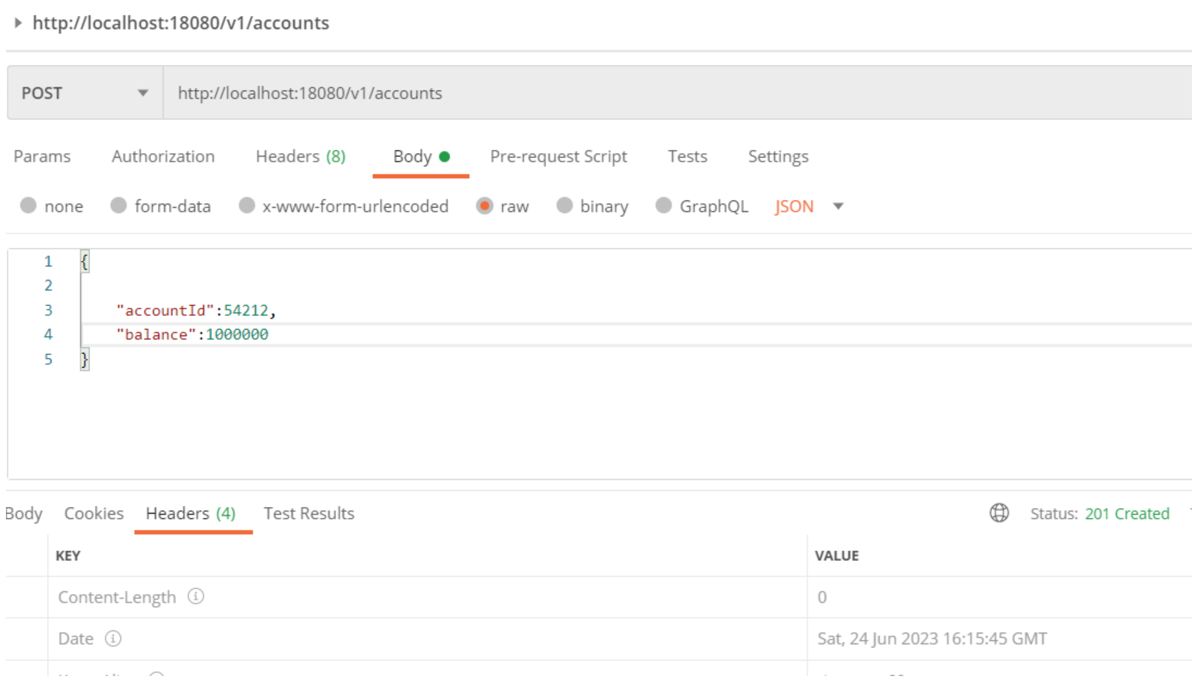
APPENDIX 1: ALL SCREENSHOT FOR THE CODE

APPENDIX 2: EXPLANATION OF CODE THE CODE, BEST PRACTICES IN DETAILS

APPENDIX 3: JUNIT AND SONAR TEST, MAKE CODE PRODUCTION READY

APPENDIX 1: ALL SCREENSHOT FOR THE CODE

Account Created:



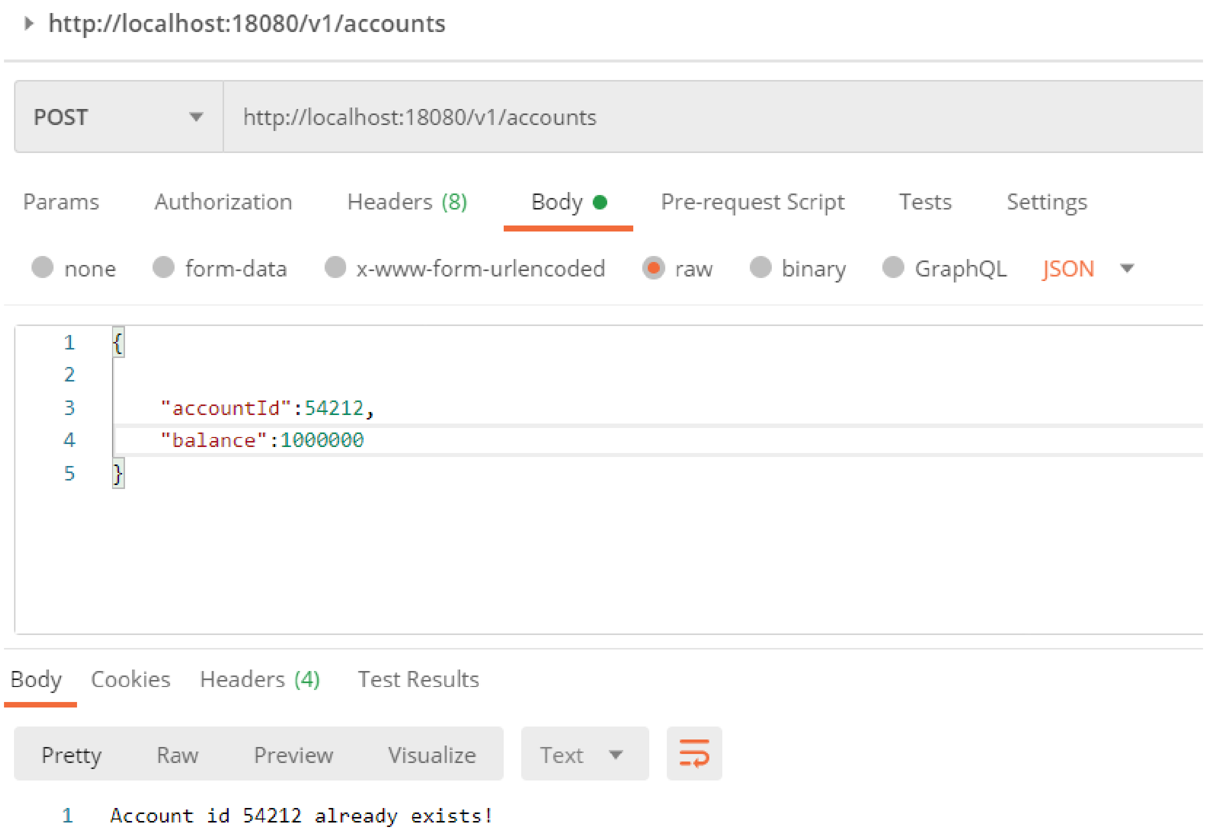
POST http://localhost:18080/v1/accounts

```
{

    "accountId":54212,
    "balance":1000000
}
```

Status: 201 Created

KEY	VALUE
Content-Length	0
Date	Sat, 24 Jun 2023 16:15:45 GMT

Dublicated Account validated:

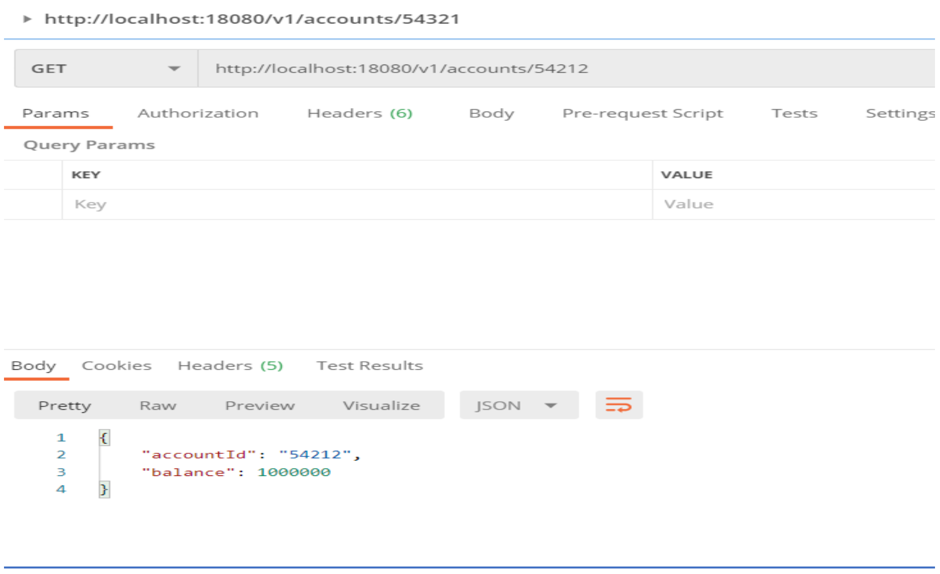


POST http://localhost:18080/v1/accounts

```
{

    "accountId":54212,
    "balance":1000000
}
```

```
Account id 54212 already exists!
```

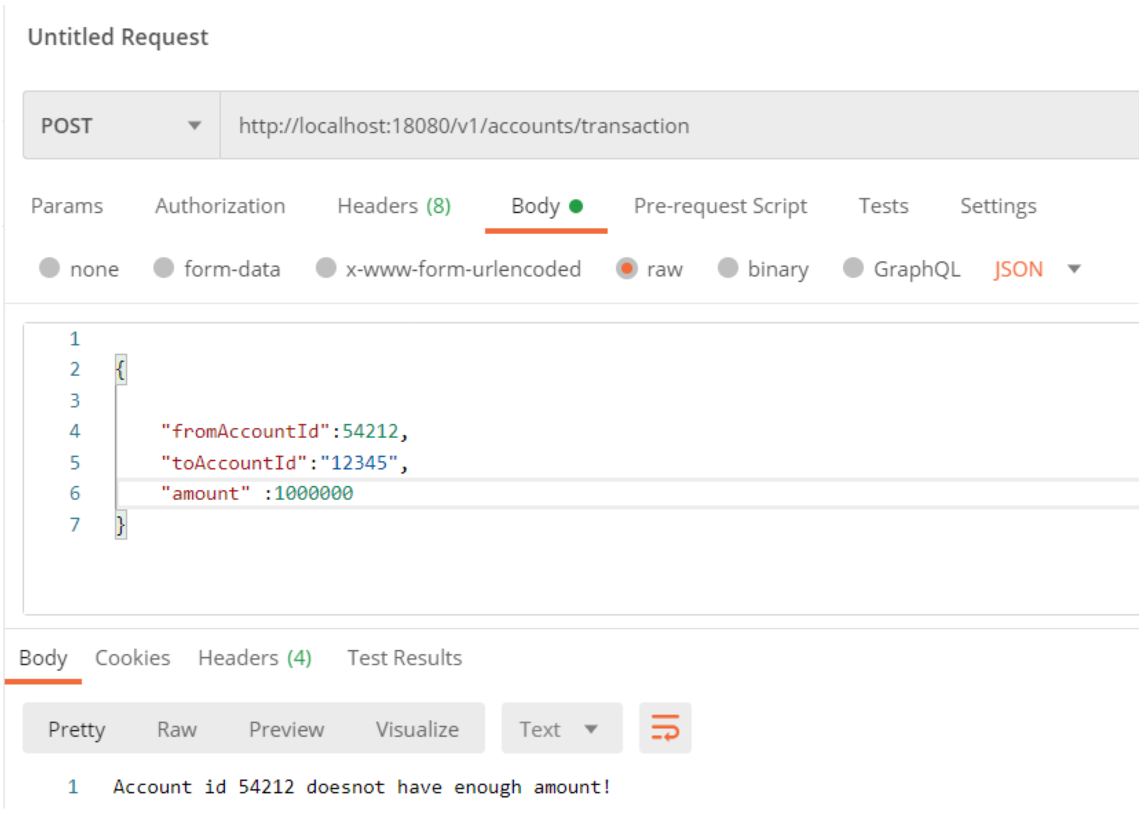
To Find Newly created Account :



GET http://localhost:18080/v1/accounts/54212

```
{
    "accountId": "54212",
    "balance": 1000000
}
```

Validation : If account does not have sufficent balance



Untitled Request

POST http://localhost:18080/v1/accounts/transaction

Params Authorization Headers (8) Body ● Pre-request Script Tests Settings

none form-data x-www-form-urlencoded [✓ raw] binary GraphQL JSON

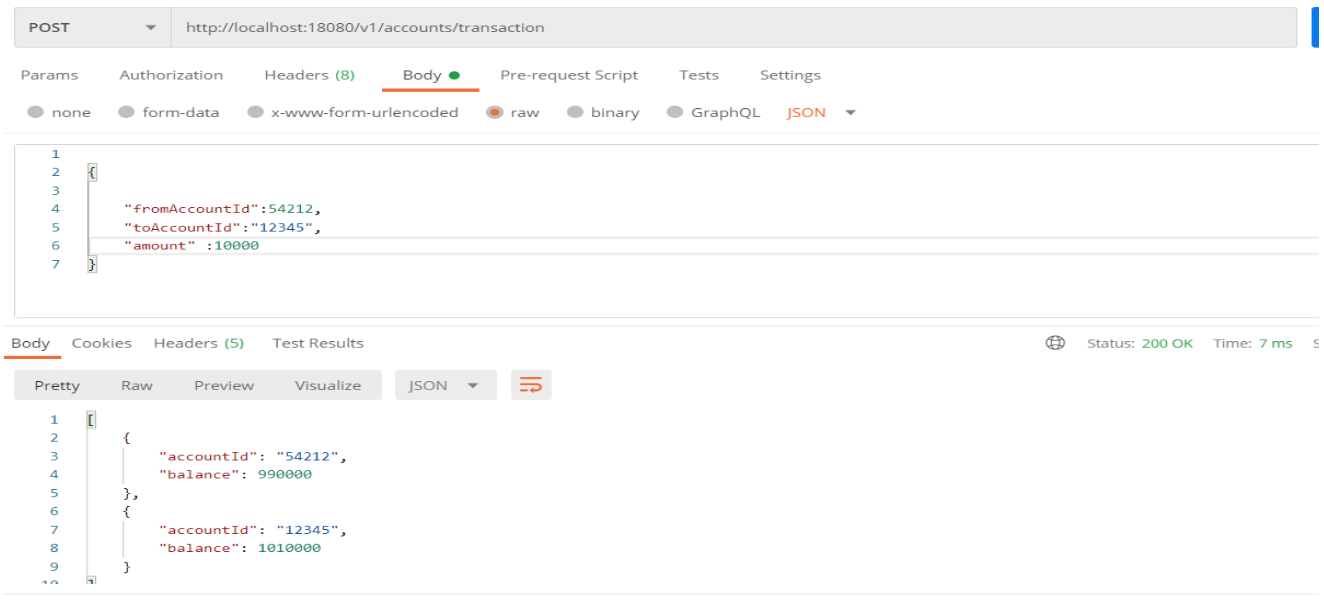
```
{

    "fromAccountId":54212,
    "toAccountId":"12345",
    "amount" :1000000
}
```

Body Cookies Headers (4) Test Results

Pretty Raw Preview Visualize Text

```
Account id 54212 doesnot have enough amount!
```

Transfer balance to other account sucessully :



POST http://localhost:18080/v1/accounts/transaction

Params Authorization Headers (8) Body ● Pre-request Script Tests Settings

none form-data x-www-form-urlencoded [✓ raw] binary GraphQL JSON

```
{

    "fromAccountId":54212,
    "toAccountId":"12345",
    "amount" :10000
}
```

Body Cookies Headers (5) Test Results Status: 200 OK Time: 7 ms

Pretty Raw Preview Visualize JSON

```
[
    {
        "accountId": "54212",
        "balance": 990000
    },
    {
        "accountId": "12345",
        "balance": 1010000
    }
]
```

APPENDIX 2: EXPLANATION OF CODE THE CODE, BEST PRACTICES IN DETAILS

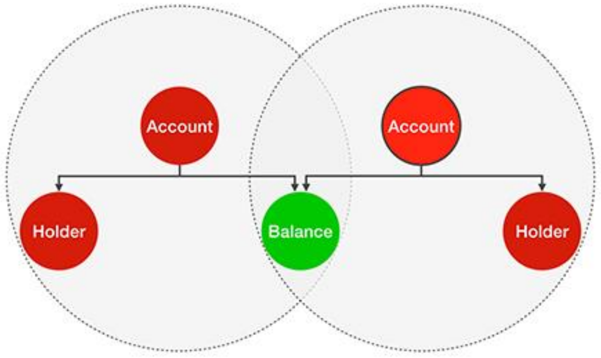
As given -->The amount to transfer should always be a positive number. It should not be possible for an account to end up with negative balance (we do not support overdrafts!)

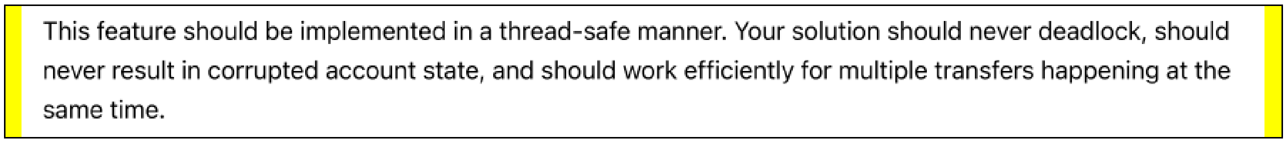
- For this context we decide to go for **immutability** because of below reason:

1. In a typical class, this validate() method would be called anytime a user's balance is changed. If the user makes a withdrawal, pays their debt, or transfers money from their account we would have to call the validate method. However, with an immutable class we only have to call the validate method only once.

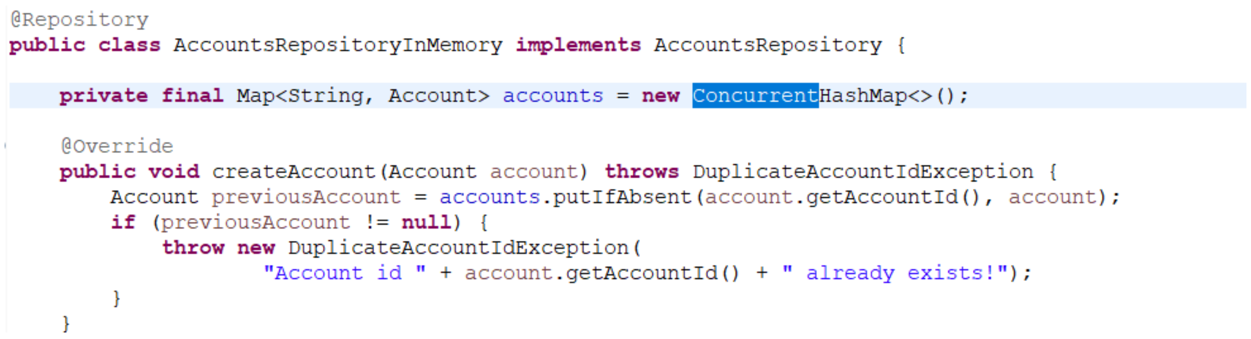
2. In case if any error occurs then immutable preventing user account from losing money before he physically receives it. In a normal class, money would be gone forever once account object was changed.
3. Also, immutable object can never get into an inconsistent state, even in the case of an exception. This stabilizes our system and removes the threat of an unforeseen error destabilizing an entire system.

So on, immutable is that they can be shared freely between objects and Immutable can even be shared freely when using a lock-free algorithm in a multithreaded environment, where multiple actions happen in parallel.




This feature should be implemented in a thread-safe manner. Your solution should never deadlock, should never result in corrupted account state, and should work efficiently for multiple transfers happening at the same time.

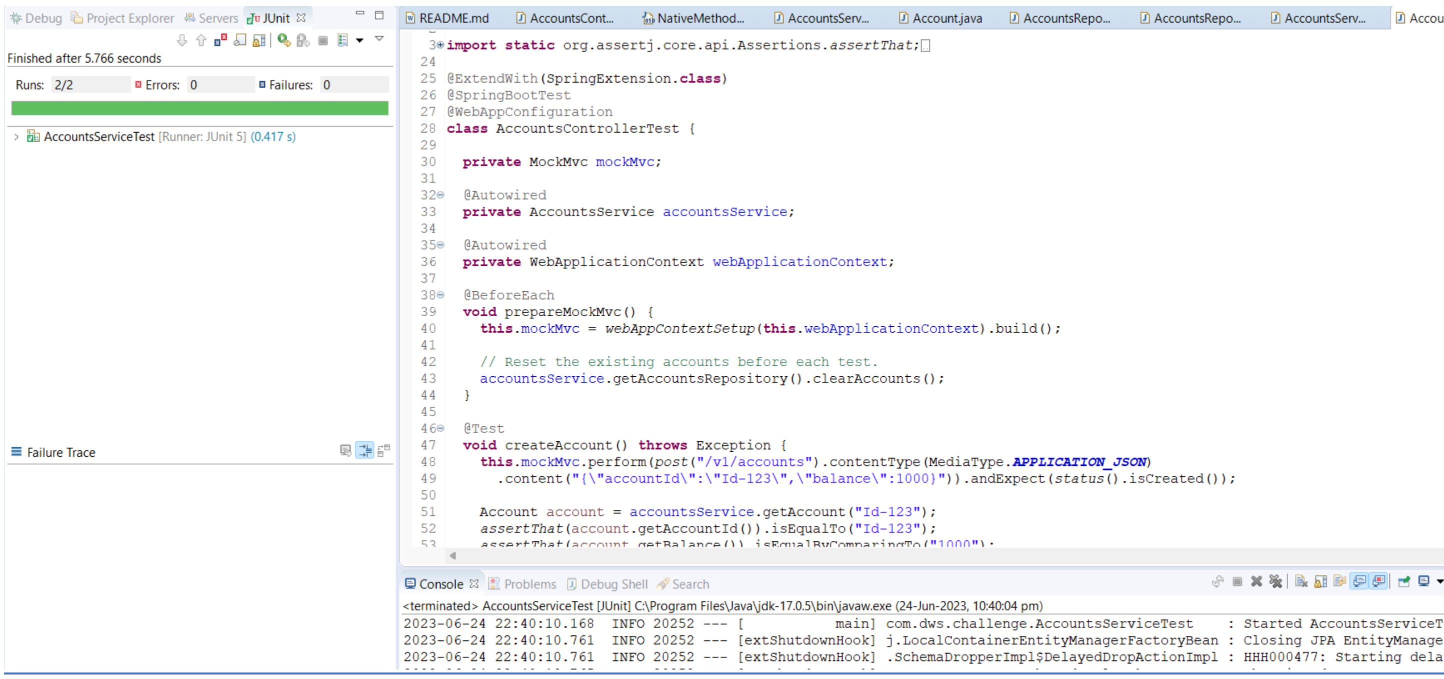
- For this,With in the same Transactional boundary we preventing dirty read ,phantom reed ,thred safty and dead lock .
- Also for the perfoarmance improvatnt we use concurrentHashMap rather then go for entire bucket level lock.


```
@Repository
public class AccountsRepositoryInMemory implements AccountsRepository {

    private final Map<String, Account> accounts = new ConcurrentHashMap<>();

    @Override
    public void createAccount(Account account) throws DuplicateAccountIdException {
        Account previousAccount = accounts.putIfAbsent(account.getAccountId(), account);
        if (previousAccount != null) {
            throw new DuplicateAccountIdException(
                    "Account id " + account.getAccountId() + " already exists!");
        }
    }
```

- Also, I wanted to implementing double checking for more memory efficient but time constraint it’s not part of this release.

Appendix 3: JUnit and Sonar TEST, MAKE code production ready

Mock all the service : JUnit Result

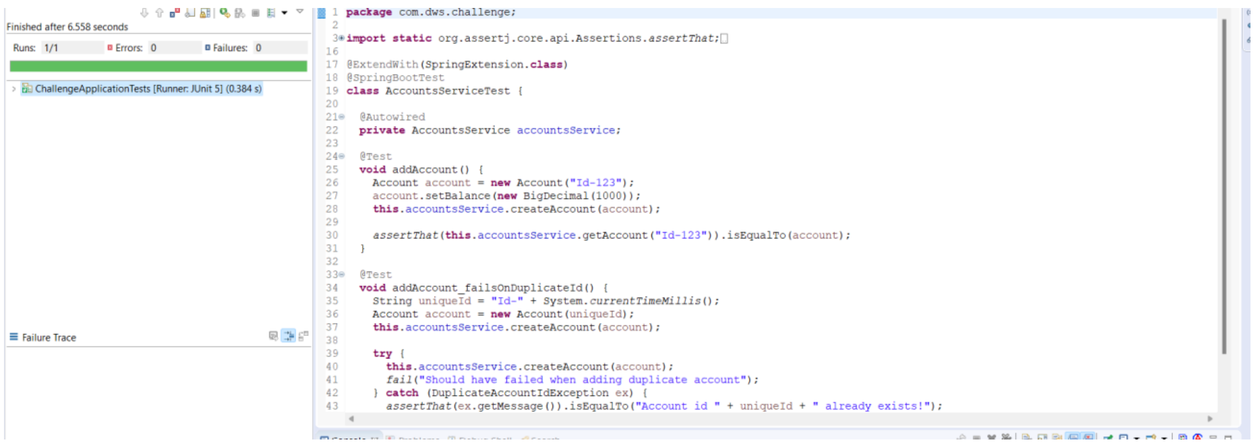


```
Finished after 5.766 seconds
Runs: 2/2    Errors: 0    Failures: 0
> AccountsServiceTest [Runner: JUnit 5] (0.417 s)
Failure Trace
```

```java
import static org.assertj.core.api.Assertions.assertThat;

@ExtendWith(SpringExtension.class)
@SpringBootTest
@WebAppConfiguration
class AccountsControllerTest {

  private MockMvc mockMvc;

  @Autowired
  private AccountsService accountsService;

  @Autowired
  private WebApplicationContext webApplicationContext;

  @BeforeEach
  void prepareMockMvc() {
    this.mockMvc = webAppContextSetup(this.webApplicationContext).build();

    // Reset the existing accounts before each test.
    accountsService.getAccountsRepository().clearAccounts();
  }

  @Test
  void createAccount() throws Exception {
    this.mockMvc.perform(post("/v1/accounts").contentType(MediaType.APPLICATION_JSON)
      .content("{\"accountId\":\"Id-123\",\"balance\":1000}")).andExpect(status().isCreated());

    Account account = accountsService.getAccount("Id-123");
    assertThat(account.getAccountId()).isEqualTo("Id-123");
    assertThat(account.getBalance()).isEqualByComparingTo("1000");
```

```
<terminated> AccountsServiceTest [JUnit] C:\Program Files\Java\jdk-17.0.5\bin\javaw.exe (24-Jun-2023, 10:40:04 pm)
2023-06-24 22:40:10.168  INFO 20252 --- [           main] com.dws.challenge.AccountsServiceTest     : Started AccountsServiceT
2023-06-24 22:40:10.761  INFO 20252 --- [extShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManage
2023-06-24 22:40:10.761  INFO 20252 --- [extShutdownHook] .SchemaDropperImpl$DelayedDropActionImpl : HHH000477: Starting dela
```

Test Cases for accountSevice

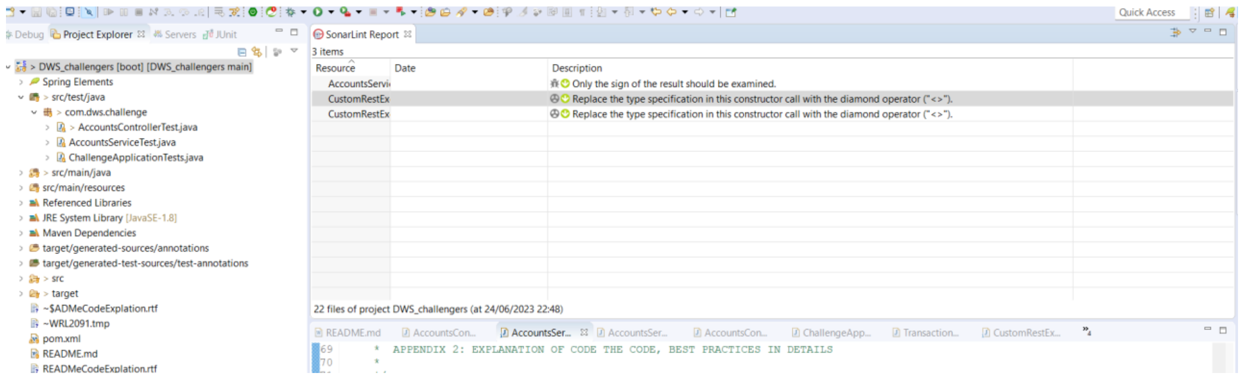


```
Finished after 6.558 seconds
Runs: 1/1    Errors: 0    Failures: 0
> ChallengeApplicationTests [Runner: JUnit 5] (0.384 s)
Failure Trace
```

```java
package com.dws.challenge;

import static org.assertj.core.api.Assertions.assertThat;

@ExtendWith(SpringExtension.class)
@SpringBootTest
class AccountsServiceTest {

  @Autowired
  private AccountsService accountsService;

  @Test
  void addAccount() {
    Account account = new Account("Id-123");
    account.setBalance(new BigDecimal(1000));
    this.accountsService.createAccount(account);

    assertThat(this.accountsService.getAccount("Id-123")).isEqualTo(account);
  }

  @Test
  void addAccount_failsOnDuplicateId() {
    String uniqueId = "Id-" + System.currentTimeMillis();
    Account account = new Account(uniqueId);
    this.accountsService.createAccount(account);

    try {
      this.accountsService.createAccount(account);
      fail("Should have failed when adding duplicate account");
    } catch (DuplicateAccountIdException ex) {
      assertThat(ex.getMessage()).isEqualTo("Account id " + uniqueId + " already exists!");
```

Sonar Complince

Code are almost Sonar Complince .



```
> DWS_challengers [boot] [DWS_challengers main]
  Spring Elements
  > src/test/java
    > com.dws.challenge
      > AccountsControllerTest.java
        AccountsServiceTest.java
        ChallengeApplicationTests.java
  > src/main/java
  src/main/resources
  Referenced Libraries
  JRE System Library [JavaSE-1.8]
  Maven Dependencies
  target/generated-sources/annotations
  target/generated-test-sources/test-annotations
  > src
  > target
  ~$ADMeCodeExplation.rtf
  ~WRL2091.tmp
  pom.xml
  README.md
  READMeCodeExplation.rtf
```

SonarLint Report

3 items

Resource	Date	Description
AccountsServi		Only the sign of the result should be examined.
CustomRestEx		Replace the type specification in this constructor call with the diamond operator ("<>").
CustomRestEx		Replace the type specification in this constructor call with the diamond operator ("<>").

22 files of project DWS_challengers (at 24/06/2023 22:48)

README.md | AccountsCon... | AccountsSer... | AccountsSer... | AccountsCon... | ChallengeApp... | Transaction... | CustomRestEx...

```
     * APPENDIX 2: EXPLANATION OF CODE THE CODE, BEST PRACTICES IN DETAILS
     *
```

Thanking You