**Exercise 3.1: Mocking and Stubbing**

**CODE:**

**BankClient Interface: -**

package org.example;  
  
public interface BankApiClient {  
 String transfer(String FromAccount,String ToAccount,int Amount);  
}

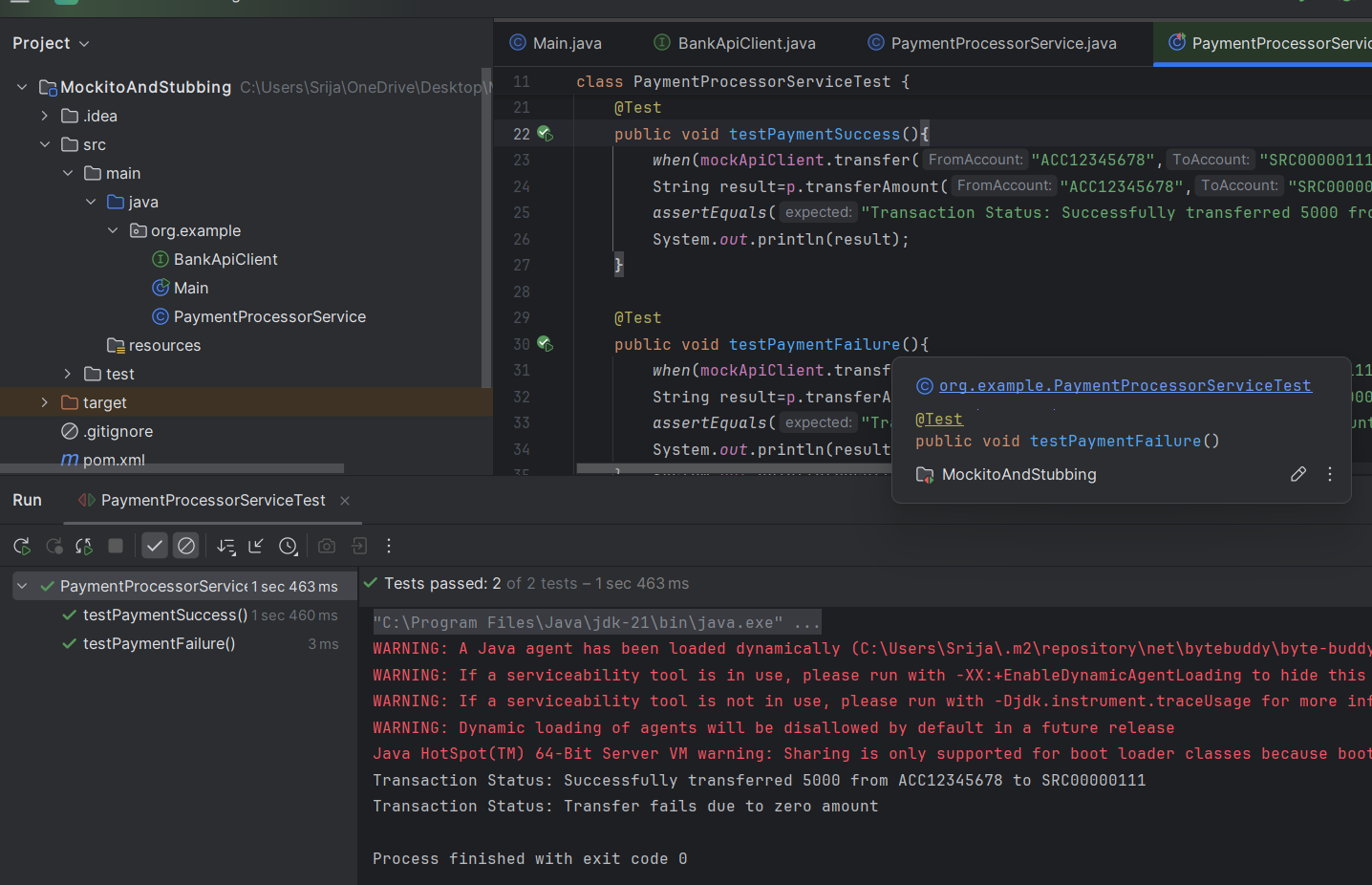
**PaymentProcessorService Class: -**

package org.example;  
  
public class PaymentProcessorService{  
 public BankApiClient bankApiClient;  
  
 public PaymentProcessorService(BankApiClient bankApiClient) {  
 this.bankApiClient = bankApiClient;  
 }  
  
 public String transferAmount(String FromAccount,String ToAccount,int Amount){  
 return "Transaction Status: "+bankApiClient.transfer(FromAccount,ToAccount,Amount);  
 }  
}

**PaymentProcessorServiceTest Class: -**

package org.example;  
  
import org.junit.jupiter.api.BeforeAll;  
import org.junit.jupiter.api.BeforeEach;  
import org.junit.jupiter.api.Test;  
  
import static org.junit.jupiter.api.Assertions.\*;  
import static org.mockito.Mockito.*mock*;  
import static org.mockito.Mockito.*when*;  
  
class PaymentProcessorServiceTest {  
  
 PaymentProcessorService p;  
 BankApiClient mockApiClient;  
 @BeforeEach  
 public void setUp(){  
 mockApiClient=*mock*(BankApiClient.class);  
 p=new PaymentProcessorService(mockApiClient);  
 }  
  
 @Test  
 public void testPaymentSuccess(){  
 *when*(mockApiClient.transfer("ACC12345678","SRC00000111",5000)).thenReturn("Successfully transferred 5000 from ACC12345678 to SRC00000111");  
 String result=p.transferAmount("ACC12345678","SRC00000111",5000);  
 *assertEquals*("Transaction Status: Successfully transferred 5000 from ACC12345678 to SRC00000111",result);  
 System.*out*.println(result);  
 }  
  
 @Test  
 public void testPaymentFailure(){  
 *when*(mockApiClient.transfer("ACC12345678","SRC00000111",0)).thenReturn("Transfer fails due to zero amount");  
 String result=p.transferAmount("ACC12345678","SRC00000111",0);  
 *assertEquals*("Transaction Status: Transfer fails due to zero amount",result);  
 System.*out*.println(result);  
 }  
  
}

**OUTPUT:**

****