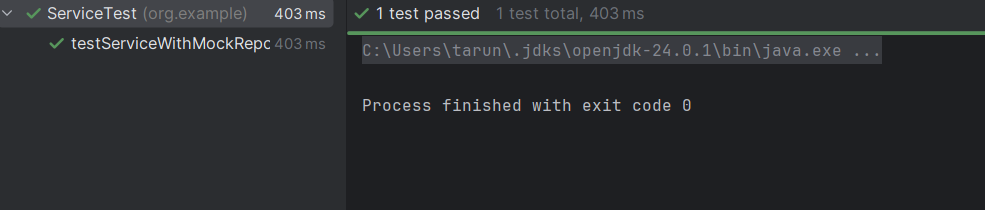
EXERCISE 1 (Mocking Databases and Repositories)

package org.example;  
public interface Repository {  
 String getData();  
}

package org.example;  
public class Service {  
 private final Repository repository;  
 public Service(Repository repository) {  
 this.repository = repository;  
 }  
 public String processData() {  
 String data = repository.getData();  
 return "Processed " + data;  
 }  
}

package org.example;  
import org.junit.jupiter.api.Test;  
import static org.mockito.Mockito.\*;  
import static org.junit.jupiter.api.Assertions.\*;  
public class ServiceTest {  
 @Test  
 public void testServiceWithMockRepository() {  
 Repository mockRepository = *mock*(Repository.class);  
 *when*(mockRepository.getData()).thenReturn("Mock Data");  
 Service service = new Service(mockRepository);  
 String result = service.processData();  
 *assertEquals*("Processed Mock Data", result);  
 }  
}

OUTPUT



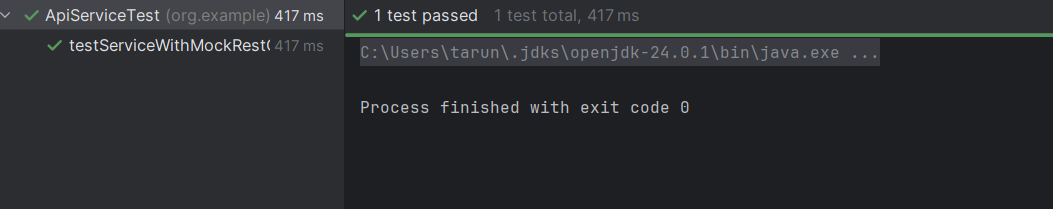
EXERCISE 2(Mocking External Services (RESTful APIs))

package org.example;  
public interface RestClient {  
 String getResponse();  
}

package org.example;  
public class ApiService {  
 private final RestClient restClient;  
 public ApiService(RestClient restClient) {  
 this.restClient = restClient;  
 }  
 public String fetchData() {  
 return "Fetched " + restClient.getResponse();  
 }  
}

package org.example;  
import org.junit.jupiter.api.Test;  
import static org.mockito.Mockito.\*;  
import static org.junit.jupiter.api.Assertions.\*;  
public class ApiServiceTest {  
 @Test  
 public void testServiceWithMockRestClient() {  
 RestClient mockRestClient = *mock*(RestClient.class);  
 *when*(mockRestClient.getResponse()).thenReturn("Mock Response");  
 ApiService apiService = new ApiService(mockRestClient);  
 String result = apiService.fetchData();  
 *assertEquals*("Fetched Mock Response", result);  
 }  
}

OUTPUT



EXERCISE 3 (Mocking File I/O)

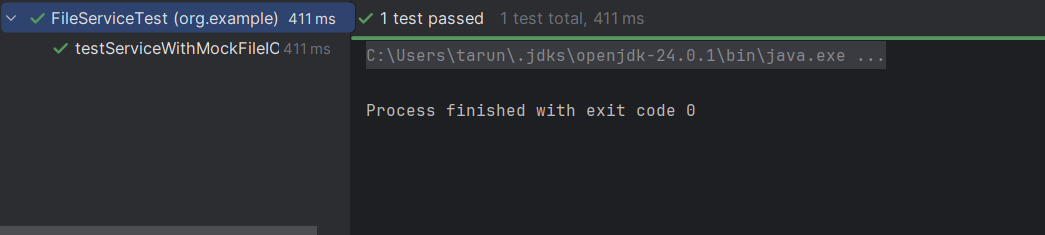
package org.example;  
public interface FileWriter {  
 void write(String content);  
}

package org.example;  
public interface FileReader {  
 String read();  
}

package org.example;  
public class FileService {  
 private final FileReader fileReader;  
 private final FileWriter fileWriter;  
 public FileService(FileReader fileReader, FileWriter fileWriter) {  
 this.fileReader = fileReader;  
 this.fileWriter = fileWriter;  
 }  
 public String processFile() {  
 String content = fileReader.read();  
 String processed = "Processed " + content;  
 fileWriter.write(processed);  
 return processed;  
 }  
}

package org.example;  
import org.junit.jupiter.api.Test;  
import static org.mockito.Mockito.\*;  
import static org.junit.jupiter.api.Assertions.\*;  
public class FileServiceTest {  
 @Test  
 public void testServiceWithMockFileIO() {  
 FileReader mockFileReader = *mock*(FileReader.class);  
 FileWriter mockFileWriter = *mock*(FileWriter.class);  
 *when*(mockFileReader.read()).thenReturn("Mock File Content");  
 FileService fileService = new FileService(mockFileReader, mockFileWriter);  
 String result = fileService.processFile();  
 *verify*(mockFileWriter).write("Processed Mock File Content");  
 *assertEquals*("Processed Mock File Content", result);  
 }  
}

OUTPUT



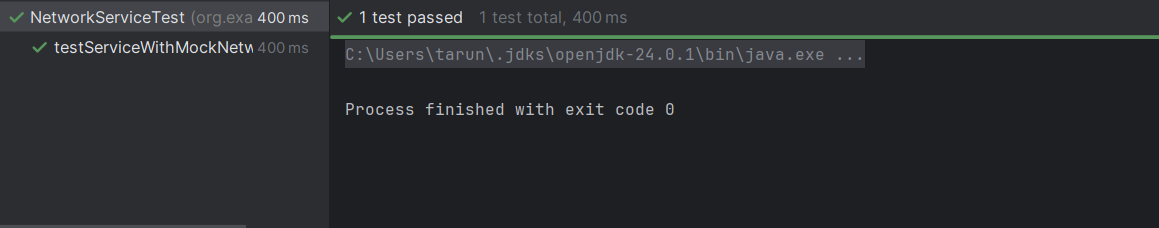
EXERCISE 4(Mocking Network Interactions)

package org.example;  
  
public interface NetworkClient {  
 String connect();  
}

package org.example;  
public class NetworkService {  
 private final NetworkClient networkClient;  
 public NetworkService(NetworkClient networkClient) {  
 this.networkClient = networkClient;  
 }  
 public String connectToServer() {  
 String connection = networkClient.connect();  
 return "Connected to " + connection;  
 }  
}

package org.example;  
  
import org.junit.jupiter.api.Test;  
import static org.mockito.Mockito.\*;  
import static org.junit.jupiter.api.Assertions.\*;  
public class NetworkServiceTest {  
 @Test  
 public void testServiceWithMockNetworkClient() {  
 NetworkClient mockNetworkClient = *mock*(NetworkClient.class);  
 *when*(mockNetworkClient.connect()).thenReturn("Mock Connection");  
 NetworkService networkService = new NetworkService(mockNetworkClient);  
 String result = networkService.connectToServer();  
 *assertEquals*("Connected to Mock Connection", result);  
 *verify*(mockNetworkClient).connect();   
 }  
}

OUTPUT



EXERCISE 5 (Mocking Multiple Return Values)

package org.example;  
public interface DataSource {  
 String fetch();  
}

package org.example;  
public class Processor {  
 private final DataSource dataSource;  
 public Processor(DataSource dataSource) {  
 this.dataSource = dataSource;  
 }  
 public String handleData() {  
 return "Handled " + dataSource.fetch();  
 }  
}

package org.example;  
import org.junit.jupiter.api.Test;  
import static org.junit.jupiter.api.Assertions.\*;  
import static org.mockito.Mockito.\*;  
public class ProcessorTest {  
 @Test  
 public void testMultipleReturnsFromMock() {  
 DataSource mockDataSource = *mock*(DataSource.class);  
 *when*(mockDataSource.fetch())  
 .thenReturn("First Value")  
 .thenReturn("Second Value");  
 Processor processor = new Processor(mockDataSource);  
 String firstResult = processor.handleData();  
 String secondResult = processor.handleData();  
 *assertEquals*("Handled First Value", firstResult);  
 *assertEquals*("Handled Second Value", secondResult);  
 }  
}

OUTPUT

