**Exercise 1 : Mocking and Stubbing**

**ExternalApi.java**

package com.example;

public interface ExternalApi {

String getData();

}

**MyService.java**

package com.example;

public class MyService {

private ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData();

}

}

**MyServiceTest.java**

package com.example;

import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.assertEquals;

import static org.mockito.Mockito.\*;

public class MyServiceTest {

public void testExternalApi() {

ExternalApi mockApi = mock(ExternalApi.class);

when(mockApi.getData()).thenReturn("Mock Data");

MyService service = new MyService(mockApi);

String result = service.fetchData();

assertEquals("Mock Data", result);

}

}

**Pom.xml**

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>mockito-demo</artifactId>

<version>1.0-SNAPSHOT</version>

<packaging>jar</packaging>

<dependencies>

<dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter-api</artifactId>

<version>5.9.3</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter-engine</artifactId>

<version>5.9.3</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-core</artifactId>

<version>5.12.0</version>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-surefire-plugin</artifactId>

<version>3.0.0-M9</version>

<configuration>

<useModulePath>false</useModulePath>

</configuration>

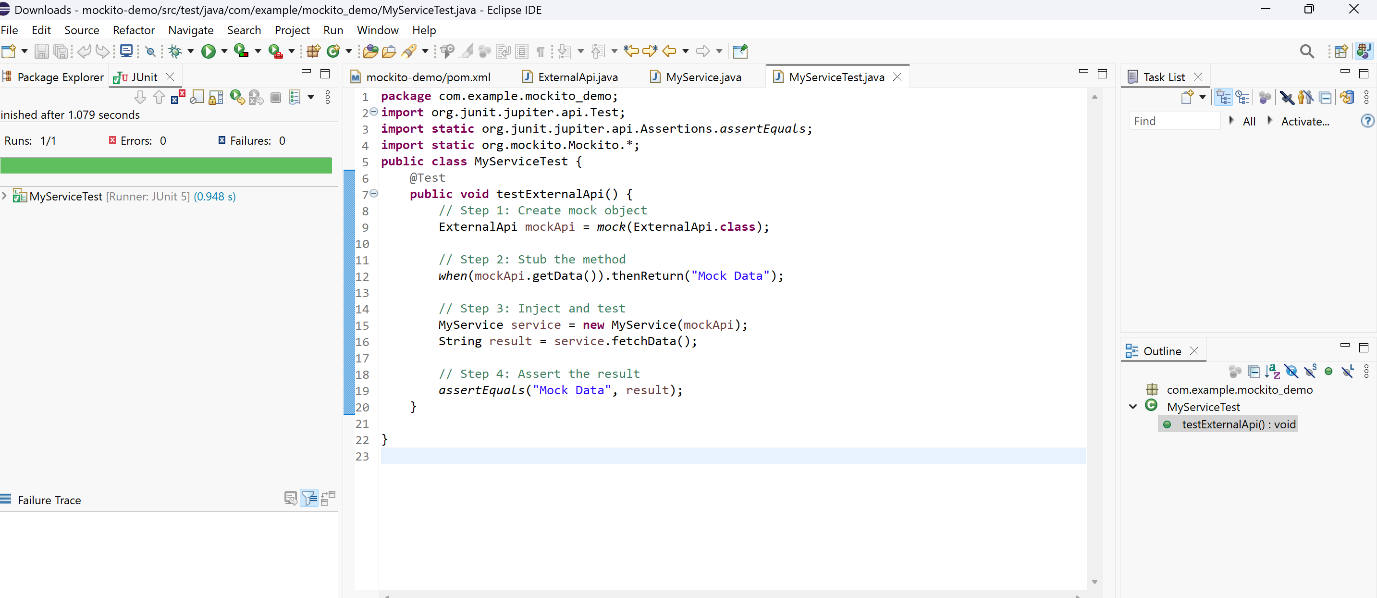
</plugin>

</plugins>

</build>

</project>

**OUTPUT**



**Exercise 2 : Verifying Interactions**

**ExternalApi.java**

package com.example;

public interface ExternalApi {

String getData();

}

**MyService.java**

package com.example;

public class MyService {

private ExternalApi api;

public MyService(ExternalApi api) {

this.api = api;

}

public String fetchData() {

return api.getData();

}

}

**MyServiceTest.java**

package com.example;

import org.junit.jupiter.api.Test;

import static org.mockito.Mockito.\*;

public class MyServiceTest {

@Test

public void testVerifyInteraction() {

ExternalApi mockApi = mock(ExternalApi.class);

MyService service = new MyService(mockApi);

service.fetchData();

verify(mockApi).getData();

}

}

**Pom.xml**

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>mockito-demo</artifactId>

<version>1.0-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter-api</artifactId>

<version>5.9.3</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter-engine</artifactId>

<version>5.9.3</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-core</artifactId>

<version>5.12.0</version>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-surefire-plugin</artifactId>

<version>3.0.0-M9</version>

<configuration>

<useModulePath>false</useModulePath>

</configuration>

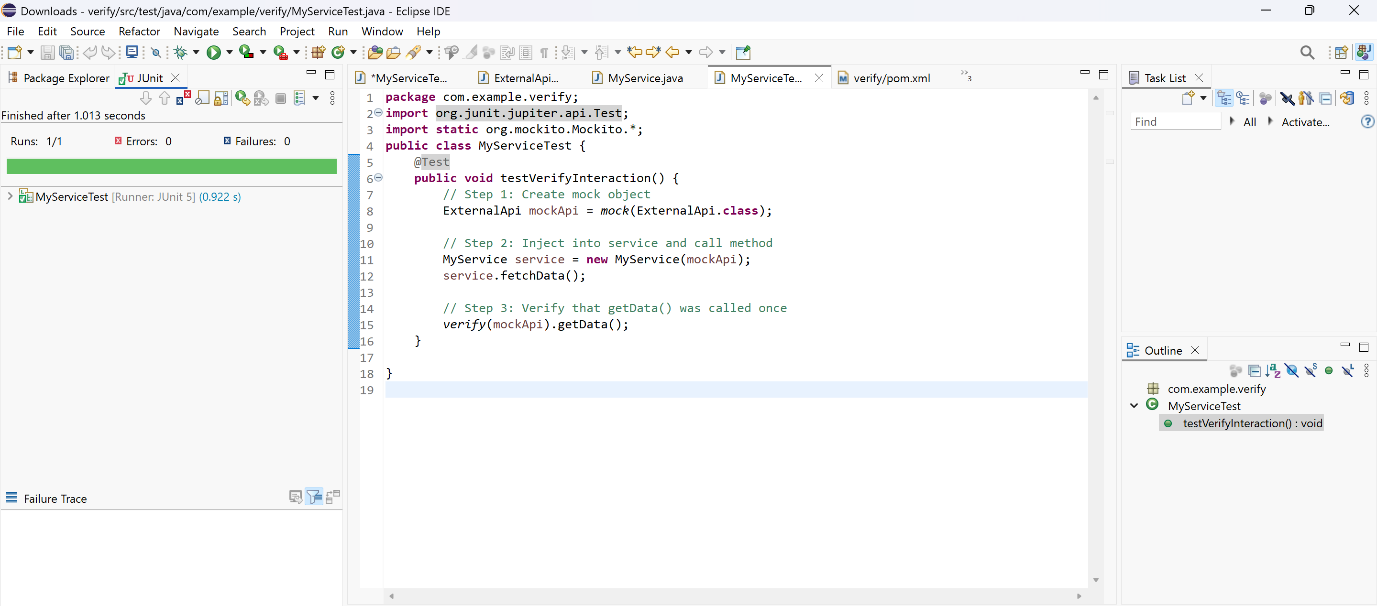
</plugin>

</plugins>

</build>

</project>

**OUTPUT**

****

**Exercise 3 : Argument Matching**

**NotificationService.java**

package com.example;

public interface NotificationService {

void sendMessage(String userId, String message);

}

**Userservice.java**

package com.example;

public class UserService {

private NotificationService notificationService;

public UserService(NotificationService notificationService) {

this.notificationService = notificationService;

}

public void notifyUser(String userId) {

notificationService.sendMessage(userId, "Welcome!");

}

}

**UserServiceTest.java**

package com.example;

import org.junit.jupiter.api.Test;

import static org.mockito.Mockito.\*;

import static org.mockito.ArgumentMatchers.\*;

public class UserServiceTest {

@Test

public void testSendMessageCalledWithCorrectArguments() {

NotificationService mockNotification = mock(NotificationService.class);

UserService userService = new UserService(mockNotification);

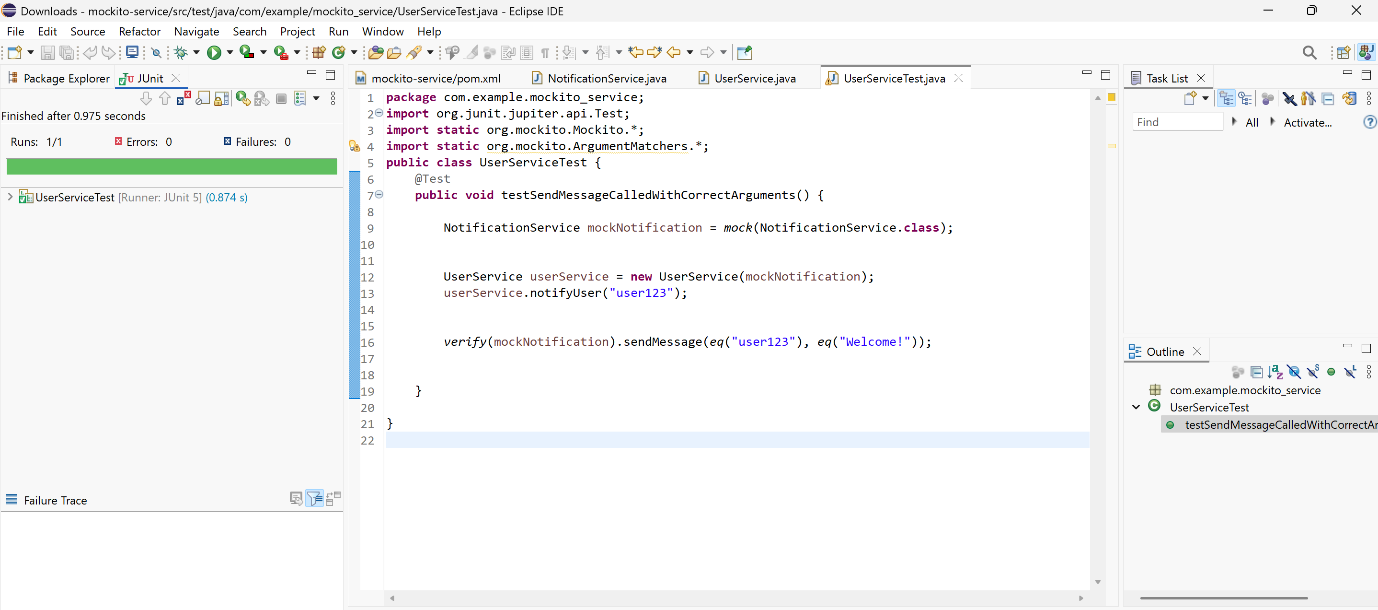
userService.notifyUser("user123");

verify(mockNotification).sendMessage(eq("user123"), eq("Welcome!"));

}

}

**OUTPUT**



**Exercise 4 : Handling Void Methods**

**Logger.java**

package com.example;

public interface Logger {

void log(String message);

}

**AuditService.java**

package com.example;

public class AuditService {

private Logger logger;

public AuditService(Logger logger) {

this.logger = logger;

}

public void recordEvent(String event) {

logger.log("Event recorded: " + event);

}

}

**AuditServiceTest.java**

package com.example;

import org.junit.jupiter.api.Test;

import static org.mockito.Mockito.\*;

public class AuditServiceTest {

@Test

public void testLogMethodCalled() {

Logger mockLogger = mock(Logger.class);

doNothing().when(mockLogger).log(anyString());

AuditService service = new AuditService(mockLogger);

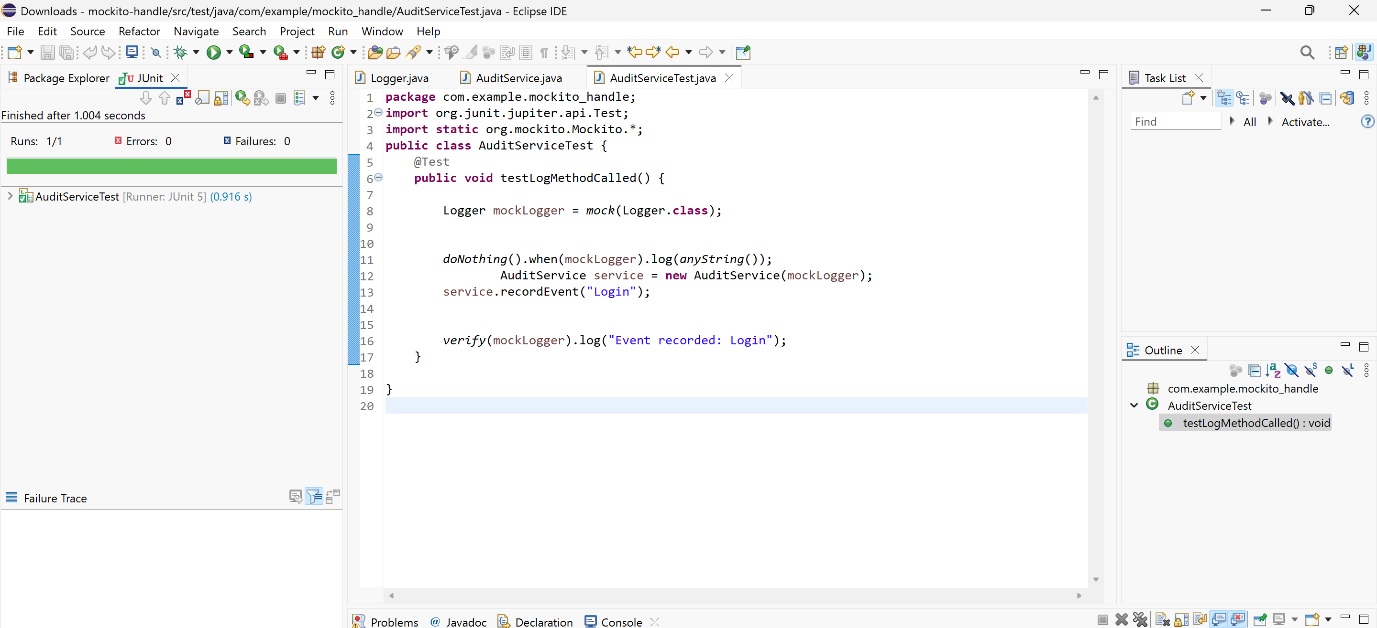
service.recordEvent("Login");

verify(mockLogger).log("Event recorded: Login");

}

}

**OUTPUT**



**Exercise 5 : Mocking and Stubbing with Multiple Returns**

**ExternalApi.java**

package com.example;

public interface ExternalApi {

String getStatus();

}

**StatusService.java**

package com.example;

public class StatusService {

private ExternalApi api;

public StatusService(ExternalApi api) {

this.api = api;

}

public String[] fetchStatuses() {

return new String[] {

api.getStatus(),

api.getStatus(),

api.getStatus()

};

}

}

**StatusServiceTest.java**

package com.example;

import org.junit.jupiter.api.Test;

import static org.mockito.Mockito.\*;

import static org.junit.jupiter.api.Assertions.\*;

public class StatusServiceTest {

@Test

public void testMultipleReturnValues() {

ExternalApi mockApi = mock(ExternalApi.class);

when(mockApi.getStatus())

.thenReturn("Online")

.thenReturn("Busy")

.thenReturn("Offline");

StatusService service = new StatusService(mockApi);

String[] result = service.fetchStatuses();

assertEquals("Online", result[0]);

assertEquals("Busy", result[1]);

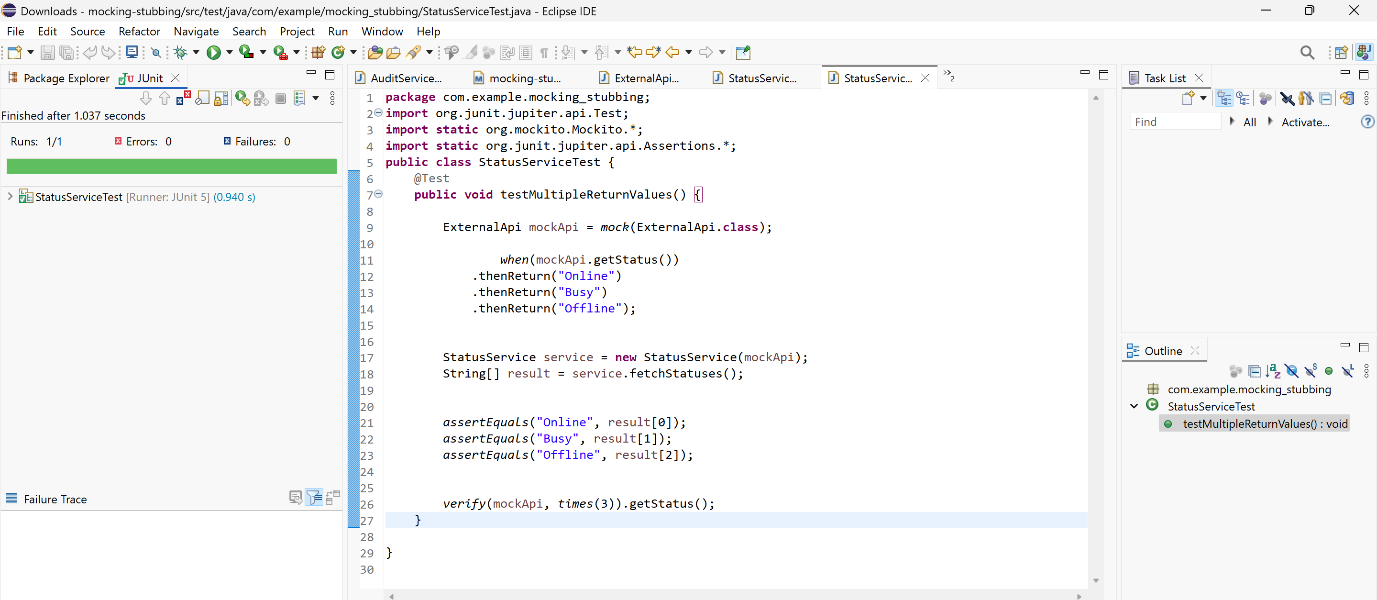
assertEquals("Offline", result[2]);

verify(mockApi, times(3)).getStatus();

}

}

**OUTPUT**



**Exercise 6 : Verifying Interaction Order**

**BankService.java**

package com.example;

public interface BankService {

void authenticate(String userId);

void withdraw(String userId, double amount);

void logout(String userId);

}

**ATM.java**

package com.example;

public class ATM {

private BankService bankService;

public ATM(BankService bankService) {

this.bankService = bankService;

}

public void processTransaction(String userId, double amount) {

bankService.authenticate(userId);

bankService.withdraw(userId, amount);

bankService.logout(userId);

}

}

**ATMTest.java**

package com.example;

import org.junit.jupiter.api.Test;

import static org.mockito.Mockito.\*;

public class ATMTest {

@Test

public void testMethodCallOrder() {

BankService mockBankService = mock(BankService.class);

ATM atm = new ATM(mockBankService);

atm.processTransaction("user123", 100.0);

InOrder inOrder = inOrder(mockBankService);

inOrder.verify(mockBankService).authenticate("user123");

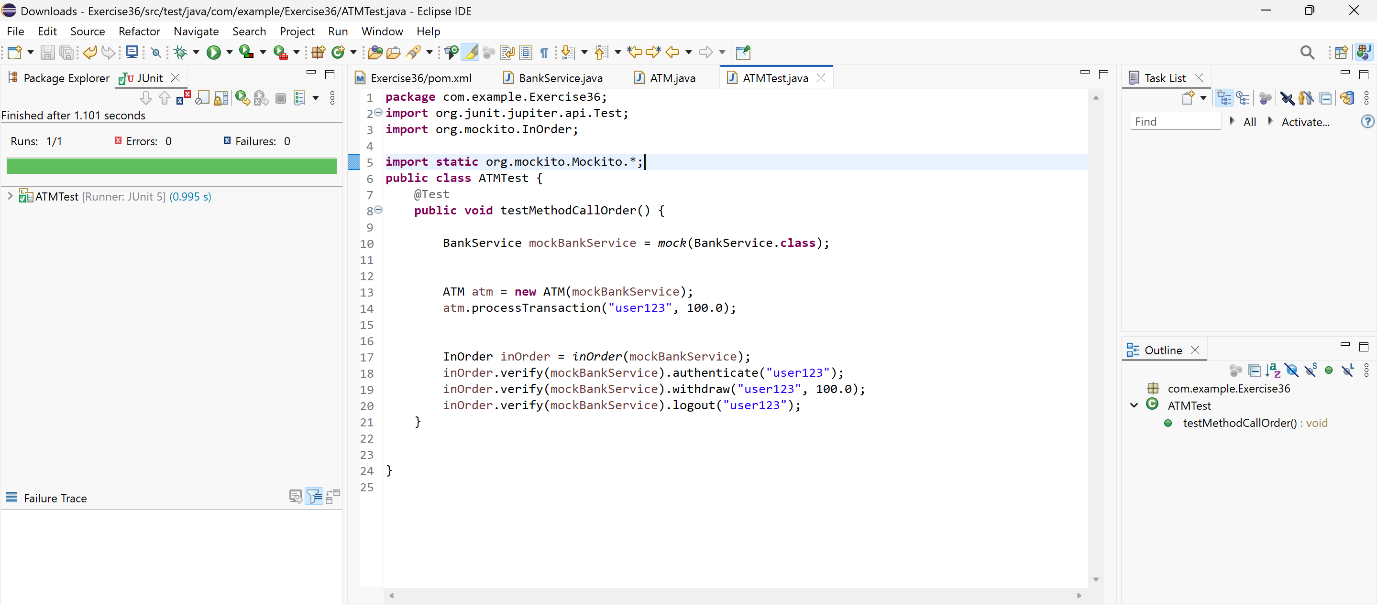
inOrder.verify(mockBankService).withdraw("user123", 100.0);

inOrder.verify(mockBankService).logout("user123");

}

}

**OUTPUT**

****

**Exercise 7 : Handling Void Methods with Exceptions**

**Logger.java**

package com.example;

public interface Logger {

void log(String message);

}

**ErrorReporter.java**

package com.example;

public class ErrorReporter {

private Logger logger;

public ErrorReporter(Logger logger) {

this.logger = logger;

}

public void reportError(String error) {

logger.log(error);

}

}

**ErrorReporterTest.java**

package com.example;

import org.junit.jupiter.api.Test;

import static org.mockito.Mockito.\*;

import static org.junit.jupiter.api.Assertions.\*;

public class ErrorReporterTest {

@Test

public void testVoidMethodThrowsException() {

Logger mockLogger = mock(Logger.class);

doThrow(new RuntimeException("Log failed"))

.when(mockLogger).log("ERROR");

ErrorReporter reporter = new ErrorReporter(mockLogger);

RuntimeException exception = assertThrows(RuntimeException.class, () -> {

reporter.reportError("ERROR");

});

assertEquals("Log failed", exception.getMessage());

verify(mockLogger).log("ERROR");

}

}

**OUTPUT**

