Testing Exceptions:
Properly Testing Exceptions in Java

Lesson Objectives

- Understand the need for testing exceptions in Java
- Get acquainted with the JUnit 5 and Mockito support for exception testing
- Learn various strategies for effective exception testing

Exception Testing Basics in JUnit 5

assertThrows Example

```
@Test
void whenExceptionThrown_thenAssertionSucceeds() {

    Exception exception = assertThrows(NumberFormatException.class, () -> {
        Integer.parseInt("1a");
    });

    assertEquals("For input string: \"1a\"", exception.getMessage());

}
```

Mockito Support for Exception Testing

thenThrow Example

```java
@Test
void whenReadFile_thenThrowException() {
    File file = mock(File.class);
    when(file.read()).thenThrow(new IOException());

    MyFileReader fileReader = new MyFileReader(file);

    assertThrows(IOException.class, () -> fileReader.readFile());
}
```

Advanced Exception Testing Strategies

Parameterized Tests

```
@ParameterizedTest
@ValueSource(strings = {"", "  ", "\t", "\n"})
void whenGivenBlankString_thenIllegalArgumentExceptionIsThrown(String input) {
    MyParser parser = new MyParser();

    assertThrows(IllegalArgumentException.class, () => parser.parse(input));
}
```

Parameterized Tests with sources of exceptions

```
@ParameterizedTest
@MethodSource("provideExceptionsForTesting")
void whenVariousExceptionsThrown_thenHandleGracefully(Exception e, Class<?> expectedType) {
    PaymentProcessor processor = new PaymentProcessor();

    assertThrows(
        expectedType,
        () -> processor.processPayment(e)
    );
}

static Stream<Arguments> provideExceptionsForTesting() {
    return Stream.of(
        Arguments.of(new NetworkException("Timeout reached"), NetworkException.class),
        Arguments.of(new PaymentDeclinedException("Payment declined by bank"),
PaymentDeclinedException.class),
        Arguments.of(new UserNotLoggedInException("User must be logged in"),
AuthenticationException.class)
    );
}
```

Combining Mockito and JUnit 5

```
@Test
void whenAuthorizationServiceIsUnavailable_thenAuthServiceCatchesException() {
    AuthorizationService authzService = mock(AuthorizationService.class);
    when(authzService.authorize(anyString())).thenThrow(new ServiceUnavailableException("Authorization
service is unavailable"));

    AuthService authService = new AuthService(authzService);

    ServiceUnavailableException thrown = assertThrows(
        ServiceUnavailableException.class,
        () -> authService.authenticate("user", "password")
    );

    assertTrue(thrown.getMessage().contains("Authorization service is unavailable"));
    verify(authzService).authorize(anyString());
}
```