Mockito を Java テストで使用する

Java のテストで Mockito を使用するには、モックオブジェクトを作成し、メソッドをスタブ化し、インタラクションを検証するための以下の手順に従ってください：

1. 依存関係の追加

`pom.xml` (Maven) に Mockito と JUnit を含めます：

```
<dependency>
    <groupId>org.mockito</groupId>
    <artifactId>mockito-core</artifactId>
    <version>5.3.1</version>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter</artifactId>
    <version>5.9.2</version>
    <scope>test</scope>
</dependency>
```

2. テストクラスの設定

JUnit 5 と Mockito の拡張機能を使用します：

```
import org.junit.jupiter.api.extension.ExtendWith;
import org.mockito.junit.jupiter.MockitoExtension;

@ExtendWith(MockitoExtension.class)
public class UserServiceTest {
    // ...
}
```

3. モックの作成と依存関係の注入

- モックしたい依存関係には `@Mock` を使用します。
- モックをテスト対象のクラスに注入するには `@InjectMocks` を使用します。

```
@Mock
private UserRepository userRepository;
```

```
@InjectMocks
private UserService userService; // UserService は UserRepository に依存しています
```

4. メソッド呼び出しのスタブ化

`when().thenReturn()` を使用してモックの動作を定義します：

```
@Test
void getUserById_ReturnsUser_WhenUserExists() {
    User mockUser = new User(1L, "Alice");
    // リポジトリのメソッドをスタブ化
    when(userRepository.findById(1L)).thenReturn(mockUser);

    User result = userService.getUserById(1L);

    assertEquals("Alice", result.getName());
}
```

5. インタラクションの検証

モックメソッドが期待通りに呼び出されたかどうかを確認します：

```
@Test
void getUserById_CallsRepository() {
    userService.getUserById(1L);
    // リポジトリメソッドが ID 1 で一度呼び出されたことを確認
    verify(userRepository, times(1)).findById(1L);
}
```

一般的な使用例

例外のスタブ化

```
@Test
void getUserById_ThrowsException_WhenRepositoryFails() {
    when(userRepository.findById(anyLong())).thenThrow(new RuntimeException("DB Error"));

    assertThrows(RuntimeException.class, () -> userService.getUserById(1L));
}
```

引数マッチャー `any()`, `eq()` などを使用して引数を柔軟にマッチングします：

```
when(userRepository.findById(anyLong())).thenReturn(new User(1L, "Bob"));
```

BDD（行動駆動開発）スタイル 読みやすさを高めるために `given().willReturn()` を使用します：

```
import static org.mockito.BDDMockito.*;

@Test
void bddStyleExample() {
    given(userRepository.findById(1L)).willReturn(new User(1L, "Alice"));

    User result = userService.getUserById(1L);

    then(userRepository).should().findById(1L);
    assertEquals("Alice", result.getName());
}
```

引数キャプチャ 詳細なアサーションを行うために引数をキャプチャします：

```
@Test
void saveUser_CapturesArgument() {
    ArgumentCaptor<User> userCaptor = ArgumentCaptor.forClass(User.class);

    userService.saveUser("Charlie");

    verify(userRepository).save(userCaptor.capture());
    assertEquals("Charlie", userCaptor.getValue().getName());
}
```

重要なポイント

- **初期化**: `@ExtendWith(MockitoExtension.class)` (JUnit 5) または `@BeforeEach` で `MockitoAnnotations.openMocks(this)` を使用します。
- **静的モック**: 静的メソッド/コンストラクタをモックするために `mockito-inline` 依存関係を使用します。
- **スパイ**: 実際のオブジェクトをラップするために `@Spy` を使用します（部分的なモック）。

トラブルシューティング

- **Null モック**: モックが初期化されていることを確認します（`@ExtendWith` または `openMocks()` を使用）。
- **スタブエラー**: テストで呼び出される前にメソッドをスタブ化します。

高度な使用については、Mockito ドキュメントを参照してください。