1. Pesquisa

Definições:

Dummy Object: Um objeto que é apenas passado como argumento para preencher a assinatura do método, sem interagir ou afetar o comportamento do teste.

Fake Object: Um objeto funcional, mas com uma implementação simplificada ou substituída, que simula o comportamento de um componente real em um teste.

Test Spy: Um objeto que grava e verifica interações com ele, mas possui uma implementação real, permitindo que se verifique o comportamento de chamadas de método.

2. Diferenças principais:

Dummy Object: Não faz nada, serve apenas para compilar o teste.

Fake Object: Implementa funcionalidade real, porém simplificada.

Test Spy: Além de implementar funcionalidades, grava e permite a verificação de interações.

3. Exemplos práticos:

Dummy Object: Usado em testes de APIs que exigem parâmetros, mas esses parâmetros não são utilizados no teste.

Fake Object: Um banco de dados em memória que simula o comportamento de um banco real.

Test Spy: Um objeto de "logging" que grava chamadas feitas a métodos para que o teste possa verificar se esses métodos foram chamados corretamente.

2 EasyMock

3 teste e comentários

BDDMockito para Behavior-Driven Development (BDD): facilita a integração do estilo BDD com given, when e then, estruturando melhor os testes.

@Captor, @Spy e @InjectMocks: simplificam a criação de capturadores, espiões e injeção de dependências, agilizando a configuração de testes.

Verificação com timeout: permite verificar interações com um limite de tempo, útil para testes em sistemas concorrentes.

4 resumos comparativo

Mockito: Permite criar mocks, spies e stubs facilmente com uma API fluida e legível. É amplamente usado para verificar comportamentos (mocks) e criar stubs. Suas anotações (@Mock, @Spy, @InjectMocks) simplificam configurações complexas, e sua integração com BDD é destacada.

JMock: Usa um estilo mais declarativo, adequado para programação orientada a mensagens, permitindo verificar interações precisas entre objetos. Facilita a criação de dublês focados em comportamento, embora a sintaxe seja mais difícil.

EasyMock: Baseado em expectativas, torna fácil verificar chamadas em uma sequência específica, útil para dublês focados em interações. No entanto, exige a definição explícita de todas as interações esperadas, o que pode dificultar a configuração em testes mais dinâmicos.