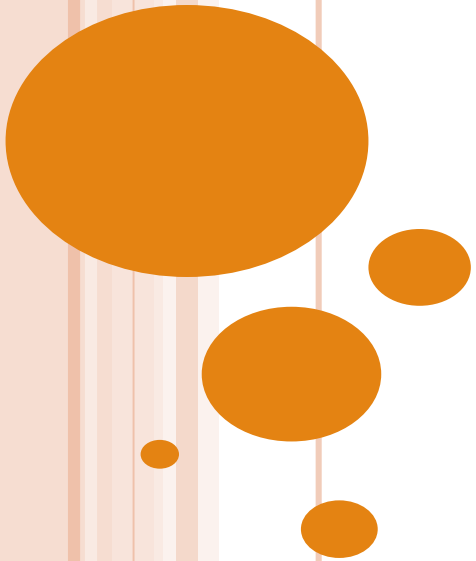
Automação de Teste - EasyMock

Teste de Software

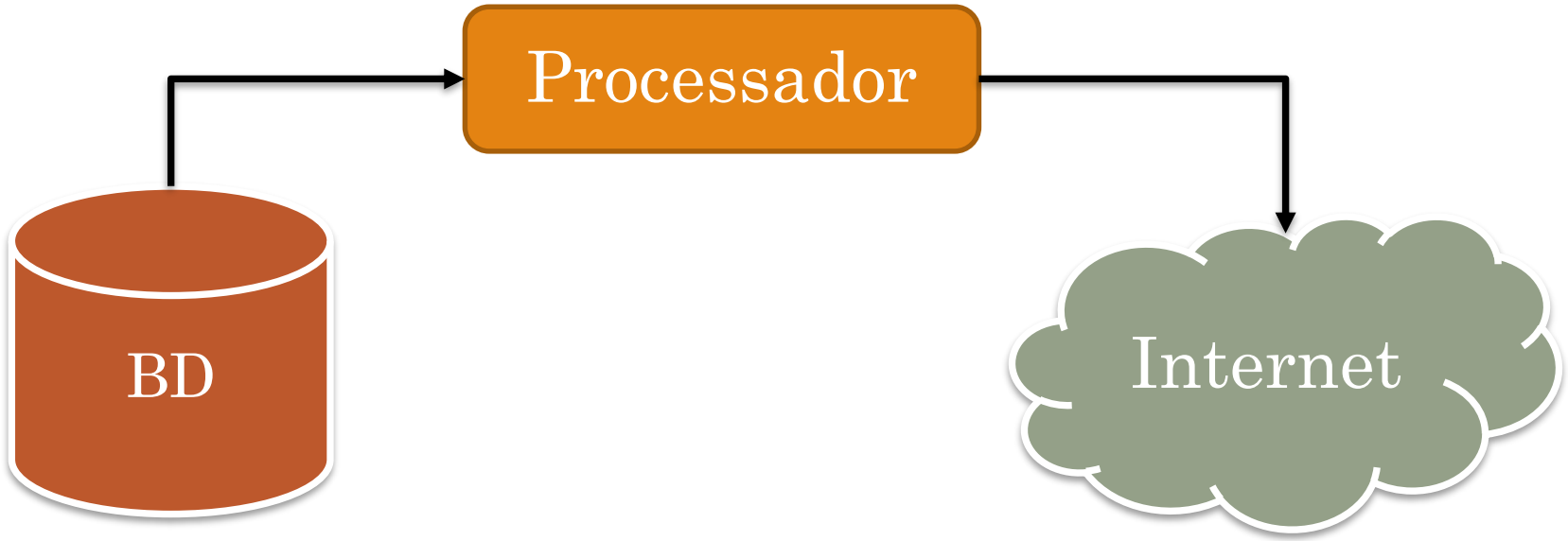


Roteiro

- Estudo de Caso
- Objetos Mock
  - Introdução
- EasyMock
  - Introdução
  - Configuração do EasyMock
  - Passos para Trabalhar com EasyMock
  - Exemplo prático

Estudo de Caso

- Imaginemos um programa **"Processador"** recebendo informações de uma "**Fonte de Origem**", em seguida processando os dados e finalmente enviando os dados para uma "**Fonte de Destino**".



Estudo de Caso

- Além da classe **Processador**, podemos abstrair o seguinte:

Uma classe **FonteOrigem** que implementa a interface FonteOrigem com alguma responsabilidade;

Exemplo:
Obter dados de um banco de dados, ou até mesmo um arquivo txt;

Uma classe **FonteDestino** que implementa a interface FonteDestino com alguma outra responsabilidade;

Exemplo:
Enviar dados da fonte origem pela rede usando WebSockets;

Como podemos criar testes para esse exemplo ?

Estudo de Caso

- A princípio para fazer esse teste deveríamos implementar além da classe Processador:

  - A classe **FonteOrigem**:
    - Acrescentar os drivers necessários para acessar o banco de dados,
    - Criar os arquivos de configuração,
    - Fazer a conexão com o banco de dados,
    - Criar alguma consulta e retorna-la para Processador.

  - E também a classe **FonteDestino**:
    - Adicionar os drivers necessários,
    - Criar possíveis arquivos de configuração,
    - Criar todo o código que faça a movimentação dos dados pela rede,
    - Etc.

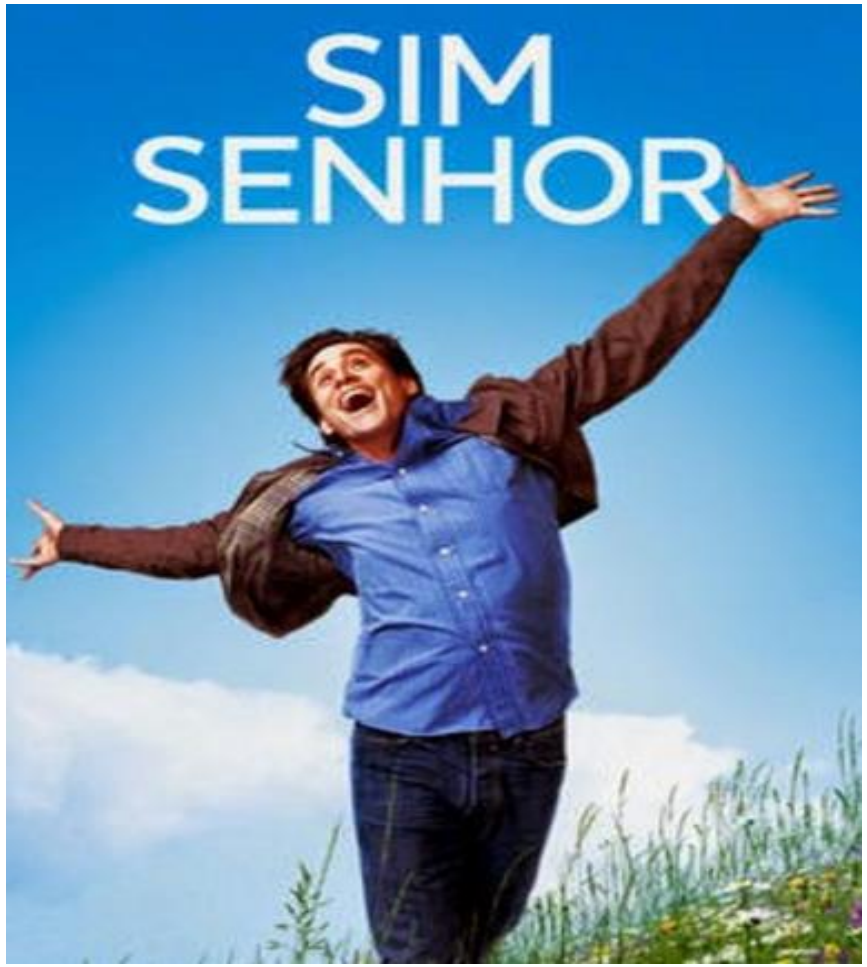
Estudo de Caso

- Depois de fazer todas essas implementações ainda poderemos ter problemas

- Por exemplo:
  - Banco de Dados pode não estar disponível em algum momento do teste;
  - A Rede pode também não estar disponível em um momento do teste;
  - Ou ainda, ter algum teste problemático atrasando o desenvolvimento;
  - Etc...

Estudo de Caso

- Será se poderíamos testar somente a classe **Processador** sem as implementações das classes **FonteOrigem** e **FonteDestino** ?



Estudo de Caso

- Precisamos de uma solução que não precise implementar nenhuma classe de dependência, nesse caso, as classes,
  - **FonteOrigem**
  - **FonteDestino**

- Para isso, podemos começar as implementações de Objetos Mock

- Esses objetos permitem simular o comportamento de outros objetos ainda não implementados
  - Simular não é implementar, e sim, emular objetos reais.

Introdução aos Objetos Mock

- São objetos vazios que simplesmente simulam as dependências dos métodos testados
- Permitem testar métodos de uma classe cujas dependências ainda não foram implementadas;
  - Objetos mocks não contém lógica de programação, apenas simulam um comportamento.
- Na realização dos testes (em tempo de execução), as dependências são substituídas por Objetos Mock
  - Processo conhecido como injeção de dependência
- Podem ser criados pelo próprio desenvolvedor
  - Ou a partir de alguma biblioteca existente

Introdução aos Objetos Mock

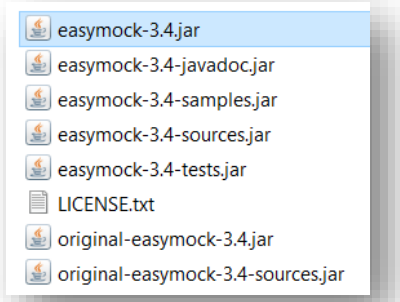
- Diversas linguagens de programação possuem bibliotecas para criação de Objetos Mock
- Algumas linguagens
  - .NET
    - NMockLib, Rhino, Nmock e TypeMock
  - Ruby
    - RSpec e FlexMock
  - PHP
    - SimpleTest, SnapTest e PHPUnit
  - Delphi
    - Pascal Mock
  - Java
    - Mockito, JMock, MockLib, HibernateMock e **EasyMock**

Introdução ao EasyMock

- Uma biblioteca Java para criação de Objetos Mock, em tempo de execução
- Objetos Mock gerados pelo EasyMock implementam qualquer classe ou interface da aplicação
  - Uma boa prática é programar para interfaces, nunca para classes
- EasyMock:
  - http://easymock.org/

Configuração do EasyMockt

- Clique em Download para baixar o arquivo (easymock-3.4.jar)

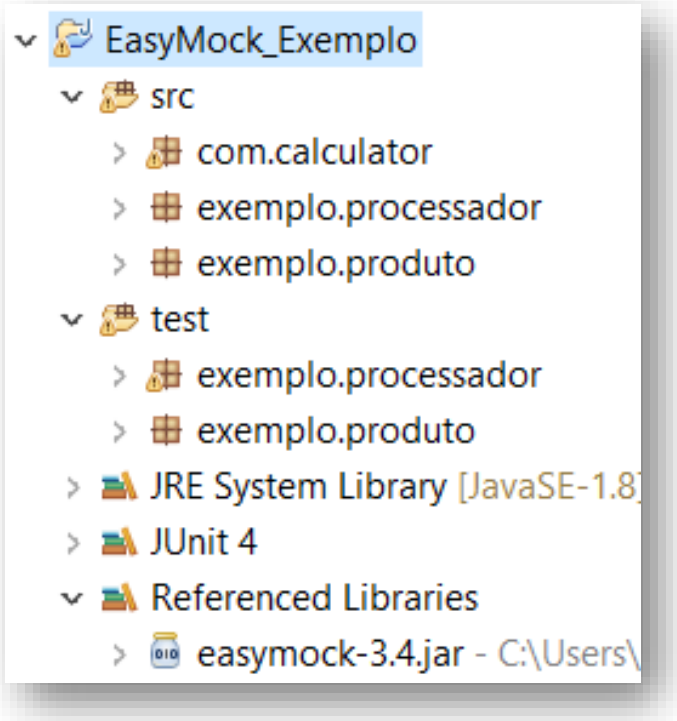


- Ou Dependência **Maven**

```
<dependency>
  <groupId>org.easymock</groupId>
  <artifactId>easymock</artifactId>
  <version>3.4</version>
  <scope>test</scope>
</dependency>
```

Configuração do EasyMockt

- Exemplo de esquema de projeto



Exemplo Prático

- Criação das interfaces
  - IFonteOrigem.java
  - IFonteDestino.java

```java
public interface IFonteOrigem {
    int lerDados();
}
```

```java
public interface IFonteDestino {
    void gravarDados(int valor);
}
```

Exemplo Prático

- Criação da classe
  - Processador.java

```
public class Processador {
    private IFonteOrigem iFonteOrigem;
    private IFonteDestino iFonteDestino;

    public Processador(IFonteOrigem iFonteOrigem, IFonteDestino iFonteDestino){
        this.iFonteOrigem = iFonteOrigem;
        this.iFonteDestino = iFonteDestino;
    }
    public void processarDados() throws Exception{
        int valor = 0;
        try{
            valor = iFonteOrigem.lerDados();
        }catch(Exception e){
            throw new Exception("Erro ao ler os dados...");
        }
        if(valor > 0){
            int resultado = tranformarValor(valor);
            try{
                iFonteDestino.gravarDados(resultado);
            }catch(Exception e){
                throw new Exception("Erro ao gravar os dados...");
            }
        }
    }
    private int tranformarValor(int valor) {
        int resto = 0;
        int valor_i = 0;
        while(valor != 0){
            resto = valor % 10;
            valor = valor / 10;
            valor_i = (valor_i * 10) + resto;
        }
        return valor_i;
    }
```

Exemplo Prático

- Na criação de objetos Mock ainda usamos a biblioteca do JUnit, então criamos a classe de teste com o JUnit Test Case:
  - TesteProcessador.java

```
package exemplo.processador;

import org.junit.Test;

public class TesteProcessador {

    @Test
    public void test() {

    }
```

Passos para Trabalhar com EasyMock

1. Criar o OBJETO MOCK
2. Realizar a injeção de dependência para a classe dependente
3. Criar o comportamento do OBJETO MOCK
4. Chamar o método a ser testado
5. Verificar o retorno do método testado
   - Se necessário
6. Verificar se o OBJETO MOCK foi chamado

Exemplo Prático EasyMock

- Classe testadora
  - TesteProcessador.java

```java
public class TesteProcessador {

    @Test
    public void testProcessarDados() {
        // 1-> Criar os OBJETOS MOCK
        IFonteOrigem fonteOrigem = EasyMock.createMock(IFonteOrigem.class);
        IFonteDestino fonteDestino = EasyMock.createMock(IFonteDestino.class);

        // 2 -> Realizar a injeção de dependência
        Processador processador = new Processador(fonteOrigem, fonteDestino);

        // 3 -> Criar o comportamento dos OBJETOS MOCK
        EasyMock.expect(fonteOrigem.lerDados()).andReturn(2345);
        fonteDestino.gravarDados(5432);
        EasyMock.replay(fonteOrigem, fonteDestino); // Ativação dos OBJETOS MOCK

        // 4 -> Chamar o metodo a ser testado
        try {
            processador.processarDados();
        } catch (Exception e) {
            e.printStackTrace();
        }

        // 5 -> Ferificar se os OBJETOS MOCKs foram chamados
        EasyMock.verify(fonteOrigem, fonteDestino);
    }
```

Fim

Referências

- Mocks: Introdução a Automatização de Testes com Mock Object. Disponível em: < http://www.devmedia. com.br/ mocks-introducao-a-automatizacao-de-testes-com-mock-object/30641 >