



xUnit e JUnit

JUnit

- JUnit é um framework para testes de unidade em Java.
- Foi escrito por Erich Gamma and Kent Beck.
- Este framework fornece suporte para a escrita dos testes de unidade, execução destes testes e apresentação dos resultados do teste, facilitando a repetição dos testes.

xUnit

A ideia do JUnit foi portada para outras linguagens, incluindo C# (NUnit), Python (PyUnit), Fortran (fUnit) and C++ (CPPUnit).

Esta família de frameworks para testes de unidade é referenciada como xUnit.

xUnit

Os frameworks do xUnit compartilham um conjunto de características:

- Especificam um teste como um método de teste (*Test Method*).
- Especificam os resultados esperados dentro do método de teste como chamadas a métodos de asserção (*Assertion Methods*).
- Agregam os testes dentro de suítes de testes que podem ser rodadas como uma única operação.
- Rodam um ou mais testes e mostram um relatório com os resultados.

xUnit

Cada teste é representado por um *Test Method* que implementa as 4 fases:

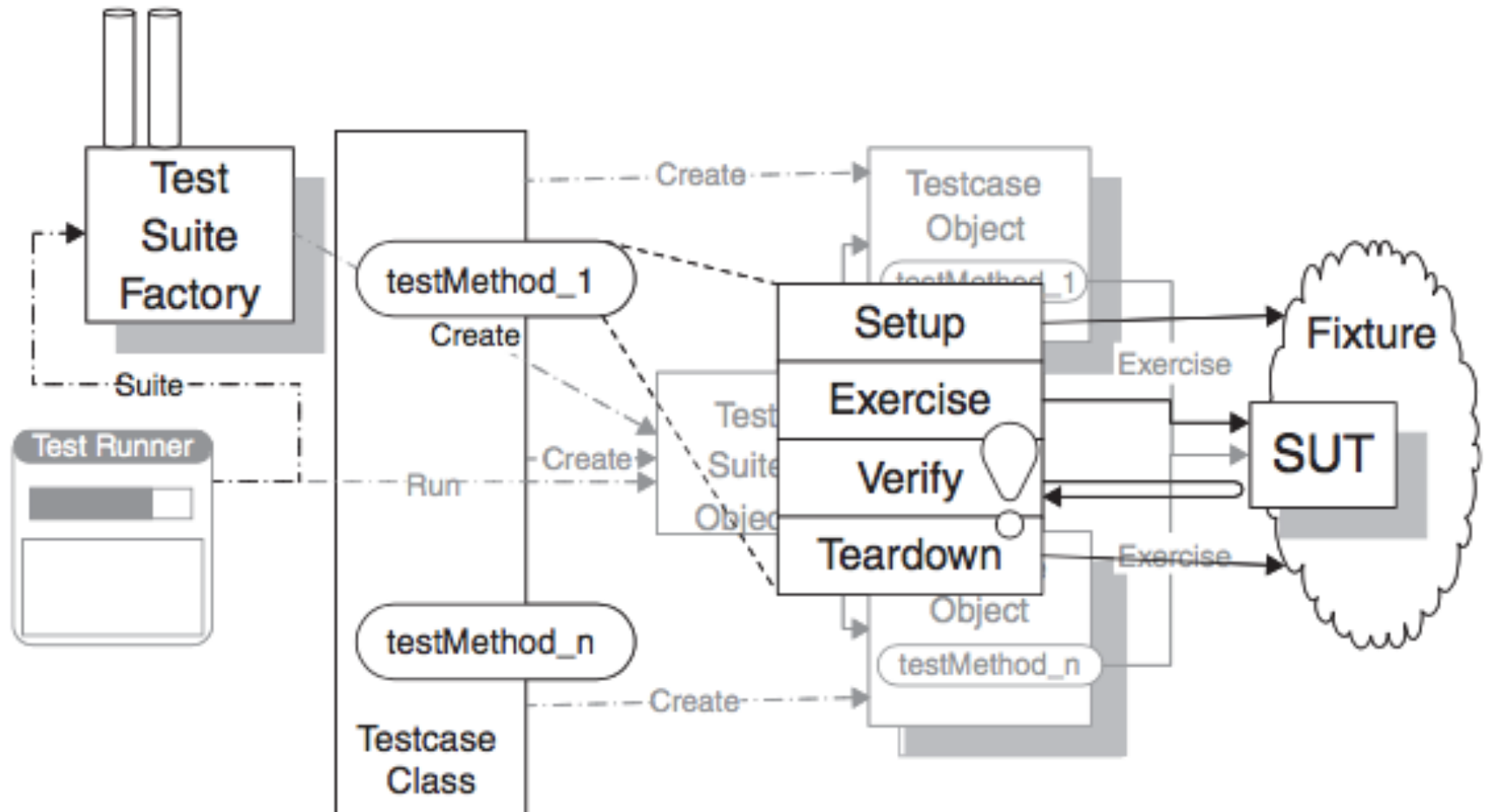
- Fixture setup,
- Exercise SUT,
- Result verification,
- Fixture teardown.

xUnit



- xUnit trabalha melhor quando a API de um sistema está sendo testada e são testadas classes ou pequenos grupos de classes isoladas.

xUnit



xUnit

Os testes são rodados usando um *Test Runner*. Diferentes Test Runners são disponíveis:

- *Graphical Test Runner*: fornece uma maneira visual para o usuário especificar, invocar e observar os resultados de uma suíte de testes que foram rodados.
- *Command-Line Test Runner*: os testes são executados por comando de linha.

xUnit

Os resultados dos testes são verificados automaticamente. Em geral é seguido o princípio de Hollywood (“Don’t call us; we’ll call you”), ou seja, os testes irão chamar você quando um problema ocorrer.

xUnit

Os resultados dos testes são classificados dentro de 3 categorias:

- Quando um teste roda sem erro ou falha, ele é considerado um sucesso. Nada acontece neste caso.
- Um teste apresenta uma falha quando uma assertção falha.
- Um teste apresenta um erro quando o SUT ou o teste falha de uma maneira inesperada.

Exemplo de Teste com Junit (1)

```
package testes;
```

```
import org.junit.*;  
import static org.junit.Assert.*;  
import modelo.*;
```

```
public class TesteCartorioEleitoral {
```

```
    private CartorioEleitoral cartorio;
```

```
    @Before
```

```
    public void iniciaTestes() {  
        cartorio = new CartorioEleitoral();  
    }
```

```
    @Test
```

```
    public void testa1Eleitor(){  
        cartorio.cadastraEleitor("Eleitor 1", 10001, 1001, "01/01/1990");  
        assertEquals(1, cartorio.numeroEleitores());  
    }
```

Exemplo de Teste com Junit (2)

@Test

```
public void testa2EleitoresDiferentes(){
    cartorio.cadastraEleitor("Eleitor 1", 10001, 1001, "01/01/1990");
    cartorio.cadastraEleitor("Eleitor 2", 10002, 1002, "02/02/1990");
    assertEquals(2, cartorio.numeroEleitores());
}
```

@Test

```
public void testa2EleitoresIguais() {
    cartorio.cadastraEleitor("Eleitor 1", 10001, 1001, "01/01/1990");
    try {
        cartorio.cadastraEleitor("Eleitor 1", 10001, 1001, "01/01/1990");
    }
    catch (RuntimeException e){}
    assertEquals(1, cartorio.numeroEleitores());
}
}
```


JUnit

- Classe **Assert**: possui os métodos de asserção.

```
import org.junit.*;
```

```
import static org.junit.Assert.*;
```

```
import modelo.*;
```



O “import static” permite chamar os métodos da classe sem o nome da classe na frente.

Isso permite chamar os métodos de asserção sem o nome da classe **Assert**.

JUnit

- **@Before:** O método com a anotação @Before é executado antes de cada teste.
- Os atributos do teste que são setados no método com a notação @Before representam informações comuns a todos os testes.

@Before

```
public void iniciaTestes() {  
    cartorio = new CartorioEleitoral();  
}
```

O atributo cartório é reinicializado antes da execução de cada teste.

JUnit

- **@After:** O método com anotação @After é executado depois de cada teste.
- Normalmente ele “limpa” tudo o que for necessário após cada teste.

JUnit

- **@Test:** Todos os métodos com a anotação @Test são considerados casos de teste pelo TestRunner do JUnit.

@Test

```
public void testa1Eleitor(){  
    cartorio.cadastraEleitor("Eleitor 1", 10001, 1001, "01/01/1990");  
    assertEquals(1, cartorio.numeroEleitores());  
}
```


JUnit

- **@BeforeClass:** O método estático com anotação `@BeforeClass` é executado uma vez antes da execução de todos os testes.
- **@AfterClass:** O método estático com anotação `@AfterClass` é executado uma vez após a execução de todos os testes.

JUnit - Asserts

- `assertTrue(Boolean condition)`
- `assertEquals(Object expected, Object actual)`
- `assertEquals(int expected, int actual)`
- `assertEquals(double expected, double actual, double tolerance)`
- `assertSame(Object expected, Object actual)`
- `assertNull(Object testobject)`
- `assertFalse(Boolean condition)`