**Estudando TDD com Delphi**

Nesse post vou demonstrar com alguns desafios de programação a forma que estou utilizando para estudar TDD no Delphi. Vou utilizar o Delphi X4, mas qualquer versão do Delphi pode ser utilizada para isso. Deem uma lida nesse post <http://www.johnidouglas.com.br/instando-o-dunit-no-delphi5-delphi-6-delphi7/>. A ideia é bem simples, vamos criar um grupo de projetos na IDE do Delphi com o projeto da aplicação e o projeto de teste. Sugiro fazer dessa forma para garantir a organização do seu projeto. Então vamos lá.

Na IDE do Delphi crie o projeto da aplicação: File->VCL Forms Applications -Delphi.

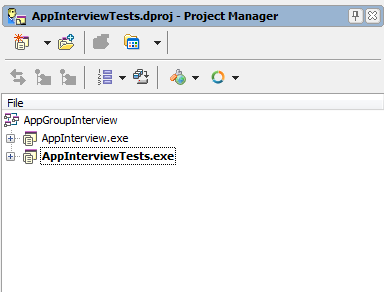
Depois crie o projeto de teste: File->New->Other->Unit Test->Test Project.

Observe que ao criar o projeto de teste uma wizard ira nos auxiliar nas configurações, utilize as opções sugeridas pela wizard.

Nota: Lembre-se que Test Runner deve ser configurado como **GUI**.

Salve todos os arquivos do projeto em um local e com o nome de sua preferência.

Veja um exemplo de como deve ficar a estrutura de seu projeto.

[](http://www.johnidouglas.com.br/wp-content/uploads/2013/08/Project-Mananger.png)

A estrutura do projeto esta pronta, agora vamos criar as units que serão usadas para implementar os algoritmos da solução de cada desafio e seus testes.

A ideia desse projeto é utilizar métodos estáticos para implementar os algoritmos.

No projeto da aplicação crie uma nova unit (File->Unit -Delphi) . Esse unit vai ser utilizada para implementar uma classe com os métodos estáticos.

Salve essa unit com o nome **QuestionsInterview.pas**.

Nessa unidade crie uma classe com o nome **TSampleInterview**.

Veja o exemplo da unit depois de criada.

[delphi]

unit QuestionsInterview;

interface

uses

System.SysUtils;

type

TSampleInterview = class

public

// aqui vamos implementar os métodos estáticos

end;

implementation

end;

[/delphi]

Agora vamos criar um test case para a nossa unidade.

Acesse o Menu File->Other->Unit Test->Test Case.

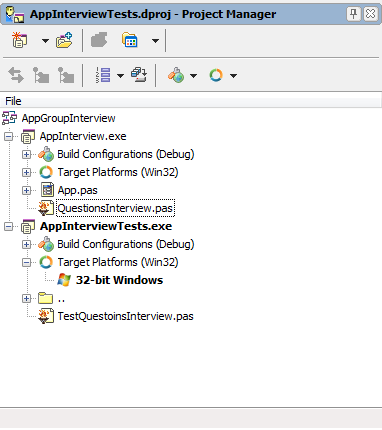
Na wizard de criação do test case escolha a unidade que vai ser a base dos testes, no nosso caso **QuestionsInterview.pas**.

Veja que a wizard já cria a estrutura da classe de teste com os métodos SetUp e TearDown.

Nota: Não há necessidade de mantermos o objeto **FSampleInterview** criado pela wizard, dessa forma exclua todas as referencias a esse objeto na unit de teste.

Salve essa unit com o nome **TestQuestoinsInterview.pas**.

Veja como ficou a estrutura de nosso projeto.

[](http://www.johnidouglas.com.br/wp-content/uploads/2013/08/Project-mananger-01.png)

Compile os dois projetos para verificar se tudo esta correto. Nesse momento seu projeto esta pronto para ser codificado. Abaixo será apresentado os desafios e como devem ser implementados usando TDD. Vamos começar a implementar os algoritmos pelos seus testes seguindo as boas práticas de TDD. Primeiro Desafio: Descobrir se um número é impar.

**Passo 01**: Na unidade **TestQuestoinsInterview.pas** seção published da classe **TestTSampleInterview** crie uma procedure de teste chamada TestIsOdd e implemente esse teste.

[delphi]

public

procedure SetUp; override;

procedure TearDown; override;

published

procedure TestIsOdd;

[/delphi]

Implementação do teste.

[delphi]

procedure TestTSampleInterview.TestIsOdd;

begin

CheckTrue( TSampleInterview.IsOdd( 5 ) );

end;

[/delphi]

**Passo 02**: Tente compilar e executar o teste. Vai dar erro na compilação, pois não temos o método IsOdd implementado. Nota: Não se esqueça de ativar o projeto de teste antes de compilar.

**Passo 03**: Implemente o método IsOdd que verifica se um número é impar. Esse método deve ser implementado na unidade QuestionsInterview.pas classe TSampleInterview seção public.

[delphi]

TSampleInterview = class

public

class function IsOdd( const ANumber: Integer ): Boolean;

end;

[/delphi]

Implementação do método IsOdd.

[delphi]

// Check number is odd

class function TSampleInterview.IsOdd(const ANumber: Integer): Boolean;

begin

Result:= (ANumber mod 2 ) = 0;

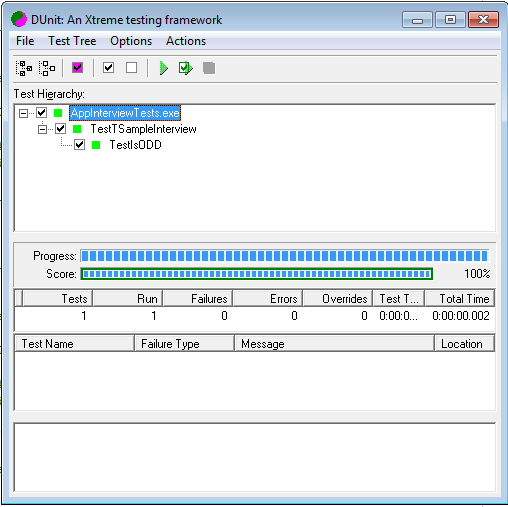
end;

[/delphi]

**Passo 04**: Compile e execute o teste novamente.

**Resultado**: O objetivo nesse ponto é que o teste não esteja falhando.

Veja uma imagem com o resultado do teste.

[](http://www.johnidouglas.com.br/wp-content/uploads/2013/08/DUNIT.png)

Como o algoritmo e bem simples crie mais um método de teste com o nome TestIsNotOdd que ira verificar se o número não é impar.

[delphi]

procedure TearDown; override;

published

procedure TestIsOdd;

procedure TestIsNotOdd;

[/delphi]

**Implementacao do teste**

[delphi]

procedure TestTSampleInterview.TestIsNotOdd;

begin

CheckFalse( TSampleInterview.IsOdd( 2 ) );

end;

[/delphi]

Compile e execute novamente os testes para ver o resultado. O resultado deve ser semelhante ao teste executado no método TestIsOdd. Segundo desafio: Descobrir se um numero é potencia de 2. Esse algoritmo é mais complexo e vai exigir mais esforço. Mas a ideia é seguir os mesmo passos do primeiro desafio.

**Passo 01**: Novamente comece pelo teste. Crie e implemente um método de teste chamado TestIsPowerTwo na classe TestTSampleInterview.

[delphi]

published

procedure TestIsODD;

procedure TestIsNotODD;

procedure TestPowerTwo;

[/delphi]

**Implementação do teste.**

[delphi]

procedure TestTSampleInterview.TestPowerTwo;

begin

CheckTrue( TSampleInterview.IsPowerTwo( 2 ) );

end;

[/delphi]

**Passo 02**: Tente compilar e executar o teste, vai dar erro de compilação, ainda não temos o método IsPowerTwo implementado.

**Passo 03**: Implemente o método IsPowerTwo na classe TSampleInterview .

[delphi]

class function TSampleInterview.IsPowerTwo(

const ANumber: Integer): Boolean;

begin

Result:= ANumber = 2;

end;

[/delphi]

**Passo 04**: Execute o teste.

**Resultado**: O resultado nesse ponto e o teste sendo executado sem falhas. Veja que ainda não implementamos o algoritmo que ira descobrir se um número é potência de 2, mas a ideia nesse momento é que o teste apenas seja executado.

Implemente mais métodos de teste para esse algoritmo.

Observe que os testes vão falhar conforme eles forem sendo implementados. Isso é normal ocorrer, o ideal e ir refatorando a função IsPowerTwo. A medida que os testes vão falhando vá implementando para que os novos testes não falhem.

Observe que os testes já implementados irão garantir que as modificações feitas durante a mudança da função continuem funcionando.

O Importante aqui é o algoritmo ser executado com a maior quantidade de possibilidades de retorno. Veja uma lista de testes sugeridos para esse algoritmo.

[delphi]

procedure TestTSampleInterview.TestIsPowerTwo0;

begin

CheckFalse( TSampleInterview.IsPowerTwo( 0 ) );

end;

procedure TestTSampleInterview.TestIsPowerTwo1;

begin

CheckFalse( TSampleInterview.IsPowerTwo( 1 ) );

end;

procedure TestTSampleInterview.TestIsPowerTwo1024;

begin

CheckTrue( TSampleInterview.IsPowerTwo( 1024 ) );

end;

procedure TestTSampleInterview.TestIsPowerTwo16;

begin

CheckTrue( TSampleInterview.IsPowerTwo( 16 ) );

end;

procedure TestTSampleInterview.TestIsPowerTwo512;

begin

CheckTrue( TSampleInterview.IsPowerTwo( 512 ) );

end;

procedure TestTSampleInterview.TestIsPowerTwo100;

begin

CheckTrue( TSampleInterview.IsPowerTwo( 100) );

end;

procedure TestTSampleInterview.TestIsPowerTwo200;

begin

CheckTrue( TSampleInterview.IsPowerTwo( 200) );

end;

procedure TestTSampleInterview.TestIsPowerTwo1000;

begin

CheckTrue( TSampleInterview.IsPowerTwo( 1000) );

end;

[/delphi]

**Implementação do algoritmo que garante a execução de todos os testes.**

[delphi]

class function TSampleInterview.IsPowerTwo(

const ANumber: Integer): Boolean;

var

Number: Integer;

begin

Result:= False;

Number:= ANumber;

while ( not (Number in [0, 1] ) ) and ( ( Number mod 2 ) = 0 ) do

begin

Number := Number – Trunc( Number / 2 );

if ( Number = 1 ) then

Result:= True;

end;

end;

[/delphi]

O projeto completo com mais desáfios esta liberado no GitHub, https://github.com/johnidm/AppInterview.

Não há dúvida que a sua vida vai ficar mais fácil com o TDD principalmente para quem trabalha com aplicações legadas e faz muita refatoração de código.

Tive a oportunidade de aplicar TDD em alguns projetos e o resultado é fantástico.

Para se aprofundar no assunto recomendo darem uma olhada nos links abaixo.

* <http://vimeo.com/17838731>
* <http://www.casadocodigo.com.br/products/livro-tdd>
* <http://imasters.com.br/artigo/24242/desenvolvimento/principios-de-test-driven-development-tdd/>
* <http://blog.thiagobelem.net/aprendendo-tdd-ou-desenvolvimento-orientado-a-testes/>
* <http://vimeo.com/13635357>
* <https://sites.google.com/site/tddproblems/>
* <http://www.infoq.com/br/articles/levison-TDD-adoption-strategy>