




Disciplina Programação Modular	Departamento Ciência da Computação	Turno Manhã	Período 9º
Professor Hugo de Paula (hugo@pucminas.br)			
Matrícula:	Aluno:		

## Exercício 1 - Classes e TDD

1. Desenvolva uma classe `Data` que possua os seguintes métodos (conforme apresentado no slides sobre TDD):

```
public Data(int ano, int mes, int dia)
public Data() // hoje
public int getDia(), getMes(), getAno()
public void adicionaDias(int dias) // avanca a data em dias
public int diasNoMes()
public String diaDaSemana() // ex. "Segunda-feira"
public boolean eAnoBisexto()
public void proximoDia() // avanca um dia
```

- (a) Utilize a abordagem de desenvolvimento dirigido por teste.
  - (b) Obedeça ao princípio da ocultação de informações.
  - (c) Desenvolva uma classe aplicação para ilustrar o funcionamento da classe `Data`.
  - (d) Desenhe a classe `Data` na notação UML. 
2. Implemente uma classe `Pessoa`. Deve-se armazenar o nome, o cpf, a idade e o sexo. Deve haver métodos para: construir a classe com valores padrões ou entrados pelo usuário; possuir atributos encapsulados; possuir métodos de acesso para permitir alterar o valor desses atributos, possuir métodos para permitir consultar o valor dos atributos; possuir método para informar se é maior de idade. Utilize a abordagem de desenvolvimento dirigido por teste. Desenvolva uma classe aplicação para ilustrar o funcionamento da classe `Pessoa`. Desenhe a classe `Pessoa` na notação UML.
  3. Implemente uma classe `Conta` que representa uma conta bancária. Ela deve possuir no mínimo o número da conta e o saldo. Adicione novos atributos que julgar necessários. Utilize a abordagem de desenvolvimento dirigido por teste. Desenvolva uma classe aplicação para ilustrar o funcionamento da classe `Conta`. Desenhe a classe `Conta` na notação UML.
  4. Desenvolva uma classe conversora de unidades de distância. Ela deve possuir os métodos:
    - `converterPesParaMetros(valor:double):double`  
recebe uma distância em pés e retorna o equivalente em metros.
    - `converterPesParaPolegadas(valor:double):double`  
recebe uma distância em pés e retorna o equivalente em polegadas.
    - `converterPesParaCentimetros(valor:double):double`  
recebe uma distância em pés e retorna o equivalente em centímetros.

Matrícula:	Aluno:
------------	--------

- converterPolegadasParaMetros(valor:**double**):**double**  
recebe uma distância em polegadas e retorna o equivalente em metros.
- converterPolegadasParaPes(valor:**double**):**double**  
recebe uma distância em polegadas e retorna o equivalente em pés.
- converterPolegadasParaCentimetros(valor:**double**):**double**  
recebe uma distância em polegadas e retorna o equivalente em centímetros.
- converterMetrosParaPes(valor:**double**):**double**  
recebe uma distância em metros e retorna o equivalente em pés.
- converterMetrosParaPolegadas(valor:**double**):**double**  
recebe uma distância em metros e retorna o equivalente em polegadas.
- converterMetrosParaCentimetros(valor:**double**):**double**  
recebe uma distância em metros e retorna o equivalente em centímetros.
- converterCentimetrosParaPes(valor:**double**):**double**  
recebe uma distância em centímetros e retorna o equivalente em pés.
- converterCentimetrosParaPolegadas(valor:**double**):**double**  
recebe uma distância em centímetros e retorna o equivalente em polegadas.
- converterCentimetrosParaMetros(valor:**double**):**double**  
recebe uma distância em centímetros e retorna o equivalente em metros.

As taxas de conversão devem ser adequadamente armazenadas na classe. Utilize a abordagem de desenvolvimento dirigido por teste. Desenvolva uma classe aplicação para ilustrar o funcionamento da classe. Desenhe a classe na notação UML.