**UFF – Universidade Federal Fluminense**
**TIC – Instituto de Computação**
**TCC – Departamento de Ciência da Computação**

**TCC 00.309 | Programação de Computadores II | Turma A-1 | 2016.2**
**Professor: Leandro A. F. Fernandes**

Lista de Exercícios Sobre

# Orientação a Objetos

Antes de Começar:
Utilize o mecanismo de lançamento de exceções em todos os exercícios dessa lista, com o objetivo de garantir consistência nos estados dos objetos.

## Questão 1

Identifique as classes, atributos e métodos necessários para modelar e implementar:

a) Uma conta corrente que possui um número, um saldo, um status que informa se ela é especial ou não, um limite e um conjunto de movimentações.

b) Uma movimentação que possui uma descrição, um valor e uma informação se ela é uma movimentação de crédito ou débito.

c) Um banco que armazene um conjunto de contas e forneça métodos que permitam que sejam feitos criações de conta, exclusão de contas, saques (uma conta corrente só pode fazer saques desde que o valor não exceda o limite de saque-limite + saldo negativo), depósitos, emissão de saldo e extrato e transferência entre contas.

Uma vez feita a modelagem, implemente em Java. Faça da forma mais completa que puder imaginar. Pense bastante antes de proceder com qualquer implementação e crie classes intermediárias na hierarquia.

## Questão 2

Escreva uma classe que represente um país. Um país tem como atributos o seu nome, o nome da capital, sua dimensão em Km2 e uma lista de países com os quais ele faz fronteira. Represente a classe e forneça os seguintes construtores e método:

a) Construtor que inicialize o nome, capital e a dimensão do país;

b) Métodos de acesso (obter/get) para as propriedades indicadas no item (a);

c) Um método que permita verificar se dois países são iguais. Dois países são iguais se tiverem o mesmo nome e a mesma capital. A assinatura deste método deve ser:

```
public boolean equals(Pais outro);
```

d) Um método que define quais outros países fazem fronteira (note que um país não pode fazer fronteira com ele mesmo);

e) Um método que retorne a lista de países que fazem fronteira;

f) Um método que receba um outro país como parâmetro e retorne uma lista de vizinhos comuns aos dois países.

## Questão 3

Implemente uma classe que represente polinômios com uma variável. Esta classe deve conter:

a) Diferentes construtores;
b) Métodos de acesso;
c) Operações de adição e multiplicação.
d) Um método que faça a avaliação do polinômio, dado um número real $x$.

Escreva, também, uma classe de testes para a classe que representa o polinômio.

## Questão 4

De forma incremental, traduza o seguinte conjunto de classes em um programa Java. Importante: Não são permitidas chamadas a `System.in`, `System.out` ou similares de dentro das classes criadas.

a) Classe: `Porta`

   Atributos: `aberta`, `cor`, `dimensaoX`, `dimensaoY`, `dimensaoZ`

   Métodos: `void abre()`, `void fecha()`, `void pinta(String s)`, `boolean estaAberta()`

   Para testar, crie uma porta, abra e feche a mesma, pinte-a de diversas cores, altere suas dimensões e use o método `estaAberta` para verificar se ela está aberta.

b) Classe: `Casa`

   Atributos: `cor`, `porta1`, `porta2`, `porta3`

   Método: `void pinta(String s)`, `int quantasPortasEstaoAbertas()`, `int totalDePortas()`

   Para testar, crie uma casa e pinte-a. Crie três portas e coloque-as na casa; abra e feche as mesmas como desejar. Utilize o método `quantasPortasEstaoAbertas` para imprimir o número de portas abertas.

c) Classe: `Edificio`

   Atributos: `cor`, `totalDePortas`, `totalDeAndares`, `portas[]`

   Métodos: `void pinta(String s)`, `int quantasPortasEstaoAbertas()`, `void adicionaPorta(Porta p)`, `int totalDePortas()`, `void adicionarAndar()`, `int totalDeAndares()`

   Para testar, crie um edifício, pinte-o. Crie seis portas e coloque-as no edifício através do método `adicionaPorta`, abra e feche-as como desejar. Utilize o método `quantasPortasEstaoAbertas` para imprimir o número de portas abertas e o método `totalDePortas` para imprimir o total de portas em seu edifício. Cria alguns andares utilizado o método `adicionarAndar` e retorne o número total de andares utilizando o método `totalDeAndares`.

d) As classes `Casa` e edifício ficaram muito parecidas. Crie a classe `Imovel` e coloque nela tudo o que `Casa` e `Edificio` tem em comum. Faça `Imovel` superclasse de `Casa` e

`Edificio`. Note que alguns métodos em comum não poderão ser implementados por `Imovel` (e.g., `quantasPortasEstaoAbertas` e `totalDePortas`). Logo, esses deverão ser declarados como métodos abstratos.

## Questão 5

Crie uma classe abstrata de polígono. Todo polígono deve saber calcular sua área. Crie três tipos de especializações, respectivamente, para círculo, retângulo e triângulo. Sobrescreva nas especializações o método abstrato de cálculo de área declarado na classe de polígono. Crie uma classe de polígono complexo que pode ser constituída por um ou mais polígonos. Forneça um método que calcule a área deste de polígonos completos. Utilize o princípio de polimorfismo sempre que possível.

## Questão 6

A integral de uma função para um determinado intervalo é a soma de sua área naquele intervalo. Ela pode ser calculada numericamente a partir de aproximações de retângulos. Implemente:

a) A classe Integral juntamente com o método

```
double calcularIntegral(Funcao f, double a, double b)
```

que recebe um objeto `Funcao` e um intervalo $[a, b]$ para calcular a integral no intervalo. A classe `Integral` tem um atributo denominado `iD` (intervalo de discretização), que corresponde a largura fixa de cada retângulo.

b) A classe `Funcao` é uma classe abstrata que tem um método abstrato

```
double avaliar(double x)
```

que recebe um valor $x$ e retorna o valor da função em $x$.

c) Classes `Exponencial` e `Cubica` como especializações da classe `Funcao` para calcular, respectivamente, as funções $f(x) = e^x$ e $f(x) = x^3$, sobrescrevendo o método `avaliar` da classe ancestral.

d) Uma classe descendente da classe `Funcao` denominada `FuncaoAgregadora`, que tem um vetor de objetos descendentes da classe `Funcao`. O valor retornado pelo método `avaliar` de `FuncaoAgregadora` é obtido por meio da soma de diversas funções que fazem parte desta função.

## Questão 7

Veja os slides de Orientação a Objetos utilizados em aula. Na parte de polimorfismo, é dado como exemplo de declaração e implementação de interface a interface `Comparavel`. Essa interface segue a mesma ideia da interface `java.lang.Comparable`, porém, sem o uso de tipos genéricos[1].

Escreva um programa que reproduz o exemplo apresentado nos slides. Implemente nesse programa um método estático de ordenação que recebe um array de objetos comparáveis como argumento e ordene o conteúdo desse array fazendo do retorno do método `compararCom`.

[1] Para saber mais sobre tipos genéricos, consulte o tutorial disponibilizado pela Oracle em https://docs.oracle.com/javase/tutorial/java/generics/