



Prof. Robson L. F. Cordeiro

### Lista 3 – Herança, polimorfismo e interfaces

**1. Criar uma estrutura hierárquica que contenha as classes Veiculo (classe abstrata), Bicicleta e Automóvel.**

Os métodos da classe Veiculo são todos abstratos e possuem a seguinte assinatura:

- listarVerificacoes( )
- ajustar( )
- limpar( )

Estes métodos são implementados nas subclasses Automóvel e Bicicleta. Acrescentar na classe Automóvel o método trocarOleo( ).

Para desenvolver a classe Teste que é apresentada a seguir é necessário criar também a classe Oficina que terá dois métodos:

- proximo() que retorna o próximo objeto do tipo Veiculo (Bicicleta ou Automóvel)
- manutencao(Veiculo v) que recebe como parâmetro um objeto do tipo Veiculo e chama os métodos definidos na classe Veiculo:
  - listarVerificacoes( )
  - ajustar( )
  - limpar( )
  - SE o veiculo for Automóvel deve também chamar o método trocarOleo( )

```
class Teste{
    public static void main(String args[])
    {
        Oficina o = new Oficina();
        Veiculo v;
        for(int i=0;i<4;++i)
        {
            v=o.proximo();
            o.manutencao(v);
        }
    }
}
```

**2. Crie uma classe “Produto” que possua os atributos “nomeloja” e “preco”, crie os métodos sets e gets para estes atributos. Crie também o atributo “descrição” e seu método chamado “getDescrição” que retorna uma string com o simples conteúdo “Produto de informática”.**

Crie duas classes filhas de “Produto”, que serão “Mouse” com o atributo “tipo” e “Livro” com o atributo “autor”, no método construtor de cada uma dessas classes passe como argumento a descrição desse produto, por exemplo, Mouse(“Mouse óptico, Saída USB, 1.600 dpi”). Crie o método “getDescrição” que retorna a descrição que foi passada no argumento do construtor concatenada com o atributo que a classe tiver, “autor” no caso de livro e “tipo” no caso de mouse, esse método deve ter a mesma assinatura do método “getDescrição” da classe pai “Produto”.

Crie uma classe “Main” que irá simular a compra de um cliente de vários mouses e livros, deve haver apenas um vector/arraylist na classe “Main” para armazenamento de todos os livros e mouses. Esse vector/arraylist deve se chamar “carrinho” que simula o carrinho de compras de produtos variados de um cliente em um e-commerce. Insira nesse “carrinho” vários mouses e livros e depois chame o método “getDescrição” de todos os objetos presentes no vector/arraylist. Para o usuário do carrinho saber as informações dos produtos em seu carrinho.

**3. Há uma preocupação atual com as pegadas de carbono (emissões anuais de gás carbônico na atmosfera) a partir de instalações que queimam vários tipos de combustíveis para aquecimento, veículos que queimam combustíveis para se mover, e assim por diante. Nesse cenário:**

- Crie três pequenas classes não relacionadas por herança - classes Prédio, Carro, e Bicicleta. Dê a cada classe alguns atributos e comportamentos (métodos) únicos que ela não tem em comum com as outras classes.
- Escreva uma interface PegadaDeCarbono com um método getPegadaDeCarbono. Faça cada uma das suas classes implementar essa interface, para que seu método getPegadaDeCarbono calcule uma pegada de carbono apropriada a cada classe de acordo com seus atributos.

- Escreva um aplicativo que crie 2 objetos de cada uma das três classes. Crie um objeto `ArrayList<PegadaDeCarbono>` e insira as referências dos objetos instanciados nessa coleção. Finalmente, itere pela coleção, chamando polimorficamente o método `getPegadaDeCarbono` de cada objeto. Modifique o código, tornando `Prédio` uma classe abstrata, e implementando duas novas subclasses concretas `Casa` e `Escola`.
- O aplicativo que cria a coleção de objetos vai continuar funcionando após a modificação na estrutura das classes?
- Modifique o aplicativo para que passe a instanciar diretamente objetos `Casa` e `Escola`, incluindo-os na coleção.

**Para entrega: códigos dos projetos NetBeans referentes aos exercícios acima em um arquivo zip → entregar via Tidia→Atividades**

#### 4. Herança

- Defina a classe `Produto`
  - Os atributos de um produto são: código, preço unitário, descrição e quantidade no estoque – todos com visibilidade protegida;
  - O construtor deve receber todos os atributos como parâmetros;
  - Deve oferecer métodos “get” e “set” para os atributos preço e descrição;
  - Deve oferecer métodos “get” para código e quantidade;
  - Deve oferecer um método onde se informa certa quantidade a ser retirada do estoque e outra onde se informa uma certa quantidade a ser acrescentada ao estoque;
  - O método onde se informa uma quantidade a ser retirada do estoque deve retornar a quantidade que efetivamente foi retirada (para os casos em que havia menos produtos do que o solicitado);
  - Deve oferecer um método que imprime a descrição básica do produto incluindo dados de todos os atributos, por exemplo: “Produto 3, arroz, custo de R\$ 5, quantidade 100”.
- Defina a classe `ProdutoPerecivel`
  - Esta deve ser derivada de `Produto`;
  - Possui um atributo extra que guarda a data de validade do produto;
  - Deve possuir métodos get/set para a data de validade;
  - O método de retirada deve receber também por parâmetro a data do dia corrente; se a data de validade do produto for expirar dentro de 30 dias o método deve zerar o estoque e devolver 0, pois produtos vencidos são jogados fora → pesquise aritmética com o tipo `Date` do Java;
  - O método de acréscimo no estoque só deve acrescentar os novos produtos caso o estoque esteja zerado, de maneira a evitar misturar produtos com prazos de validade diferentes. O método retorna `true` ou `false` caso tenha sido possível, ou não, alterar a quantidade.
- Defina a classe `ProdutoPerecivelEspecial`
  - Esta é derivada de `ProdutoPerecivel`;
  - Oferece um método de impressão de dados capaz de imprimir uma nota de controle onde consta o código, a descrição, a quantidade em estoque e a data de validade do produto.
- Defina a classe `ProdutoNaoPerecivel`
  - Esta é derivada de `Produto`;
  - Deve possuir campos para armazenar o tempo de garantia (em número de anos) do produto;
  - A classe deve oferecer métodos para permitir obter o tempo de garantia, mas não para alterar.
- Defina a classe `Estoque`
  - Esta mantém uma lista com os produtos em estoque (do tipo `Produto`) – usar array simples ou a classe `ArrayList` do Java;
  - A classe deve ter métodos para cadastrar produtos, consultar produtos pelo código (caso não encontre, retorna `null`), e retirar produtos do estoque, bem como para informar o custo total do estoque armazenado, e imprimir a descrição básica de todos os produtos.
- Escreva um pequeno programa que cria um produto de cada tipo especializado (perecível, perecível especial, e não perecível). Acrescente estes produtos a um estoque. Teste a sua busca por um produto. Imprima o total do estoque, e liste todos os produtos.

→ Para todos os itens, defina o construtor da classe com os atributos necessários para sua inicialização considerando as necessidades da super classe.

**5. (Sobrecarga)** Considere uma classe com funcionalidades de exibição de texto na tela do computador. Esta classe recebe o texto e o exibe de diferentes maneiras:

- sem formatação;
- com formatação de estilo: itálico, negrito, ou underline;
- com fonte especial, neste caso o nome da fonte deve ser passado;
- com cor, sendo que um código de cor deve ser passado.

Considere que quem usa esta classe não quer se preocupar com todos os parâmetros necessários toda vez que for usá-la, mas apenas com os parâmetros que interessam para um determinado uso.

Escreva o código desta classe definindo as assinaturas dos métodos preenchendo-os sem usar código de formatação propriamente dito, mas apenas escrevendo quais estilos foram usados. Escreva um conjunto de métodos suficiente para suportar as várias combinações de como o texto pode ser exibido.

Escreva um programa que evoca cada um dos métodos definidos e que imprima texto como nos seguintes exemplos:

“TEXTO DE TESTE -> sem formatação”

“TEXTO DE TESTE -> negrito, com fonte Times New Roman”

“TEXTO DE TESTE -> itálico, com fonte Courier New, na cor azul”

**6. (Classe abstrata/Interface)** Imagine um sistema operacional para computadores pessoais. Este sistema usa vários dispositivos, como vídeo, impressora, mouse, e teclado. São todos dispositivos distintos, mas que para trabalhar com um dado sistema operacional devem ter as seguintes funcionalidades em seus drivers:

- ligar/desligar;
- checar status, o que retorna um número indicado a condição do dispositivo;
- calibrar.

**a)** Qual solução de projeto você adotaria para que os desenvolvedores de dispositivos pudessem desenvolver dispositivos para este sistema sem que o código do sistema fosse revelado?

Escreva o correspondente código.

**b)** Considere o caso em que o driver de um determinado tipo de dispositivo, além de satisfazer aos requisitos do sistema operacional considerado, também será usado para a criação de toda *uma família* de drivers que tem funcionalidades comuns e que possuem especificidades em cada uma de suas variações. Escolha um dos tipos de dispositivo e escreva o código correspondente – sem preencher os métodos.

**7. (Classe abstrata/Interface)** Considere o seguinte código:

```
class System{
    public void execute(Executable e){
        int iStatus = -1;
        if(e.initialize( )){
            this.setLoad(e.LOAD * this.timeUnits);
            iStatus = e.execute( );
        }
        e.close( );
    }
}
```

Escreva o código correspondente à definição do contrato **Executable** usando:

- a) Classe abstrata
- b) Interface

**8. (Interface)** Considere uma classe TimeFutebol a qual guarda informações de times de futebol que participam de um dado campeonato, as informações são: número de vitórias, derrotas, e empates, número de gols marcados e de gols sofridos, número de cartões amarelos e vermelhos. Crie um método que retorna a quantidade de pontos de um dado time = vitórias\*3 + empates\*1.

Em seguida, usando a interface Comparable, ordene os times de acordo com os critérios de desempate da federação paulista de futebol ([https://pt.wikipedia.org/wiki/Campeonato\\_Paulista\\_de\\_Futebol\\_de\\_2020\\_-\\_Série\\_A1](https://pt.wikipedia.org/wiki/Campeonato_Paulista_de_Futebol_de_2020_-_Série_A1)), inclusive o item 6 (Sorteio).

Escreva um programa principal (main) com vários times cujos atributos irão acionar cada um dos critérios de desempate para ordenação.