Lista de Exercícios: Abstração com Classes e Métodos Abstratos

1. Refatorando a Frota da "Mobilidade Urbana"

Na locadora "Mobilidade Urbana", não é possível alugar um "veículo genérico". Os clientes alugam Carros ou Motos. Além disso, a regra para o cálculo do seguro diário (calcularSeguro()) é sempre específica para cada tipo de veículo; não existe um cálculo "padrão".

- **Sua Tarefa:**
    - **Parte A:** Modifique a superclasse Veiculo que você criou, tornando-a abstract para impedir que seja instanciada diretamente.
    - **Parte B:** Transforme o método calcularSeguro() (ou calcularPedagio() da aula anterior) em um método abstract, pois não há uma implementação lógica para ele na superclasse. Lembre-se de remover o corpo do método.
    - **Parte C:** Verifique no main que a linha new Veiculo(...) agora causa um erro de compilação. Confirme que suas classes Carro e Moto ainda compilam, pois elas já cumprem o "contrato" ao sobrescrever o método.

2. O Contrato das Formas Geométricas no "CanvasPro"

No software "CanvasPro", o conceito de FormaGeometrica é puramente abstrato; um usuário desenha Circulos, Quadrados, etc. Adicionalmente, é um requisito obrigatório que toda e qualquer forma geométrica no sistema saiba como calcular sua própria área e seu perímetro.

- **Sua Taref_a:**
    - Refatore sua hierarquia de FormaGeometrica:
    - Torne a classe FormaGeometrica abstract.
    - Transforme os métodos calcularArea() e calcularPerimetro() em métodos abstract, forçando todas as subclasses a fornecerem suas próprias implementações.

3. Contas no "NextGen Bank"

O banco digital "NextGen Bank" possui vários tipos de contas (ContaPoupanca, ContaCorrente), mas não existe uma "Conta genérica". Cada tipo de conta tem uma maneira específica de calcular as taxas de manutenção mensal. A diretoria quer garantir que, ao criar um novo tipo de conta no futuro, os programadores não se esqueçam de implementar essa lógica de cálculo de taxas.

- **Sua Tarefa:**
    - Crie uma superclasse public abstract class Conta com atributos como titular e saldo.
    - Adicione um método public abstract void calcularTaxas().
    - Crie duas subclasses concretas, ContaPoupanca e ContaCorrente, que herdam de Conta. Implemente o método calcularTaxas() em cada uma com uma lógica simples (ex: a poupança não tem taxa, a corrente desconta R$ 15,00 do saldo).

4. A Essência da Abstração (Conceitual)

Um colega de equipe pergunta: "Eu entendi como usar abstract, mas não entendi quando. Qual é a principal pergunta que eu devo me fazer para decidir se uma classe deve ser abstrata ou concreta?".

- **Sua Tarefa:** Em um comentário no seu código, escreva uma resposta para ele. Foque na diferença entre um "conceito" e uma "coisa real" no contexto do seu sistema.

5. Caça ao Erro de Compilação (Leitura de Código)

Analise o código abaixo. Ele não compila. Identifique a classe que contém o erro, explique em um comentário por que o erro acontece e como corrigi-lo.

```java
public abstract class InstrumentoMusical {
    public abstract void tocar();
}

public class Violao extends InstrumentoMusical {
    @Override
    public void tocar() {
        System.out.println("Soando cordas de nylon...");
    }
}

public class Piano extends InstrumentoMusical {
    // A classe Piano tem um erro de compilação. Por quê?

    public void afinar() {
        System.out.println("Afinando o piano...");
    }
}
```

6. Justificando a Abstração (Leitura de Código)

Seu colega escreveu a classe ProcessoJudicial abaixo. Ao tentar remover a palavra abstract, o código para de compilar.

```java
public abstract class ProcessoJudicial {
    private int numeroProcesso;

    public void iniciarProcesso() {
        validarDocumentos();
        distribuirParaVara();
    }

    // Método que depende de cada tipo de processo
    public abstract void validarDocumentos();

    private void distribuirParaVara() {
        // ... lógica comum ...
    }
}
```

- **Sua Tarefa:** Em um comentário, explique por que a classe ProcessoJudicial **precisa** ser abstract. Qual linha de código a torna obrigatoriamente abstrata?

7. Decisão de Design: Sistema de E-commerce

Você está projetando um sistema de e-commerce. Você tem a classe Produto. Você decide criar subclasses como Livro, Eletronico e Vestuario. A classe Produto deve ser concreta ou abstrata? Justifique sua decisão de design. O que aconteceria se você permitisse que um vendedor cadastrasse um "Produto genérico" no sistema?

8. Hierarquia de Ingressos para Eventos

Um sistema de venda de ingressos vende diferentes tipos de Ingresso. Todo ingresso tem um valor base. Um IngressoVIP tem um valor adicional, e um IngressoPista não. O método imprimirValor() deve exibir o valor final do ingresso.

- **Sua Tarefa:**
    - Crie uma superclasse public abstract class Ingresso com o atributo protected double valor.
    - Adicione um método public abstract void imprimirValor() nela.
    - Crie as subclasses IngressoVIP e IngressoPista.
    - IngressoVIP deve ter um atributo valorAdicional. Sua implementação de imprimirValor() deve imprimir valor + valorAdicional.
    - IngressoPista não tem atributos extras. Sua implementação de imprimirValor() deve imprimir apenas o valor.

9. A Regra de Ouro da Abstração (Conceitual)

Um método abstract pode existir dentro de uma classe concreta (não-abstrata)? Em um comentário, explique por que sim ou por que não, pensando no que aconteceria se fosse possível criar uma instância (new) dessa classe.

10. Desafio: Um Mini-Framework de Relatórios

Você foi encarregado de criar um esqueleto para gerar diferentes tipos de relatórios (HTML, Texto, etc.). O fluxo de geração é sempre o mesmo: construir um cabeçalho, construir o corpo e construir um rodapé. No entanto, o conteúdo de cada parte muda drasticamente para cada tipo de relatório.

- **Sua Tarefa:**
    - Crie uma classe public abstract class GeradorDeRelatorio.
    - Nela, crie um método **concreto** e final (que não pode ser sobrescrito) chamado gerarRelatorio(), com a seguinte lógica:

```
public final void gerarRelatorio() {
    String cabecalho = construirCabecalho();
    String corpo = construirCorpo();
    String rodape = construirRodape();
    System.out.println(cabecalho + corpo + rodape);
}
```

    - Agora, declare os métodos construirCabecalho(), construirCorpo() e construirRodape() como public abstract.
    - Crie uma subclasse concreta RelatorioHTML que herda de GeradorDeRelatorio e implementa os três métodos abstratos para retornar tags HTML (ex: "<header>...</header>").
    - No main, crie uma instância de RelatorioHTML e chame o método gerarRelatorio() para ver o resultado. Este padrão de design é chamado de **Template Method**.