PROGRAMAÇÃO ORIENTADA A OBJETOS

Exercício 1: Sistema de Tarefas Agendáveis

Contexto:

Você está desenvolvendo um sistema para gerenciar tarefas automatizadas em uma empresa. Algumas dessas tarefas podem ser **agendadas** para serem executadas em momentos específicos, e é importante monitorar o status de cada uma. O objetivo é criar um sistema que permita agendar, executar e monitorar essas tarefas de maneira polimórfica, utilizando uma interface.

Requisitos:

1. **Interface Agendável**:
    - Crie uma interface chamada Agendavel, que possua os seguintes métodos:
        - agendar(String horario): para agendar a execução da tarefa em um determinado horário.
        - executar(): para executar a tarefa.
        - monitorar(): para verificar o status da tarefa (agendada, em execução ou concluída).
2. **Implementação da Interface**:
    - Crie classes que representem diferentes tipos de tarefas, cada uma implementando a interface Agendavel:
        - **BackupDeDados**: Representa uma tarefa de backup de dados que pode ser agendada e monitorada.
        - **RelatorioFinanceiro**: Representa uma tarefa de geração de relatório financeiro.
        - **LimpezaDeSistema**: Representa uma tarefa de limpeza de arquivos temporários e dados obsoletos no sistema.
3. **Classe SistemaDeTarefas**:
    - Crie uma classe SistemaDeTarefas, que gerencia uma lista de tarefas agendáveis e tenha os seguintes métodos:
        - adicionarTarefa(Agendavel tarefa): para adicionar uma tarefa ao sistema.
        - agendarTodas(String horario): para agendar todas as tarefas registradas em um horário comum.
        - executarTodas(): para executar todas as tarefas agendadas.
        - monitorarTarefas(): para verificar o status de todas as tarefas.

4. **Teste**:
    - No método main, crie instâncias de diferentes tarefas (backup de dados, relatório financeiro, limpeza de sistema), adicione-as ao sistema e realize operações de agendamento, execução e monitoramento.

Exercício 2: Sistema de Simulação de Veículos Autônomos

Contexto:

Você está desenvolvendo uma simulação de um sistema de transporte autônomo que gerencia diferentes tipos de veículos (como carros, caminhões e drones). Cada tipo de veículo possui características únicas, mas todos podem ser controlados remotamente e têm a capacidade de **navegar**, **carregar** e **descarregar** itens. O objetivo é criar um sistema que gerencie esses veículos de maneira polimórfica, utilizando interfaces para as capacidades comuns, e classes específicas para os tipos de veículos.

Requisitos:

1. **Interface Navegável**:
    - Crie uma interface chamada Navegavel, que possua os seguintes métodos:
        - iniciarRota(String destino): para iniciar a navegação até um determinado destino.
        - parar(): para interromper a navegação.
        - monitorarPosicao(): para verificar a posição atual do veículo (simulada).
2. **Interface Carregável**:
    - Crie uma interface chamada Carregavel, que possua os seguintes métodos:
        - carregar(int peso): para carregar um item com um determinado peso.
        - descarregar(): para descarregar o item transportado.
3. **Classe abstrata VeiculoAutonomo**:
    - Crie uma classe abstrata VeiculoAutonomo que implemente ambas as interfaces Navegavel e Carregavel. Esta classe deve conter propriedades comuns, como:
        - String tipo: o tipo de veículo.
        - int capacidade: a capacidade máxima de carga.
        - String destinoAtual: o destino atual do veículo.
        - boolean emRota: indica se o veículo está em rota ou não.
        - boolean carregado: indica se o veículo está carregado ou não.
        - Métodos abstratos exibirStatus(), que serão implementados pelas subclasses para exibir o estado específico de cada tipo de veículo.

4. **Classes Específicas de Veículos**:
    - Crie classes específicas que herdem de VeiculoAutonomo e implementem os métodos abstratos:
        - **CarroAutonomo**: um carro autônomo que pode transportar até 500kg de carga.
        - **CaminhaoAutonomo**: um caminhão autônomo que pode transportar até 5000kg de carga.
        - **DroneAutonomo**: um drone autônomo que pode transportar até 50kg de carga.
5. **Classe SistemaDeTransporte**:
    - Crie uma classe SistemaDeTransporte, que gerencia uma lista de veículos autônomos e tenha os seguintes métodos:
        - adicionarVeiculo(VeiculoAutonomo veiculo): para adicionar veículos ao sistema.
        - iniciarRotaVeiculo(int indice, String destino): para iniciar a rota de um veículo específico.
        - carregarVeiculo(int indice, int peso): para carregar um veículo com um determinado peso.
        - descarregarVeiculo(int indice): para descarregar um veículo.
        - monitorarTodos(): para monitorar o status e a posição de todos os veículos.
6. **Simulação**:
    - No método main, crie instâncias de diferentes veículos (carro autônomo, caminhão autônomo, drone autônomo), adicione-os ao sistema e simule uma sequência de operações como carregar, iniciar rotas, parar e monitorar as posições dos veículos.

-X-