Pontifícia Universidade Católica de Minas Gerais
Bacharelado em Engenharia de *Software* - *Campus* Lourdes
Algoritmos e Estruturas de Dados II (Laboratório)
Prof.ª Glender Brás - Semestre 2/2025
Atividade: Estruturas de Dados Lineares em Sistemas de *Software* -
Valor: 3 pontos - **Data:** 27/10/2025

INSTRUÇÕES

- Esta é uma **atividade individual**, em laboratório, da disciplina "Algoritmos e Estruturas de Dados II".
- Utilize seu repositório, criado no **GitHub Classroom**, para resolver as questões de programação propostas e enviar suas respostas para correção. **Nenhuma outra forma de entrega da atividade será aceita**.
- O prazo **limite para entrega** é o dia **27/10**, ao final da aula. *Commits* no repositório com data posterior a esse limite serão desconsiderados no momento da correção da atividade.
- Além do **conteúdo do seu repositório de trabalho** e dos **materiais disponíveis no Canvas citados na Tarefa 0**, **não é permitida a consulta a nenhum outro material**, porém o professor está **disponível para esclarecer dúvidas** e debater propostas de soluções vindas dos alunos.
- Sendo uma atividade individual, **espera-se dos alunos comprometimento e ética** na produção de sua solução. Soluções produzidas em conjunto, entregas contendo cópias, mesmo que parciais, em código ou estrutura, e consultas a materiais e ferramentas não permitidos são **faltas graves**, **resultando em nota 0 para todos os envolvidos** e posterior **notificação à Coordenação do Curso de Engenharia de *Software*** por desrespeito às normas acadêmicas.

Tema: Estruturas de dados lineares em sistemas de *software*

Temos um sistema de software simples que permite a venda de produtos de escritório agrupados em pedidos, com funcionalidades de consulta aos produtos cadastrados, recuperação de dados de arquivos-texto, e algumas funções gerenciais implementadas recentemente com uso de pilhas e filas. Vamos agora utilizar uma nova estrutura de dados – listas encadeadas –, melhorar nossa classe "Pedido" empregando essa estrutura e incluir novas funções gerenciais no sistema usando uma lista de pedidos.

Tarefa 0: (preparação)

- **Adicione uma nova ramificação** (*branch*) no seu repositório GitHub Classroom correspondente a esta atividade, com o nome ***atividade2710***, a partir de seu *branch* anterior. Certifique-se de que as classes ***Celula***, ***Pilha*** e ***Fila*** estejam neste novo *branch*.
- Em seguida, **descompacte o código-base** fornecido no Canvas e inclua a nova classe ***Lista*** e a nova versão da classe ***Pedido*** em sua pasta de código.
- Estude o código da classe ***Lista***. Esta classe tem as operações básicas de uma lista simplesmente encadeada e algumas outras operações úteis para o uso dessa estrutura de dados, a serem implementadas nessa atividade.
- Observe o código da classe ***Pedido***. Um pedido, agora, contém uma lista de produtos e, portanto, alguns de seus métodos foram atualizados.

Tarefa 1: (1 ponto)
Implemente o método **`public E obterElemento(int posicao)`** na classe **`Lista<E>`**. Esse método deve ser capaz de retornar, sem remover, o elemento da lista armazenado na posição indicada por seu único parâmetro. A primeira posição da lista é considerada a posição 0 e, assim, a última é (tamanho atual da lista - 1). O parâmetro **`posicao`** indica a posição do elemento da lista a ser consultado. Deve ser maior ou igual a 0 e menor do que o tamanho atual da lista. Esse método deve lançar a exceção **`IllegalStateException`** se a lista estiver vazia e **`IndexOutOfBoundsException`** em caso de posição inválida.

Tarefa 2: (0,5 ponto)
Implemente o método **`public int contar(Predicate<E> condicional)`** na classe **`Lista<E>`**. Esse método deve ser capaz de contar e retornar a quantidade de elementos da lista que satisfazem uma condição específica. Apresenta o parâmetro **`condicional`**, predicado com a condição para verificação de elementos da lista.

Tarefa 3: (0,5 ponto)
Implemente o método **`public double obterSoma(Function<E, Double> extrator)`** na classe **`Lista<E>`**. Esse método deve ser capaz de calcular e retornar a soma de um dos atributos dos elementos da lista,

utilizando uma função de extração fornecida pelo parâmetro **`extrator`**, que extrai um valor numérico (**`Double`**) de cada elemento da lista.

Tarefa 4: (1 ponto)

Altere o código da classe **`App`** desenvolvido na atividade anterior permitindo agora que os pedidos sejam armazenados em uma lista.

Acrescente ao *menu* do sistema de comércio as seguintes novas opções: **`Exibir o faturamento do comércio de produtos`**; e **`Exibir a quantidade de pedidos realizados em um determinado período`**. Implemente os métodos correspondentes a essas novas funcionalidades do sistema de comércio, a saber:

- **`public static void obterFaturamento()`**: exibe o faturamento total do comércio de produtos, formatado na moeda corrente brasileira. O método deve **obrigatoriamente** utilizar a função **`obterSoma`**, da classe **`Lista<E>`** para calcular a soma dos valores finais de todos os pedidos cadastrados.
  Exemplo de saída: **Faturamento do comércio de produtos: R$ 12.345,67**

- **`public static void contarPedidosPorData()`**: lê da entrada padrão uma data inicial e uma data final e exibe a quantidade de pedidos realizados entre essas datas. Somente pedidos com data estritamente posterior à data inicial e anterior à data final devem ser considerados (ou seja, não inclui os limites). Esse método deve **obrigatoriamente** utilizar a função **`contar`**, da classe **`Lista<E>`**, para obter a contagem de pedidos realizados entre as datas informadas.
  Exemplo de uso:
  Informe a data inicial dos pedidos: 01/10/2025
  Informe a data final dos pedidos: 31/10/2025
  Quantidade de pedidos realizados entre as datas informadas: 12

Empregue métodos das classes **`Lista<E>`** e **`Pedido`**.

Instruções e observações:

- O projeto deve estar hospedado na tarefa correspondente do GitHub Classroom. Endereço para aceitar a tarefa:
  - Grupo G1 (7:00): https://classroom.github.com/a/jhBkotPS
  - Grupo G2 (10:40): https://classroom.github.com/a/lOupw6WI

- As atividades pontuadas da disciplina podem depender direta ou indiretamente dos códigos desenvolvidos nas aulas. Portanto, é essencial o comprometimento no acompanhamento das atividades semanais.
- Para a correção das atividades pontuadas, serão considerados todos os *commits/pushes* realizados ao longo das semanas, não somente o último com a resposta final do exercício.