Pontifícia Universidade Católica de Minas Gerais
Instituto de Ciências Exatas e Informática
Algoritmos e Estruturas de Dados III - Prof. Felipe Soares

Trabalho Prático I

15 pontos - felipesoares@pucminas.br

O trabalho deve ser feito **individualmente** ou em grupos de no **máximo 2 alunos**.

Data de entrega: 14/10/2022

Penalidade por atraso: a cada dia corrido de atraso, a nota será penalizada em 2 pontos.

Penalidade por cópia: trabalhos iguais não são aceitos (nota 0).

Descrição:

Neste trabalho, você deverá implementar um sistema responsável por realizar operações de CRUD (create, read, update e delete) em um arquivo indexado. Para a realização dessas operações, você deve usar o mesmo contexto da implementação da lista de exercícios 1: contas bancárias.

Orientações:

- O sistema deve ser implementado em Java. Todo o código deve ser de autoria do grupo (com exceção para bibliotecas/classes relacionadas a aberturas e escritas/leituras de arquivos e conversões entre atributos e campos).

- Todo o código deve ser comentado de modo a se compreender a lógica utilizada. A não observância deste critério implica na redução da nota final em 50%.

- A estrutura do arquivo deve ser a seguinte:
  - Deve-se utilizar um int no cabeçalho para armazenar o último valor de id utilizado.
  - Os registros do arquivo devem ser compostos por:
    - Lápide - Byte que indica se o registro é válido ou se é um registro excluído;
    - Indicador de tamanho do registro - Número inteiro que indica o tamanho do vetor de bytes;
    - Vetor de bytes - Bytes que descrevem o objeto.

- Os objetos utilizados devem possuir os seguintes atributos:
  - **idConta** (deve ser incremental à medida que novos registros forem adicionados) (int)
  - **nomePessoa** (string de tamanho variável)
  - **email** (1 ou mais) (strings de tamanhos variáveis com indicador de quantidade)

- **nomeUsuario** (string de tamanho variável)
- **senha** (string de tamanho variável)
- **cpf** (string de tamanho fixo igual a 11)
- **cidade** (string de tamanho variável)
- **transferenciasRealizadas** (int)
- **saldoConta** (float)

Parte 1 - aproveitada da Lista de Exercícios 1:

O sistema deverá oferecer uma tela inicial (com uso pelo terminal) com um menu com as seguintes opções:

- Criar uma conta bancária -> essa escolha deve, a partir da leitura dos dados da conta bancária pelo terminal (nomePessoa, cpf, cidade, emails (1 ou mais), nomeUsuario e senha), criar uma nova conta no arquivo (saldoConta e transferenciasRealizadas devem ser iniciados com “valor informado pelo terminal” e 0, respectivamente). Obs: o nomeUsuario deve ser único para cada cliente.

- realizar uma transferência -> essa escolha deve atualizar dados em duas contas no arquivo.
    - Para isso, é necessário permitir ao usuário cadastrar uma operação de transferência, ou seja, informar duas contas (uma para debitar e outra para creditar o valor) e o valor da transferência. Assim, a conta para debitar deve ter uma redução em saldoConta do valor da transferência, enquanto que a conta para creditar deve receber um acréscimo. Então, o programa deve atualizar o campo saldoConta e o campo transferenciasRealizadas (em +1) das duas contas.

- Ler um registro (id) -> esse método deve receber um id como parâmetro, percorrer o arquivo e retornar os dados do id informado.

- Atualizar um registro -> esse método deve receber novas informações sobre um objeto e atualizar os valores dele no arquivo. Observe duas possibilidades que podem acontecer:
    - O registro mantém seu tamanho - Nenhum problema aqui. Basta atualizar os dados no próprio local.
    - O registro aumenta ou diminui de tamanho - O registro anterior deve ser apagado (por meio da marcação lápide) e o novo registro deve ser escrito no fim do arquivo.

- Deletar um registro (id) -> esse método deve receber um id como parâmetro, percorrer o arquivo e colocar uma marcação (lápide) no registro que será considerado deletado.

- Realizar a ordenação do arquivo. Para isso, deve ser realizada a implementação da ordenação externa, considerando a memória principal com limitação de “m” registros (parametrizável) e usando “n” caminhos (parametrizável). Implemente os seguintes algoritmos:
    a) Intercalação balanceada comum

b) Intercalação balanceada com blocos de tamanho variável
c) Intercalação balanceada com seleção por substituição
d) Intercalação balanceada usando n+1 arquivos
e) Intercalação Polifásica

Parte 2 - Novas implementações:

Nesta nova etapa, você deverá implementar um sistema responsável por ajustar/alterar a parte 1, para:

- a criação de um arquivo de índices (usando Árvore B+);
- a criação de um arquivo de índices (usando Hashing Estendido);
- criação de dois arquivos contendo lista invertida.

- Orientações para a criação do arquivo de índices usando **Árvore B+**:
  - O arquivo de índices deve usar a estrutura de Árvore B+, usando como chave o campo *id*. Utilize Árvore B+ de ordem 05.
  - O arquivo de índices deve conter o id (da conta bancária) e a posição do registro (referente a esse id) no arquivo de dados.
  - Sempre que acontecerem alterações no arquivo de dados, novas alterações devem ser feitas no arquivo de índices, mantendo sempre a coerência entre esses arquivos.
  - Deve existir a possibilidade de realizar buscas por um ou vários ids usando a estrutura de índices de Árvore B+.

- Orientações para a criação do arquivo de índices usando **Hashing Estendido**:
  - O arquivo de índices deve usar a estrutura de Hashing Estendido, usando como chave o campo *id*. Deve-se usar a função hash h(k) = $k \ mod$ 2p, em que p é o número de bits (profundidade) usado no diretório, sendo que cada bucket pode armazenar até 04 registros.
  - O arquivo de índices deve conter o id (da conta bancária) e a posição do registro (referente a esse id) no arquivo de dados.
  - Sempre que acontecerem alterações no arquivo de dados, novas alterações devem ser feitas no arquivo de índices, mantendo sempre a coerência entre esses arquivos.
  - Deve existir a possibilidade de realizar buscas por um *id* usando a estrutura de índices de Hashing Estendido.

- Orientações sobre a criação da lista invertida.
    - *Deve-se criar um arquivo contendo a lista invertida para o nome da pessoa.*
    - *Deve-se criar um arquivo contendo a lista invertida para a cidade da pessoa.*
  - O sistema deverá realizar alterações nas listas invertidas sempre que novos registros forem inseridos, alterados ou deletados no arquivo de dados.

- O sistema deve ser capaz de receber uma busca por nome e/ou cidade. A partir do nome e/ou cidade digitados, o sistema deve ser capaz de retornar em quais *ids* de registros a pesquisa foi capaz de encontrar os termos informados.

O que deve ser entregue:

Deve ser entregue um relatório (**usando LaTeX - utilizar "SBC Conferences Template"**) como resultado da realização deste trabalho prático (máximo 10 páginas). Anexo a esse relatório deve ser entregue o projeto do sistema desenvolvido (formato .zip). Esse relatório deve conter a seguinte estrutura:

- Título
- Resumo
- Introdução
- Desenvolvimento
- Testes e Resultados
- Conclusão

Além do relatório, o grupo deve criar um vídeo (duração máxima de 10 minutos), com:

- Explicação das principais decisões de implementação dos códigos criados.
- Demonstração da execução do sistema.
- Testes e resultados realizados.

Critérios para avaliação

- Implementação do sistema (15 pontos)
  - Correção e robustez dos programas
  - Conformidade às especificações
  - Clareza de codificação
  - Critérios de escolha
- Documentação (1 ponto)
- Vídeo (1 ponto)

NOTA FINAL = Implementação x Documentação x Vídeo

Observação final: ponto(s) extra(s) pode(m) ser dado(s) para trabalhos considerados excelentes.