
Curso	Sistemas de Informação
Disciplina	Programação Orientada a Objetos 1
Professor	José Gustavo de Souza Paiva

1. Tema central

Coleções e arquivos em Java.

2. Conteúdo

Para cada um dos problemas abaixo, construa o programa completo na linguagem de programação Java capaz de verificar a funcionalidade de cada uma das funções solicitadas.

- a) Construa uma classe `Empregado`, com atributos CPF, idade, e salário. Pesquise na Internet como dotar a classe `Empregado` da capacidade de decidir, dados dois empregados, qual possui a maior idade (DICA: pesquise sobre a interface `Comparable`). Depois escreva uma classe principal que:
- Instancia uma lista de 5 empregados;
 - Pede ao usuário os dados dos 5 empregados, instancia os objetos e armazena na lista;
 - Realiza a busca de um usuário dado o seu CPF;
 - Pede ao usuário um valor de salário, e retorna todos os empregados com salário maior do que o salário informado;
 - Utilize a Classe `Collections` para ordenar o vetor em ordem crescente, de acordo com o valor definido no comparador.
- b) Utilizando as informações do exercício anterior, utilize um `HashMap` para criar uma coleção indexada pelo CPF dos empregados, e realize as seguintes operações no programa principal:
- Instancia um `hashmap` com 5 empregados;
 - Pede ao usuário os dados dos 5 empregados, instancia os objetos e armazena no `hashmap`;
 - Realiza a busca de um usuário dado o seu cpf;
 - Retorna os CPFs dos Empregados com menor e maior salários.
- c) Construa um programa que leia dois arquivos texto (`salbruto.dat` e `desc.dat`) contendo, respectivamente, os salários brutos e os descontos de até 100 funcionários, identificados pelo CPF, sendo os dados de cada registro em uma linha separada. Considere que os dados podem não estar na mesma ordem nos dois arquivos. Para cada registro, deve-se calcular o salário líquido (sal. bruto – descontos) e gravá-lo no arquivo texto `salliq.txt`.

- d) Pretende-se simular um sistema de gestão de contatos. Para isso, faça um programa em Java, seguindo os passos seguintes:
- Implemente uma classe `Contato` composta pelos seguintes atributos: Nome, Idade, Sexo, Profissão, Telefone, Email. Essa classe deve conter um construtor que recebe parâmetros de inicialização de todos os atributos, e um método `String toString()`, que devolve uma String com o seguinte formato:
"`<Nome>`, `<Idade>` anos, sexo `<Sexo>`, `<Profissão>`, telefone n.º`<Telefone>`, e-mail: `<Email>`"
 - Implemente uma classe `GestaoContatos`, que deve conter três *ArrayList's*: Família, Amigos e Profissional. Essas *ArrayList's* destinam-se, respectivamente, a contatos de familiares, amigos e de âmbito profissional. Para essa classe, implemente os seguintes métodos:
 - `adicionaContato`, que recebe um contato a ser criado e um *ArrayList* no qual o contato será inserido;
 - `eliminaContato`, que recebe o nome do contato e o elimina do *ArrayList* em que este se encontrar. Se não encontrar em nenhum, lance uma exceção `ContatoNaoEncontradoException`, que deve ser criada;
 - `listaContatos`, que recebe um argumento correspondente à *ArrayList* pretendida e escreve na tela todos os contatos dessa lista;
 - `toString` que retorne o conteúdo de todas as *ArrayList's*.
 - `maisVelho`, que devolve o contato mais velho da lista correspondente ao argumento que este recebe. Implemente, também, um método `maisNovo`.
 - Crie um programa principal que permita ao usuário gerenciar seus contatos pessoais. Faça funcionalidades para utilizar as estruturas criadas, e permita que o usuário salve/carregue as informações em um arquivo (texto ou binário), de forma a persistir os dados corretamente;
- e) Implemente o mesmo programa, substituindo o uso dos *ArrayList's* por um *HashMap* ou *TreeMap*. Faça alterações para que o uso dessas estruturas otimize a organização dos dados.