Descrição de Exercício

Strings e Arrays

Objetivo: Este exercício tem como objetivo a familiarização com objetos do tipo *String* e com *Arrays*.

Observação

Para este exercício, utilize as definições de `Conta` e `Cliente` do Exercício 3.

Strings

1. Criar dentro uma nova classe, chamada `StringTest`. Esta classe conterá apenas um método `main()`.
2. Declarar no método `main` duas Strings (`s1` e `s2`). Inicializar cada uma delas utilizando o construtor da classe String que recebe como parâmetro apenas uma String ("José" ou seu nome, por exemplo). Colocar o mesmo parâmetro para a inicialização de ambas as Strings.
3. Imprimir na tela a soma dos tamanhos de `s1` e `s2` (dica: utilizar o método `length()` da classe String).
4. Imprimir na tela se o conteúdo de `s1` e `s2` é o mesmo. Para isso, utilizar o método `equals()`.
5. Imprimir na tela se `s1` e `s2` são o mesmo objeto. Para isso, utilizar o operador de igualdade (`==`). Você consegue compreender a diferença entre o resultado do item 5 e o deste item?
6. Declarar agora duas novas Strings (`s3` e `s4`). Inicialize-as não utilizando um construtor como no item 3, e sim atribuindo diretamente a elas uma String, como você faria normalmente com uma variável de tipo primitivo.
7. Imprimir na tela se `s3` e `s4` possuem mesmo conteúdo (`equals`) e se são o mesmo objeto (`==`). Analisar os resultados.
8. Atribuir um novo valor a `s3` e repetir o item 8. Analisar os resultados.

Arrays

(Crie um novo projeto para executar as tarefas abaixo)

0. Estude a classe `RepositorioContasArray` vista em sala de aula, crie um arquivo `.java` contendo o código desta classe. Certifique-se que há métodos `get` e `set` para os atributos (exceto a constante) . Caso estes não existam, defina-os.

1. Criar uma nova classe intitulada `RepositorioClientesArray`. Esta classe deve conter como atributos um *array* de `Clientes`, um índice (inteiro) que identifica o tamanho atual do *array* e uma variável estática do tipo inteiro, intitulada (`tamanhoCache`), que armazena a capacidade máxima do *array*. Inicialize-a com 100. Este atributo deve ser uma constante pública.

2. Criar um construtor para `RepositorioClientesArray`, que não recebe nenhum parâmetro e inicializa o índice e o *array* de clientes. Crie métodos `get` e `set` para os atributos (exceto a constante) .
3. Criar o método `inserir`, que recebe um cliente como parâmetro e o adiciona ao *array* de clientes. Observe a variável índice (que identifica o tamanho atual do *array*) também deve ser atualizada.
4. Criar o método `procurarIndice`, que recebe como parâmetro o cpf de um cliente e retorna a posição desse cliente no *array* de clientes. Este método deve retornar (-1) caso o cliente não seja encontrado.
5. Criar o método `existe`, que recebe como parâmetro o cpf de um cliente e retorna um `boolean` identificando se o cliente existe ou não no *array* de clientes. Dica: utilize o método `procurarIndice` aqui.
6. Criar o método `procurar`, que recebe como parâmetro o cpf de um cliente e retorna o cliente correspondente. Caso o cliente não exista, imprimir uma mensagem de erro na tela e retornar o valor `null` . Dica: utilize os métodos `existe` e/ou `procurarIndice` aqui.
7. Criar o método `atualizar`, que recebe como parâmetro um cliente (que substituirá o cliente que possui o mesmo cpf no *array* de clientes). Caso o cliente a ser substituído não exista, imprimir uma mensagem de erro na tela. Dica: utilize os métodos `existe` e/ou `procurarIndice` aqui.
8. Criar o método `remover`, que recebe como parâmetro o cpf do cliente a ser removido do *array*. Caso o cliente a ser removido não exista, imprima uma mensagem de erro na tela. Dica: utilize os métodos `existe` e/ou `procurarIndice` aqui.
9. Criar uma classe de teste contendo o método `main()` de tal forma que ele implemente a seguinte interface textual com o usuário:

```
*** Aplicação bancária ***

Operações disponíveis:

1- Criar cliente
2- Consultar cliente
3- Atualizar cliente
4- Remover cliente
5- Criar conta
6- Consultar conta
7- Atualizar conta
8- Remover conta
9- Creditar em conta
10- Debitar de conta
11- Transferir entre contas
12- Exibir os dados da conta de um determinado cliente
13- Exibir os dados de todas as contas
14- Exibir os dados de todos os clientes
15- Sair da aplicação

Favor escolher uma opção:_
```

O usuário deverá fornecer uma opção; após isso, a aplicação deve perguntar a ele os dados necessários para a execução da operação; a aplicação, então, executa a operação selecionada pelo usuário e imprime novamente a tela acima esperando nova seleção de opção pelo usuário. Aplicação deve terminar apenas quando o usuário selecionar a opção 15. **Abaixo seguem os dados que devem perguntados ao usuário pela aplicação** e as restrições que devem ser satisfeitas pelas operações:

Operação 1: Nome do cliente `(String)`, CPF do cliente `(String)`

Operações 2,3,4: CPF do cliente `(String)`
(Restrição: se o cliente não existir, exibir mensagem de erro. A operação 2 deve exibir os dados do cliente, CPF e nome; a operação 3 deve perguntar um CPF de um cliente existente e um novo nome.)

Operação 5: Número da conta `(String)`, CPF do cliente `(String)`
(Restrição: não se deve criar conta para um cliente inexistente; ou seja, caso não exista cliente com o CPF fornecido, deverá ser exibida uma mensagem de erro na interface textual.)

Operações 6,7,8: Número da conta `(String)`
(Se a conta não existir, exibir mensagem de erro. A operação 6 deve exibir os dados da conta: nome, saldo, e os dados dos cliente associado).

Operações 9,10: Número da conta `(String)`, valor `(double)`
(se a conta não existir, exibir mensagem de erro).

Operação 11: Número da conta de origem `(String)`, Número da conta de destino `(String)` valor `(double)` a ser transferido.
(se alguma conta não existir, exibir mensagem de erro e não alterar o estado das contas).

Operação 12: CPF do cliente `(String)`
(se o cliente não existir, exibir mensagem de erro; caso contrário, exibir os dados da Conta associada a ele.)

Para fazer leitura dos dados, dentro do método `main()`, faça chamadas a alguns métodos da classe `Scanner`, cujo uso é exemplificado no arquivo `TestScanner.zip`, que se encontra no diretório do exercício.

Operações 13,14: a aplicação não pergunta coisa alguma ao usuário e simplesmente imprime o que é solicitado (*Dica: pense em usar métodos `get` dos repositórios*).