PUC Minas
Instituto de Ciências Exatas e Informática

Pontifícia Universidade Católica de Minas Gerais
Bacharelado em Engenharia de Software
Laboratório de Computação III
Prof.: Henrique Batista da Silva
Lista de Exercícios 1

Orientações:

- Os exercícios devem ser feitos em linguagem Java

Parte I

1- Considere a classe **Vetor**, cuja assinatura está definida abaixo:

```
public class Vetor {

	public static final int TAMANHO = 10;

	private int v [] = new int[TAMANHO];  // Armazena os elementos
	private int numElementos;   // Informação sobre o número de elementos inseridos

	public Vetor() { }

	public int obtemTamanho() { return 0; }

	public void insereNoFinal(int novoElemento) { }

	public int posicaoDe (int elemento) { return 0; }

	public void alteraEm (int pos, int novoValor) { }

	public int elementoDe (int pos) { return 0; }

	public int elementoEm (int pos) { return 0; }

	public void reverte() { }

	public void imprime() { }

}
```

Implemente todos os métodos da classe **Vetor** acima, obedecendo às seguintes descrições dos métodos:

a. Vetor(): Construtor que inicializa o atributo numElementos com zero e zera todas as posições do vetor;
b. obtemTamanho(): Retorna o número de elementos armazenados no vetor;
c. insereNoFinal(novoElemento): Insere novoElemento na primeira posição vazia do vetor;
d. posicaoDe(elemento): Retorna a posição da primeira ocorrência de elemento no vetor ou o valor -1, caso o elemento não seja encontrado.
e. alteraEm(pos, novoValor): Atribui novoValor ao elemento da posição pos do vetor, caso esta seja uma posição válida, retorna -1, caso contrário.

f. elementoEm(pos): Retorna o valor armazenado na posição pos, caso esta seja uma posição válida, retorna -1, caso contrário.
g. imprime(): Imprime os elementos do vetor, separados por espaço.
h. reverte(): Reverte os elementos do vetor. Por exemplo, o vetor inicial {1,3,4,1,2}, após a chamada desta função, o vetor ficará {2,1,4,3,1,};

Após a implementação da classe Vetor, a função main, a seguir, deverá funcionar:

```
public class Program {

	public static void main(String[] args) {
		// TODO Auto-generated method stub

	 Vetor V = new Vetor();

	  V.insereNoFinal(10); V.insereNoFinal(8);
	  V.insereNoFinal(16); V.insereNoFinal(7);
	  V.insereNoFinal(5);  V.insereNoFinal(13);

	  V.imprime();
	  V.alteraEm(3,19); V.alteraEm(15,9);
	  for (int i = 0; i < V.obtemTamanho(); i++)
	       System.out.println("Elemento na posicao " + i + ": " + V.elementoEm(i));

	  V.reverte();
	  V.imprime();

	 Vetor v = new Vetor();

	}
}
```

2- Nesta aula, você deverá alterar a classe Vetor (definida anteriormente) para:

a. Permitir a realocação do vetor ao se tentar inserir um elemento e não houver posições vazias. A cada realocação você deverá dobrar o tamanho do vetor. Inicie com 10 elementos.

b. Permitir a retirada de todas as ocorrências de um elemento. Ao remover um elemento, se este não for o último elemento, deve-se deslocar todos os demais para não deixar posições vazias no vetor. Se o número de elementos após a retirada ficar menor que a metade do tamanho do vetor, então você deverá realocar o vetor com a metade do número de elementos.

c. Modifique o programa principal (função main) para testar se novo programa.

3- Considere que você foi contratado para desenvolver um sistema para um Banco.

No sistema bancário precisamos armazenar as informações dos clientes, como número e nome. Crie uma classe para modelar os objetos que representarão os clientes.

```
public class Cliente {

	private int Numero;
	private String Nome;
```

```
}
```

Crie um construtor para a classe Cliente

```
public Cliente (String nome, int numero)
{
	this.Nome = nome;
	this.Numero = numero;
}
```

Crie também a classe conta com os seguintes atributos:

```
public class Conta {

	private int Numero;
	private double Saldo;
	public Cliente Titular;
}
```

Crie também o construtor da classe conta

```
public Conta (Cliente cliente, int numero)
{
	this.Titular = cliente;
	this.Numero = numero;
	this.Saldo = 0
}
```

Suponha que o banco precise realizar operações de saque, depósitos e transferências. No entanto, as operações de saque e transferência possuem algumas regras. Há uma taxa de 10% sobre o valor da operação para cada retirada (seja ela do tipo saque ou transferência).

Para a classe Conta inclua os métodos deposita, saque e transfere.

```
public void deposita(double valor) { }

public double saque(double valor) { return 0.0; }

public double transfere (double valor, Conta destino) { return 0.0; }
```

Outra regra é que o banco não permite que o cliente tenha saldo negativo em sua conta. As operações de saque e transferência devem tratar para que o saldo não fique negativo antes de realizar a operação.

Implemente o programa conforme especificado abaixo:

a. Implemente as operações de saque, transferência e depósito de cada conta. Atente-se que para as operações de saque e transferência é necessário verificar se há saldo suficiente antes de realizar a operação.

b. Implemente uma operação para verificar o saldo do cliente (observe a visibilidade do atributo saldo da classe conta).

c. Implemente um programa principal para criar dois clientes e duas contas para cada um dos clientes e proceda com as operações de depósito para cada cliente, saque e transfira uma quantia

de valor de um cliente para o outro. Seu programa deve imprimir na tela a ocorrência de cada operação (ex.: cliente A transferiu R$ 10,00 para o cliente B).

d. Ao final, seu programa deverá imprimir na tela o nome de cada cliente e o saldo de suas respectivas contas.

e. Crie testes unitários usando o jUnit para testar os métodos *deposita*, *saque* e *transfere* (opcional).

Parte II

1- Implemente um procedimento que receba como parâmetro duas Filas e retorne uma terceira contendo a concatenação das duas. Seu procedimento deve usar apenas os métodos já implementados (Fila e/ou Pilha). O nome da função será concatenaFila.

2- Implemente um procedimento que receba como parâmetro duas Pilhas e retorne uma terceira contendo a concatenação das duas. Isto é, o fundo da terceira pilha é o mesmo funda da primeira e o topo da terceira pilha é o mesmo topo da segunda. Seu procedimento deve usar apenas os métodos já implementados (Pilha e/ou Fila). O nome da função será concatenaPilha.

Parte III

Faça um programa de criptografa de dados, ou seja, um programa capaz de ler um arquivo texto, codificar este arquivo através de alguma técnica de alteração do código ASCII (exemplo: somar 1 ao valor ASCII de cada caractere), e escrever em disco o arquivo codificado. Crie um outro programa que descriptografe um arquivo criado pelo programa de criptografa, realizando a operação inversa: ler o arquivo do disco, decodificar e escrever o novo arquivo em disco descriptografado. Lembre-se que para que seja possível criptografar/descriptografar um arquivo a função de codificação deve possuir uma função inversa. Exemplo de código de criptografia:

```
public static void criptografa (String texto, int key, String path) throws IOException {

          FileWriter file = new FileWriter(path);
        BufferedWriter bw = new BufferedWriter(file);

        String textoCriptografado = "";

        for (int i = 0; i < texto.length(); i++)
        {
             textoCriptografado += (char)(texto.charAt(i) + key);
        }

        bw.write(textoCriptografado);

        bw.newLine();

        bw.close();
}
```

```
      public static String abreTextoCriptografado (int key, String path) throws IOException
{

            String textoAberto ="";

            FileReader file = new FileReader(path);
          BufferedReader br = new BufferedReader(file);

          String textoCriptografado = br.readLine();

          br.close();

            return textoAberto;
}

public static void main(String[] args) throws IOException, ClassNotFoundException {

        Scanner input = new Scanner(System.in);
        String texto = input.nextLine();

        String path = "arquivoTexto.txt";
        criptografa(texto, 1, path);

        System.out.println( abreTextoCriptografado(1, path) );

}
```

Você consegue desenvolver uma função de criptografia/descriptografia que seja mais sofisticada e menos óbvia do que esta?