Fundação CECIERJ - Vice Presidência de Educação Superior a Distância

Curso de Tecnologia em Sistemas de Computação
Disciplina: Programação Orientada a Objetos
AP2 2° semestre de 2016.

Nome –

Assinatura –

Observações:

1. Prova sem consulta e sem uso de máquina de calcular.
2. Use caneta para preencher o seu nome e assinar nas folhas de questões e nas folhas de respostas.
3. Você pode usar lápis para responder as questões.
4. Ao final da prova devolva as folhas de questões e as de respostas.
5. Todas as respostas devem ser transcritas nas folhas de respostas. As respostas nas folhas de questões não serão corrigidas.

Questão 1) (3.0 pontos)

Considere o gabarito da questão 2 da AP1, a qual pedia para implementar um carrinho de compras. Crie agora uma classe CarrinhoDeCompras2, usada método main() abaixo, o qual implementa o carrinho como uma lista e não um vetor. Além dos métodos adiciona() e soma(), crie o método ***boolean retira (int cod)***, o qual usa o método getCodigo() da classe Produto e remove o produto com o respectivo código. Este método deve retorna **true** quando a remoção acontecer, e **falso** caso contrário.

```
class CarrinhoDeCompras {
	Produto produtos[];
	int ultimo = 0;
	public CarrinhoDeCompras() {
		produtos = new Produto[1000];
	}
	public void adiciona (Produto p) {
		produtos[ultimo] = p;
		ultimo++;
	}
	public double soma () {
		double s = 0;
		for (int i=0; i<ultimo; i++)
			s = s + produtos[i].getPreco();
		return s;
	}
}

public class AP2_2016_2_Q2 {
	public static void main(String[] args) {
```

```
                Produto tv = new Produto(1, "TV Plasma", 1500);
                Produto geladeira = new Produto(2, "Geladeira Frost Free", 1000);
                Produto sofa = new Produto(3, "Sofa 3 lugares", 500);
                CarrinhoDeCompras2 carrinho = new CarrinhoDeCompras2();
                carrinho.adiciona(tv);
                carrinho.adiciona(geladeira);
                carrinho.adiciona(sofa);
                System.out.println("Valor do carrinho: " + carrinho.soma());
        }
}
```

RESPOSTA:

```
import java.util.ArrayList;
import java.util.List;

class CarrinhoDeCompras2 {
        List<Produto> produtos;

        public CarrinhoDeCompras2() {
                produtos = new ArrayList<Produto>();
        }

        public void adiciona (Produto p) {
                produtos.add(p);
        }

        public double soma () {
                double s = 0;
                for (Produto p : produtos)
                        s = s + p.getPreco();
                return s;
        }

        public boolean retira (int cod) {
                for (Produto p : produtos) {
                        if (p.getCodigo() == cod) {
                                produtos.remove(p);
                                return true;
                        }
                }
                return false;
        }
}
```

Questão 2) (3.0 pontos)

Suponha um sistema de controle de pedidos num restaurante. Neste sistema, um cliente pode ser identificado pelo número de sua mesa ou pelo número do seu telefone, no caso do pedido ser entregue em domicílio. Um pedido contém o código do prato, o preço deste e o cliente correspondente. Os pedidos a serem entregues em domicílio tem acréscimo de 10% (Para acrescer

um valor de x%, basta multiplicar este valor por (1 + x/100). Baseado no código do método main(), informado abaixo, apresente uma implementação para as classes utilizadas neste código.

```
import java.util.ArrayList;
import java.util.List;

public class AP2_2016_2_Q2 {
	public static void main(String[] args) {
		List<Pedido> pedidos = new ArrayList<Pedido>();
		pedidos.add(new PedidoLocal(1, 50.0, 1)); // Código do prato, valor e
código do cliente
		pedidos.add(new PedidoLocal(2, 20.0, 1));
		pedidos.add(new PedidoLocal(3, 100.0, 1));
		pedidos.add(new PedidoViagem(1, 50.0, "9999-9999")); // Código do prato,
valor e telefone do cliente
		pedidos.add(new PedidoViagem(2, 20.0, "8888-8888"));
		double soma = 0;
		for (Pedido p : pedidos) {
			soma = soma + p.getPrecoPrato();
		}
		System.out.println("A soma é: " + soma);
	}
}
```

RESPOSTA:

```
import java.util.ArrayList;
import java.util.List;

class Pedido {
	int codPrato;
	double precoPrato;
	public Pedido(int codPrato, double precoPrato) {
		this.codPrato = codPrato;
		this.precoPrato = precoPrato;
	}

	public double getPrecoPrato() {
		return precoPrato;
	}
}

class PedidoViagem extends Pedido {
	String telefone;

	public PedidoViagem(int codPrato, double precoPrato, String tel) {
		super(codPrato, precoPrato);
		this.telefone = tel;
	}

	public double getPrecoPrato() {
		return super.getPrecoPrato()*1.1;
	}
}
```

```
class PedidoLocal extends Pedido {
	int mesa;

	public PedidoLocal(int codPrato, double precoPrato, int mesa) {
		super(codPrato, precoPrato);
		this.mesa = mesa;
	}
}
```

Questão 3) (4.0 pontos)
Implemente, em Java, o algoritmo de *Balance Line*. Dados dois arquivos informados como parâmtero de entrada, o arquivo mestre e o arquivo de transações, será gerado um novo aqruivo mestre, cujo o nome será **mestre-<nome do arquivo mestre de entrada>**, que será formado pela implementação das transações no arquivo mestre original. Caso contrário, se a operação não puder ser realizada, será gerado o arquivo **erro-<nome do arquivo mestre de entrada>**, contendo as transações não realizadas. As operações, **I (inclusão), R (retirada) e A (alteração)**, são realizadas sobre matrículas únicas.

Por exemplo, se o arquivo mestre for (onde cada linha é composta de duas informações, a matrícula de um aluno e o seu CR):

```
1  10.0
2  0.3
3  7.6
4  5.5
10 7.2
15 2.3
```

E o arquivo de transações for (cada linha é composta da matrícula, da transação e do novo CR, para as operações **A** e **I)** :

```
1  R
2  A 5.0
3  I 0.5
4  R
5  I 8.9
6  R
9  A 10.0
20 R
```

O novo arquivo mestre será:

```
2  5.0
3  7.6
5  8.9
10 7.2
15 2.3
```

O arquivo de erros será (isto é, as transações que não puderam ser realizadas):

```
3  I 0.5
6  R
9  A 10.0
20 R
```

SEU PROGRAMA DEVE EXECUTAR COM QUAISQUER ARQUIVOS INFORMADOS COMO PARÂMETROS DE ENTRADA. SE O SEU PROGRAMA RESOLVER SOMENTE O PROBLEMA DO EXEMPLO SUPRACITADO, SUA QUESTÃO SERÁ TOTALMENTE DESCONTADA.

RESPOSTA:

```
import java.io.*;

public class AP2_POO_2016_2_Q3{
  public static void main(String[] args) throws IOException{
    BufferedReader in_m, in_t;
    in_m = new BufferedReader(new FileReader(args[0]));
    in_t = new BufferedReader(new FileReader(args[1]));
    BufferedWriter out_m, out_e;
    out_m = new BufferedWriter(new FileWriter("mestre-" + args[0]));
    out_e = new BufferedWriter(new FileWriter("erro-" + args[0]));

    try {
      String s_m, vs_m[], s_t, vs_t[];
      s_m = in_m.readLine();
      s_t = in_t.readLine();
      while((s_m != null) && (s_t != null)){ //mat iguais...
        vs_m = s_m.split(" ");
        vs_t = s_t.split(" ");
        if(vs_m[0].equals(vs_t[0])){
          if(vs_t[1].equals("I")){
            out_e.write(s_t + "\n");
            out_m.write(s_m + "\n");
          }
          else if(vs_t[1].equals("A"))
            out_m.write(vs_t[0] + " " + vs_t[2] + "\n");
          s_m = in_m.readLine();
          s_t = in_t.readLine();
        }
        //matricula do mestre e menor...
        else if(Integer.parseInt(vs_m[0])<Integer.parseInt(vs_t[0])){
          out_m.write(s_m + "\n");
          s_m = in_m.readLine();
        }
        //matricula do arquivo de transacoes e menor...
        else{
          if(vs_t[1].equals("I"))
            out_m.write(vs_t[0] + " " + vs_t[2] + "\n");
```

```
          else
            out_e.write(s_t + "\n");
          s_t = in_t.readLine();
        }
      }
      //acabou o arquivo de transacoes: copia o resto no novo mestre...
      while (s_m != null){
        out_m.write(s_m + "\n");
        s_m = in_m.readLine();
      }
      //acabou o arquivo mestre: deve-se analisar todas as transicoes...
      while (s_t != null){
        vs_t = s_t.split(" ");
        if(vs_t[1].equals("I")){
          out_m.write(vs_t[0] + " " + vs_t[2] + "\n");
        }
        else out_e.write(s_t + "\n");
        s_t = in_t.readLine();
      }
      in_m.close();
      in_t.close();
      out_e.close();
      out_m.close();
    }
    catch (Exception e){
      System.out.println("Excecao\n");
    }
  }
}
```