Fundação CECIERJ - Vice Presidência de Educação Superior a Distância

Curso de Tecnologia em Sistemas de Computação
AD2 de Programação III
2° semestre de 2010.

Nome:
Matrícula:
Pólo:

Obs: *A solução para o exercício proposto deve ser entregue por escrito e em formato digital (arquivos .java).*

Suponha o código abaixo que manipula um objeto que representa um carrinho de compras.

```
1 public class AD2_2010_2 {
2   public static void main(String[] args) {
3      EstoqueProdutos produtos = new EstoqueProdutos();
4      produtos.adicionaProduto(new ProdutoEstoque ("monitor", 500, 100));
5      produtos.adicionaProduto(new ProdutoEstoque ("telefone", 150, 300));
6      produtos.adicionaProduto(new ProdutoEstoque ("teclado", 70, 50));
7      produtos.adicionaProduto(new ProdutoEstoque ("mouse", 50, 50));
8
9      CarrinhoCompra carrinho = new CarrinhoCompra(produtos);
10     carrinho.adicionaItem("monitor", 2);
11     carrinho.adicionaItem("telefone", 5);
12     carrinho.adicionaItem("teclado", 2);
13     carrinho.finalizaCompra();
14
15     System.out.println("Soma dos produtos: " + carrinho.calculaTotal());
16   }
17 }
```

Para o programa acima, realize as seguintes inserções (utilize os conceitos de OO vistos sempre que possível):

a) Defina a classe *ProdutoEstoque*, cujo objetos são criados das linhas 4 a 7. Um produto, representado por esta classe, possui um nome, um valor e a quantidade deste produto em estoque.
b) Crie a classe *EstoqueProdutos*, cujo objeto é criado na linha 3. Esta classe deve permitir que produtos possam ser armazenados, como indica o método *adicionaProduto*, chamado das linhas 4 a 7.

c) Crie a classe CarrinhoCompra, a qual permitirá a adição de produtos à partir de um estoque. Esta classe precisará implementar, pelo menos, 3 métodos: adicionaItem(), (linhas 10 a 12), que adiciona um item ao carrinho; finalizaCompra(), (linha 13), que debita efetivamente a quantidade do produto no estoque da quantidade comprada, e; calculaTotal(), (linha 15), que dá o cômputo total dos produtos (soma dos valores * a quantidade destes).

RESPOSTA:

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

/*
 * Classe produto contendo o valor e o nome do produto
 */
class Produto {
	private float valor;
	private String nome;
	public Produto(String n, float v) {
		nome = n;
		valor = v;
	}
	public float getValor() {
		return valor;
	}
	public void setValor(float valor) {
		this.valor = valor;
	}
	public String getNome() {
		return nome;
	}
	public void setNome(String nome) {
		this.nome = nome;
	}
}

/*
 * Classe ProdutoEstoque pedida no item a)
 */

class ProdutoEstoque extends Produto {
	private int quantidade;
	public ProdutoEstoque(String n, float v, int qtd) {
		super(n, v);
		quantidade = qtd;
	}
	public int getQuantidade() {
		return quantidade;
	}
	public void setQuantidade(int quantidade) {
		this.quantidade = quantidade;
	}
```

```
}

/*
 * Classe que modela um item a ser inserido no carrinho.
 * Esta não foi explicitamente pedida, mas facilita a
 * criação da classe CarrinhoCompra
 * Além dos campos da classe Produto, declaramos um campo
 * quantidade, o qual define a quantidade de itens do
 * produto no carrinho
 */
class ItemCompra {
	private int quantidade;
	private ProdutoEstoque produto;
	public ItemCompra(ProdutoEstoque prod, int quant) {
		produto = prod;
		quantidade = quant;
	}
	public int obtemQuantidade () {
		return quantidade;
	}
	public ProdutoEstoque getProduto() {
		return produto;
	}
	public void setProduto(ProdutoEstoque produto) {
		this.produto = produto;
	}
}

/*
 * Classe que representa o estoque. Contém uma lista
 * de produtos e métodos para adição e obtenção de um
 * produto.
 * Naturalmente, para uma modelagem mais realista, deveríamos
 * criar outros métodos, como a remoção de produtos
 */
class EstoqueProdutos {
	private List<ProdutoEstoque> produtos;
	public EstoqueProdutos() {
		produtos = new ArrayList<ProdutoEstoque>();
	}
	public void adicionaProduto (ProdutoEstoque prod) {
		produtos.add(prod);
	}
	public ProdutoEstoque obtemProduto (String nome) {
		// Forma simplificada de percorrermos uma coleção
		// Alternativamente, poderíamos usar a interface Iterator
		for(ProdutoEstoque prod : produtos) {
			if (prod.getNome() == nome)
				return prod;
		}
		return null;
	}
	public void removeEstoque (String nome, int qtd) {
		ProdutoEstoque prod = this.obtemProduto(nome);
```

```
                prod.setQuantidade(prod.getQuantidade() - qtd);
        }
}

/*
 * Classe que modela um carrinho de compras.
 * Esta possui uma referência para o estoque sobre o qual
 * o carrinho se baseará e uma lista de itens
 */
class CarrinhoCompra {
        private List<ItemCompra> produtos;
        private EstoqueProdutos estoque;
        public CarrinhoCompra(EstoqueProdutos e) {
                produtos = new ArrayList<ItemCompra>();
                estoque = e;
        }
        public void adicionaItem (String nome, int quant) {
                ProdutoEstoque prod = estoque.obtemProduto(nome);
                if (prod != null && prod.getQuantidade() >= quant)
                        produtos.add(new ItemCompra(prod, quant));
        }
        public void adicionaItem (ItemCompra item, int quant) {
                ProdutoEstoque prod =
estoque.obtemProduto(item.getProduto().getNome());
                if (prod.getQuantidade() >= quant)
                        produtos.add(item);
        }
        public float calculaTotal () {
                float soma = 0;
                for (ItemCompra item : produtos) {
                        soma = soma + (item.getProduto().getValor() *
item.obtemQuantidade());
                }
                return soma;
        }
        public void finalizaCompra() {
                for (ItemCompra item : produtos) {
                        estoque.removeEstoque(item.getProduto().getNome(),
item.obtemQuantidade());
                }
        }
}

public class AD2_2010_2 {
        public static void main(String[] args) {
                EstoqueProdutos estoque = new EstoqueProdutos();
                estoque.adicionaProduto(new ProdutoEstoque ("monitor", 500, 100));
                estoque.adicionaProduto(new ProdutoEstoque ("telefone", 150,
300));
                estoque.adicionaProduto(new ProdutoEstoque ("teclado", 70, 50));
                estoque.adicionaProduto(new ProdutoEstoque ("mouse", 50, 50));

                CarrinhoCompra carrinho = new CarrinhoCompra(estoque);
                carrinho.adicionaItem("monitor", 2);
```

```
            carrinho.adicionaItem("telefone", 5);
            carrinho.adicionaItem("teclado", 2);

            carrinho.finalizaCompra();

            System.out.println("A soma dos produtos Ã©: " +
carrinho.calculaTotal());
      }
}
```