Fundação CECIERJ - Vice Presidência de Educação Superior a Distância

Curso de Tecnologia em Sistemas de Computação
Disciplina: Programação III
AP3 1° semestre de 2015.

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) (5.0 pontos)

Os dados das vendas de um supermercado estão armazenados num arquivo texto passado como parâmetro de entrada. O arquivo contém na primeira linha o total de vendas realizadas (valor inteiro positivo) e nas linhas seguintes os registros das vendas. Cada registro ocupa seis linhas, onde cada uma dessas linhas possui uma das seguintes informações, nesta ordem: *número da venda* (valor inteiro positivo), *data da venda* (compatível com java.util.Date), *código do produto vendido* (valor inteiro), *nome do produto vendido* (string de tamanho não conhecido, podendo conter espaços em branco), *quantidade vendida* (valor inteiro positivo) e *preço unitário do produto* (valor real positivo).

Desenvolva uma aplicação Java que execute um algoritmo composto exatamente pelos seguintes passos: (1) ler os dados do arquivo de entrada, criando objetos dos tipos Produto e Venda, armazenando-os em alguma(s) estrutura(s) de dado(s); e (2) fazer uso dos dados armazenados nesta(s) estrutura(s) para retornar, em ordem decrescente, a quantidade de cada produto adquirido pelos clientes (isto é, o(s) produto(s) mais vendido(s) será(ão) listado(s) primeiramente).

Note que o código de um produto qualquer pode aparecer em mais de uma venda registrada no arquivo de entrada. Porém, a coleção de produtos não poderá conter mais de um objeto com mesmo código, bem como vendas de um mesmo produto não poderão estar associadas a objetos distintos.

Isso implica na implementação de um mecanismo de unicidade de objetos de produtos incluídos na coleção e na correta associação das vendas a esses objetos únicos. Você deve assumir que os únicos atributos de um produto são: *código do produto, o preço do produto* e *nome do produto*, sendo o *código do produto* uma chave primária (isto é,

nenhum outro produto possuirá o mesmo código deste). Os demais atributos armazenados no arquivo de entrada pertencem à venda.

RESPOSTA:

```
import java.io.*;
import java.util.Date;

class Venda {
  int numero;
  Date data;
  Produto produto;
  int quantidade;

  public Venda(int numero, Date data, Produto produto, int quantidade) {
    if ((numero <= 0) || (data == null) || (produto == null) || (quantidade <=
0))
      throw new IllegalArgumentException("Verifique o valor dos
argumentos informados.");
    this.numero = numero;
    this.data = data;
    this.produto = produto;
    this.quantidade = quantidade;
  }

  public String toString(){ return "(" + numero + ") " + data + ":\t" +
produto.toString() + " " + quantidade + "\n"; }
}


class Produto {
  final int codigo;
  final String nome;
  final double preco;

  public Produto(int codigo, String nome, double preco) {
    if ((codigo <= 0) || (nome == null) || (preco <= 0))
      throw new IllegalArgumentException("Verifique o valor dos
argumentos informados.");
    this.codigo = codigo;
    this.nome = nome;
    this.preco = preco;
  }
```

```
  public String toString() { return "(" + this.codigo + ") " + this.nome + " "
+ this.preco;
  }
}


public class AP3_2015_1 {
  public static void main(String[] args) throws Exception{
    int n;
    try (BufferedReader in = new BufferedReader(new
FileReader(args[0]))) {
      n = Integer.parseInt(in.readLine());
      Produto[] colecaop = new Produto[n];
      Venda[] colecaov = new Venda[n];

      for (int i = 0; i < n; i++) {
        int numero = Integer.parseInt(in.readLine());
        Date data = new Date(in.readLine());
        int codigo = Integer.parseInt(in.readLine());
        String nome = in.readLine();
        int quantidade = Integer.parseInt(in.readLine());
        double valorUnitario = Double.parseDouble(in.readLine());

        for(int j = 0; j < i; j++)
          if(colecaop[j].codigo == codigo)
            throw new IllegalArgumentException("Verifique o valor do
arquivo de entrada informado.");
        Produto produto=new Produto(codigo, nome, valorUnitario);
        colecaop[i] = produto;

        for(int j = 0; j < i; j++)
          if(colecaov[j].numero == numero)
            throw new IllegalArgumentException("Verifique o valor do
arquivo de entrada informado.");
        colecaov[i]=new Venda(numero, data, produto, quantidade);
      }

      for(int i = 0; i < n; i++) System.out.print(colecaov[i]);
      System.out.println("Depois de Ordena...");
      Ordena(colecaop, colecaov);
```

```
    for(int i = 0; i < n; i++) System.out.print(colecaov[i]);
   }
   catch (FileNotFoundException erro) {
    System.err.println("O arquivo de vendas não foi encontrado.");
    return;
   }
   catch (IOException erro) {
    System.err.println("Houve algum erro na leitura do arquivo de
vendas.");
    return;
   }
  }


  static void Ordena(Produto[] col_prod, Venda[] col_venda){
   int i, j, maior;
   for(i = 0; i < col_prod.length; i++){
    maior = i;
    for(j = i + 1; j < col_prod.length; j++)
     if(col_venda[maior].quantidade < col_venda[j].quantidade)
      maior = j;
    if(maior != i){
     Produto temp = col_prod[i];
     col_prod[i] = col_prod[maior];
     col_prod[maior] = temp;
     Venda tempv = col_venda[i];
     col_venda[i] = col_venda[maior];
     col_venda[maior] = tempv;
    }
   }
  }
}
```

Questão 2) (5.0 pontos)

Considere a classe abaixo, a qual implementa um suposto carrinho de compras:

```
class Carrinho {
	List <Produto> produtos;
	public Carrinho() {
		produtos = new ArrayList<Produto>();
	}
	public double somaTotal() {
		double aux = 0;
```

```
                for (Produto p : produtos) {
                        aux = aux + p.getValor();
                }
                return aux;
        }
}
```

a) Implemente Produto, utilizado na classe.

b) Crie uma classe chamada Mouse, a qual é um tipo de produto possível de ser inserido no carrinho.

c) Crie uma classe chamada Teclado, similar a do item b).

d) Crie uma classe Computador, o qual possui uma lista de produtos e também pode ser inserida no carrinho.

e) Crie uma classe Main, a qual exemplifica a utilização de todas as classes acima.

Observe que tanto um teclado, um mouse, ou um computador inteiro podem ser inseridos no carrinho. Além disso, nas classes criadas somente o mínimo necessário para o funcionamento da classe Carrinho precisa ser fornecido.

RESPOSTA:

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

// item a)
// Poderia ser uma classe abstrata ou até uma classe
//concreta. Entretanto, como não havia restrição no
//código, uma interface seria o recurso recomendado
//neste caso
interface Produto {
        double getValor();
}

// item b)
// Como nada além era exigido, além de ser um tipo de
//produto, nada além era necessário. Entretanto, poderia
//se colocar mais detalhes, desde que estes fossem
//corretamente manipulados
class Mouse implements Produto {
        public double getValor() {
                return 10;
        }
}

// item c)
// Idem ao item acima
class Teclado implements Produto {
        public double getValor() {
                return 30;
        }
}
```

```java
// item d)
// Além deste possuir uma lista de produtos, computadores
//completos também podem ser inseridos no carrinho. Ou seja,
//ambos devem se comportar como produtos
class Computador implements Produto {
	List<Produto> pecas;
	public Computador(List<Produto> p) {
		this.pecas = p;
	}
	public double getValor() {
		double aux = 0;
		for (Produto p : pecas)
			aux = aux + p.getValor();
		return aux;
	}
}

// Classe fornecida no enunciado
class Carrinho {
	List <Produto> produtos;
	public Carrinho() {
		produtos = new ArrayList<Produto>();
	}
	public void adicionaProduto (Produto p) {
		produtos.add(p);
	}
	public double somaTotal() {
		double aux = 0;
		for (Produto p : produtos) {
			aux = aux + p.getValor();
		}
		return aux;
	}
}

// Item e)
// Era necessário que se criasse, ao menos, um objeto
//de cada classe manipulada
public class AP3_2015_1_Q2 {
	public static void main(String[] args) {
		Produto m = new Mouse();
		Produto t = new Teclado();
		List<Produto> pecas = new ArrayList<Produto>();
		pecas.add(m);
		pecas.add(t);
		Produto c = new Computador(pecas);
		Produto m2 = new Mouse();
		Produto t2 = new Teclado();
		Carrinho carrinho = new Carrinho();
		carrinho.adicionaProduto(c);
		carrinho.adicionaProduto(m2);
		carrinho.adicionaProduto(t2);
		System.out.println("A soma dos produtos no carrinho é: " +
carrinho.somaTotal());
    }
}
```