Fundação CECIERJ - Vice Presidência de Educação Superior a Distância

Curso de Tecnologia em Sistemas de Computação
Disciplina: Programação III
AP2 1° semestre de 2011.

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 pontos)

Considerando as seguintes declarações de nó e de uma lista encadeada (estes métodos já estão prontos. Isto é, você não precisa escrevê-los):

```
class no{
      int info;
      no prox;
      no(int info){ this.info = info; }
      public String toString(){ return "info: " + info; }
}

class lista{
      no prim;
      lista(){ ... };
      void ins_ini(int info){ ... }
      void ins_fim(int info){ ... }
      void imprime(){ ... }
      no busca(int info){ ... }
      void retira(int info){ ... }
}
```

Escreva um método que, dada uma lista circular, inverta todos os elementos desta lista. O protótipo deste método é `static void inverte(lista l)`.

RESPOSTA:

```
static void inverte (lista l){
  if(l.prim == null) return null;

  lista aux = new lista();
  aux.ins_ini(l.prim.info);
  no p = l.prim.prox;
  while(p != l.prim){
    aux.ins_ini(p.info);
    p = p.prox;
  }

  l.prim = aux.prim;
}
```

Questão 2) (3 pontos)

Supondo que você tenha uma empresa de software, e que uma empresa de hardware contrate sua empresa para escrever um programa que armazene todos os comandos utilizados no terminal de um computador. Seu sistema deve armazenar os comandos usados até o atual momento. Quando seu programa receber o comando **`hist -f`**, seu sistema deve listar, em ordem, todos os comandos usados, desde de o primeiro até o próprio **`hist -f`**, seu último comando. Se seu programa receber o comando **`hist -r`**, seu sistema deve listar todos os comandos usados em ordem reversa, partindo de **`hist -r`** até o primeiro comando. Quando seu programa receber o comando **`fflush`**, os comandos devem ser armazenados no arquivo cujo o nome é **`hist.txt`**. Toda vez que o comando **`fflush`** for usado, os comandos que forem guardados em **`hist.txt`** devem ser "esquecidos" pelo seu programa para que não haja repetição no arquivo de saída. Seu programa para quando o comando **`exit`** for usado, fazendo-se um **`fflush`** antes de terminar o programa.

RESPOSTA:

```
import java.io.*;
import java.util.*;
class no{
  String comando;
  no ant, prox;

  no(String comando){
    this.comando = comando;
    ant = prox = null;
  }

  public String toString(){
    String s = "comando: " + comando + "\n";
    return s;
  }
}

class LDE{
```

```
no prim, marca;

LDE(){ marca = prim = null; }

void imprime(){
  no aux = prim;
  while(aux != null){
    System.out.println(aux);
    aux = aux.prox;
  }
}

void imprime_contrario(){
  if(prim == null) return;

  no aux = prim;
  while(aux.prox != null) aux = aux.prox;
  while(aux != null){
    System.out.println(aux);
    aux = aux.ant;
  }
}

void ins_fim(String info){
  no novo = new no(info);

  if(prim == null) prim = novo;
  else{
    no aux = prim;
    while(aux.prox != null) aux = aux.prox;
    aux.prox = novo;
    novo.ant = aux;
  }
}

void escreve_Arq() throws IOException{
  no p;

  if(marca == null) p = prim;
  else p = marca.prox;

  if(p == null) return;

  BufferedWriter out = new BufferedWriter(new FileWriter("hist.txt", true));

  while(p.prox != null){
```

```
      out.write(p.toString());
      p = p.prox;
    }

    out.write(p.toString());
    marca = p;
    out.close();
  }
}


public class VS_Q4{
  public static void main(String[] args) throws IOException{
    try{
      Scanner in = new Scanner(System.in);
      LDE l = new LDE();
      String leitura = in.nextLine();
      while(!leitura.equals("exit")){
        l.ins_fim(leitura);
        if(leitura.equals("hist -f")) l.imprime();
        else if(leitura.equals("hist -r")) l.imprime_contrario();
        else if(leitura.equals("fflush")) l.escreve_Arq();
        leitura = in.nextLine();
      }

      l.ins_fim(leitura);
      l.escreve_Arq();
    }catch(IOException e){ System.out.println(e); }
  }
}
```

Questão 3) (4 pontos)

Suponha que você tenha que construir um sistema para ser utilizado na construção civil. Inicialmente, serão manipulados prédios mistos, ou seja, que possuem salas comerciais e apartamentos numa mesma construção. Por simplificação, tanto salas comerciais quanto apartamentos tem dimensão retangular, ou seja, possuem apenas 2 lados. Um apartamento, diferentemente de uma sala comercial, possui uma quantidade arbitrária de cômodos. Apartamentos e salas comerciais possuem a informação se estão posicionados na frente ou nos fundos do prédio, o que normalmente valoriza ou deprecia, respectivamente. Para efeitos de cálculo aproximado de custo, uma sala tem seu valor definido pelo preço do metro quadrado, o qual pode ser modificado, mas é o mesmo para todos os apartamentos e salas comerciais deste sistema. Além disso, estes têm uma acréscimo de 20% quando posicionados na parte da frente do prédio. No caso dos apartamentos, estes ainda têm um acréscimo de 10% para cada cômodo existente. Crie as classes necessárias para o cálculo do custo total do prédio. Utilize os conceitos de OO vistos sempre que possível.
Sugestões:
1) defina uma classe Predio, a qual conterá uma lista com as salas comerciais e apartamentos;
2) para aumentar um valor em 30%, basta multiplicar este valor por 1,3.

RESPOSTA:

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

class Predio {
	List<Sala> salas; // Em metros

	public Predio() {
		salas = new ArrayList();
	}

	public double custoTotal () {
		double retorno = 0;
		for (Sala s: salas) {
			retorno = retorno + s.valor();
		}
		return retorno;
	}
}

class Sala {
	double lado1, lado2;
	boolean frente; // Frente ou fundos
	static double preco;

	public double metragem () {
		return lado1 * lado2;
	}

	public static void setPreco (double p) {
		preco = p;
	}

	public double valor() {
		return this.metragem() * preco * ((frente) ? 1.2 : 1);
	}
}

class Apartamento extends Sala {
	int comodos;

	public double valor() {
		return super.valor() * (1 + (0.1 * comodos));
	}
}

public class AP2_2011_1_Q3 {
	public static void main(String[] args) {
	}
}
```