Fundação CECIERJ - Vice Presidência de Educação Superior a Distância

Curso de Tecnologia em Sistemas de Computação
Disciplina: Programação III
AP3 2° semestre de 2009.

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) (4 pontos)

Supondo que você tenha uma empresa de software, e que a CBF contrate sua empresa para escrever um programa que informe, automaticamente, os nomes dos times campeões do Campeonato Brasileiro.

O vencedor de um campeonato é determinado a partir do número de pontos do mesmo, e caso os times tenham o mesmo número de pontos, o critério de desempate é o número de vitórias, seguido do saldo de gols (o saldo de gols é dado pela diferença entre os gols feitos e os gols sofridos). Se mais de um time estão empatados em todos os critérios descritos acima, estes times são considerados campeões.

Os dados de entrada são o número de times e os placares dos jogos. Todos os times jogam entre si. Um time que ganha um jogo recebe três pontos e um time que perde não ganha pontos. Já quando os times empatam, eles recebem um ponto cada. Para o exemplo do arquivo que segue:

```
4                          /* quatro times */
Fla  Flu    3 0
Fla  Bota   2 0
Fla  Vasco  1 0
Flu  Bota   1 1
Flu  Vasco  1 2
Bota Vasco  0 0
```

O seu software deve informar que o Fla é o campeão.

Um exemplo de uso desse programa seria java Campeao arq.txt, onde arq.txt é o nome do arquivo de entrada que contém o número de times e os placares dos jogos.

Resposta:

```
import java.io.*;  class no{
  String nome;
  int pontos, num_vit, saldo;
  no prox;

  no (String n, int p, int nv, int s){
    nome = n;
    pontos = p;
    num_vit = nv;
    saldo = s;
    prox = null;
  }
};

class lista{
  no prim;

  lista(){
    prim = null;
  }

  void insere (String n, int pontos, int nv, int s){
    int r = busca(n);
    if (r == 1){
      no p = prim;
      while (!(n.equals(p.nome))) p = p.prox;
      p.pontos += pontos;
      p.num_vit += nv;
      p.saldo += s;
    }
    else{
      no q = new no(n, pontos, nv, s);
      q.prox = prim;
      prim = q;
    }
  }

  int busca (String n){
    no p = prim;
    while ((p != null) && (!(n.equals(p.nome)))) p = p.prox;
    if (p == null) return 0;
    return 1;
  }

  void calculaCampeao(){
    no maior = prim;
    no p = prim;

    while(p != null){
      if(maior.pontos < p.pontos) maior = p;
      else{
        if(maior.pontos == p.pontos){
          if(maior.num_vit < p.num_vit) maior = p;
```

```
          else{
            if(maior.num_vit == p.num_vit){
              if (maior.saldo < p.saldo) maior = p;
            }
          }
        }
      }
      p = p.prox;
    }

    p = prim;
    while (p != null){
      if((p.pontos == maior.pontos) && (maior.num_vit == p.num_vit) && (maior.saldo == p.saldo))
        System.out.println(p.nome);
      p = p.prox;
    }
  }
};

public class Campeao {
 public static void main(String[] args) throws IOException {
   BufferedReader in = new BufferedReader(new FileReader(args[0]));
   lista l = new lista();

   try{
     String s = in.readLine();
     int n = Integer.parseInt(s);
     int total = (n * (n - 1)) / 2;
     String vs[];
     int aux_g1, aux_g2;
     while (total > 0){
       total--;
       s = in.readLine();
       vs = s.split(" ");
       aux_g1 = Integer.parseInt(vs[2]);
       aux_g2 = Integer.parseInt(vs[3]);
       if (aux_g1 > aux_g2){
         l.insere(vs[0], 3, 1, (aux_g1 - aux_g2));
         l.insere(vs[1], 0, 0, (aux_g2 - aux_g1));
       }
       else if (aux_g2 > aux_g1){
         l.insere(vs[0], 0, 0, (aux_g1 - aux_g2));
         l.insere(vs[1], 3, 1, (aux_g2 - aux_g1));
       }
       else{
         l.insere(vs[0], 1, 0, 0);
         l.insere(vs[1], 1, 0, 0);
       }
     }
     in.close();
   }
   catch (Exception e){
     System.out.println(e);
   }
   finally{
     in.close();
   }
```

```
  l.calculaCampeao();
 }
}
```

Questão 2) (3 pontos)

Considere as classes abaixo que modelam um sistema simples para conta bancária, as quais foram utilizadas na AP3 do 1º. semestre deste ano.

```
class OperacaoIlegal extends Exception {
      enum TipoOperacao {saque, deposito};
      ContaCorrente cc;
      float valor;
      TipoOperacao operacao;
      public OperacaoIlegal(ContaCorrente contaCorrente, float valor,
TipoOperacao op) {
            this.cc = contaCorrente;
            this.valor = valor;
            this.operacao = op;
      }
}

class ContaCorrente {
      int conta;
      float saldo;
      public ContaCorrente(int pConta, float pSaldo) {
            this.conta = pConta; this.saldo = pSaldo;
      }
      public float obtemSaldo() {
            return saldo;
      }
      public void realizaDeposito(float valor) throws OperacaoIlegal {
            if (valor < 0)
                  throw new OperacaoIlegal(this, valor,
OperacaoIlegal.TipoOperacao.deposito);
            else
                  this.saldo = this.saldo + valor;
      }
      public void realizaSaque(float valor) throws OperacaoIlegal {
            if (valor >= this.saldo)
                  throw new OperacaoIlegal(this, valor,
OperacaoIlegal.TipoOperacao.saque);
            else
                  this.saldo = this.saldo - valor;
      }
}

public class AP3_2009_1_Q1 {
      public static void main(String[] args) {
            ContaCorrente c = new ContaCorrente(1, 500);
            try {
                  c.realizaSaque(1000);
                  c.realizaDeposito(3500);

            } catch (OperacaoIlegal e) {
                  System.out.println("Operação de " + e.operacao + " com
```

```
valor " + e.valor + " inválida para a conta " + e.cc.conta + "!");
            }
      }
}
```

Crie uma (ou mais) classe(s) que modele(m) uma transação (movimentação). Esta transação deve ser utilizada para exibição do histórico do cliente, o qual deve conter: a operação realizada, o valor movimentado e a hora em que a operação foi realizada. Para a manipulação da hora, pode ser utilizada a classe java.util.GregorianCalendar, cujo construtor padrão retorna a hora corrente. A classe ContaCorrente deve conter o método ***public void*** *exibeExtrato();*, o qual irá exibir todas as movimentações realizadas sobre a respectiva conta.

Resposta:

Nesta questão era necessário criar uma classe chamada Transacao (listada abaixo), além de alterarmos a classe ContaCorrente para criar novas transações a cada operação realizada (saque e depósito, neste caso). Além disso, a classe ContaCorrente também contém o método exibeExtrato(), o qual exibe todas as movimentações sobre uma determinada conta. A classe que modela exceções não foi alterada, então não está listada abaixo.

```
Import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.GregorianCalendar;
import java.util.List;

class Transacao {
      ContaCorrente conta;
      TipoOperacao operacao;
      float valor;
      GregorianCalendar hora;
      public Transacao(ContaCorrente c, TipoOperacao op, float v) {
            conta = c;
            operacao = op;
            valor = v;
            hora = new GregorianCalendar();
      }
      public void exibeTransacao () {
            System.out.println("***");
            System.out.println("Cliente: " + conta.numConta);
            System.out.println("Operacao: " + operacao);
            System.out.println("Valor: " + valor);
            SimpleDateFormat sdf = new SimpleDateFormat("hh:mm:ss -
dd.MM.yyyy");
            System.out.println("Hora: " + sdf.format(hora.getTime()));
      }
}

class ContaCorrente {
      int numConta;
      float saldo;
      List<Transacao> transacoes;
      public ContaCorrente2(int pConta, float pSaldo) {
            this.numConta = pConta; this.saldo = pSaldo;
            transacoes = new ArrayList<Transacao>();
```

```
	}
	public float consultaSaldo() {
		return this.saldo;
	}
	public void realizaDeposito(float valor) throws OperacaoIlegal {
		if (valor < 0)
			throw new OperacaoIlegal(this, valor,
TipoOperacao.deposito);
		else {
			this.saldo = this.saldo + valor;
			transacoes.add(new Transacao(this,
TipoOperacao.deposito, valor));
		}
	}
	public void realizaSaque(float valor) throws OperacaoIlegal2 {
		if (valor >= this.saldo)
			throw new OperacaoIlegal2(this, valor,
TipoOperacao.saque);
		else {
			this.saldo = this.saldo - valor;
			transacoes.add(new Transacao(this, TipoOperacao.saque,
valor));
		}
	}
	public void exibeExtrato() {
		for (Transacao trans : transacoes) {
			trans.exibeTransacao();
		}
	}
}

public class AP3_2009_2_Q2 {
	public static void main(String[] args) {
		ContaCorrente c1 = new ContaCorrente(1, 500);
		ContaCorrente c2 = new ContaCorrente(1, 500);
		try {
			c1.realizaDeposito(50);
			c1.realizaSaque(100);
			c1.exibeExtrato();
		} catch (OperacaoIlegal e) {
			System.out.println("Operação de " + e.operacao + " com
valor " + e.valor + " inválida para a conta " + e.cc.numConta + "!");
		}
	}
}
```

Questão 3) (3 pontos)

Suponha o código abaixo que desenha figuras geométricas básicas:

```
import java.awt.*;
import javax.swing.JFrame;

class Olimpiada {
	int pos_x, pos_y; // Posição base da figura
```

```
	int raio;
	public Olimpiada(int x, int y, int r) {
		pos_x = x; pos_y = y;
		raio = r;
	}
	public void desenha (Graphics g, JFrame frame) {
		Graphics2D g2d = (Graphics2D)g;
		int x = pos_x;
		int x2 = pos_x + (raio/4);
		int y = pos_y;
		g2d.drawOval(x, y, raio, raio);
		g2d.drawOval(x = x + (raio/2), y, raio, raio);
		g2d.drawOval(x = x + (raio/2), y, raio, raio);
		g2d.drawOval(x2, y = y + (raio/2), raio, raio);
		g2d.drawOval(x2 = x2 + (raio/2), y, raio, raio);
	}
}

class Janela extends JFrame {
	public Janela() {
		this.setTitle("Figuras Geométricas");
		this.setBounds(0, 0, 400, 400);
	}
	public void paint(Graphics g) { // Método sobrecarregado da JFrame
para desenho de um frame
		super.paint(g);
		Graphics2D g2d = (Graphics2D)g;
		g2d.drawLine(100, 100, 200, 200); // Figuras básicas - Linha
		g2d.drawOval(100, 150, 100, 50);  // Elipse
		g2d.drawRect(200, 150, 50, 100);  // Retângulo
	}
}

public class AP3_2009_2_Q3 {
	public static void main(String[] args) {
		(new Janela()).setVisible(true);
	}
}
```

Altere o código para que o frame passe a desenhar (método Janela.paint()) à partir de uma lista de objetos. Imagine que novas classes que representam figuras, como a classe Olimpíada (esta classe cria instâncias com o formato do símbolo olímpico no *frame*), poderão ser criadas.

Resposta:

No gabarito abaixo foi criado a classe abstrata Objeto, a qual contém as características mínimas e essenciais de um objeto a ser desenhado: uma coordenada, um construtor para inicialização e um método abstrato chamado desenha(), a ser implementado pelas classes de objeto concretas. Na classe Janela foi criado um objeto do tipo lista (figuras), o qual conterá os objetos a serem desenhados. Esta lista armazena objetos do tipo Objeto (`List<Objeto> figuras;`), ou seja, somente objetos deste tipo podem ser adicionados a lista. Assim, obrigamos que qualquer objeto que precise ser desenhado tenha que oferecer uma implementação do método desenha(). Somente para efeitos de ilustração, criamos uma classe chamada Audi, a qual desenhará os símbolos similares ao desta marca de automóveis.

```
abstract class Objeto {
	int pos_x, pos_y;
	public Objeto(int x, int y) {
		pos_x = x;
		pos_y = y;
	}
	abstract public void desenha (Graphics g, JFrame frame);
}

class Olimpiada extends Objeto {
	int raio;
	public Olimpiada(int x, int y, int r) {
		super(x, y);
		raio = r;
	}
	public void desenha (Graphics g, JFrame frame) {
		Graphics2D g2d = (Graphics2D)g;
		int x = pos_x;
		int x2 = pos_x + (raio/4);
		int y = pos_y;
	  g2d.drawOval(x, y, raio, raio);
	  g2d.drawOval(x = x + (raio/2), y, raio, raio);
	  g2d.drawOval(x = x + (raio/2), y, raio, raio);
	  g2d.drawOval(x2, y = y + (raio/2), raio, raio);
	  g2d.drawOval(x2 = x2 + (raio/2), y, raio, raio);
	}
}

class Audi extends Objeto {
	int raio;
	public Audi(int x, int y, int r) {
		super(x, y);
		raio = r;
	}
	public void desenha (Graphics g, JFrame frame) {
		Graphics2D g2d = (Graphics2D)g;
		int x = pos_x;
		int y = pos_y;
	  g2d.drawOval(x, y, raio, raio);
	  g2d.drawOval(x = x + (raio/2), y, raio, raio);
	  g2d.drawOval(x = x + (raio/2), y, raio, raio);
	  g2d.drawOval(x = x + (raio/2), y, raio, raio);
	}
}

class Janela extends JFrame {
	List<Objeto> figuras;
	public Janela(List l) {
		// Atribui título da janela
		this.setTitle("Figuras Geométricas");
		// Define tamanho padrão da janela
		this.setBounds(0, 0, 800, 600);
		// Termina o processo no fechamando da janela
		this.setDefaultCloseOperation(EXIT_ON_CLOSE);
		figuras = l;
	}
	public void paint(Graphics g) {
```

```
            super.paint(g);

        // Desenho dos objetos do jogo
        for (Objeto figura : figuras)
            figura.desenha(g, this);
      }
}

public class AP3_2009_2_Q3 {
      public static void main(String[] args) {
            List<Objeto> figuras = new ArrayList<Objeto>();
            figuras.add(new Olimpiada(300, 150, 50));
            figuras.add(new Audi(200, 200, 30));
            figuras.add(new Audi(300, 300, 30));
            (new Janela(figuras)).setVisible(true);
      }
}
```