Fundação CECIERJ - Vice Presidência de Educação Superior a Distância

Curso de Tecnologia em Sistemas de Computação
Disciplina: Programação III
AP2 1° semestre de 2008.

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) (2.5 pontos)
Escreva um programa que, ao receber como parâmetro de entrada um número de 1 a 9 (você deve testar se o número informado segue este intervalo) e o nome de um arquivo de saída, crie uma pirâmide de números neste arquivo de saída. Por exemplo, se o número passado for **três** e o nome do arquivo for **teste.txt**, será escrito em **teste.txt** as seguintes seqüências, uma por linha:

```
1
121
12321
```

RESPOSTA:

```
import java.io.*;

public class Piramide{

  public static void main (String[] args) throws IOException{
    int num = Integer.parseInt(args[0]);

    if ((num >= 1) && (num <= 9)){
      BufferedWriter out = new BufferedWriter(new FileWriter(args[1]));
      String s;
```

```
      try{
        for (int k = 1; k <= num; k++){
          s = "";

          for (int i = 1; i <= k; i++)
            s += i;

          for (int j = k - 1; j >= 1; j--)
            s += j;

          out.write(s +"\n");
        }
      }

      catch (Exception e){
        System.out.println("Excecao\n");
      }

      finally{
        out.close();
      }
    }
  }
}
```

Questão 2) (2.5 pontos)
Considere o seguinte trecho de código em JAVA.

```
class InfracaoTransito extends Exception {}

class ExcessoVelocidade extends InfracaoTransito {}

class AltaVelocidade extends ExcessoVelocidade {}

class Acidente extends Exception {}

class Defeito extends Exception {}

abstract class Dirigir {
  Dirigir() throws InfracaoTransito {}
  void irTrabalhar () throws InfracaoTransito {}
  abstract void viajar() throws ExcessoVelocidade, Defeito;
  void caminhar() {}
}

public class DirecaoPerigosa extends Dirigir {
  DirecaoPerigosa() throws Acidente {}
  void caminhar() throws AltaVelocidade {}
  public void irTrabalhar() {}
  void viajar() throws AltaVelocidade {}
  public static void main(String[] args) {
    try {
          DirecaoPerigosa dp = new DirecaoPerigosa ();
          dp.viajar ();
    } catch(AltaVelocidade e) {
```

```
    } catch(Acidente e) {
    } catch(InfracaoTransito e) {
   }
    try {
          Dirigir d = new DirecaoPerigosa();
          d.viajar ();
    } catch(Defeito e) {
    } catch(ExcessoVelocidade e) {
    } catch(Acidente e) {
    } catch(InfracaoTransito e) {}
  }
}
```

O trecho de código acima apresenta dois erros identificáveis em tempo de compilação. Que erros são esses? Justifique sua resposta.

JAVA estabelece a seguinte regra para garantir o uso apropriado do mecanismo de exceções: os construtores devem necessariamente propagar as exceções declaradas no construtor da superclasse. Se o construtor da superclasse pode propagar exceções, o da subclasse também deverá propagá-las pois o último necessariamente chama o primeiro. Assim, o primeiro erro é que o construtor de *DirecaoPerigosa* não propaga a exceção *InfracaoTransito*, quando deveria fazê-lo, pois *DirecaoPerigosa* é subclasse de *Dirigir* e nesta classe o construtor propaga a exceção *InfracaoTransito*.

O outro erro identificável em tempo de compilação é a implementação do método *caminhar* na classe *DirecaoPerigosa*, pois sua implementação dispara a exceção *AltaVelocidade*, que não está listada na especificação desse método na superclasse *Dirigir*. O compilador de JAVA impede que isso possa ser feito porque no caso de se chamar o método *caminhar* de *DirecaoPerigosa* através de uma referência a superclasse *Dirigir*, isso poderia ocasionar o disparo da exceção *AltaVelocidade*, sem que ela fosse devidamente tratada.

Questão 3) (2.5 pontos)

Defina um Dicionário através de uma interface Java. Um dicionário, também chamado de tabela associativa, é um TAD (Tipo Abstrato de Dados) que permite a armazenagem de valores associados a chaves (você pode encarar um dicionário como um array que é indexado por chaves ao invés de números). Projete seu dicionário para usar objetos quaisquer como chaves e valores.
Note que a forma de se obter um valor do dicionário é fazer uma consulta através da chave que foi usada para armazenar esse valor. Além dessa consulta normal, por chave, um dicionário deve também permitir que todos os pares (chave/valor) existentes sejam obtidos, um a um. Dessa forma é possível descobrir todas as informações contidas em um dicionário, mesmo sem conhecer as chaves. Definam classes e interfaces para que o programa abaixo funcione.

```
public class Teste {
	public static void main(String[] args) {
		InterfaceDicionario d = new Dicionario();
		d.insere("João", "512-1313");
		d.insere("Maria", "512-2299");
		System.out.println("O telefone de João é:
"+d.consulta("João"));
		System.out.println("O telefone de Maria é:
"+d.consulta("Maria"));
	}
}
```

RESPOSTA:

```
import java.util.HashMap;
import java.util.Map;

interface InterfaceDicionario {
	void insere(String string, String string2);
	String consulta(String string);
}

class Dicionario implements InterfaceDicionario {
	Map<String, String> dicionario = new HashMap<String, String>();
	public String consulta(String string) {
		return (String)dicionario.get(string);
	}
	public void insere(String string, String string2) {
		dicionario.put(string, string2);
	}
}

public class AP2_2008_1_Q3 {
	public static void main(String[] args) {
		InterfaceDicionario d = new Dicionario
d.insere("João", "512-1313");
		d.insere("Maria", "512-2299");
		System.out.println("O telefone de João é:
"+d.consulta("João"));
		System.out.println("O telefone de Maria é:
"+d.consulta("Maria"));
	}
}
```

Questão 4) (2.5 pontos)

Escreva um programa que exiba apenas uma janela com 1 botão. Clicando neste botão, a cor de fundo da janela é alterada segundo este ciclo (vermelho → verde → azul → vermelho), um clique por vez.

RESPOSTA:

```
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import java.util.List;
import javax.swing.*;

public class AP2_2008_1_Q4 extends JFrame implements ActionListener {
	JPanel panel;
	JButton botao;
	List<Color> cores;
	Iterator<Color> it_cores;

	public AP2_2008_1_Q4() {
		// Define configurações gerais da janela principal
		this.setBounds(0, 0, 800, 600);
		this.setDefaultCloseOperation(EXIT_ON_CLOSE);

		// Cria os objetos gráficos
		panel = new JPanel();
		botao = new JButton("Troca Cor");

		// Configura esta própria classe para tratar os eventos do
botão
		botao.addActionListener(this);
		panel.add(botao);

		// Cria coleção que irá armazenar as cores desejadas
		cores = new ArrayList<Color>();
		cores.add(Color.RED);
		cores.add(Color.GREEN);
		cores.add(Color.BLUE);

		// Inicializa iterador que será utilizado para percorrer as
cores
		it_cores = cores.iterator();
		panel.setBackground((Color)it_cores.next());
		this.add(panel);
	}

	public void actionPerformed(ActionEvent e) {
		// Trat os cliques no botão
		if (e.getSource() == botao) {
			if (!it_cores.hasNext())
				it_cores = cores.iterator();
			panel.setBackground((Color)it_cores.next());

		}
	}

	public static void main(String[] args) {
		AP2_2008_1_Q4 janela = new AP2_2008_1_Q4();
		janela.setVisible(true);
	}
}
```