

Bacharelado em Ciência da Computação
MAC0242 - Laboratório de programação
Segunda prova – 12/11/2011

Nome : Pedro Paulo Vezzà Campos_____

Assinatura : Pedro Paulo Vezzà Campos_____

Questão	Valor	Nota
Q1	1.5	
Q2	2.0	
Q3	1.5	
Q4	3.0	
Q5	2.0	
total	10.0	

1. A prova é **individual** e pode ser feita a lápis.
2. É permitido a consulta a livros, apontamentos ou Internet.
3. Não é necessário apagar rascunhos no caderno de questões.

Boa prova !

Q1.(1.5) A questão refere-se às classes abaixo:

```
abstract class Cidadao {
    public abstract String getRG() ;

    public void print() {
        System.out.println("Impressão do cidadão") ;
    }
}

class ZéMané extends Cidadao {
    public String nome = "José Manoel" ;
}

class TestaZé {
    public static void main(String[] args) {
        ZéMané ze = new ZéMané();
        System.out.println("O nome do Zé é: "+ze.nome) ;
    }
}
```

- Q1a.(0.5)** Existe um problema relacionado à herança com a classe ZéMané. Diga qual é e apresente uma possível correção.
- Q1b.(1.0)** Quais os possíveis motivos para se utilizar herança? Qual é o motivo mais apropriado para descrever a herança a partir de classes abstratas e interfaces? Explique como este tipo de herança funciona.
-

Q1a. O problema relacionado à classe ZéMané é que ela é concreta mas não implementa o método abstrato `getRG()` da classe mãe `Cidadao`, o que é inválido segundo a herança de objetos. Implementar o método na classe ZéMané resolve o problema. Uma possível implementação é a seguinte:

```
class ZéMané extends Cidadao {
    public String nome = "José Manoel";

    public String getRG(){
        return "123456-X SP";
    }
}
```

Q1b. Retirado da apostila FJ-11 da Caelum:

“Note que o uso de herança aumenta o acoplamento entre as classes, isto é, o quanto uma classe depende de outra. A relação entre classe mãe e filha é muito forte e isso acaba fazendo com que o programador das classes filhas tenha que conhecer a implementação da classe pai e vice-versa - fica difícil fazer uma mudança pontual no sistema.”

Q2.(2.0) O sistema ao qual trata esta questão foi retirado da primeira fase de um projeto que foi entregue por alunos (veja a figura 1):


diagramaFeio3.png

Figura 1: Números arbitrariamente grandes.

- Na implementação destes alunos, os números inteiros arbitrariamente compridos foram representados como listas duplamente ligadas.
- A classe `ListaLigada` é uma lista duplamente ligada. Seus nós são elementos da classe `NóLista`, mostrada no diagrama.
- As classes `Subtração`, `Multiplicação` e `Soma` servem para aplicar operações às listas ligadas que representam os números. Os métodos destas classes recebem como argumentos os nós iniciais das listas ligadas sobre as quais deve ser realizada a operação, e retorna o nó inicial de uma lista ligada que contém o resultado.
- A classe `Principal`, no projeto em questão, fazia a leitura dos arquivos de entrada, construía listas ligadas e aplicava as operações a elas. Esta classe depende de todas as demais classes, e a relação de dependência foi omitida do diagrama para deixá-lo mais limpo.

Q2a.(0.5) Aponte dois defeitos deste projeto. Você deve levar em conta princípios como abstração de dados, modularidade e encapsulamento, assinaturas dos métodos, etc.

Q2b.(1.5) Proponha correções para os defeitos que você mencionou acima, desenhando um diagrama UML do sistema após a correção E o esqueleto das classes do sistema de acordo com sua proposta.

-
- Q2a.**
- * Violação do encapsulamento das classes `ListaLigada` e `NóLista` graças ao uso de atributos públicos.
 - * Forte acoplamento entre as classes do projeto com a `ListaLigada` e `NóLista`.
 - * `Principal` é pouco coesa já que ela possui múltiplas atribuições como leitura de dados, construção das listas e aplicação de operações.
 - * `ListaLigada` não utiliza tipos de dados genéricos sendo restrita a armazenar inteiros.
 - * Métodos de `Subtração`, `Soma` e `Multiplicação` deveriam receber como parâmetros e retornar instâncias de uma classe, por exemplo `NumeroGrande`, que abstraia a implementação de números arbitrariamente grandes.

Q2b.

```
public class NumeroGrande {
    private ListaLigada<Integer> num;

    public NumeroGrande(String numero) {
        construirLista(numero);
    }

    private void construirLista(String numero) {

    }

    public NumeroGrande somar(NumeroGrande x) {

    }

    public NumeroGrande subtrair(NumeroGrande x) {
```

```

    }

    public NumeroGrande multiplicar(NumeroGrande x) {

    }

}

public class InterfaceUsuario {
    public void lerDados() {

    }

    public void invocarOperacao(){

    }

    public void exhibirResultado(NumeroGrande num) {

    }
}

public class ListaLigada<T> {
    private NóLista<T> primeiro;

    public ListaLigada() {
    }

    public void inserir(T elemento) {

    }

    public T obter(int posicao) {

    }

    public void remover(int posicao) {

    }
}

public class NóLista<T> {
    private T conteúdo;
    private NóLista<T> próximo;
    private NóLista<T> anterior;

    public NóLista(T conteúdo) {
        this.conteúdo = conteúdo;
    }

    public T getConteúdo() {
        return conteúdo;
    }
}

```

```
    public void setConteúdo(T conteúdo) {
        this.conteúdo = conteúdo;
    }

    public NóLista<T> getPróximo() {
        return próximo;
    }

    public void setPróximo(NóLista<T> próximo) {
        this.próximo = próximo;
    }

    public NóLista<T> getAnterior() {
        return anterior;
    }

    public void setAnterior(NóLista<T> anterior) {
        this.anterior = anterior;
    }
}
```

Q3.(1.5) Modele a seguinte aplicação usando a técnica dos substantivos e verbos e os cartões CRC (especifique três cenários e desenhe os cartões na sua prova).

O José desenvolve e instala sistemas de monitoramento (para segurança) que consistem de zonas, sensores e uma estação centralizada de monitoramento (veja figura 2). Uma zona é uma coleção de sensores, tipicamente localizados dentro de uma sala, andar, linha de montagem ou prédio. Um sensor é um dispositivo de aquisição de dados que mede uma dada variável de sistema e provê um sinal de saída para outros dispositivos/sistemas lerem. Por exemplo, diferentes tipos de sensores podem detectar temperatura, pressão, nível de fumaça, ou movimento.

Figura 2: Sistema de monitoramento do José

Q4.(3.0) Responda às questões conceituais abaixo.

Q4a.(1.0) Explique o que são "inner classes", qual a razão de usá-las e dê um exemplo.

Q4b.(1.0) Quais são as principais diferenças entre C# e Java? Dê um exemplo de código de uma das diferenças citadas.

Q4c.(1.0) Explique como funciona o tratamento de exceções em Java. Por que não se deve usar no caso do código abaixo?

```
...
try {
    for (i=0, i < N, i++)
        a[i++].f() ;
}
catch(ArrayIndexOutOfBoundsException e) {
    ...
}
```

-
- Q4a.** * Adaptado de <http://docs.oracle.com/javase/tutorial/java/java00/nested.html>
Em Java há o conceito de "Nested class", uma classe definida dentro de uma outra classe. Inner classes são uma versão não estática de uma nested class, logo, inner classes tem acesso aos atributos do objeto que a contem, mesmo os privados.
- * Retirado de http://www.de9.ime.eb.br/~madeira/teaching/labprog2/classes_internas.pdf:

"As classes internas são úteis por sua relação íntima com a classe externa. Por exemplo, se você precisa de um objeto que será usado apenas pela classe que você está escrevendo e por mais nenhuma outra, você não precisa escrever uma nova classe dentro do seu modelo de framework, basta tornar este código uma classe interna. Isto irá permitir que a classe interna tenha acesso a todas as variáveis e métodos (inclusive os privados) da classe que a contém, ou seja, a classe externa."

- * Exemplo adaptado de <http://www.javaworld.com/javaworld/javaqa/2000-03/02-qa-innerclass.html?page=2>:

```
public class InterfaceUsuario extends JFrame {
    //Inner class
    class TratadorBotao implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            // Faz algo
        }
    }

    private void buildGUI() {
        JButton botao = new JButton();
        botao.addActionListener(new TratadorBotao());
    }
}
```

Q4b. Retirado de <http://stackoverflow.com/questions/295224/what-are-major-differences-between-c-sharp-and-java>

- * Java roda em quase qualquer plataforma através da JRE, C# é uma linguagem de propriedade da Microsoft e depende de esforços externos (Projeto Mono) para rodar em outras plataformas;
- * C# não possui exceções checked;

- * Java não possui sobrecarga de operadores;
- * C# não possui inner classes, apenas nested classes estáticas;
- * Java não possui properties como parte da linguagem, apenas uma convenção de métodos get/set/is;
- * Java não possui comandos de pré-processamento (#define, #if etc do C#);
- * C# não possui inner classes anônimas.
- * Java não possui structs

Exemplo de código retirado de http://www.harding.edu/fmccown/java_csharp_comparison.html para o fato de Java não possuir properties:

Java	C#
<pre>private int mSize; public int getSize() { return mSize; } public void setSize(int value) { if (value < 0) mSize = 0; else mSize = value; } int s = shoe.getSize(); shoe.setSize(s+1);</pre>	<pre>private int mSize; public int Size { get { return mSize; } set { if (value < 0) mSize = 0; else mSize = value; } } shoe.Size++;</pre>

Q4c. Retirado da apostila FJ-11 da Caelum: O sistema de exceções do Java funciona da seguinte maneira: quando uma exceção é lançada (throws), a JVM entra em estado de alerta e vai ver se o método atual toma alguma precaução ao tentar executar esse trecho de código. Como podemos ver, o metodo2 não toma nenhuma medida diferente do que vimos até agora. Como o metodo2 não está tratando esse problema, a JVM pára a execução dele anormalmente, sem esperar ele terminar, e volta um stackframe pra baixo, onde será feita nova verificação: o metodo1 está se precavendo de um problema chamado `ArrayIndexOutOfBoundsException`? Não... volta para o main, onde também não há proteção, então a JVM morre (na verdade, quem morre é apenas a Thread corrente, veremos mais para frente).

Q5.(2.0) Resolva a questão 5 da segunda lista de exercícios (que está no paca, em exercícios preparatórios para a prova).

Q5a. A ideia principal é tornar o ato de ser informado de algum acontecimento um processo assíncrono, ou seja, evitar a necessidade de verificar periodicamente (polling) o estado atual de um programa. Num programa baseado em eventos, em geral temos um objeto que é o gerador dos eventos, e outros objetos que se registram junto a ele, interessados naquele evento.

Q5b. Código retirado das notas de aula:

```
import java.awt.event.*;
class EscutaBotao implements ActionListener {
    public void actionPerformed(ActionEvent a){
        System.out.println("Botão apertado");
    }
}

public Janela() {
    EscutaBotao eb = new EscutaBotao();
    botao.addActionListener(eb);
    try {
        jbInit();
    } catch (Exception e) {
        e.printStackTrace(e);
    }
}
```

Q5c.