Laboratório de Programação 2
2015.2

Lista de exercícios

2 de Dezembro de 2015

Exercícios de Tratamento de Exceções

Exercício 1

O programa abaixo lança uma exceção aritmética. Explique o erro e corrija o programa.

```
1 class Teste {
2   public static void main(String[] args) {
3     int a = 10;
4     int b = 0;
5     int c = 0;
6
7     c = a/b;
8     System.out.println("divisao:" + c);
9   }
10 }
```

Exercício 2

Escreva um programa que lê do usuário o raio de um círculo e imprime o diâmetro do círculo, circunferência e área, utilizando o valor de ponto flutuante para PI. Ao final, o programa deverá perguntar se o usuário deseja calcular os dados novamente ou sair do programa.

- diâmetro = 2r
- circunferência = $2\pi r$
- área = πr^2

Utilize os blocos *try, catch e finally* para lidar com os casos em que o usuário informe valores inapropriados (!%*$x@, etc). **Dica**: Utilize a classe InputMismatchException e a constante *Math.PI*.

Exercício 3

O programa abaixo é abortado devido a um erro durante sua execução. O que poderia ter sido feito para evitar que o programa fosse abortado?

```
1 import java.util.*;
2 class Teste {
3   public static void main(String[] args) {
4     int a[100];
5     Random rand = new Random();
6
7     a[0] = 0;
8     for(int i = 1 ; i <= 100 ; i++)
9       a[i] = rand.nextInt(100);
10  }
11 }
```

Exercício 4

A classe abaixo especifica um novo tipo de exceção "MinhaExcecao", que é uma especialização da classe RuntimeException.

```
1 class MinhaExcecao extends RuntimeException{
2   private int id = 0;
3   public MinhaExcecao(int i) {
4     super("Valor invalido");
5     id = i;
6   }
7   public int getId() {
8     return id;
9   }
10 }
```

Explique o que acontece em um objeto Conta quando o usuário invocar o método "saca", segundo a especificação da classe Conta abaixo.

```
1 class Conta {
2   private double saldo;
3   private Cliente cliente;
4
5   public Conta(Cliente cliente) {
6     this.cliente = cliente;
7     this.saldo = 0;
8   }
9
10  public boolean deposita(double valor)  {
11    this.saldo += valor;
12    return true;
13  }
14
15  public void saca(double valor) {
16    if(valor > saldo || valor < 0)
17      throw new MinhaExcecao(1);
18
19    this.saldo -= valor;
20  }
21
22  public double getSaldo() {
23    return saldo;
24  }
25 }
```

Exercício 5

Explique o que é o "NullPointerException".

Componentes GUI

Exercício 1

Escreva a classe *Criptografia*, que conterá alguns métodos estáticos para codificação e decodificação de strings. Escreva nessa classe o método *codificaRot13*, que receberá uma string como argumento e retornará uma string codificada com o algoritmo *rot13*, que substitui cada caracter da string pelo valor do caracter mais treze, subtraindo vinte e seis caso o resultado seja maior que a última letra, de forma que "abCde" seja substituída por "noPqr", "kLmnoPq" seja substituída por "xYzabCd", e "UVWxyz" seja substituída por "HIJklm". Somente os caracteres alfabéticos não-acentuados devem ser modificados. Por exemplo, se a string "Revolução de 1930" for passada como argumento para esse método, ele retornará "Eribyhçãb qr 1930". Uma característica interessante do algoritmo *rot13* é que, se uma string codificada por ele for passada de novo pelo próprio algoritmo, a string original será retornada. Escreva também um método *decodificaRot13* que seja somente uma chamada para o método *codificaRot13*. Para este exercício, crie uma GUI com dois JTextArea, e os respectivos JButton para converter o texto de um JTextArea e apresentar sua codificação no outro JTextArea, e vice-versa.

Exercício 2

Para que servem os Layouts (ex: FlowLayout, GridLayout, etc) em Java?

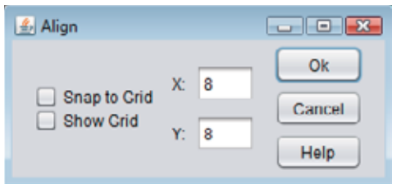
Exercício 3

Explique detalhadamente o que o programa abaixo faz.

```
1 import java.awt.*;
2 import javax.swing.*;
3
4 class LabelFrame extends JFrame {
5   private JLabel label1;
6   private JLabel label2;
7   private JLabel label3;
8
9   public LabelFrame() {
10    super("Testando a classe JLabel");
11    setLayout(new FloatLayout() );
12
13    label1 = new JLabel("Label com texto");
14    label1.setToolTipText("Este é o label1");
15    add(label1);
16
17    Icon bug = new ImageIcon("bug1.png");
18    label2 = new JLabel("Label com texto e ícone",bug,SwingConstants.LEFT);
19    label2.setToolTipText("Este é o label2");
20    add(label2);
21  }
22
23  public static void main(String args[])
24  {
25    LabelFrame labelframe = new LabelFrame();
26    labelframe.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
27    labelframe.setSize(260,180);
28    labelframe.setVisible(true);
29  }
30 }
```

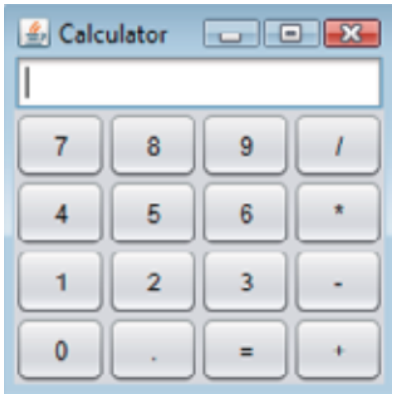
Exercício 4

Criea a interface gráfica abaixo. Não é preciso implementar nenhuma funcionalidade.



Exercício 5

Criea a interface gráfica abaixo. Não é preciso implementar nenhuma funcionalidade.



Exercício 6

Implemente um programa para conversão de temperaturas Fahrenheit (F) para Celsius (C). A temperatura em Fahrenheit deve ser fornecida pelo teclado (usando um JTextField). Um JLabel deverá ser usado para apresentar a temperatura convertida. Utilize a fórmula $C = \frac{5}{9}(F - 32)$.

Exercício 7

Acrescente a escala Kelvin ao programa do exercício 6. O novo programa deverá permitir que o usuário escolha o tipo de conversão que deseja calcular. Utilize também a fórmula $K = C + 273.15$.

Exercício 8

Escreva o programa de um jogo de adivinhação de números. O programa deve escolher o número a ser adivinhado através da class Random (gerando um número aleatório entre 1 e 1000). O programa deve apresentar em um JLabel a seguinte frase:

> *Eu tenho um número entre 1 e 1000. Você consegue adivinhar qual? Por favor, informe o seu palpite.*

Um JTextField deve ser usado para entrada do palpite. A cada palpite, a cor de fundo deve mudar entre azul e vermelho. Vermelho indica "quente", ou seja, está perto de acertar. Azul indica "frio", o oposto. Um JLabel deve também indicar "Muito alto" ou "Muito baixo" para ajudar o usuário. Quando o usuário acertar, o JLabel deve apresentar o texto "Correto!", e o JTextField utilizado para entrada dos palpites deve ser desabilitado. Um JButton deve ser usado para permitir que o usuário jogue novamente. Quando o JButton for clicado, um novo número aleatório deve ser gerado e o JTextField de entrada deve ser habilitado novamente.

Exercício 9

Implemente as funcionalidades da calculadora cuja interface gráfica foi implementada no exercício 5.

Exercício 10

Explique por que interfaces não podem ter construtores.

Exercício 11

Identifique e explique o(s) erro(s) na classe abaixo.

```
1 public interface Resetavel{
2   public void reseta();
3 }
4
5 public interface Modificavel
6 {
7   public void reseta(int origem);
8   public void modifica(int tamanho);
9 }
10
11 public class Contador implements Resetavel, Modificavel
12 {
13   int valor;
14
15   public void reseta()
16   {
17     valor = 1;
18   }
19
20   public void modifica(int tam)
21   {
22     valor = tam;
23   }
24 }
```

Arquivos

Exercício 1

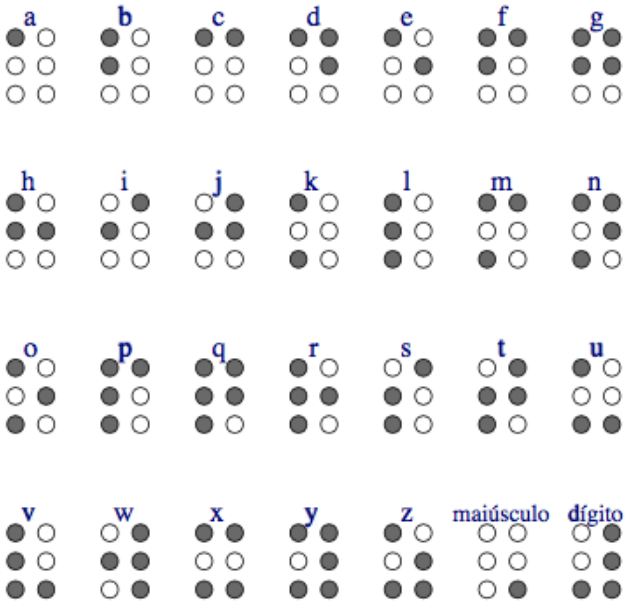
Escreva um programa que leia um arquivo texto qualquer. Ao final da leitura, o programa deverá apresentar a quantidade de caracteres, palavras, espaços, exclamações, interrogações presentes no arquivo.

Exercício 2

Escreva um programa que leia um arquivo texto qualquer em busca de uma palavra informada pelo usuário. O programa deverá apresentar o número de ocorrência destas palavras, além da posição de cada uma delas (linha e coluna).

Exercício 3

Escreva a classe *StringBraille* em Java. Essa classe deve representar internamente uma string e ser capaz de imprimi-la no alfabeto Braille, devendo ter ao menos o construtor (que recebe uma string a ser encapsulada, como argumento) e o método *toString* que imprimirá a string encapsulado em Braille.
A figura abaixo mostra o alfabeto Braille simplificado, onde um círculo preenchido significa uma marca em relevo no papel. Cada letra maiúscula que aparecer no texto deve ser precedida pelo caracter maiúsculo do alfabeto Braille. No caso dos dígitos, os caracteres Braille correspondentes a 'a', 'b', 'c', ..., 'i', 'j' são usados para representar os dígitos '1', '2', '3', ..., '9', '0'. Para simplificar considere que as strings a serem convertidas não contêm acentos nem símbolos, e que um espaço em Braille pode ser representado por um caracter em Braille sem nenhuma marca em relevo.



A saída do programa pode ser feita usando os caracteres de texto 'X' para representar uma marca em relevo, '.' para representar uma posição onde não há marca em relevo, e o espaço para separar uma letra do alfabeto Braille de outra. Assim, se a string "Java 123" for entrada, a saída deve ser:

```
.. .X X. X. X. .. .X X. .X X. .X XX
.. XX .. X. .. .. .X .. .X X. .X ..
.X .. .. XX .. .. XX .. XX .. XX ..
```

Crie um programa usando a classe *StringBraille* para ler um arquivo texto, convertê-lo para Braille e gravá-lo em um arquivo, segundo o formato acima.