#Principal#

package Prova;

import javax.swing.JOptionPane;

public class Principal {

public static void main (String [ ] args){

Teste teste = new Teste();

JOptionPane.showMessageDialog(null, teste.calcular(

Integer.parseInt (JOptionPane.showInputDialog("Informe n1 inteiro: ")),

Double.parseDouble (JOptionPane.showInputDialog("Informe n2 double: ")),

(JOptionPane.showInputDialog("Informe o nome: "))

));

}//Fecha main

}// Fecha chave

#Teste#

package Prova;

public class Teste{

private int n1;

private double n2;

private String resposta, nome;

public String calcular (int n1, double n2, String nome){

this.n1 = n1;

this.n2 = n2;

this.nome = nome;

this.resposta = ("n1: " + this.n1 + "\nn2: " + this.n2 + "\nnome: " + this.nome);

return this.resposta;

}

}// Fecha chave

**Conjunto de Letras**  
─ JVM ─ JRE  
─ API ─ JDK

**JVM (Java Virtual Machine)**  
– Significa Máquina Virtual Java  
– Software que emula uma CPU e Memória para a execução de programas Java

**API (Application Programming Interface)**  
– Significa Interface de Programação de Aplicativos  
– Biblioteca (ou uma série delas) com funções e procedimentos públicos que permitem aos programadores desenvolverem aplicações fazendo uso de recursos já definidos.

**JRE (Java Runtime Environment)**  
– Significa Ambiente de Tempo de Execução  
– É um pacote de softwares, que é executado como um aplicativo do sistema operacional e que interpreta a execução de programas Java  
– A JRE é composta pela JVM somada ao conjunto de API’s. (JVM + API’s = JRE)

**JDK (Java Development Kit) ou SDK**  
(Software Development Kit)  
– Significa Kit de Desenvolvimento Java  
– Conjunto de ferramentas para a compilação, documentação e debug de aplicativos Java.  
– Composto pela JRE somada as ferramentas de desenvolvimento

**Características**

• **Portabilidade**  
– Uma mesma aplicação pode ser executada em diferentes plataformas (hardware e software) sem a necessidade de adaptação de código.  
• **Multithreading**  
– Possibilidade de execução de diferentes processos simultaneamente

• **Suporte a Comunicação**  
– Oferece um conjunto de classes para desenvolvimentos de aplicações rodando em rede.  
**• Orientação a Objetos**– Técnica de programação que modela componentes de softwares em termos de objetos do mundo real

**Classes**  
• Uma Classe, em linguagem Orientada a Objetos, **é a possibilidade de combinar num único registro, campos de dados e campos que são métodos para operar os campos de dados**.

• Em outras palavras, as classes são os blocos de  
construções mais importantes de qualquer sistema  
orientado a objetos. Uma classe **é uma descrição de um conjunto de objetos que compartilham os mesmos atributos, operações, relacionamentos e semântica**.

**Classes (cont)**  
• Todos os objetos são instâncias de classes, onde a classe descreve as propriedades e comportamentos daquele objeto.  
• Em UML as classes são representadas por um retângulo dividido em três compartimentos:  
**- Nome:** que conterá apenas o nome da classe modelada;  
**- Atributos:** que possuirá a relação de atributos que a classe possui em sua estrutura interna;  
**- Métodos:** que serão os métodos de manipulação de dados e de comunicação de uma classe com outras do sistema.  
A sintaxe usada em cada um destes compartimentos é independente de qualquer linguagem de programação

**Objetos**  
• Os objetos são elementos que podemos manipular, acompanhar seu comportamento, criar, interagir com ele, ou até destruí-lo. Corresponde a qualquer coisa que tenha algum significado para uma dada aplicação.  
• Um objeto é simplesmente alguma coisa que faz sentido no contexto de uma aplicação, é uma entidade independente, assíncrona e concorrente, armazena dados, encapsula serviços, troca mensagens

**Tipos primitivos**  
- String - int  
- float - boolean  
- double - date

**Conversão de tipos**  
- String **→ float**  
Float.parseFloat(“12.90”);  
- String → **double**  
Double.parseDouble(“12.90”);  
- String → **int**  
Integer.parseInt(“12”);  
- float, double e int → **String**  
String.valueOf(12.90);

**Lógicos**  
&& ||  
**Relacionais**  
>= ==  
> !=  
<= equals(“”)   
< equalsIgnoreCase(“”)