LTP 1 – Java Básico

Aula 5

Tópicos

1 - Solução dos exercícios da aula anterior

a) Crie uma classe com estes métodos:

- **um método que recebe o comprimento de três pedaços de madeira, verifica se os mesmos podem formar um triângulo e retorna 0 em caso afirmativo e 1 em caso negativo;**
- **um outro método que verifica e informa na tela qual o tipo de triângulo que pode ser formado.**

Crie uma segunda classe, com o método main, que fará a leitura do comprimento de três pedaços de madeira e chama o método que verifica se os mesmos podem formar um triângulo ou não. Se puder formar um triângulo, esse método deverá chamar o outro que verifica que tipo de triângulo pode ser formado, caso contrário deverá informar a seguinte frase "não pode formar um triângulo". Use os métodos printf ou println da classe System e os métodos da classe Scanner.

```
import java.util.*;
public class MinhaClasse {
   public static void main(String[] args) {
      Scanner aux = new Scanner(System.in);
      ValidaTriangulo var = new ValidaTriangulo();
      int a,b,c,valor;
      System.out.println("Entre com o comprimento dos 3 pedaços de madeira");
      a=aux.nextInt();
      b=aux.nextInt();
      c=aux.nextInt();
      valor=var.formaTriangulo(a,b,c);
      if(valor==0)
         var.tipoTriangulo(a,b,c);
      else
         System.out.println("Não forma um triângulo");
   }
}
```

```
public class ValidaTriangulo {
   public int formaTriangulo(int ladoA, int ladoB, int ladoC){
      if(ladoA<ladoB+ladoC && ladoB<ladoA+ladoC && ladoC<ladoA+ladoB)
         return 0;
```

```
        else
            return 1;
    }
    public void tipoTriangulo(int ladoA, int ladoB, int ladoC){
        if(ladoA==ladoB && ladoA==ladoC)
            System.out.println("triângulo equilátero");
        else{
            if(ladoA==ladoB || ladoA==ladoC || ladoB==ladoC)
                System.out.println("triângulo isósceles");
            else
                System.out.println("triângulo escaleno");
        }

    }
}
```

b) Refaça o exercício acima usando os métodos da classe JOptionPane.

```
import javax.swing.JOptionPane;
public class ValidaTriangulo {
    public int formaTriangulo(int ladoA, int ladoB, int ladoC){
        if(ladoA<ladoB+ladoC && ladoB<ladoA+ladoC && ladoC<ladoA+ladoB)
            return 0;
        else
            return 1;
    }
    public void tipoTriangulo(int ladoA, int ladoB, int ladoC){
        if(ladoA==ladoB && ladoA==ladoC)
            JOptionPane.showMessageDialog(null, "triângulo equilátero");
        else{
            if(ladoA==ladoB || ladoA==ladoC || ladoB==ladoC)
                JOptionPane.showMessageDialog(null, "triângulo isósceles");
            else
                JOptionPane.showMessageDialog(null, "triângulo escaleno");
        }
    }
}
```

```
import javax.swing.JOptionPane;
public class MinhaClasse {
    public static void main(String[] args) {
        ValidaTriangulo var = new ValidaTriangulo();
        int a,b,c,valor;
        String aux;
        aux=JOptionPane.showInputDialog("Entre com o primeiro lado");
        a=Integer.parseInt(aux);
        aux=JOptionPane.showInputDialog("Entre com o segundo lado");
        b=Integer.parseInt(aux);
        aux=JOptionPane.showInputDialog("Entre com o terceiro lado");
        c=Integer.parseInt(aux);
        valor=var.formaTriangulo(a,b,c);
        if(valor==0)
            var.tipoTriangulo(a,b,c);
        else
```

```
            JOptionPane.showMessageDialog(null, "Não pode formar um triângulo");
        }
    }
```

c) Crie uma classe chamada Fatura para que uma loja de suprimentos em informática possa apresentar a fatura de um produto vendido na loja. Essa classe deverá ter uma variável de instância denominada totalFatura. Na referida classe deverá ter um método que recebe estas informações:

- **Número da fatura (String)**
- **Nome do produto (String)**
- **Quantidade de produtos (int)**
- **Preço de cada produto (double)**

Então, calcule o total da fatura (preço x quantidade) e guarde na variável de instância criada anteriormente. Nessa mesma classe, crie um segundo método que mostra uma tela com todos os itens da fatura, inclusive o total da mesma.

```
import javax.swing.JOptionPane;
public class MinhaClasse {
    public static void main(String[] args) {

        String minhaFatura;
        String nome;
        int quant;
        double preco;
        String aux;

        minhaFatura = JOptionPane.showInputDialog(null,"Número da fatura?");
        nome = JOptionPane.showInputDialog(null,"Nome do produto?");
        aux = JOptionPane.showInputDialog(null,"Quantidade de produto?");
        quant=Integer.parseInt(aux);
        aux = JOptionPane.showInputDialog(null,"Preço do produto?");
        preco=Double.parseDouble(aux);

        Fatura var = new Fatura();
        var.recebFatura(minhaFatura, nome, quant, preco);
    }
}
```

```
import javax.swing.JOptionPane;

public class Fatura {
    private double totalFatura;
    public void recebFatura(String nroFatura,
            String nomeProduto,
            int quantProduto,
            double precoProduto){
        totalFatura=precoProduto*quantProduto;
        String aux;
        aux = String.format(
                "Fatura número: "+nroFatura
                +"\nProduto: "+nomeProduto
                +"\nQuantidade:"+quantProduto
                +"\nPreço unitário: R$ "+precoProduto
```

```
            +"\n_____"
            +"\nTotal da Fatura: R$ "+totalFatura);
        JOptionPane.showMessageDialog(null,aux);
    }
}
```

d) Crie uma classe com um método chamado maiorValor, que recebe 2 valores e retorna o maior. Cria uma classe com o método main que lê dois números pelo teclado, chama o método maiorValor e mostra uma frase informando que é o maior valor, a partir do retorno do método maiorValor. Use os métodos da classe JOptionPane.

```
public class Calculo
{
   public int maiorValor(int a, int b)
   {
      if(a>b)
         return a;
      else
         return b;
   }
}
```

```
import javax.swing.JOptionPane;
public class Exemplo
{
   public static void main(String[] args)
   {
      String aux;
      int n1,n2,maior;
      Calculo var = new Calculo();
      aux=JOptionPane.showInputDialog(null,"Entre com o primeiro valor");
      n1=Integer.parseInt(aux);
      aux=JOptionPane.showInputDialog(null,"Entre com o segundo valor");
      n2=Integer.parseInt(aux);
      maior=var.maiorValor(n1,n2);
      aux=String.format("Maior valor = %d",maior);
      JOptionPane.showMessageDialog(null,aux);
   }
}
```

2 - Introdução ao encapsulamento, gets e sets

Declarar variáveis de instância com o modificador de acesso private é conhecido como ocultamento de dados (encapsulamento). Em O.O. encapsulamento significa separar o programa em partes, o mais isoladas possível. A ideia é tornar o software mais flexível, fácil de modificar e de criar novas implementações. Para exemplificar, podemos pensar em uma dona de casa (usuário) utilizando um liquidificador (sistema). O usuário não necessita conhecer detalhes do funcionamento interno do sistema para poder utilizá-lo, precisa apenas conhecer a interface, no caso, os botões que controlam o liquidificador.

Uma grande vantagem do encapsulamento é que toda parte encapsulada pode ser modificada sem que os usuários da classe em questão sejam afetados. No exemplo do liquidificador, um técnico poderia substituir o motor do equipamento por um outro totalmente diferente, sem que a dona de casa seja afetada.

O encapsulamento protege o acesso direto (referência) aos atributos de uma instância fora da classe onde estes foram declarados. Esta proteção consiste em se usar modificadores de acesso mais restritivos sobre os atributos definidos na classe. Depois devem ser criados métodos para manipular de forma indireta os atributos da classe.

É comum usar o padrão get<nomeDoAtributo> para o método que retorna o valor atual do atributo e set<nomeDoAtributo> para o método que modifica o valor de um atributo do objeto.

os JavaBeans são componentes reutilizáveis de software que podem ser manipulados visualmente com a ajuda de uma ferramenta de desenvolvimento. São usados para encapsular muitos objetos em um único objeto (o bean), assim eles podem ser transmitidos como um único objeto em vez de vários objetos individuais. O JavaBean permite acesso às suas propriedades através de métodos getter e setter.

2.1 - Refazendo o exemplo com get e set

Alterando a classe BoasVindas

```
public class BoasVindas
{
   private String nomeProfessor;
   public void setNomeProfessor(String nome)
   {
      nomeProfessor = nome;
   }
   public String getNomeProfessor()
   {
      return nomeProfessor;
   }
   public void mostraMensagem()
   {
      System.out.printf("Bem vindo ao curso de Java do %s\n",getNomeProfessor());
   }
}
```

Alterando a classe Exemplo

```
public class Exemplo
{
   public static void main(String args[ ])
   {
      BoasVindas msg=new BoasVindas( );
      msg.setNomeProfessor("Salvador");
      msg.mostraMensagem( );
   }
}
```

Observação:

- O acesso direto a um atributo protegido implicará erro de compilação

msg. nomeProfessor = "Salvador";//ERRO

Toda instância (objeto) de classe contém uma cópia da variável de instância. Então, a partir da mesma variável de instância da classe BoasVindas, posso fazer o seguinte sem problema:

```
 public static void main(String args[ ])
 {
       BoasVindas msg=new BoasVindas( );
       BoasVindas outra=new BoasVindas( );
       msg.setNomeProfessor("Salvador");
       msg.mostraMensagem( );
       outra.setNomeProfessor("Melo");
       outra.mostraMensagem( );
       msg.mostraMensagem( );
}
```

Na tela:
Bem vindo ao curso de Java do Salvador
Bem vindo ao curso de Java do Melo
Bem vindo ao curso de Java do Salvador

3 - Usando this

Se acontecer do nome de um argumento ou variável declarada por um método coincidir com o nome de um campo da classe (variável de instância), você deverá usar this.<nome da variável> para indicar, deixar claro, que você está se referindo ao campo da classe e não à variável do método, por exemplo. Esse this é inserido automaticamente guando se gera o gets e sets pelas IDE's do tipo Eclipse ou NetBeans.

Exemplo:

```
public class BoasVindas
{
   private String nomeProfessor;
   public String getNomeProfessor()
  {
      return nomeProfessor;
   }
   public void setNomeProfessor(String nomeProfessor)
  {
      this.nomeProfessor = nomeProfessor;
   }
   public void mostraMensagem()
   {
      System.out.printf("Bem vindo ao curso de Java do %s\n",getNomeProfessor());
   }
}
```

4 - Inicializando objetos com construtores

Vimos anteriormente que é simples criar uma classe. Mas, para realmente conseguirmos utilizar a classe, ela deve conter pelo menos um método construtor. O método construtor é desenvolvido da mesma forma que uma função, a diferença é que ele tem o mesmo nome da classe, e não podem retornar valores. Portanto não podem especificar tipo de retorno algum, nem mesmo void.

Isso se deve ao fato de que um objeto deve ser construído cada vez que chamamos a classe. E a responsabilidade de fazer isso é do construtor. Isso parte do princípio que podemos ter dois objetos com a mesma característica, mas que não são os mesmos objetos.

Exemplo:

```
public class Exemplo
{
   public static void main(String args[ ])
   {
      Conta consulta;
      consulta = new Conta(5730);
      consulta.mostraSaldo( );
   }
}
```

```
public class Conta
{
   private double saldoConta;
   public Conta(double valor)//método construtor
   {
      saldoConta = valor;
   }
   public void setSaldoConta(double valor)
   {
      saldoConta=valor;
   }
   public double getSaldoConta( )
   {
      return saldoConta;
   }
   public void mostraSaldo( )
   {
       System.out.println("Saldo = R$ "+getSaldoConta( ));
   }
}
```

5 - Métodos static, campos static e classe Math

Como já foi dito anteriormente, um método que não depende de instanciar um objeto da classe onde o método se encontra é declarado e conhecido como método static ou método da classe. Lembrando que a classe Math faz parte do pacote Java.lang, que é implicitamente importado pelo compilador.

Exemplo:

```
import java.math.*;
import java.util.Scanner;
public class Main {
   public static void main(String[] args)
   {
double valor;
System.out.println("Valor?");
Scanner x = new Scanner(System.in);
```

```
valor = x.nextDouble();
System.out.println("raiz = "+Math.sqrt(valor));
        //Math é a classe onde o método static sqrt( ) foi declarado
    }
}
```

1.1 Outros métodos da classe Math

Método	Descrição	Exemplo
abs(x)	valor absoluto de x	abs(23.7) é 23,7 abs(0.0) é 0,0 abs(-23.7) é 23,7
ceil(x)	arredonda x para o menor inteiro não menor que x	ceil(9.2) é 10,0 ceil(-9.8) é -9,0
cos(x)	co-seno trigonométrico de x (x em radianos)	cos(0.0) é 1,0
exp(x)	método exponencial e^x	exp(1.0) é 2,71828 exp(2.0) é 7,38906
floor(x)	arredonda x para o maior inteiro não maior que x	floor(9.2) é 9,0 floor(-9.8) é -10,0
log(x)	logaritmo natural de x (base e)	log(Math.E) é 1,0 log(Math.E * Math.E) é 2,0
max(x, y)	maior valor de x e y	max(2.3, 12.7) é 12,7 max(-2.3, -12.7) é -2,3
min(x, y)	menor valor de x e y	min(2.3, 12.7) é 2,3 min(-2.3, -12.7) é -12,7
pow(x, y)	x elevado à potência de y (isto é, x^y)	pow(2.0, 7.0) é 128,0 pow(9.0, 0.5) é 3,0
sin(x)	seno trigonométrico de x (x em radianos)	sin(0.0) é 0,0
sqrt(x)	raiz quadrada de x	sqrt(900.0) é 30,0
tan(x)	tangente trigonométrica de x (x em radianos)	tan(0.0) é 0,0

Dentro da classe Math temos ainda duas constantes matemáticas:

- Math.PI = relação entre a circuferência e o diâmetro de um circulo qualquer (3.1415...)
- Math.E = é a base para logaritmos naturais (2.718...)

Esses campos são declarados na classe Math como sendo public, final e static:

- Public: permite que outros programadores utilizem esse campo;
- Static: não exige a criação do objeto para ser utilizado;
- Final: para indicar que é uma constante, que seu valor não pode ser alterado depois que o campo é inicializado.

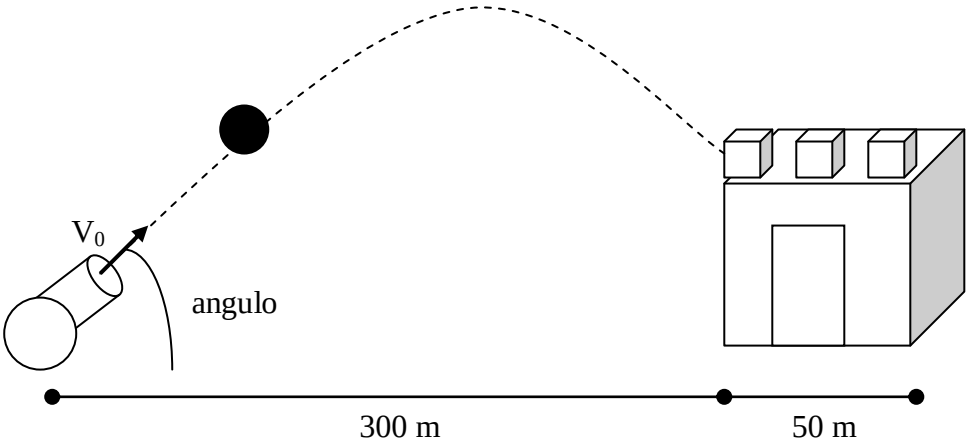
6 - Novos Exercícios

a) Imagine que você está tentando fazer um jogo de luta medieval com canhões, e tenta simular a realidade. Para isso você precisa utilizar equações da física para determinar o alcance da bala do canhão. Imagine também que o alvo está a 300 metros do canhão e tem 50 metros de comprimento. Faça um programa que calcula o alcance a partir da velocidade de lançamento e do ângulo de tiro, e verifica se o tiro acertará alguma parte do alvo, utilizando as seguintes fórmulas:

$$alcance = \frac{V_0^2 * Sen(2 * \theta)}{g}$$

onde:
alcance = alcance máximo que o tiro de canhão poderá alcançar

V_0 = velocidade inicial do tiro de canhão
θ = ângulo de tiro
g = aceleração da gravidade (considere como sendo 9,8 m/s^2)



Crie uma classe com um método com duas variáveis de instância: velocidade e ângulo, depois crie os métodos gets e sets para cada um desses campos da classe, crie também um método que calcula e retorna o alcance.

Crie outra classe com o método main que lê os valores da velocidade e ângulo e chama o método da primeira classe que calcula o alcance. Por fim mostre a informação se atingiu o alvo ou não.

b) Imagine que você está fazendo um jogo sobre Corrida de Dragsters (um tipo corrida envolvendo um veículo leve com motores extremamente potentes, especialmente projetados para provas de arrancadas em retas). O código abaixo visa criar um programa onde você deve entrar com o valor da aceleração e da distância e o referido programa calcula a velocidade final, usando a Equação de Torricelli:

$$V2 = \sqrt{V1^2 + 2 * a * distância}$$

Observações:

- Considera-se que a velocidade inicial (V1) é zero.
- A velocidade é calculada originalmente em m/s e deve ser convertida para km/h antes de mostrar o resultado na tela. Para isso o programa deve multiplicar a velocidade calculada por 3.6.

Crie uma classe com um método com duas variáveis de instância: aceleração e distância, depois crie os métodos gets e sets para cada um desses campos da classe, crie também um método que calcula e retorna a velocidade final.

Crie outra classe com o método main que lê os valores da aceleração e distância e chama o método da primeira classe que calcula a velocidade final. Por fim mostre a informação se a velocidade final bateu um novo recorde ou não.

c) Cria um jogo de adivinhação usando gets e sets. Um primeiro jogador entra com um número, depois um segundo jogador terá 5 chances (por exemplo) para acertar o número. Os resultados possíveis serão os seguintes:

- Se o palpite apresentado baixo, o programa deve informar: TENTE UM NÚMERO MAIOR.
- Se o palpite for alto, o programa deve informar: TENTE UM NÚMERO MENOR.
- Se for exato deve informar: PARABÉNS!!! ADVINHOU!!!
- Se ultrapassar as 5 tentativas, o programa deve informar: VOCÊ PERDEU!!!!