LTP 1 – Java Básico
Aula 6

Tópicos

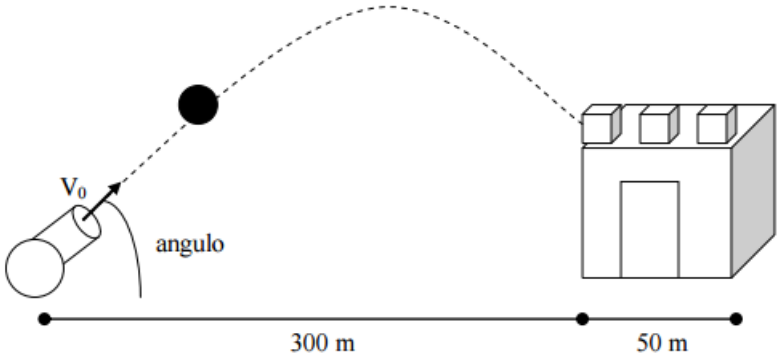
1 - Resolvendo os exercícios da aula passada

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/s2)



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.

```
import javax.swing.JOptionPane;//importando o pacote que contém a classe JOptionPane
public class Classe {// definindo a classe chamada Classe
   public static void main(String args[]){//definindo o método main
      String vel,ang;//declarando vel e ang como sendo do tipo String
      Calculo var = new Calculo();//instanciando um método da classe Calculo

      //lê um valor de uma caixa de entrada e retorna um String
      vel = JOptionPane.showInputDialog("Qual a velocidade?");
      ang = JOptionPane.showInputDialog("Qual o valor do ângulo?");

      //converte vel e ang de String para double e passa como argumento dos métodos set
      var.setVelocidade(Double.parseDouble(vel));
      var.setAngulo(Double.parseDouble(ang));

     //mostra o retorno do método calculaAlcance()
      System.out.println("Alcance = "+var.calculaAlcance());

      //verifica se você atingiu o alvo ou não
      if(var.calculaAlcance()>=300 && var.calculaAlcance()<=350){
         JOptionPane.showMessageDialog(null, "Acertou o alvo!!");
         System.out.println("Alcance = "+var.calculaAlcance());}
      else{
         JOptionPane.showMessageDialog(null, "Errou o alvo!!");
         System.out.println("Alcance = "+var.calculaAlcance());}
   }
}
```

```
public class Calculo
{
   private double velocidade;
   private double angulo;
   public double getVelocidade() {//retorna o valor da velocidade
      return velocidade;
   }
   public void setVelocidade(double velocidade) {//define o valor da velocidade
      this.velocidade = velocidade;
   }
   public double getAngulo() {//retorna o valor do angulo
      return angulo;
   }
   public void setAngulo(double angulo) {//define o valor do angulo
      this.angulo = angulo;
   }
   public double calculaAlcance(){
      double rad = angulo*Math.PI/180;//converte de graus para radianos
      double alcance = Math.pow(velocidade,2)*Math.sin(rad);//calcula o alcance
      return alcance;
   }
}
```

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.

```
import javax.swing.JOptionPane;//importa o pacote com a classe JOptionPane
public class Classe {
   public static void main(String args[]){
      String acel,dist;//declara duas variáveis do tipo String
      Calculo var = new Calculo();//instancia um novo objeto da classe Calculo

      //lê a aceleração e a distância pelo teclado, usando uma janela de entrada de dados
      acel = JOptionPane.showInputDialog("Qual a aceleração?");
      dist = JOptionPane.showInputDialog("Qual a distância?");

      //converte de String para double e o resultado é passado como parâmetro do método set
      var.setAceleracao(Double.parseDouble(acel));
      var.setDistancia(Double.parseDouble(dist));

      //mostra na tela o retorno do método calculaVelocidade()
      System.out.println("Velocidade = "+var.calculaVelocidade());

      //verifica se o retorno do método calculaVelocidade() é maior do que 500
      if(var.calculaVelocidade()>500)
         JOptionPane.showMessageDialog(null, "Quebrou o recorde!!");
      else
         JOptionPane.showMessageDialog(null, "Não quebrou o recorde!!");
   }
}
```

```
public class Calculo
{
   private double aceleracao;//variável de instância
   private double distancia; //variável de instância
   public double getAceleracao() {//retorna o valor armazenado em aceleracao
      return aceleracao;
```

```
    }
    public void setAceleracao(double aceleracao) {//define o valor de aceleracao
        this.aceleracao = aceleracao;
    }
    public double getDistancia() {//retorna o valor armazenado em distancia
        return distancia;
    }
    public void setDistancia(double distancia) {//define o valor de distancia
        this.distancia = distancia;
    }
    public double calculaVelocidade(){//define o método calculaVelocidade( )
        double velocidade;
        velocidade = Math.sqrt(2*aceleracao*distancia);
        return velocidade;
    }
}
```

2 - Gerando números aleatórios

Para gerar um número aleatório podemos utilizar a classe Radom (pacote Java.util) ou a classe Math.

2.1 - Usando a Classe Random

Os objetos da classe Random podem produzir valores aleatórios dos seguintes tipos:

- **Boolean**: um tipo de dado que contém literal lógico. Serve para armazenar um único bit de informação. Este bit pode ser representado pelas palavras false (falso) ou true (verdadeiro);

- **Float**: tipo de dado capaz de armazenar números reais de precisão simples, ou seja, 32 bits de informação representando um número real;

- **Double**: um tipo de dado capaz de armazenar números reais de precisão dupla, ou seja, 64 bits de informação em forma de número real. É usado para representar valores nos quais é preciso uma precisão maior que a de float;

- **Int**: um tipo de dado capaz de armazenar 32 bits, ou seja, de representar um número inteiro qualquer entre -2.147.483.648 e 2.147.483.647;

- **Long**: um tipo de dado capaz de armazenar 64 bits de informação, ou seja, que pode representar um número inteiro qualquer entre -9.223.372.036.854.775.808 e 9.223.372.036.854.775.807.;

- **Gaussianos**: para gerar um número aleatório (double) com distribuição gaussiana (ou "normal" - veja que "normal" não é a mesma coisa que "uniforme"!), com média 0.0 e desvio-padrão 1.0. Fim da página 11.

Exemplo:

```
import java.util.*;
public class Main {
   public static void main(String[] args)
   {
      Random var = new Random();//instancia um objeto da classe Random
      boolean v1=var.nextBoolean();//chama o método nextBoolean( ) da classe Random
      float v3=var.nextFloat();//chama o método nextFloat( ) da classe Random
```

```
        double v4=var.nextDouble();//chama o método nextDouble( ) da classe Random
        int v5=var.nextInt();//chama o método nextInt( ) da classe Random
        long v6=var.nextLong();//chama o método nextLong( ) da classe Random
        double gauss=var.nextGaussian();//chama o método nextGaussian( ) da classe Random
        System.out.println("boolean = "+v1);
        System.out.println("float = "+v3);
        System.out.println("double = "+v4);
        System.out.println("int = "+v5);
        System.out.println("long = "+v6);
        System.out.println("gaussiano = "+gauss);
    }
}
```

Observação:

Se quiser limitar o número de inteiros gerados, faça algo assim:
int v5=var.nextInt(6);//Neste caso os números gerados seriam de 1 a 6.

2.2 - Usando a Classe Math

O método Math.random pode produzir somente números do tipo double no intervalo entre 0.0<=aleatório<=1.0.

3 - Um pouco mais sobre o método main

3.1 - Por que método main é declarado como static?

A JVM tenta invocar o método main da classe que você especifica, sem ter que criar uma instância de classe.

3.2 - Por que o método main tem passagem de parâmetros?

Ao executar o método main, a JVM poderiam utilizar os argumentos desse método, normalmente String arg[ ], para especificar opções, por exemplo o nome de um arquivo que será necessário para a execução do aplicativo.

3.3 - A rigor só podemos ter um método main por programa?

Na verdade, qualquer classe pode ter um método main. Você pode colocar um método main em cada classe que você declara. A JVM invoca o método main somente na classe utilizada para executar o aplicativo. Mas isso, por enquanto pode gerar confusão. Mantenha um único método main por solução.

4 - Alguns erros comuns ao se utilizar métodos

- Declarar um método fora do corpo de uma declaração de classe;
- Declarar um método dentro do corpo de outro método;
- Omitir o tipo de retorno (mesmo que void) em uma declaração de método;
- Colocar um ponto e vírgula no final da linha de declaração de algum método;
- Redeclarar um parâmetro de um método com o mesmo nome de uma variável local;
- Informar um tipo de retorno (mesmo que void) quando se declara um método construtor.

5 - Sobrecarga de método

Sobrecarga de métodos ocorre quando, diferentes métodos com o mesmo nome podem ser declarados na mesma classe, contanto que tenham diferentes conjuntos de parâmetros

(determinados pelo número, tipos e ordem dos parâmetros). Quando um método sobrecarregado é chamado o compilador chama o método adequado, de acordo com os parâmetros passados.

Exemplo:

```
public class Main {
   public static void main(String[ ] args)
   {
      Sobrecarga var = new Sobrecarga();//instancia um objeto da classe Sobrecarga
      var.calcula(2.5); //chama o método calcula( ) e passa 1 parâmetro
      var.calcula(7,3); //chama o método calcula( ) e passa 2 parâmetros
   }
}
```

```
import javax.swing.*;
public class Sobrecarga
{
   public void calcula (double valor)//defini o método calcula que recebe 1 parâmetro
   {
      double dobro=valor*2;
      String x = String.format("Dobro = "+dobro);//formata toda a saída para um único String
      JOptionPane.showMessageDialog(null,x);
   }
   public void calcula (int valor1, int valor2) // defini o método calcula que recebe 2 parâmetros
   {
      int soma=valor1+valor2;
      String x = String.format("Soma = "+soma); //formata toda a saída para um único String
      JOptionPane.showMessageDialog(null,x);
   }
}
```

Observação:

Chamadas de métodos sobrecarregados não podem ser diferenciadas apenas pelo tipo de retorno dos métodos. Os métodos sobrecarregados podem ter diferentes tipos de retorno, se os métodos tiverem diferentes listas de parâmetros.

Exemplo:

- public int calculo(int x)
- public double calculo(int y)

6 - Exercícios

6.1 - Exercício a

Leia as coordenadas x e y de dois pontos e em seguida calcule a distância entre os dois pontos:
d = raiz ((x2 – x1) ^2 + (y2 – y1) ^2).

6.2 - Exercício b

Dizemos que um número é perfeito se a soma de seus fatores, incluindo o 1 (mas não o próprio número) é igual ao número. Por exemplo, 6 é um número perfeito porque 6=1+2+3. Escreva um método chamado perfeito que determina se o parâmetro numero é um número perfeito. Exiba os fatores de cada número perfeito para confirmar que o número é de fato perfeito. Teste todos os números de 1 a 1000.

6.3 - Exercício c

Escreva um método que calcula a distância entre dois pontos (x1,y1) e (x2,y2), com a leitura de parâmetros no próprio método. Escreva outro método com o mesmo nome, mas que passa os valores de x1, x2, y1 e y2, e que também calcula a distância entre dois pontos.

6.4 - Exercício d

Os computadores estão desempenhando um crescente papel na educação. Escreva um programa que ajudará um estudante do ensino fundamental a aprender multiplicação. Utilize algum gerador de números aleatórios para produzir dois números inteiros positivos. O Programa deve então fazer ao usuário o seguinte tipo de pergunta: Quanto é 6 vezes 7? Depois o programa deve ler a resposta e informar se o usuário acertou ou não.