Programação

Métodos

Por Carla Macedo

Métodos

- Métodos são pequenos blocos de código que, juntos compõem um sistema maior.
- Os métodos recebem um determinado nome e podem ser chamados várias vezes durante a execução de uma classe.
- Algumas das vantagens são a redução do tamanho total de código e modularização do sistema.

Criação de Métodos

- Um método pode invocar outro método, isto é o metodo1, pode executar o metodo2, e este por sua vez pode executar o metodo3.
- Todos os métodos tem que ter obrigatoriamente, um **qualificador**, um **tipo de retorno** e **nome**.
- Pode ter ou não lista-de-parametros.

```
qualificador tipo-de-retorno nome-do-metodo ([lista-de-parametros])
{
    codigo-do-metodo
}
```

Métodos sem Retorno s/ parâmetros

metodo1.java *

```
1  class metodo1
2
3  {
4      public static void main(String args[])
5      {
6          imprime();
7          imprime();
8          imprime();
9      }
10
11
12     public static void imprime()
13     {
14         System.out.println("Ola, primeiro metodo em  Java.");
15
16     }
17
18
19 }
```

Método main

Método imprime

qualificador modificador retorno nome parâmetros

Métodos (qualificador)

- Define a visibilidade do método, pode ser:
  - public: visível em qualquer parte da classe
  - private: visível apenas pela própria class
  - protected: visível pela propria class, por suas subclasses e pelas classes do mesmo pacote.

Métodos (modificador)

- O modificador do método permite especificar como classes derivadas podem ou não redefinir ou alterar o método, e de que forma esse método será visível.

  public: O método declarado com este moderador é público e pode ser chamado a partir de métodos contidos em qualquer outra classe. Esta é a condição de menor restrição possível.

Métodos tipo-de-retorno

- É obrigatório definir **SEMPRE** o tipo de retorno

referente ao tipo de dado retornado pelo método.

- Métodos que não retornam valores devem possuir neste parâmetro a palavra **void**.
- Um método pode ter como retorno qualquer tipo primitivo (int, float, double, etc).
- O valor é retornado pelo comando **RETURN**

Métodos nome-do-metodo

- Pode ser qualquer palavra ou frase, desde que iniciada por uma letra, termina ()
- Se for uma frase, não pode ter espaços em branco entre as palavras. Como padrão da linguagem Java, o nome de um método começa sempre com uma palavra com letra minúscula.

  Exp: public void **imprime()**

  private void **imprimeFrase()**

  protected void graveFicheiroTexto()

Metodos lista-de-parametros

- Trata-se de uma lista de variáveis opcionais, que podem ser recebidas pelo método para tratamento interno.
- Quando um método é invocado, este pode receber valores de quem o chamou.
- Um método pode ter 0, 1 ou mais parâmetros
- Um mesmo método pode receber diferentes tipos de variáveis.

Métodos sem Retorno

Não retornam valores, são definidos como void.

void=vazio

Exemplos:

public static void imprime()

public static void imprimeFrase()

public static void gravaFicheiroTexto()

public static void limpaEcra()

Métodos sem Retorno s/ param

metodo1.java *

```
1  class metodo1
2
3  {
4      public static void main(String args[])
5      {
6          imprime();
7          imprime();
8          imprime();
9      }
10
11
12     public static void imprime()
13     {
14         System.out.println("Ola, primeiro metodo em  Java.");
15
16     }
17
18
19 }
```

Métodos sem Retorno c/ param

```
public class beepm
{

    public static void main(String args[])
    {
        beep(3);

    }

    public static void beep(int a)
    {
        for (int i=0;i<a;i++)
            System.out.println((char)7);
    }
}
```

Métodos sem Retorno c/ param

metodo2.java *

```
1  class metodo1
2
3  {
4      public static void main(String args[])
5      {
6          imprimeA(5);
7          System.out.println();
8          imprimeA(10);
9          System.out.println();
10         imprimeA(15);
11         System.out.println();
12     }
13
14     public static void imprimeA(int n)
15     {
16         for (int i=0;i<=n;i++)
17             System.out.print("*");
18
19     }
20 }
```

Métodos - Exercícios

J25 - Crie uma classe composta por 3 métodos. Um que leia uma variável, outro que limpe o ecrã e outro que mostre a variável lida no ecrã.

J26 – Utilizando métodos crie uma classe que gere um vector com números aleatórios (t=10), ordene e o mostre ordenado e desordenado.

J27 - Utilizando vários métodos, crie uma classe que mostre um menu com possibilidade de executar repetir e terminar 3 programas á sua escolha.

Métodos com Retorno

A declaração deste método informa que ira receber 2 parâmetros inteiros (x,y) e que retorna um numero inteiro do tipo int.

```
public static int soma(int x, int y)
{
    return x+y;
}
```

O retorno é feito com o comando **return**

Métodos com Retorno

```
import java.util.Scanner;


class a1
{
    public static double a,b;

    public static void main(String args[])
    {
        areaTriangulo();
    }

    public static void areaTriangulo()
    {
        Scanner kbd = new Scanner(System.in);
        System.out.print("Base: ");
        b=kbd.nextInt();
        System.out.print("Altura: ");
        a=kbd.nextInt();
        System.out.println("Area triangulo: "+(b*a)/2);
    }
}
```

Advertência:

- É **PROIBIDO** utilizar operações de entrada e saída de dados fora da classe Main().

Exemplo: proibido a utilização do println() e leitura de dados fora da classe Main().

Métodos com Retorno

```
import java.util.Scanner;

class a2
{
    public static void main(String args[])
    {
        double a, b;
        Scanner kbd = new Scanner(System.in);
        System.out.print("Base: ");
        b=kbd.nextInt();
        System.out.print("Altura: ");
        a=kbd.nextInt();
        System.out.println("Area triangulo:" + areaTriangulo(b,a));
    }

    public static double areaTriangulo(double bt, double at)
    {
        return (bt*at)/2;
    }
}
```

Funções matemáticas (Math)

Math.PI = 3,1415926535 e Math.E=2.718281824

- Math.ceil() - arredonda tipo double para inteiro
- Math.sqrt() - calcula a raiz quadrada
- Math.pow() - eleva um numero a outro (2^3)
- Math.random() - gera numero entre 0.0 e 1.0
- Math.max() - retorna maior de 2 valores
- Math.min() - retorna menos de 2 valores

Classe Calendar

Esta classe permite manipular a data e hora do sistema.

```
import java.io.*;
import java.util.*;

class t1
{
    public static void main(String args[])
    {
        Calendar agora = Calendar.getInstance();

        System.out.println(agora.getTime());

        System.out.println(agora.get(Calendar.YEAR));
        System.out.println(agora.get(Calendar.MONTH)); // 0 a 11
        System.out.println(agora.get(Calendar.DAY_OF_MONTH));
        System.out.println(agora.get(Calendar.HOUR));
        System.out.println(agora.get(Calendar.HOUR_OF_DAY));
        System.out.println(agora.get(Calendar.MINUTE));
        System.out.println(agora.get(Calendar.SECOND));

    }
}
```

Métodos - Exercícios

J28 - Crie uma classe com um método que recebe tempo em **horas, minutos e segundos**, e retorne o numero de segundos correspondente.

J29 - Crie uma classe com um método que recebe um mês em formato numérico e retorna o **número de dias desse mês** (com validações na classe).

J30 - Crie 2 métodos, um que gera um **nº aleatórios** e outro que avalia se um número é **par ou impar**. A classe deve perguntar quantos números quer gerar, mostrar e avaliar se são pares ou impares.

Métodos chamam outros métodos

É possível um método chamar outros métodos.

```
class m3
{
    public static void main(String args[])
    {
        System.out.println(media(quadrado(dobro(3)), 4, 5));
    }

    public static double media(double a, double b, double c)
    {
        return (a+b+c)/3;
    }

    public static double quadrado(double a)
    {
        return a*a;
    }

    public static double dobro(double a)
    {
        return 2*a;
    }

}
```

Acesso a Métodos de Outras Classes

- Esta característica permite executar métodos criados noutras classes.
- Qualquer método do tipo static criado numa classe pode ser utilizado noutra classe pelo formato nome-da-classe.nome-do-metodo()
- Permite a reutilização de métodos noutras classes.

Acesso a Métodos de Outras Classes

- Exemplo:

```
class pp
{

     public static int soma(int x, int y)
     {
          return x+y;
     }
}
```

```
class ppp
{

     public static void main(String args[])
     {
          System.out.println(pp.soma(2,3));
     }


}
```

chama o método soma da classe pp com parâmetros 2 e 3

Métodos - Exercícios

J31 - Crie uma classe com vários métodos de forma a calcular factorial(dobro(raiz(49)));

J32 - Crie uma classe sem o método **main** com 7 métodos; **soma**, **subtrai**, **divide**, **multiplica**, **dobra**, **quadrado** e **factorial** para números reais.

J33 - Crie uma classe **apenas como o método main**, com um menu (0 termina) que permita simular uma maquina de calcular utilizando a classe J32.

Excepções em Java

As excepções Java referem-se a erros que podem ser gerados durante a execução de um programa. Esses erros podem e devem ser controlados (tratados) dentro de um programa, pois é comum que diversos tipos de execução ocorram durante o processo de execução.

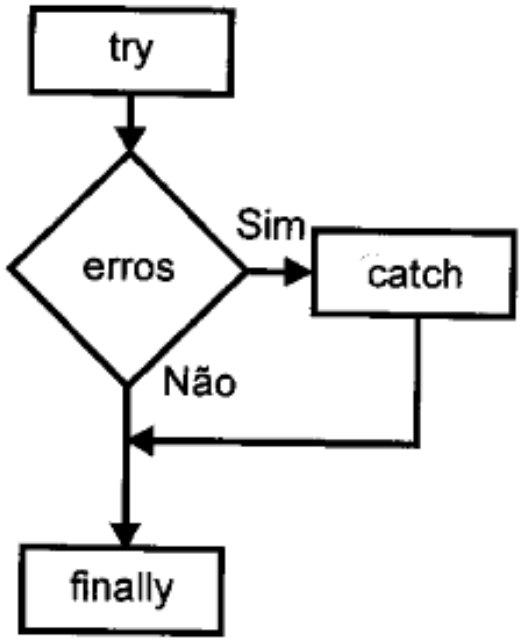
Em java existem 2 estruturas de controle de erros: **try-catch-finally** e **throws**.

try-catch-finally

Esta estrutura tem como função desviar a execução de um programa caso ocorram certos tipo de erro, predefinidos durante o processamento das linhas, e evitar que o programador precise fazer testes de validações. Quando um erro ocorre, ele gera uma excepção que pode ser tratada pelo programa.

try-catch-finally

Sintaxe:



try-catch-finally

Escreva o seguinte programa:

```
import java.util.Scanner;

class tc // escrever num ficheiro
{
    public static void main(String args[])
    {
        int x,y;
        Scanner kbd = new Scanner(System.in);
        System.out.print("Insira o primeiro numero:");
        x=kbd.nextInt();
        System.out.print("Insira o segundo numero:");
        y=kbd.nextInt();
        System.out.println("Divisao de " + x + " por " + y  +" = " + x/y);
    }
```

Teste o programa com segundo valor = ZERO

Como eliminar este erro com esta estrutura ?

try-catch-finally

```
import java.util.Scanner;

class tc // escrever num ficheiro
{
    public static void main(String args[])
    {
        String a,b;
        int x,y;
        Scanner kbd = new Scanner(System.in);
        System.out.print("Insira o primeiro numero:");
        a=kbd.nextLine();
        System.out.print("Insira o segundo numero:");
        b=kbd.nextLine();

        try
        {
            x=Integer.parseInt(a);
            y=Integer.parseInt(b);
            System.out.println("Divisao de " + x + " por " + y  +" = " + x/y);
        }
        catch (ArithmeticException erro)
        {
            System.out.println("ERRO de divisao por zero ("+ erro + ")");
        }
        catch (NumberFormatException erro)
        {
            System.out.println("ERRO digite apenas numeros inteiros !");
        }
        finally
        {
            System.out.println("Fim do programa");

        }
    }
}
```

Excepções – Exercícios

J34 - Crie uma classe que leia o nome de um filme e o ano, validando o ano.

J35 - Utilizando a classe J34, altere a mesma de forma a que receba apenas valores entre 1900 e 2011 para o ano.

J36 - Crie uma classe que simule uma maquina de calcular com opções (1-Soma, 2-Subtrai, 3-Multiplica, 4-Divide, 0-Sair) com números inteiros, validando excepções.

Operadores condicionais

```
import java.util.Scanner;

class oc
{


    public static void main(String args[])
    {
        int n;
        Scanner kbd = new Scanner(System.in);

        System.out.print("Insira um numero: ");
        n=kbd.nextInt();

//      if (n>0)
//          System.out.println("Numero positivo");
//      else
//          System.out.println("Numero negativo");


        System.out.println(n >0 ? "Numero positivo" : "Numero negativo");


    }
}
```

Ciclos avançados

```
import java.util.Scanner;

class e1
{
    public static void main(String args[])
    {
        int ar[]={11,22,33,44,55};

        for (int x=0;x<ar.length;x++)
            System.out.println(ar[x]);

        for(int x:ar)
            System.out.println(x);

    }
}
```

Métodos com n parâmetros

```
import java.util.Scanner;

class mn
{
    public static void main(String args[])
    {
        System.out.println(media(1,2,3,4));
    }

    public static int media(int...numeros)
    {
        int total=0;
        for(int i:numeros)
            total=total+i;
        return (total/numeros.length);
    }


}
```

FIM

ORACLE®

Sun microsystems