Sobrecarga de métodos

P. O. O.

Prof. Grace

Maratona InterFatecs
Participem e divulguem!

- Enviar nome do time + dados dos participantes (nome + e-mail + CPF + tamanho camiseta)
- Prazo: até dia 14/04.
- Enviar pelo chat do teams ou e-mail graceapborges@fatecsp.br

- Saiba mais: Palestra dia 01/04 (2ª feira) das 18h às 19h
- Lab. Informática – Sala 6 (Azul)

Agenda para Aula 05

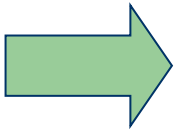
- Correção de atividades
- Sobrecarga de métodos
- Mais exercícios

Membros de uma Classe

Uma classe é composta por:

- Comportamentos ou operações
- Características ou atributos

(modelo)



- Métodos ou funções
- Dados ou variáveis

(implementação)

Atividade: Classe Círculo

- Classe: Círculo
- Atributos (variáveis de instância)
  - Raio
- Métodos (tarefas)
  - Alterar/ informar raio
  - Calcular diâmetro
  - Calcular área
  - Calcular circunferência
  - Exibir Dados

E o construtor, precisa implementar?
Para que serve?
E se eu não implementar, a classe funciona?

Classe Circulo (com construtor)

```
//Declaracao da classe Circulo.java
public class Circulo
{
    // atributo privado
    private double raio;

    // método construtor
    public Circulo(double r)
    {
        setRaio(r);
    }

    // método alterar raio
    public void setRaio(double r)
    {
        if (r < 0)
            System.out.println("O raio não pode ser negativo.");
        else
            raio = r;
    }

    // método informar raio
    public double getRaio()
    {
        return raio;
    }
```

Atributos

Aloca memória inicializa atributos do obj.

Métodos

Altera atributo com segurança (encapsulamento)

Acessa valor armazenado no atributo

Classe Circulo.java (cont.)

```
    // método calcular diametro
    public double diametro()
    {
        return raio*2;
    }

    // método calcular area
    public double area()
    {
        return Math.PI * Math.pow(raio,2);
    }

    // método calcular circunferencia
    public double circunferencia()
    {
        return 2 * Math.PI * raio;
    }

    // método exibir dados
    public void exibeDados()
    {
        System.out.printf("\n=====================================\n");
        System.out.printf("\n Dados do circulo de raio %.2f", getRaio());
        System.out.printf("\n Diametro      : %6.2f", diametro());
        System.out.printf("\n Circunferencia: %6.2f", circunferencia());
        System.out.printf("\n Area          : %6.2f", area());
        System.out.printf("\n\n=====================================\n");
    }
}// fim da classe
```

devolve o diâmetro

devolve a área

circunferência

Uso da classe implementada

- Programa java (possui método main)
  - Instancia um círculo ***c*** de raio 5 e exibe seus dados
  - Altera o raio de ***c*** para 15 e exibe seus dados novamente

```
//TesteCirculo.java
// utiliza classe circulo
public class TesteCirculo
{
    public static void main (String args[])
    {
        Circulo c = new Circulo(5);
        c.exibeDados();

        c.setRaio(15);
        c.exibeDados();
    }
}
```

Saída jGrasp

```
----jGRASP exec: java TesteCirculo


========================================

 Dados do circulo de raio 5,00
 Diametro       :  10,00
 Circunferencia:  31,42
 Area           :  78,54

========================================

========================================

 Dados do circulo de raio 15,00
 Diametro       :  30,00
 Circunferencia:  94,25
 Area           : 706,86

========================================

 ----jGRASP: operation complete.
```

Atividade
Classe ContaCorrente

- Atributos (variáveis de instância)
  - Número da conta
  - Titular
  - Saldo

- Métodos (operações/ tarefas)
  - Construtor: inicializa titular, numero da conta e saldo (sempre maior ou igual a zero);
  - Depósito (atualizar saldo acrescido da quantia depositada);
  - Saque (atualizar saldo decrescido da quantia sacada);
  - Exibir dados da conta

Atividade – Conta corrente

- Implemente a classe ContaCorrente
  - O valor inicial do saldo deve ser sempre maior ou igual a 0;
  - Não esqueça de validar os valores de saque e depósito (não devem ser menores que zero).

Implementação

```
public class ContaCorrente
{
    private int numeroConta;
    private String titular;
    private double saldo;

    public ContaCorrente(int n, String t, double s)
    {
        setTitular (t);
        if (n < 0)
            System.out.printf("\nNumero de conta invalido!");
        else
            numeroConta = n;

        if (s < 0)
            System.out.printf("\nSaldo inicial invalido!");
        else
            saldo = s;
    }
```

Não apresenta tipo de retorno

Usa método set

Não possui método set

Não possui método set

Saque e deposito

```
public void saque(double vlrSaque)
{
    if (vlrSaque < 0)
        System.out.printf("\nValor de saque inválido!");
    else
        if (vlrSaque > saldo)
            System.out.printf("\nRecurso insuficiente!");
        else
            saldo = saldo - vlrSaque;
}

public void deposito (double vlrDeposito)
{
    if (vlrDeposito < 0)
        System.out.printf("\nValor de deposito invalido!");
    else
        saldo = saldo + vlrDeposito;
}
```

verDados( ) e setTitular( )

```
public void verDados()
{
    System.out.printf("\n=======================================");
    System.out.printf("\nConta  : %07d", getConta());
    System.out.printf("\nTitular: %s", getTitular());
    System.out.printf("\nSaldo  : R$ %.2f", getSaldo());
    System.out.printf("\n=======================================");
}

public void setTitular (String s)
{
    titular = s;
}
```

Obs.: Caso opte em criar o método setSaldo( ) para ser usado no construtor, Saque( ) e Deposito( ), não esqueça de deixar com acesso privado (private)

Gets

```
    public String getTitular ()
    {
        return titular;
    }

    public int getConta ()
    {
        return numeroConta;
    }

    public double getSaldo ()
    {
        return saldo;
    }
}
```

Programa teste

```
import java.util.Scanner;
public class TesteCCorrente
{
    public static void main(String args[])
    {
        ContaCorrente cc1 = new ContaCorrente(12345, "Joao da Silva", 0);
        cc1.verDados();

        ContaCorrente cc2;
        cc2 = new ContaCorrente(54321, "Maria dos Santos", 500);
        cc2.verDados();

        Scanner entrada = new Scanner(System.in);

        System.out.printf("\nValor para deposito em c1: ");
        double vlr = entrada.nextDouble();
        cc1.deposito(vlr);
        cc1.verDados();

        System.out.printf("\nValor de saque em c2: ");
        cc2.saque(entrada.nextDouble());
        cc2.verDados();
    }
}
```

Dúvidas

Sobrecarga de métodos

- O que é?
  - Recurso que permite que vários métodos, com mesmo nome sejam definidos
  - Obrigatório diferentes parâmetros: quantidade, tipos e ordem dos argumentos
  - Apesar de ser bastante usada para definir vários construtores, a sobrecarga também pode ser usada para sobrescrever outros métodos que não sejam construtores.

Quantos construtores uma classe pode ter?

Nenhum (construtor padrão: não recomendável)

Apenas 1 construtor

Vários construtores!!!

Para isso usaremos o recurso de sobrecarga

Exemplo inicial: Classe Data

```
1  public class Data {
2
3      private int mes; // 1 a 12
4      private int dia; // 1 a 31, de acordo com o mês
5      private int ano; // qualquer ano
6      public Data (int d, int m, int a)
7      {
8          setData(d, m, a);
9      }
10
11     public void setData(int d, int m, int a)
12     {   if (m > 0 && m <=12)
13             mes = m;
14         else
15         {
16             mes = 1;
17             System.out.println("Mês " + m +
18                     " inválido. Configurado mês = 1.");
19         }
20         ano = a;
21         dia = checkDia (d);
22     }
```

construtor

valida e atualiza atributos do objeto tipo Data

Criando a classe Data (cont.)

```
23
24      private int checkDia (int diaTeste)
25      {
26          int diasMes[] = {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
27          if (diaTeste > 0 && diaTeste <= diasMes[mes])
28             return diaTeste;
29          System.out.println("Dia " + diaTeste +
30                  " inválido. Configurado dia = 1.");
31          return 1;
32      }
33      public String toString()
34      {
35          return dia + "/"+ mes+ "/"+ ano;
36      }
37  }
```

Método privado: uso interno da classe

Teste da Classe Data

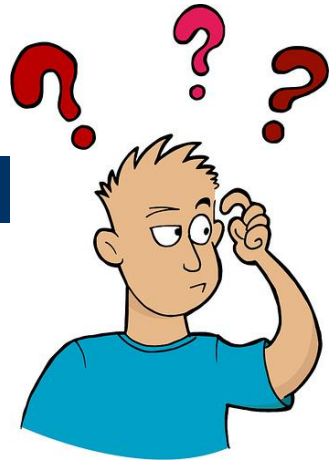
```
public class TesteData {
   public static void main (String args[])
   {
        Data data = new Data(06, 04, 2024);
        System.out.println("Meu aniversário: " + data.toString());

        data.setData(31, 02, 2024);
        System.out.println("Que data é essa? " + data.toString());

   }
}
```

Implemente e teste sua classe

E se eu quisesse usar apenas algumas informações?



```
public class TesteData {
   public static void main (String args[])
   {
        Data data1, data2, data3;
        data1 = new Data(20, 03, 2024);
        data2 = new Data(2022);
        data3 = new Data(06, 2025);

        System.out.println(
        " Hoje é " + data1.toString() +
        " ingressei no segundo semestre de " + data2.getAno() +
        " e me formarei em " +
        data3.getMes()+"/"+data3.getAno());
   }
}
```

Construtores sobrecarregados - Classe Data

```
// construtor existente
public Data (int d, int m, int a)
{
     setData(d, m, a);
}

// construtor com mês e ano
public Data (int m, int a)
{
     setData(1, m, a);
}

// construtor com apenas ano
public Data (int a)
{
     setData(1, 1, a);
}
```

Incluir métodos get

```
(...) // método público – obtém dia
    public int getDia ()
   {
        return dia;
   }
   // método público – obtém mês
    public int getMes ()
   {
        return mes;
   }
   // método público – obtém ano
    public int getAno ()
   {
        return ano;
   }
(...)
```

Teste Classe Data

```
public class TesteData {
   public static void main (String args[])
   {
        Data data1, data2, data3;
        data1 = new Data(20, 03, 2024);
        data2 = new Data(2022);
        data3 = new Data(06, 2025);

        System.out.println(
        " Hoje é " + data1.toString() +
        " ingressei no segundo semestre de " + data2.getAno() +
        " e me formarei em " +
        data3.getMes()+"/"+data3.getAno());
   }
}
```


Implemente e Teste

Pergunta importante

- Sobrecarga de métodos só funciona para método construtor???

Atividade - Classe Data

- Altere a classe Data incluindo o método bissexto( ) que informa se o ano é bissexto ou não (devolve booleano);
- Utilize este método para validar a quantidade de dias de fevereiro dentro do método checkDia( );

Ativ.: Conta Corrente com Limite de Cheque Especial

- Altere a classe conta corrente considerando um novo atributo: **Limite**
- Considere este atributo no construtor da classe;
- Durante a operação de saque atualize **saldo** considerando que:
  - Só é permitido sacar se a quantia estiver dentro do saldo mais limite do cliente;
  - Caso a quantia não esteja coberta pelo limite, informe que não foi possível realizar o saque por falta de recursos;

Ativ.: Conta Corrente com Limite de Cheque Especial

- Sobrecarga:
  - Codifique um segundo construtor que receba número da conta, titular e saldo. Neste caso, o **Limite padrão** é de R$1000,00.
  - Codifique um terceiro construtor que receba apenas o número da conta e titular. Neste caso, **saldo e limite devem ser inicializados com 0**.

Atividades disponíveis no Teams

- Classe Data com Ano Bissexto
- Classe Conta corrente com Limite

- Em caso de problemas com o teams, por gentileza enviar atividades por email: ***poo.profgrace@yahoo.com.br***