

Classes, encapsulamento e construtores

Parte 2

P. O. O.

Prof. Grace

Cuidados!!!

- Declarar mais de uma **classe public** no mesmo arquivo é um erro de compilação.
- Porém, podemos ter vários **métodos public** declarados em uma mesma classe
- Recomenda-se que atributos sejam privados (**encapsulamento**)
- Apesar de privados, os **atributos** podem ser acessados ou alterados a partir de qualquer **método**.

Boa prática de programação

- Listar os **atributos** de uma classe **antes de declarar métodos** da classe: ao ler o código, você vê os nomes e tipos das variáveis antes de usá-los nos métodos.
- É possível listar os atributos da classe em qualquer lugar na classe (**fora dos métodos**), mas sua dispersão tende a resultar em um código de difícil leitura.

Como usar/testar uma classe?

- Classe Circulo não tem método **main**
- Execução de programa em Java: classes com método **main**
- Se tentarmos executar classe sem main?
 - ERRO!
 - Apenas compile para gerar o “.class”



Programas usando classes

- **Palavra reservada “new”**: cria instância (exemplar) de classe, ou seja, um objeto na memória:
 - **new** + nome da classe + parênteses.
- Chamando (invocando) um método:
 - Nome de objeto + ponto (.) + nome do método + parênteses.

Programa usando classe Circulo

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

Instancia objeto **c**
do tipo **Circulo**

Chama método
exibeDados do objeto **c**

**Invocamos os métodos a partir do objeto.
Nesse exemplo, o que será impresso?**

Invocando métodos

- Observe que não existe obrigatoriedade de chamar os métodos na ordem em que foram implementados.
- Semelhante a um “fornecedor de serviços”, o objeto instanciado disponibiliza todos os seus métodos, entretanto, o programa só utiliza aqueles que desejar, na ordem que for relevante.
- Exemplo: Métodos da classe Circulo
 - + setRaio(double r)
 - + getRaio()
 - + exibeDados()

Alterando o raio

- Em métodos com **parâmetros**, os mesmos são usados para passar informações adicionais ao **método**.
- Neste exemplo, usamos o valor do parâmetro **r** para alterarmos o atributo **raio**.

```
// método alterar raio
public void setRaio(double r)
{
    raio = r;
}
```


Alterando o programa

```
//TesteCirculo.java
// utiliza classe circulo
public class TesteCirculo
{
    public static void main (String args[])
    {
        Circulo c = new Circulo();
        c.setRaio(5);    ← Chamando método
        c.exibeDados();  com parâmetro
    }
}
```

Teste seu código!



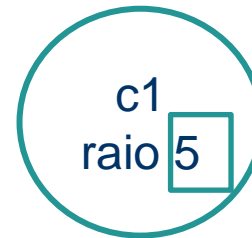
Variáveis de instância x Variáveis locais

- Atributos ou variáveis de instância:
 - Variáveis declaradas na declaração de classe;
 - Cada objeto (instância) da classe tem uma instância separada da variável;
 - Existe enquanto o objeto existir: **antes** e **depois** de chamadas aos métodos;
 - Por exemplo: ***raio***
- Variáveis locais:
 - Declaradas no corpo do método;
 - Só podem ser utilizadas nesse método;
 - Só existem **durante** a execução do método;

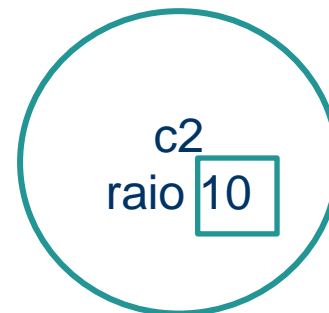
Exemplo: Classe Circulo

- Variável de instância: **raio**;
- Cada objeto do tipo **Circulo** tem seu próprio **raio**;

```
Circulo c1 = new Circulo();  
c1.setRaio(5);
```

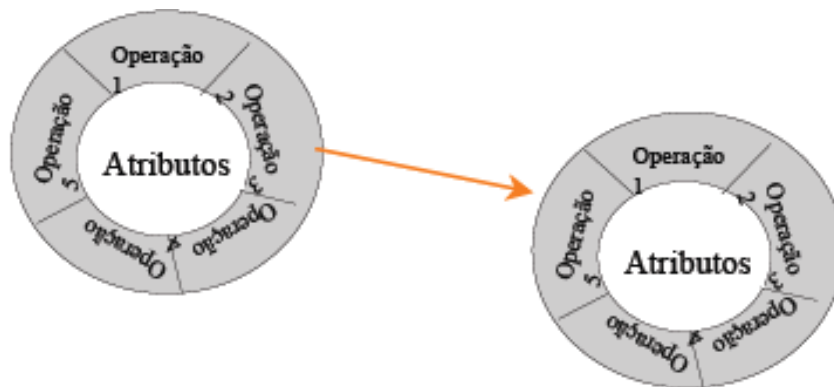


```
Circulo c2 = new Circulo();  
c2.setRaio(10);
```



Acesso aos atributos: público ou privado?

- Pela prática de encapsulamento de O.O., atributos são **privados** (somente o próprio objeto pode vê-lo);
- Para alterar ou ler seu valor, criamos métodos públicos: **set** e **get**;
- **Encapsulamento:**



Como controlamos visibilidade de atributos e métodos?

- Modificadores de acesso
 - **Public**: em geral, métodos públicos de interface com cliente;
 - **Private**: métodos ou atributos não acessíveis fora da classe;

Recomendação: todas as variáveis de instância sejam **private**

Importância do encapsulamento

Porque encapsular e ocultar?

- Criamos classes para “clientes” (reuso)
- Capacidade de usar sem conhecer detalhes internos
- Alterações na implementação não afetam cliente
 - Garantia de acesso seguro aos dados

Posso ter círculo com raio negativo?

Supondo que não, devemos validar no método set.

Alterando o método Set

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



Dúvidas



Construtor



- O que é?
 - Método especial que cria ou instancia novos objetos na memória do computador;
 - Tem o mesmo **nome** da classe;
 - Assegura **estado consistente** do objeto inicializando seus atributos;

Boa prática. Inicialize as variáveis de instância de uma classe no seu construtor.

Construtores Java

- O Java requer **um** construtor para cada classe.
- O Java **fornecerá** um construtor sem argumentos-padrão, caso **nenhum seja fornecido**.
- Ex. **classe Círculo**: Podemos implementar o construtor inicializando o valor do raio.
- Os construtores são chamados quando a palavra-chave **new** precede o **nome da classe**. Ex.:
Scanner entrada = **new** Scanner(System.in);
Circulo c = **new** Circulo();

Classe Circulo (início)

//Declaração da classe Circulo.java

public class Circulo ← Início da declaração da classe Circulo

{

// atributo privado
private double raio;

Atributos

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

← Aloca memória inicializa atributos do obj.

Métodos

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

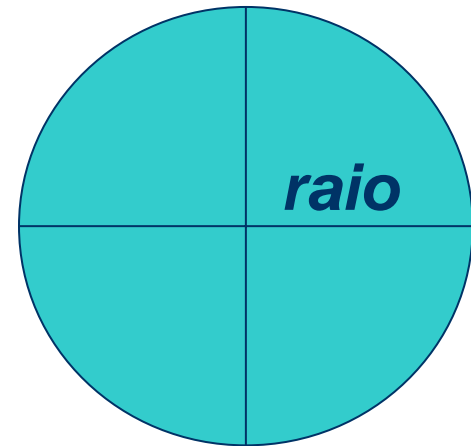
← Altera atributo com segurança
(encapsulamento)

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

← Acessa valor armazenado no atributo

Atividade: Classe Circulo

- Atributo (variáveis de instância)
 - Raio
- Métodos (tarefas)
 - Método construtor
 - Alterar/ informar raio
 - Calcular diâmetro
 - Calcular área
 - Calcular circunferência
 - Exibir Dados: informa diâmetro, área e circunferência.
- Obs.: Utilize a classe Math
 - Math.PI
 - Math.pow()



Exemplo: Uso da classe círculo

```
=====
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
=====
Press any key to continue...
```

Atividade - Classe ContaCorrente

- Atributos (variáveis de instância)
 - Número da conta
 - Titular
 - Saldo
- Métodos (operações/ tarefas)
 - Construtor: inicializa titular, número 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).

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();
    }
}
```


Exemplo de saída

```
=====
Conta   : 0012345
Titular: Joao da Silva
Saldo   : R$ 0,00
=====

Conta   : 0054321
Titular: Maria dos Santos
Saldo   : R$ 500,00
=====
Valor para deposito em c1: -100 ←
Valor de deposito invalido!
=====
Conta   : 0012345
Titular: Joao da Silva
Saldo   : R$ 0,00
=====
Valor de saque em c2: 1000 ←
Saldo insuficiente!
=====
Conta   : 0054321
Titular: Maria dos Santos
Saldo   : R$ 500,00
=====
```

Outro exemplo

```
=====
Conta   : 0012345
Titular: Joao da Silva
Saldo   : R$ 0,00
=====
```

```
=====
Conta   : 0054321
Titular: Maria dos Santos
Saldo   : R$ 500,00
=====
```

```
=====
Valor para deposito em c1: 1000
```

```
=====
Conta   : 0012345
Titular: Joao da Silva
Saldo   : R$ 1000,00
=====
```

```
=====
Valor de saque em c2: 100
```

```
=====
Conta   : 0054321
Titular: Maria dos Santos
Saldo   : R$ 400,00
=====
```

Discussão

- Precisamos dos métodos públicos:
 - Set/ get saldo?
 - Set/ get titular?
 - Set/ get número da conta?
- Sugestão:
 - Para saldo e número da conta, codifique os métodos ***get***
 - Para titular podemos ter ***set*** e ***get***