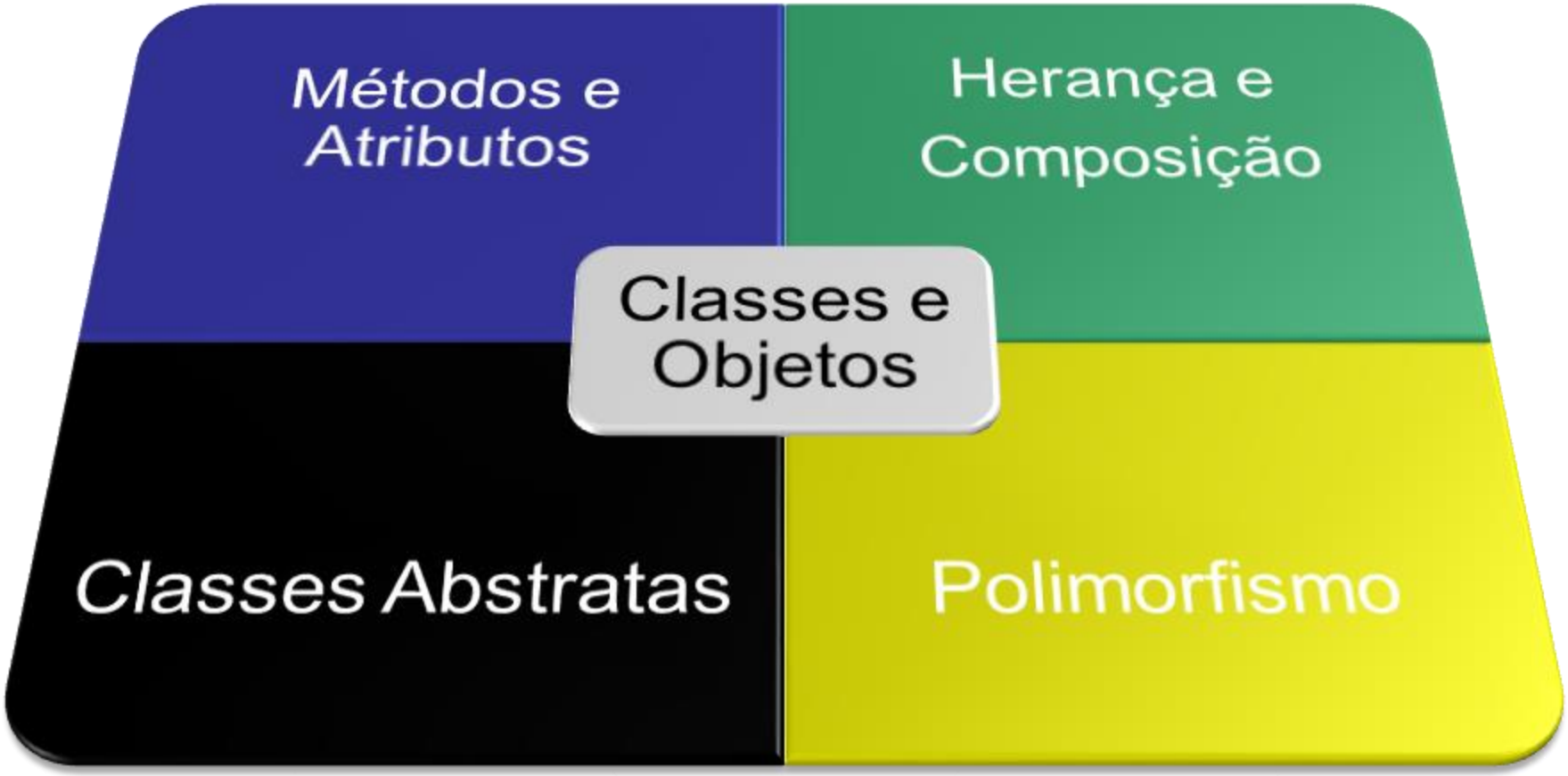
PROGRAMAÇÃO ORIENTADA A OBJETOS

Prof. Milton Palmeira Santana

Classes e objetos para programação orientada a objetos



Classes e objetos para programação orientada a objetos

- Orientação a objetos é um paradigma que ajuda na organização do código.

- Alguns problemas do paradigma procedural:
  - Vamos supor que temos 100 formulários e nesses formulários devemos ter uma função para fazer a validação de um cpf. Você é obrigado a sempre chamar esse método? Não.
  - 4 programadores. Todos devem conhecer.
  - E se entrar um novo desenvolvedor?
  - Novo método para implementar em todos os formulários.
  - Mundo ideal? Responsabilidade em só local ou um só programador.

Classes e objetos para programação orientada a objetos

- Orientação a Objetos vai te ajudar em muito a se organizar e escrever menos, além de concentrar as responsabilidades nos pontos certos.

- Se entendemos que modelos são representações simplificadas de um objeto ou de um processo, entre outras coisas, então devemos também entender que as classes são implementações desses modelos em uma linguagem orientada a objetos.

- Classes são estruturas das linguagens de POO criadas para conter os dados que devem ser representados e as operações que devem ser efetuadas com esses dados.

Classes e objetos para programação orientada a objetos

- Regras para criação de classes.
  - **Nome da classe**: início com letra maiúscula, sem acento e sem espaço entre palavras. Declarada com a palavra reservada class, seguida do nome da classe. O corpo da classe é delimitado por { e }.
  - **Nome de métodos e atributos**: início com letra minúscula; acento permitido, porém pouco recomendado; sem espaço entre palavras.

Classes e objetos para programação orientada a objetos

- Uma classe é uma “fábrica” para produzir objetos

- Podemos definir uma classe como o projeto de um determinado assunto que podemos criar a partir de analogias com os mesmos no mundo real.
  - Ex: Carro, Conta, Homo Sapiens.

Classes e objetos para programação orientada a objetos

- Confuso? Vamos verificar alguns exemplos:
  - Uma receita de bolo.
  - A planta de uma casa.
    - Podemos morar dentro da planta de uma casa?
    - Precisamos criar instâncias, ou seja, a própria casa para podermos morar nela ou pintar uma parede, por exemplo.

Classes e objetos para programação orientada a objetos

- Modificadores de acesso são utilizados para definir níveis de acesso a membros das classes

Declaração	Definição
public	Acesso ilimitado
private	Acesso limitado à classe e seus membros
protected	Acesso limitado à classe, seus membros e a tipos derivados da mesma

Classes e objetos para programação orientada a objetos

- Objetos são gerados a partir de classes
- Uma classe define as propriedades e o comportamento dos objetos gerados por ela
- Todo objeto é uma instância de uma classe

Classes e objetos para programação orientada a objetos

- Vamos elaborar um programa para um banco e criar uma entidade Conta.

- Para criá-la devemos primeiramente identificar o que ela deverá conter, ou seja, quais são suas características. Ex Conta:
  - Número da conta
  - Dono da conta
  - Saldo
  - limite

Anhanguera

Classes e objetos para programação orientada a objetos

» Definindo uma classe e seus atributos

```
class Conta {
  string numero;
  string donoDaConta;
  double saldo;
  double limite;
}
```

» Instanciando uma classe

```
Conta minhaConta = new Conta();
```

Anhanguera

Classes e objetos para programação orientada a objetos

- Para acessar um atributo utilizamos o seguinte código:
  - minhaConta.numero = “123456”;
  - minhaConta.donoDaConta = “Milton”;
  - minhaConta.saldo = 100000.00;
  - minhaConta.limite = 1000.00;

Anhanguera

Classes e objetos para programação orientada a objetos

- Métodos representam as operações associadas à classe

```
public void saque(double valor) {
     this.saldo -= valor;
}
```

- Chamando um método

```
minhaConta.saque(10000);
```

Métodos e parâmetros

- Os métodos podem ou não conter parâmetros. No exemplo do método saca temos o parâmetro do tipo **double valor**. Essa é uma variável comum chamada de temporária pois ao final da execução deixa de existir. Quando queremos sacar precisamos informar o valor do saque, portanto, temos que passar para o programa esse valor. É através desse parâmetro que fazemos isso.

- Até agora vimos os métodos e, neles, temos algum tipo de retorno.
- Todo método deve definir o que retorna, nem que defina que não há retorno.
- Ex:

```
public void deposita (double valor) {
    this.saldo += valor;
}
```

Métodos com retorno

- No exemplo utilizamos o void que significa que o método não terá retorno.

- Temos um problema com o método saca: E se tentarmos sacar um valor maior do que o saldo? Podemos usar um retorno para informar se conseguimos ou não efetuar o saque.

```
public boolean saca(double valor) {
        if (this.saldo < valor)  {
            return false;
        }
        else {
            this.saldo -= valor;
            return true;
        }
    }
```

Métodos com retorno

» No exemplo anterior temos um retorno do tipo boolean: true ou false. Agora no código principal:

```
if (c.saca(valor))
{
     System.out.println(“Saque efetuado");
}
else
{
    System.out.println(“Saque não efetuado.");
}
```

Métodos com retorno

» Outro exemplo: Podemos ter um método para efetuar um cálculo.

```
public int calculo (int valor1, int valor2) {
    int resultado;
    resultado = valor1 + valor2;
    return resultado;
}
```

E no código principal:

```
int resultado = c.calculo(valor1, valor2);
System.out.println("Resultado: " + resultado);
```

Métodos com retorno

- Vamos supor que precisamos adicionar mais dados a nossa Conta, como por exemplo, adicionar o nome, cpf e endereço do cliente.

- Se pensarmos bem, uma Conta não tem todos esses atributos. Quem tem esses atributos é um Cliente. Então o que iremos fazer? Criaremos uma nova classe com o nome de Cliente.

```
public class Cliente
{
    public String nome;
    public String cpf;
    public String endereco;
}
```

- E como fica a classe conta?

Métodos com retorno

```
public class Conta ()
    {
    public String numero;
    public double saldo;
    public double limite;
    public Cliente donoDaConta;
    }
```

Métodos com retorno

- E no código principal:

```
Conta minhaConta = new Conta();
Cliente cliente = new Cliente();
minhaConta.donoDaConta = cliente;
```

- Dessa forma estamos apenas criando um novo cliente e dizendo que o atributo donoDaConta contém o endereço para esse objeto cliente.

Objetos são acessados por referência

- E se forem criados dois ou mais objetos de conta?

Exercícios

- **1)** Elabore um programa que implemente todos os métodos que foram utilizados no projeto da classe Conta. Implemente as regras de validações dos exemplos.

- **2)** Elabore um programa que simule uma calculadora. Crie uma classe com 4 métodos: somar, subtrair, multiplicar, dividir.

- **3)** Faça um programa de agenda telefônica, com as classes Agenda e Contato.


Anhanguera