Programação Orientada a Objetos

Modificadores

Professor Me.: Gustavo Siqueira Vinhal

Modificadores

- Modificadores são palavras chaves acrescentadas a definições de classes, métodos e atributos para alterar o significado:

  - Controle de Classe, Método ou Atributo:
    - public, protected, private.
  - Criação de métodos e atributos:
    - static.
  - Finalização:
    - final.
  - Criar classes e métodos abstratos:
    - abstract.

Controle de acesso

- Modificadores mais utilizados são para controle de Classe, Método ou Atributo:
    - public, protected, private.

- Definem quais atributos e métodos de uma classe são visíveis a outras classes.

- Permite o **encapsulamento**:
    - Processo que impede que variáveis de uma classe sejam lidas ou modificadas por outras classes. A única maneira de modificar é utilizando métodos da classe, se disponíveis.

Controle de acesso - default

- O acesso padrão é denominado **default**.

- Neste caso, não há necessidade de colocar nenhum modificador na declaração do método ou atributo.

- Quando utilizado desta maneira, o método/atributo estão disponíveis para todas as outras classes.

```
String versao = "0.7a";

boolean processOrder()
{
    return true;
}
```

Controle de acesso - private

- O acesso privado é denominado **private**.

- Esse modificar impede que um método/atributo seja acessado por quaisquer outras classes que não seja a própria classe.

- **Cuidado!** Essa restrição afeta também a herança! As subclasses não herdam os métodos/atributos privados.

```
private String versao = "0.7a";

private boolean processOrder()
{
    return true;
}
```

Controle de acesso - public

- O acesso público é denominado **public**.

- Esse modificar permite que um método/atributo seja acessado por quaisquer outras classes que não seja a própria classe.

```
public String versao = "0.7a";

public boolean processOrder()
{
    return true;
}

public static void main(String[] args)
{

}
```

Controle de acesso - protected

- O acesso protegido é denominado **protected**.

- Esse modificar permite que um método/atributo seja acessado:
  - Por quaisquer subclasses; ou
  - Classes dentro de um mesmo pacote.

- É um nível intermediário entre o privado e o público.

```
protected String versao = "0.7a";

protected boolean processOrder()
{
    return true;
}
```

Controle de acesso

Visibilidade	Public	Protected	Default	Private
Da mesma classe	Sim	Sim	Sim	Sim
De qualquer classe dentro do mesmo pacote	Sim	Sim	Sim	Não
De qualquer classe fora do pacote	Sim	Não	Não	Não
De uma subclasse no mesmo pacote	Sim	Sim	Sim	Não

Controle de acesso e herança

- Quando uma subclasse é criada (herdada) os métodos dela não podem ter um nível de acesso mais restrito do que os da superclasse (classe pai).

- Algumas regras:

  - Métodos declarados como **public** em uma superclasse também precisam ser public nas subclasses.
  - Métodos declarados como **protected** em uma superclasse precisam ser **protected** ou **public** nas subclasses; não podem ser **private**.
  - Métodos sem controle de acesso, podem ser declarados mais privados nas subclasses.

- Métodos declarados como **private** não são herdados, logo tais regras não se aplicam.

Métodos acessadores

- Quando um atributo é privado ele não pode ser acessado por métodos de outras classes, diretamente.

- Se um objeto de outra classe desejar acessar atributos privados usam-se métodos acessadores: **get** e **set**.

- Tais métodos devem ser implementados de tal forma a alterar os valores dos atributos privados.

Classes, métodos e atributos finais

- O modificados final indica que não serão alterados:
  - Uma classe final não pode ser subclassificada;
  - Um método final não pode ser redefinido por quaisquer subclasses;
  - Um atributo final não pode mudar de valor.

- Todos os métodos de uma classe final são automaticamente finais.

Classes e métodos abstratos

- Em uma hierarquia de classes, quanto maior a classe, mais abstrata é sua definição.

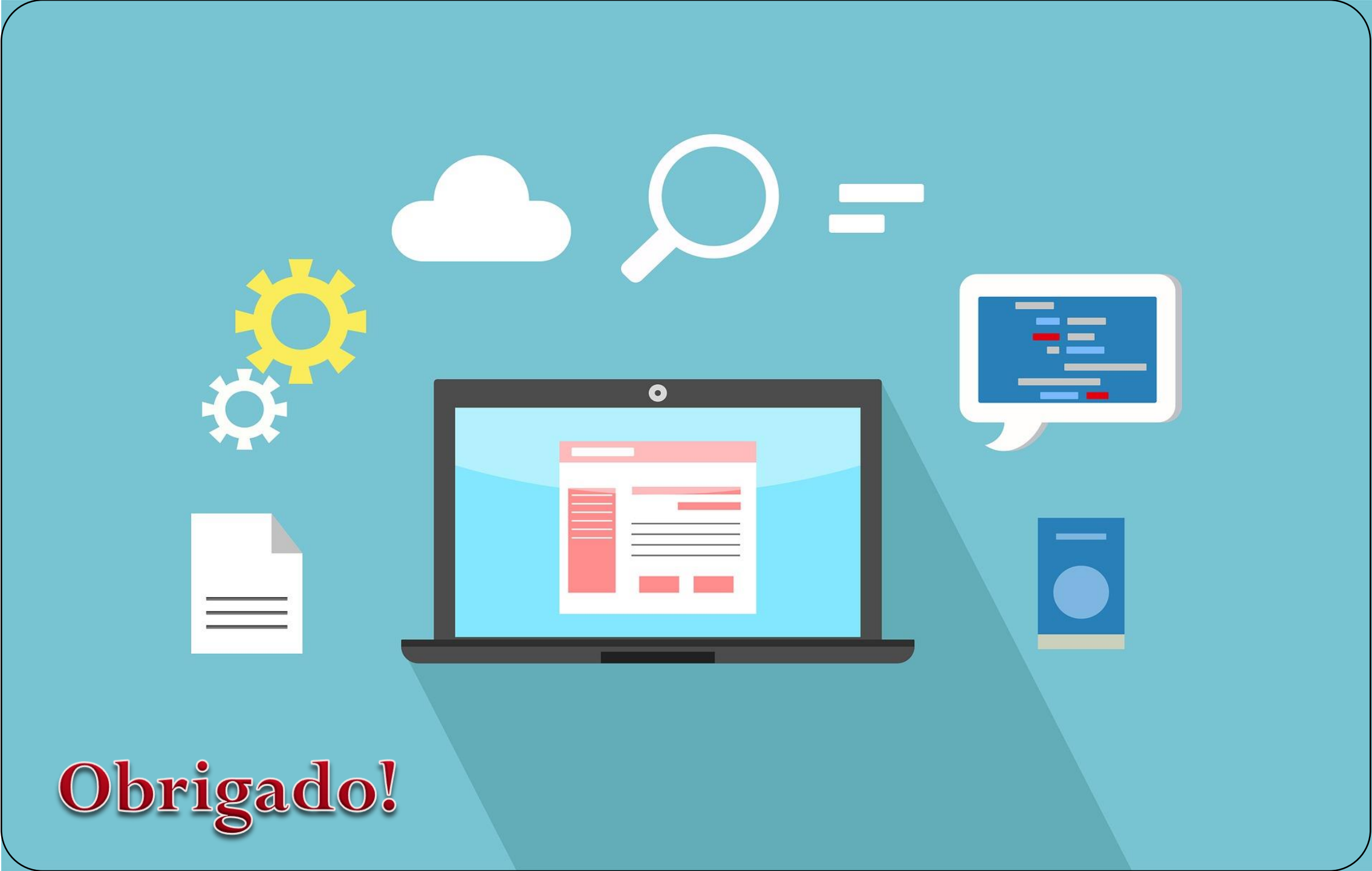
- Se uma classe possui apenas as definições de comportamento e atributos comuns, sem precisar ser instanciada, ela pode ser abstrata.

- Para criar classes abstratas utiliza-se o modificador **abstract.**

```
public abstract class Palette
{

}
```

- Métodos abstratos não podem existir fora de classes abstratas.


Obrigado!