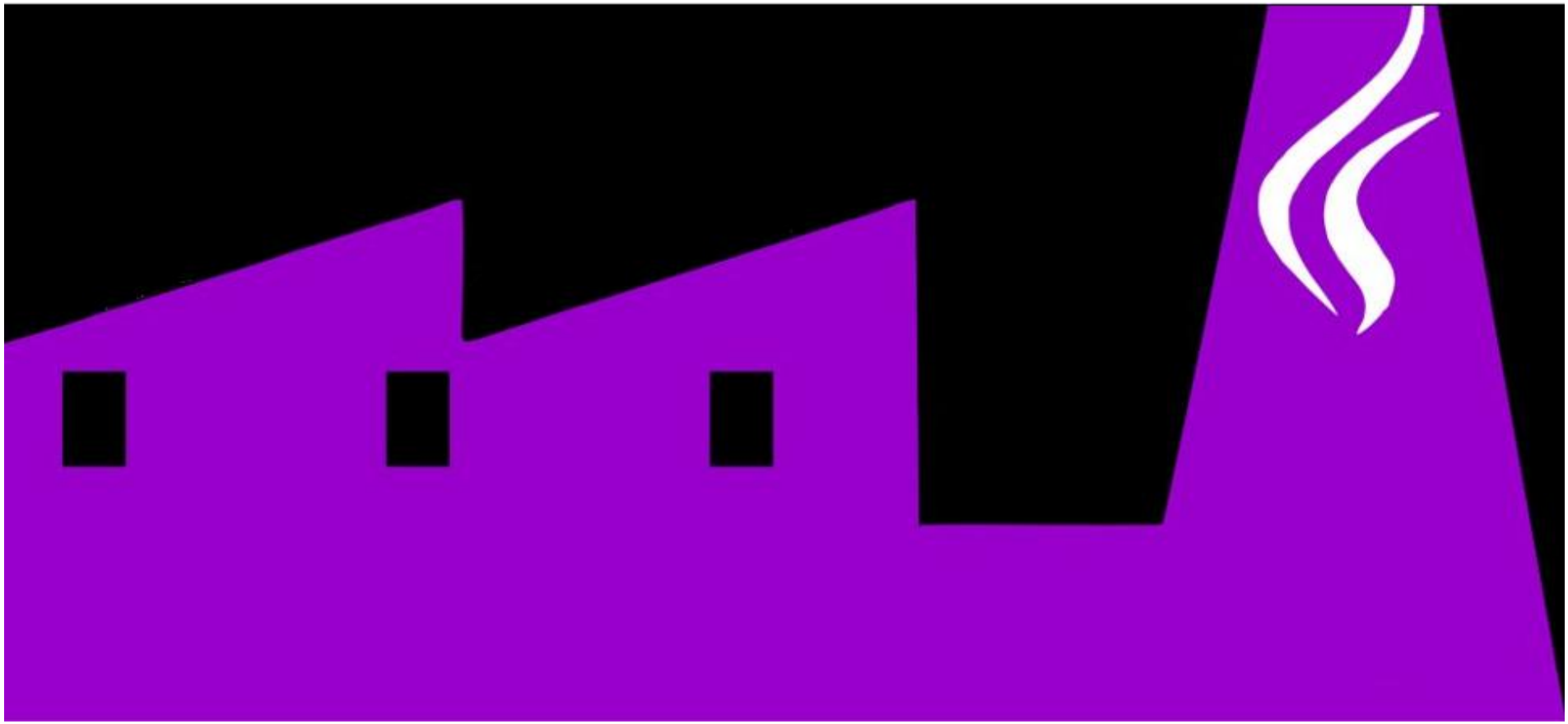
Fábrica de Software

Prof[es.] Ivan L. Süptitz e Daniela Bagatini

Visibilidade e Encapsulamento



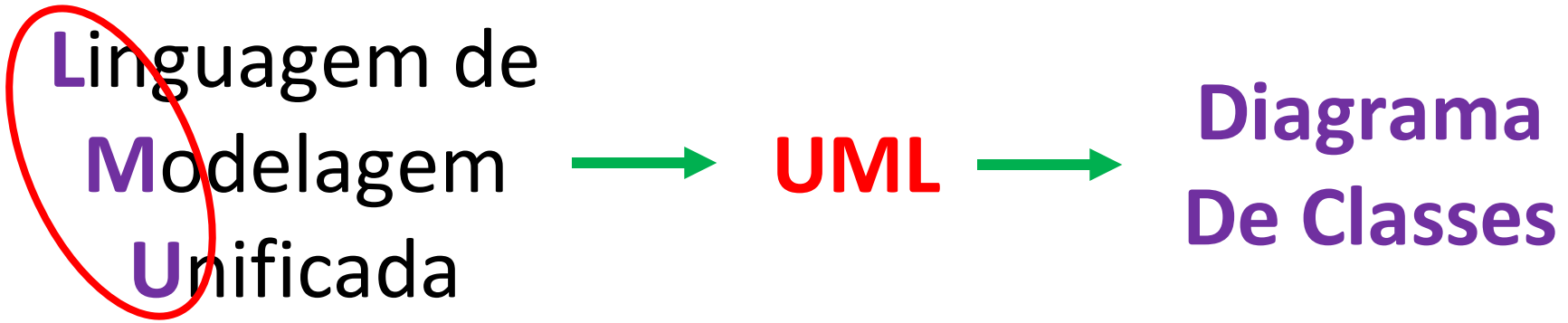
Sumário – O que veremos nesta aula?

- Notação UML – diagrama de classes
- O que é visibilidade em objetos
- Modificadores de visibilidade: + público; - privado; # protegido
- Construtores
- Getters e Setters
- Exercícios.

O que é visibilidade de um objeto?

Visibilidade é uma essência de POO. é utilizada para indicar o nível de acessibilidade de um determinado atributo ou método.

UML – o que é?



UML – Diagrama de classes

Simplificar a representatividade das classes!

Caneta
modelo cor ponta carga tampada
escrever( ) rabiscar ( ) pintar ( ) tampar ( ) destampar ( )

Nome da Classe

Atributos

Métodos

Modificadores de Visibilidade

*Indicam o nível de acesso aos componentes de uma classe.

Quais componentes?

Atributos e métodos!

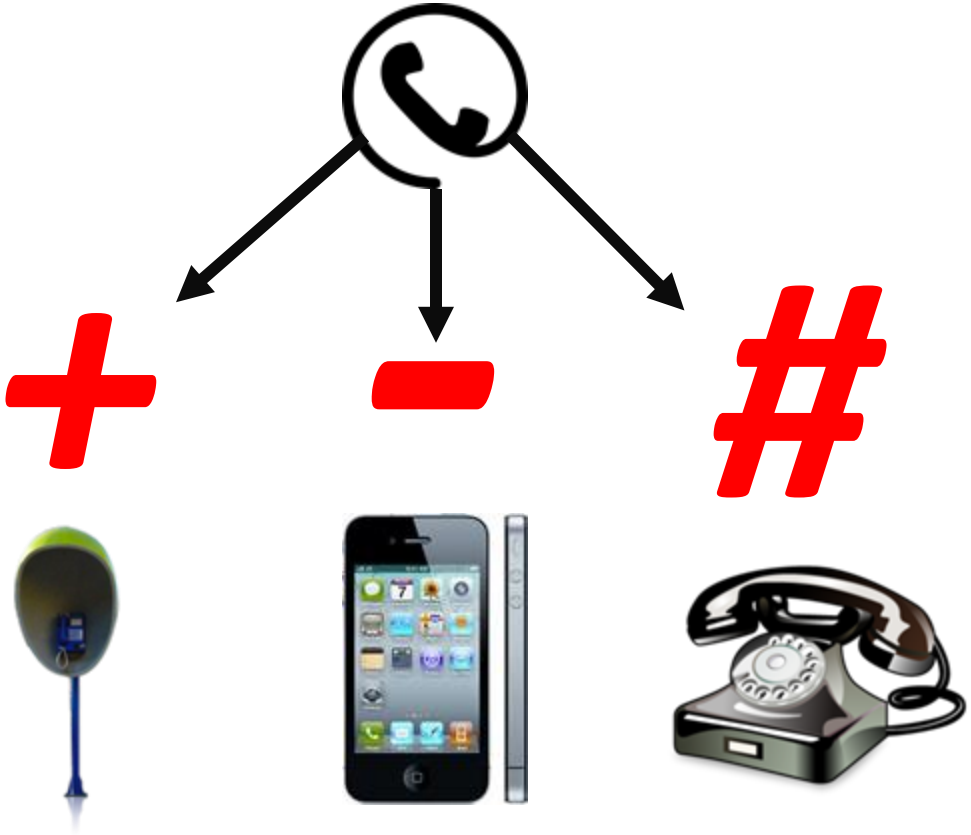
Três símbolos

\+ público

\- privado

\# protegido

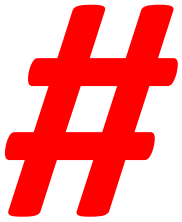
Modificadores de Visibilidade



Definição conforme POO

Público : a classe atual e todas as outras classes.

Privado : somente a classe atual.



Protegido : a classe atual e todas as suas subclasses.

Modificadores de Visibilidade


```
Classe Caneta
    publico     modelo: Caractere
    publico     cor : Caractere
    privado     ponta : Real
    protegido   carga : Inteiro
    privado     tampada : Logico

    publico     Metodo escrever ( )

                FimMetodo
    privado     Metodo rabisar ( )

                FimMetodo
FimClasse
```

Voltando a IDE ...

1 – Acrescentando visibilidade.

1.1 – Verificar a diferença entre publico e privado.

1.2 – Acessar o atributo no método main( ) com opção de publico.

1.3 – Alterar atributos para privado. Como acessar?

Encapsulamento

- Não consigo mais acessar o atributo privado *tampada*: **ÓTIMO**. Ele é um estado da Caneta que só deve ser modificado pelos métodos apropriados.

- Mas preciso atribuir valor ao atributo ponta e não consigo!!!
- Neste caso é necessário criar métodos para atribuírem valores aos atributos.
- Isso se chama **ENCAPSULAMENTO**!

Encapsulamento

- A ideia é encapsular, isto é, **esconder** todos os membros de uma classe, além de esconder como funcionam as rotinas(no caso métodos) do nosso sistema.
- Encapsular é fundamental para que seu sistema seja suscetível a mudanças: não precisaremos mudar uma regra de negócio em vários lugares, mas sim **em apenas um único lugar**, já que essa regra está encapsulada.

- **Como???**

Getters e Setters

- O modificador private faz com que ninguém "*de fora*" consiga modificar, nem mesmo ler, o atributo em questão.

- Sempre que precisamos arrumar uma maneira de fazer alguma coisa com um objeto, utilizamos de **métodos**!

- Vamos então criar um método chamado

`void setPonta(float p)` para realizar essa simples tarefa.

- Se precisarmos também ler o atributo fora do objeto, criamos um método `float getPonta()`

Construtores

- Quando usamos a palavra chave **new**, estamos construindo um objeto. Sempre quando o new é chamado, ele **executa o construtor da classe**. O construtor da classe é um bloco declarado com o **mesmo nome que a classe**:

Construtores

```
public class Caneta{

    //Construtor
    public Caneta(){
        System.out.println("Construindo a caneta");
    }
}

//então quando fazemos
Caneta c = new Caneta();
//a mensagem 'Construindo a caneta' aparecerá
```

Construtores

```
public class Caneta{

    //Construtor
    public Caneta(String cor, float ponta){
        this.cor = cor;
        this.ponta = ponta;
    }
}
```

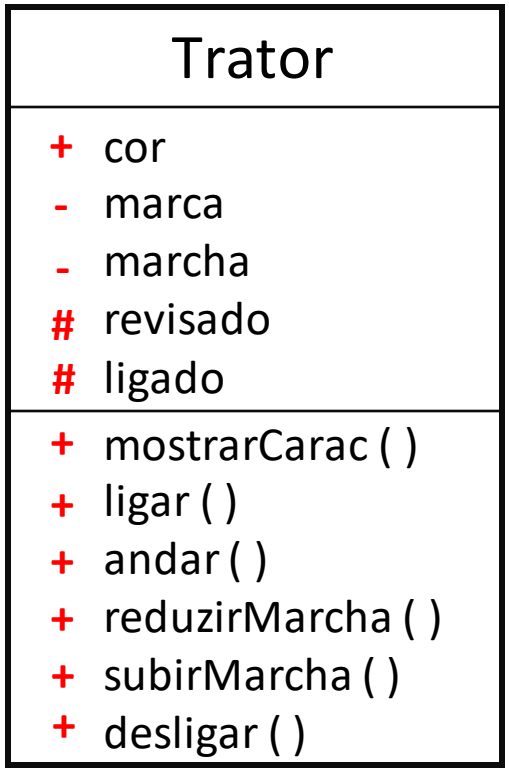
- Podemos criar mais de um construtor numa classe, inclusive passando valores na criação dela!!!!
- Estes parâmetros são para inicializar os atributos.

Agora vocês ...

Atividade J.3.1 – Novo projeto. Criar uma classe para representar o objeto Carro. Definir ao menos 3 atributos e 2 métodos.

Criar um vetor de n carros (onde n é informado pelo usuário), pedir para o usuário os seus atributos em um laço e chamar ao menos um método.

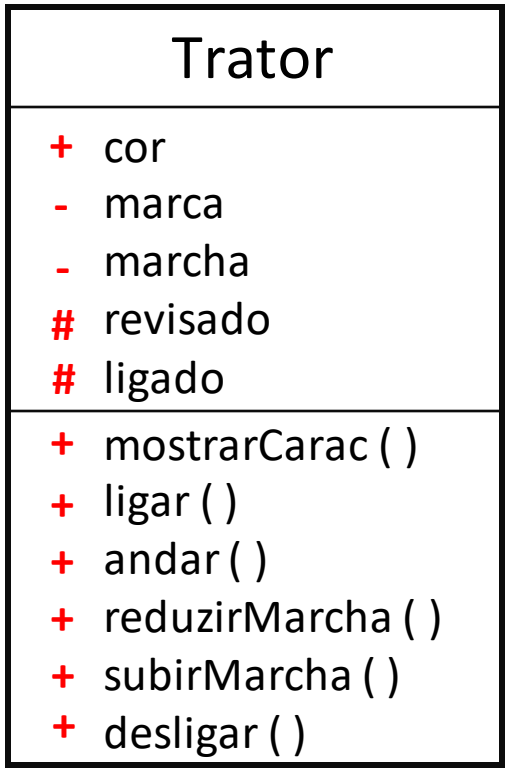
Vamos praticar ...



Atividade J.3.2 – Implementar o seguinte diagrama de classes (criar 3 objetos).

- mostrarCarac ( ) – printar os atributos do trator.
- ligar ( ) – ligar o trator, lembrar de verificar se ele já foi ligado antes.
- andar ( ) – colocar o trator em movimento, verificar se ele esta ligado.
- reduzirMarcha ( ) – reduz a marcha do trator. O trator possui 3 marchas.
- subirMarcha ( ) – engatar a próxima marcha do trator. Avisar quando chegar ao limite.
- desligar ( ) – desligar trator. Lembrar de verificar se o trator esta ligado.

Vamos praticar ...



Atividade J.3.3 – Acrescentar ao programa da atividade anterior: permitir selecionar os métodos pelo teclado (considerar apenas um trator).

- mostrarCarac ( ) – printar os atributos do trator.
- ligar ( ) – ligar o trator, lembrar de verificar se ele já foi ligado antes.
- andar ( ) – colocar o trator em movimento, verificar se ele está ligado.
- reduzirMarcha ( ) – reduz a marcha do trator. O trator possui 3 marchas.
- subirMarcha ( ) – engatar a próxima marcha do trator. Avisar quando chegar ao limite.
- desligar ( ) – desligar trator. Lembrar de verificar se o trator está ligado.

Dicas

- Como aprender programação?
  - Estude outros códigos-fonte
  - Faça todos os exercícios
    - Do início até o final

- A universidade é uma '*simulação*' do ambiente empresarial ou acadêmico;

- Estudamos **fundamentos** e exercitamos a capacidade de **resolver** quaisquer problemas do mundo real.

Referências

- Documentação oficial Java: https://docs.oracle.com/javase/tutorial/java/index.html

- DEITEL, Paul J.; DEITEL, Harvey M. Java: como programar