CENTRO UNIVERSITÁRIO DE BELO HORIZONTE - UNIBH

Modelagem de Software

Samara Soares Leal - samara.leal@prof.unibh.br

OBJETIVOS

MODELAGEM DE PROBLEMAS

Modelagem dos requisitos do software a partir de diagramas da UML e user stories

ANÁLISE E PROJETO ORIENTADO A OBJETOS

Mapeamento dos relacionamentos entre as classe do software orientado a objetos para o desenvolvimento do projeto (Apoio à codificação).

MODELAGEM DE BANCO DE DADOS

Modelo entidade e relacionamento.
Modelo relacional e normalização.
Modelo lógico e físico de banco de dados

unibh

MODELANDO O PROBLEMA A PARTIR DO PARADIGMA DA ORIENTAÇÃO A OBJETOS

- ANÁLISE E PROJETO ORIENTADO A OBJETOS
- DIAGRAMA DE CLASSES

ORIENTAÇÃO A OBJETOS

É um paradigma (forma de pensar) que representa coisas do mundo real a partir de classes e objetos dessas classes que se comunicam para realizar as funcionalidades de usuário do sistema.

Objetivo: Gerenciar a complexidade do mundo real abstraindo o conhecimento relevante e o mapeamento em objetos no mundo computacional

Vantagens:

- Reutilização e organização de código;
- Maior nível de abstração;
- Facilidade de manutenção do código.

unibh

ORIENTAÇÃO A OBJETOS

Elementos Básicos

Classe:

- Coleção de objetos com características similares;
- Tipo abstrato de dados (molde);
- Possui **atributos** e **métodos** que operam sob esses atributos.

Objeto:

- Entidade física ou conceitual do mundo real;
- Instância (um exemplar) de uma classe;
- Possui características próprias (atributos) e executa determinadas ações (métodos).

unibh

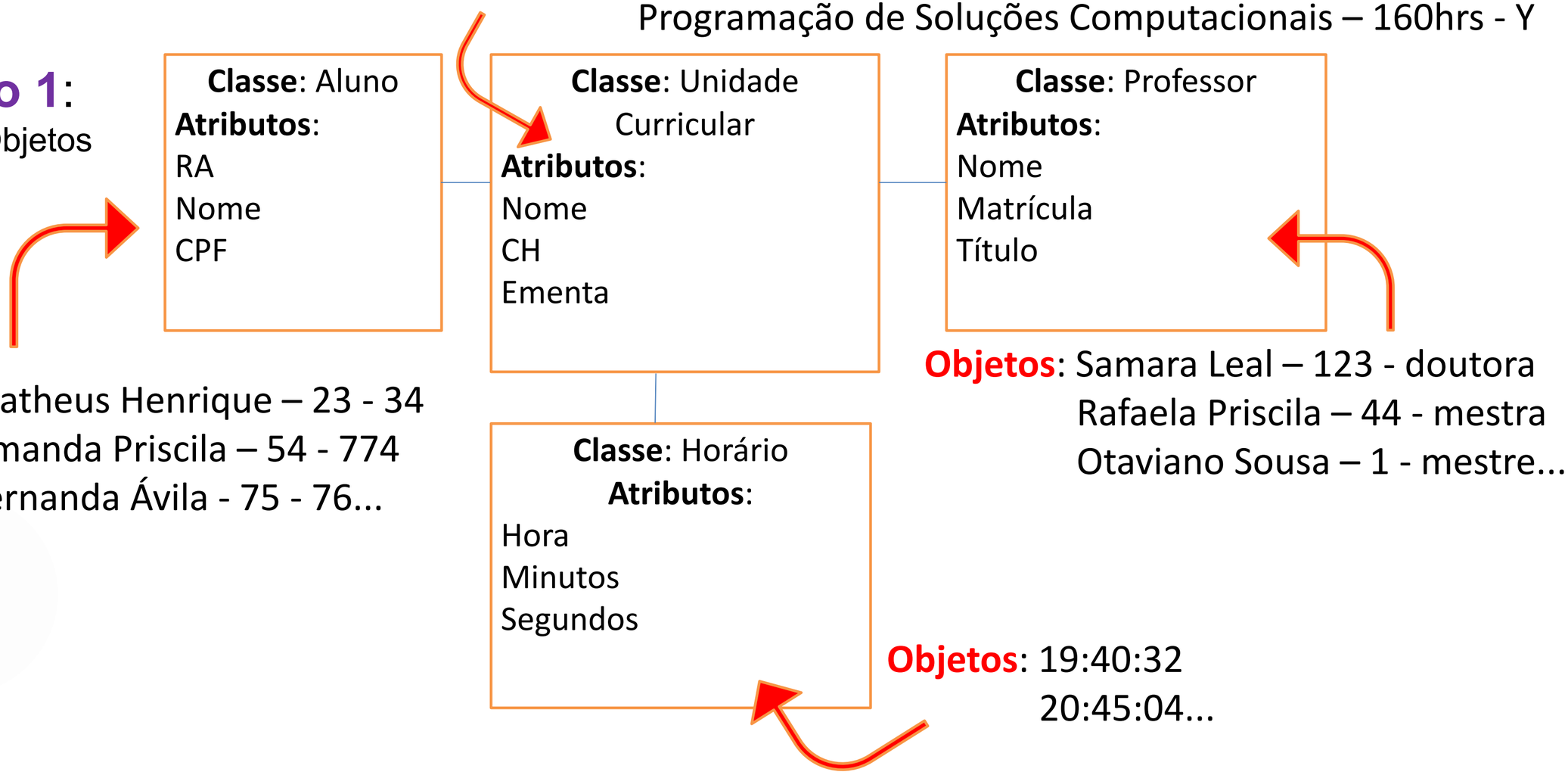
ORIENTAÇÃO A OBJETOS

Exemplo 1: **Requisitos funcionais para o gerenciamento de matrículas de alunos em disciplinas.**

- O aluno deve selecionar as disciplinas que deseja cursar
- O aluno visualiza o horário em que cada disciplina será ministrada
- O aluno visualiza o professor que irá ministrar a disciplina selecionada

ORIENTAÇÃO A OBJETOS

Exemplo 1:
Classes e Objetos



PENSANDO ORIENTADO A OBJETOS

PRINCÍPIOS DA ORIENTAÇÃO A OBJETOS

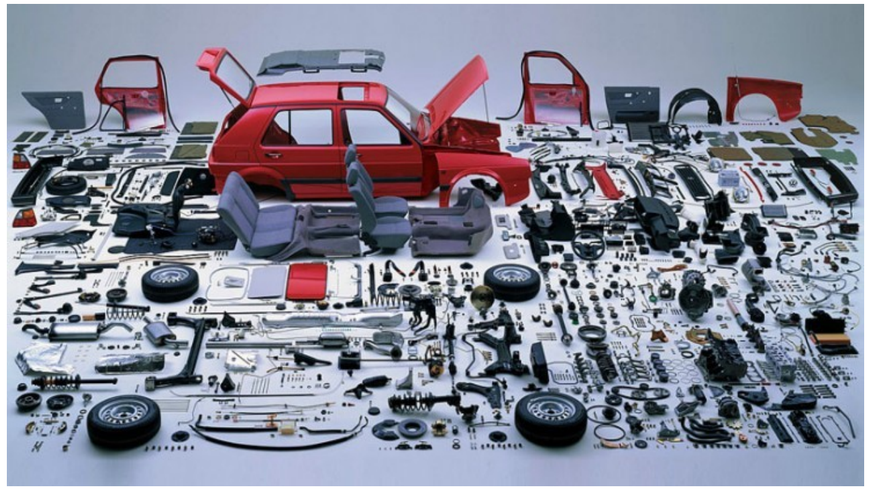
1. ABSTRAÇÃO

É a representação de objetos reais

Representação de objetos do mundo real que possuem:

- **Identidade**: necessita de um nome para identificar o objeto.
- **Propriedades**: características do objeto.
- **Métodos**: ações que o objeto executará.

Humanos gerenciam a complexidade através de abstração.



2. ENCAPSULAMENTO

Oculta os detalhes internos dos métodos de uma classe.

Uma classe pode ser visualizada de duas formas:

- **Interface**: pode ser vista e usada por outros objetos.
- **Implementação**: é escondida do objeto.

Exemplo: Frear um carro.

2. ENCAPSULAMENTO

Oculta os detalhes internos dos métodos de uma classe.

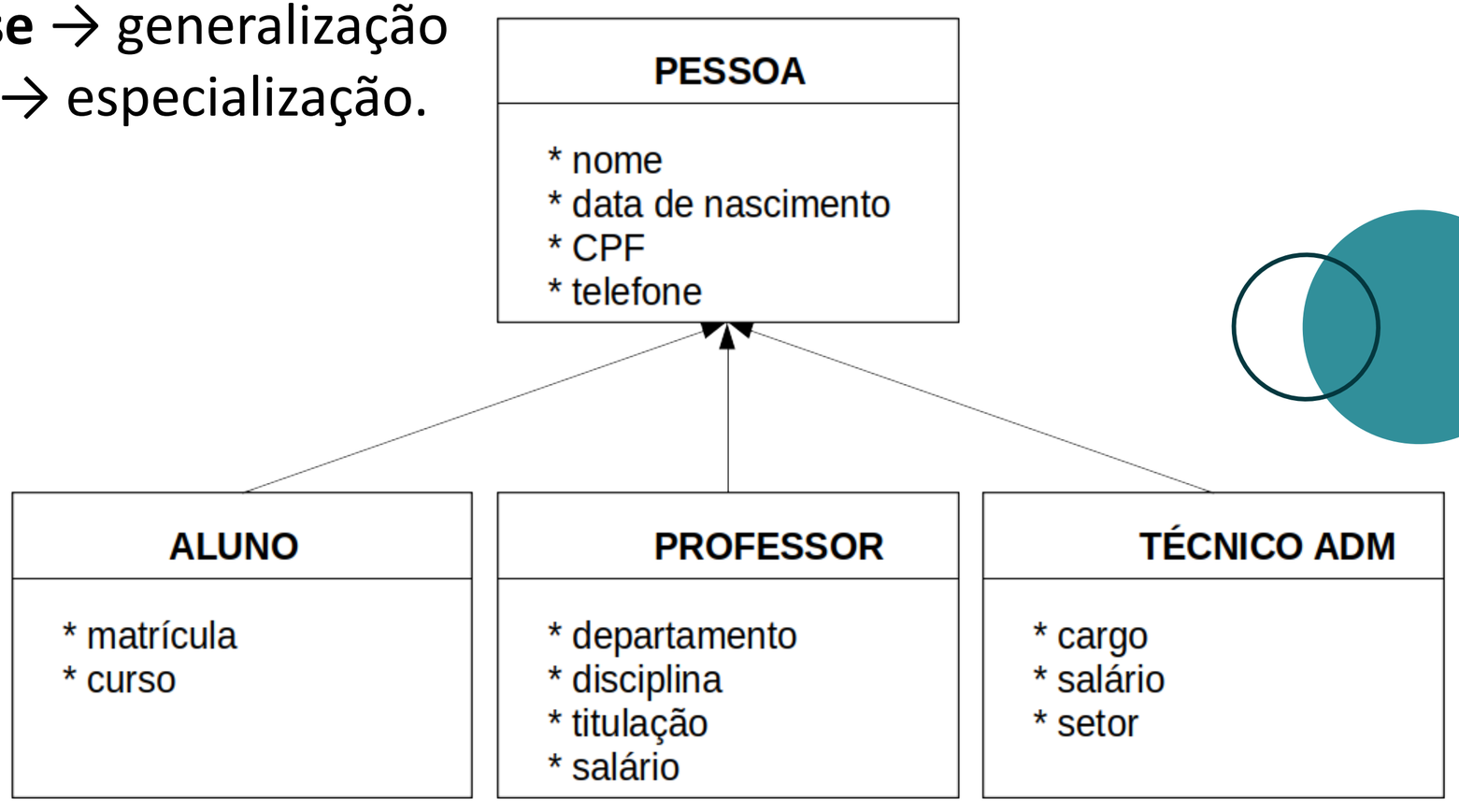
Modificadores de acesso: definem como um membro pode ser acessado.

- **Público (*public*)**: os métodos ou atributos são acessíveis por qualquer classe;
- **Privado (*private*):** os métodos ou atributos são acessíveis apenas pela própria classe.
- **Protegido (*protected*):** os métodos ou atributos são acessíveis pela própria classe, classes do mesmo pacote ou classes da mesma hierarquia.

3. HERANÇA

Classes filhas herdam atributos e métodos da classe mãe

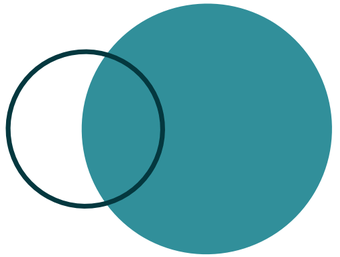
- A subclasse (classe filha) “herda” as funcionalidades da superclasse (classe mãe) e adiciona novos aspectos.
- **Superclasse** → generalização
- **Subclasse** → especialização.



4. POLIMORFISMO

Poli = muitas + Morfos = formas

- Técnica que permite a criação de múltiplas operações (**métodos**) com a capacidade de operar sobre valores distintos (várias formas).
- **Assinatura de métodos**: É a identificação do método (**nome + quantidade de parâmetros + tipo de parâmetros**).
- O retorno do método não faz parte da assinatura.
- Métodos com a mesma assinatura são **sobrescritos**. Caso contrário, são **sobrecarregados**.



4. POLIMORFISMO

Poli = muitas + Morfos = formas

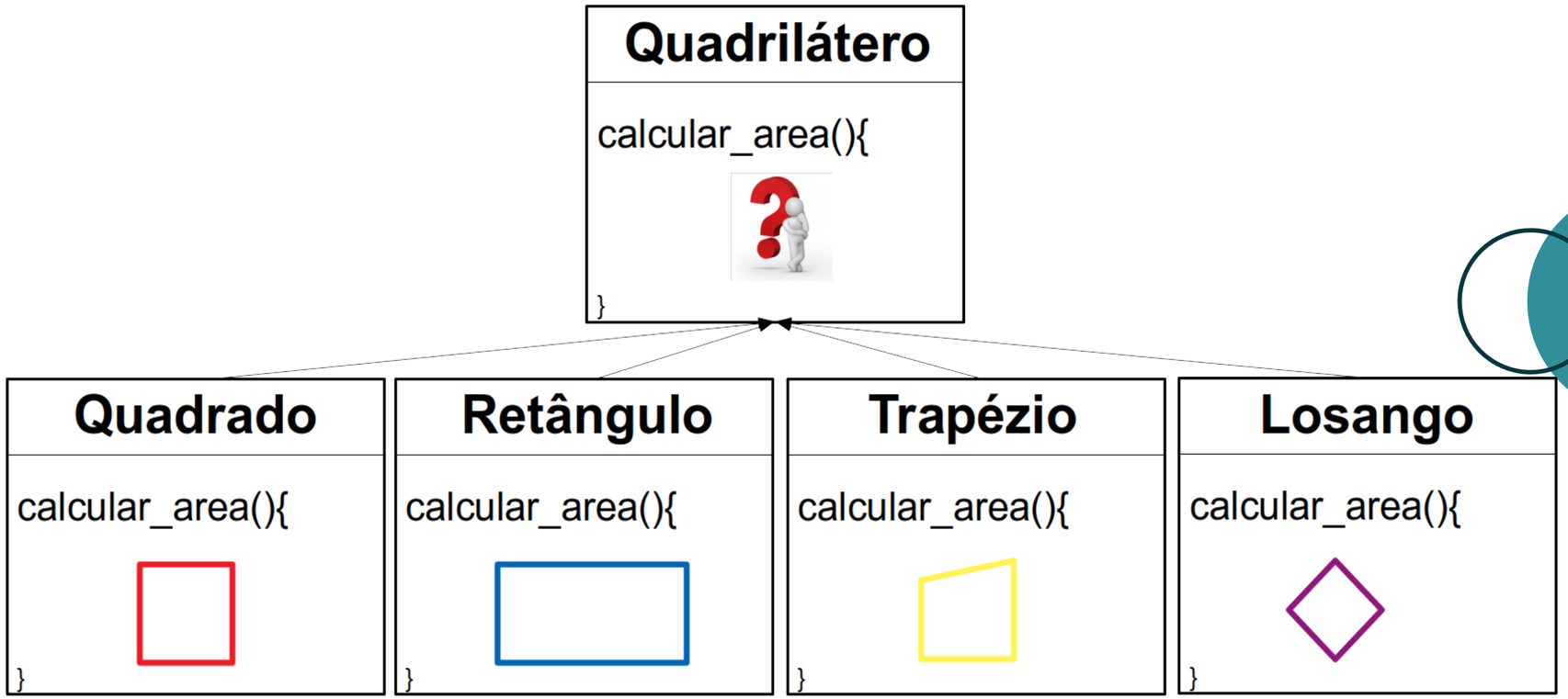
- **Sobrecarga**: permite criar, dentro da mesma classe, métodos com o mesmo nome, mas com parâmetros diferentes.

```
soma(int a, int b);

soma(double a, double b);

soma(string a, string b);
```

4. POLIMORFISMO

Poli = muitas + Morfos = formas

- **Sobrescrita**: permite reescrever na subclasse os métodos da superclasse, alterando o seu comportamento.



DÚVIDAS?

ENTRE EM CONTATO: *samara.leal@prof.unibh.br*

NÃO SE ESQUEÇA DE CONSULTAR O REFERENCIAL BIBLIIOGRÁFICO e MATERIAL COMPLEMENTAR!