Herança e Polimorfismo

PROFESSOR: EDUARDO HABIB BECHELANE MAIA

HABIB@CEFETMG.BR

Herança

Capacidade de compartilhar estruturas comuns entre diversas classes derivadas

Reaproveitamento de código da classe pai

Herança

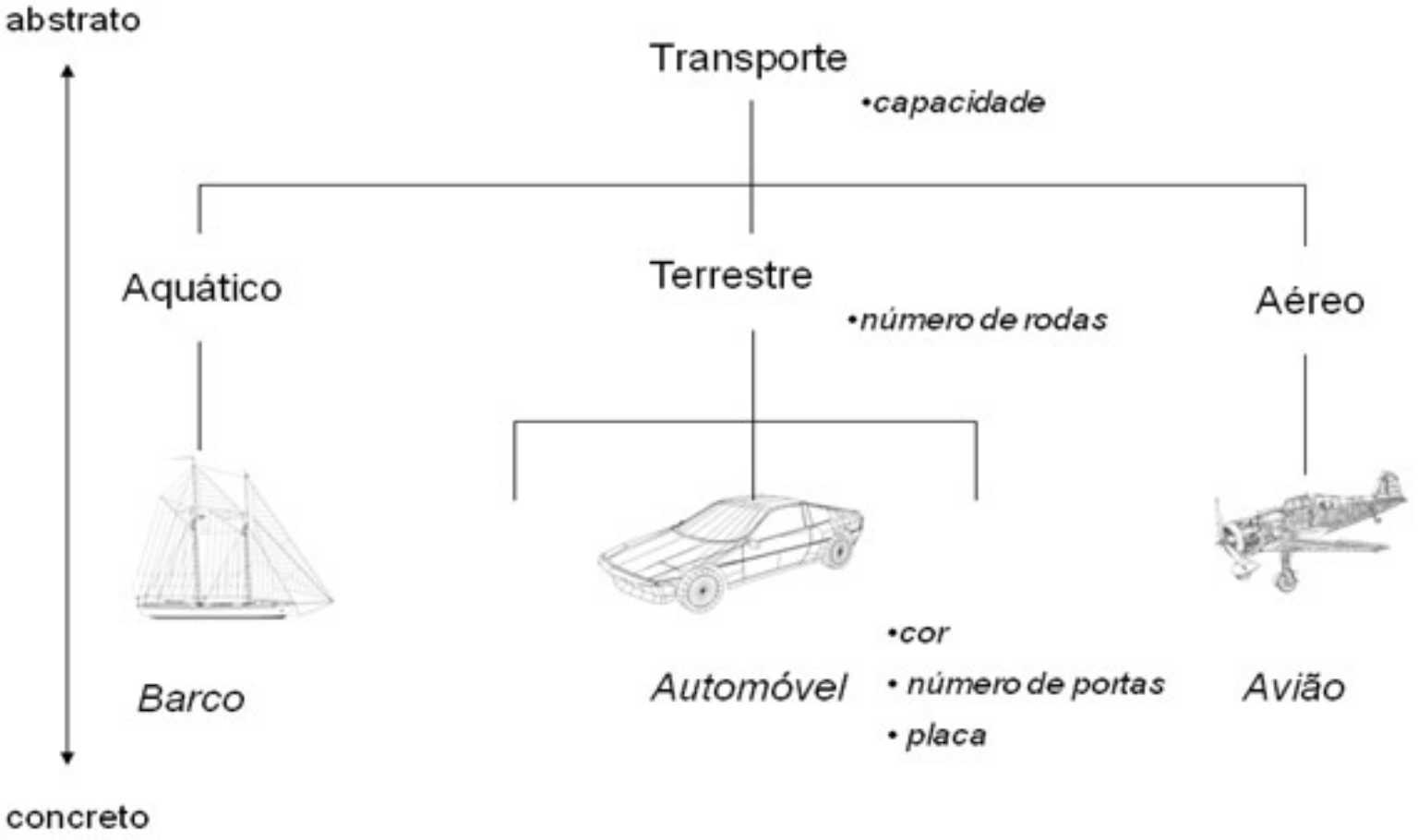
É o processo em que um objeto pode adquirir as características de outro objeto.

Herança Simples: um objeto herda as características de uma única classe.

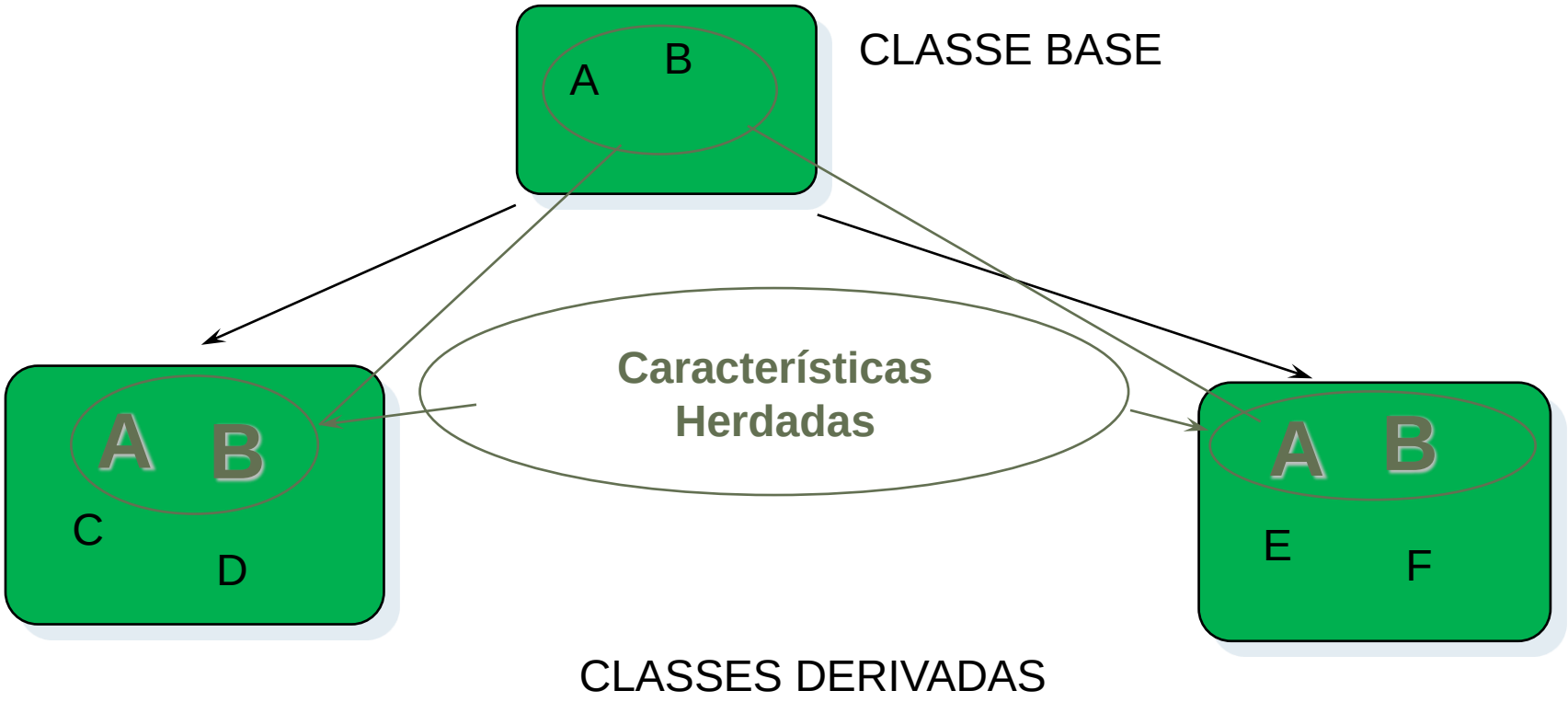
Herança Múltipla: uma objeto herda características de mais de uma classe.

- Algumas linguagens não permitem

Herança


abstrato
Transporte
•capacidade
Aquático
Terrestre
•número de rodas
Aéreo
Barco
Automóvel
•cor
• número de portas
• placa
Avião
concreto

Herança


A
B
CLASSE BASE
Características
Herdadas
A
B
C
D
A
B
E
F
CLASSES DERIVADAS

Herança

Relacionamento entre classes onde uma classe compartilha a estrutura e o comportamento de uma ou mais classes;

Define uma hierarquia de abstrações na qual a **subclasse** herda de uma ou mais **superclasses**:

Herança simples; **Herança múltipla**;

Uma herança é um relacionamento do tipo “é um tipo de”.

Herança

A **subclasse** herda os atributos, operações e relacionamentos da **superclasse**;

Cada subclasse pode definir novos atributos e/ou operações;

Cada subclasse pode redefinir operações da superclasse;

Cada subclasse pode participar de relacionamentos específicos.

Herança

Como identificar necessidade de heranças?

- Procure por similaridades entre as classes;
- Siga a regra: a subclasse é um **tipo** da superclasse;
- Evite herança de implementação, siga a regra;
- A herança **deve** ser total, pela subclasses;

Caso a regra não seja satisfeita, utilize composição.

Herança

A **herança** é uma forma de reuso de *software*

- O programador cria uma classe que absorve os dados e o comportamento de uma classe existente;
- Ainda é possível aprimorá-la com novas capacidades.

Além de reduzir o tempo de desenvolvimento, o reuso de *software* aumenta a probabilidade de eficiência de um *software*

- Componentes já debugados e de qualidade provada contribuem para isto.

Herança

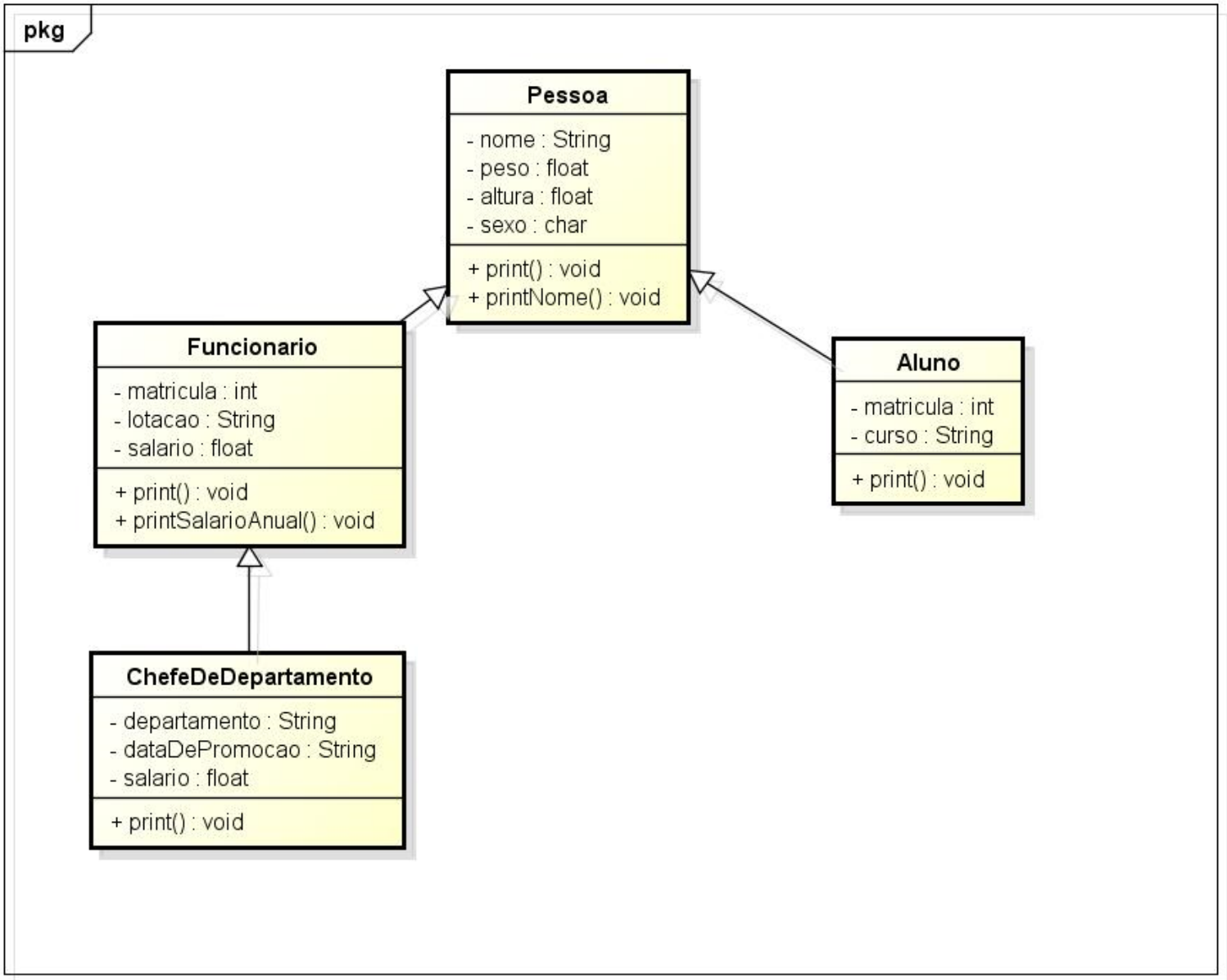
A herança define um relacionamento "**é um**"

- Um carro é um veículo
  - Todas as propriedades de um veículo são propriedades de um carro.
- Um objeto de uma classe derivada pode ser tratado como um objeto da classe base.

Os métodos de uma classe derivada podem necessitar acesso aos métodos e atributos da classe base

- Somente os membros não privados estão disponíveis;
- Ou seja, membros que não devem ser acessíveis através de herança devem ser **privados**;
  - Poderão ser acessíveis por *getters* e *setters* públicos, por exemplo.

Exemplo de herança



Pessoa.hpp

```
1  #ifndef __PESSOA_HPP
2  #define __PESSOA_HPP
3  #include <iostream>
4  using namespace std;
5
6  class Pessoa
7  {
8  private:
9      float peso, altura;
10     char sexo;
11     string nome;
12
13 public:
14     //Contrutores
15     Pessoa();
16     Pessoa(string n, float p, float a, char s);
17     //Gets
18     string getNome();
19     char getSexo();
20     float getPeso();
21     float getAltura();
22     //Sets
23     void setNome(string nome);
24     void setSexo(char sexo);
25     void setPeso(float peso);
26     void setAltura(float altura);
27     //Métodos
28     void print();
29     void printNome();
30 };
31 #endif
32
```

Pessoa.cpp

```cpp
#include "Pessoa.hpp"
using namespace std;

Pessoa::Pessoa()
{ //cria o construtor objeto
}
Pessoa::Pessoa(string n, float p, float a, char s)
{
    nome = n;
    peso = p;
    altura = a;
    sexo = s;
}

string Pessoa::getNome()
{
    return nome;
}
      class Pessoa
char Pessoa::getSexo()
{
    return sexo;
}

float Pessoa::getPeso()
{
    return peso;
}

float Pessoa::getAltura()
{
    return altura;
}

void Pessoa::setNome(string nome)
{
    this->nome = nome;
}

void Pessoa::setSexo(char sexo)
{
    this->sexo = sexo;
}

void Pessoa::setPeso(float peso)
{
    this->peso = peso;
}

void Pessoa::setAltura(float altura)
{
    this->altura = altura;
}

void Pessoa::print()
{ //mostra o valor na tela
    cout << "-------------\n";
    cout << "Nome: " << nome << endl;
    cout << "sexo: " << sexo << endl;
    cout << "Peso: " << peso << endl;
    cout << "Altura: " << altura << endl;
}

void Pessoa::printNome()
{ //mostra o valor na tela
    cout << "-------------\n";
    cout << "Nome: " << nome << endl;
}
```

Aluno.hpp

```
1  #ifndef __ALUNO_HPP
2  #define __ALUNO_HPP
3  #include <iostream>
4  #include "Pessoa.hpp"
5  #include <string.h>
6
7  using namespace std;
8
9  class Aluno : public Pessoa
10 {
11 private:
12     int matricula;
13     string curso;
14     //Construtor
15     Aluno();
16
17 public:
18     //Construtor
19     Aluno(int matricula, string curso, string nome,
20         float peso, float altura, char sexo);
21     //Gets
22     int getMatricula();
23     string getCurso();
24     //Sets
25     void setMatricula(int mat);
26     void setCurso(string curso);
27     //Métodos
28     void print();
29 };
30 #endif
```

Aluno.cpp

```cpp
#include <iostream> //biblioteca
#include "Aluno.hpp"

Aluno::Aluno(int mat, string curso, string nome,
             float peso, float altura, char sexo)
             : Pessoa(nome, peso, altura, sexo)
{
    this->matricula = mat;
    this->curso = curso;
}

int Aluno::getMatricula()
{
    return matricula;
}

string Aluno::getCurso()
{
    return curso;
}

void Aluno::setMatricula(int mat)
{
    this->matricula = mat;
}

void Aluno::setCurso(string curso)
{
    this->curso = curso;
}

void Aluno::print()
{
    Pessoa::print();
    Pessoa::printNome();
    cout << "Matricula: " << matricula << endl;
    cout << "Curso: " << curso << endl;
}

```

Funcionario.hpp

```hpp
#ifndef __FUNCIONARIO_HPP
#define __FUNCIONARIO_HPP
#include <iostream> //biblioteca
#include "Pessoa.hpp"
#include <string.h>

using namespace std;

class Funcionario : public Pessoa
{
private:
    int numCadastro;
    string lotacao;
    float salario;

public:
public:
    //Construtores
    Funcionario();
    Funcionario(int numCadastro, string lotacao, float salario,
        string n, float p, float a, char s);
    //Gets
    string getLotacao();
    int getNumCadastro();
    //Sets
    void setNumCadastro(int num);
    void setLotacao(string lot);
    //Métodos
    void print();
    void printSalarioAnual();
};
#endif
```

Funcionario.cpp

```cpp
#include <iostream> //biblioteca
#include "Funcionario.hpp"

Funcionario::Funcionario()
{
}

Funcionario::Funcionario(int numCadastro, string lotacao,
    float salario, string nome, float peso,
    float altura, char sexo) : Pessoa(nome, peso, altura, sexo)
{
    this->numCadastro = numCadastro;
    this->lotacao = lotacao;
    this->salario = salario;
}

string Funcionario::getLotacao()
{
    return lotacao;
}

int Funcionario::getNumCadastro()
{
    return numCadastro;
}

void Funcionario::setNumCadastro(int numCadastro)
{
    this->numCadastro = numCadastro;
}

void Funcionario::setLotacao(string lot)
{
    this->lotacao = lotacao;
}

void Funcionario::print()
{
    Pessoa::print();
    cout << "Número de Cadastro: " << numCadastro << endl;
    cout << "Lotacao: " << lotacao << endl;
}

void Funcionario::printSalarioAnual()
{
    float salarioAnual = salario * 13 + salario * 1 / 3;
    cout << "Salario anual = " << salarioAnual;
}
```

Main.cpp

```cpp
#include <iostream>
#include "Pessoa.hpp"
#include "Aluno.hpp"
#include "Funcionario.hpp"

int main(int argc, char** argv) {
    Pessoa z; // chama o construtor-padrão
    float peso,altura, salario;
    char sexo;
    string nome,lot;
    int numCadastro;

    cout << "------------FUNCIONÁRIO-----------";
    cout << "\nDigite o nome: ";
    getline(cin,nome);
    cout << "\nDigite o sexo: ";
    cin >> sexo;
    cout<< "\nDigite a altura: ";
    cin >> altura;
    cout << "\nDigite o peso: ";
    cin >> peso;
    cout << "\nDigite o número do cadastro: ";
    cin >> numCadastro;
    cout << "\nDigite a lotação: ";
    cin >> lot;
    cout << "\nDigite o salário: ";
    cin >> salario;

    Funcionario f1 (numCadastro,lot,salario, nome,peso,altura,sexo);
    f1.print(); //enviar o resultado para void print()

    cout << "\n\n\n\n------------ALUNO---------";
    cout << "\nDigite o nome: ";
    cin.ignore();
    getline(cin,nome);
    cout << "\nDigite o curso: ";
    string curso;
    cin.ignore();
    getline(cin,curso);
    cout << "\nDigite o sexo: ";
    cin >> sexo;
    cout<< "\nDigite a altura: ";
    cin >> altura;
    cout << "\nDigite o peso: ";
    cin >> peso;
    cout << "\nDigite a matricula: ";
    int mat=0;
    cin >> mat;

    Aluno al(mat,curso,nome,peso,altura,sexo);
    al.print();
    return 0;
}
```

Polimorfismo

Um mesmo objeto pode ser de **vários tipos**;

Exemplo:

- Uma **Pessoa** pode ser um **Estudante** ou um **Professor**;

Não é viável exigir que todos os outros objetos saibam todos os possíveis tipos de um determinado objeto;

Todos os outros objetos devem reconhecer o objeto através de um **único** tipo;

Trechos de código para tratamento de diferentes tipos são eliminados;

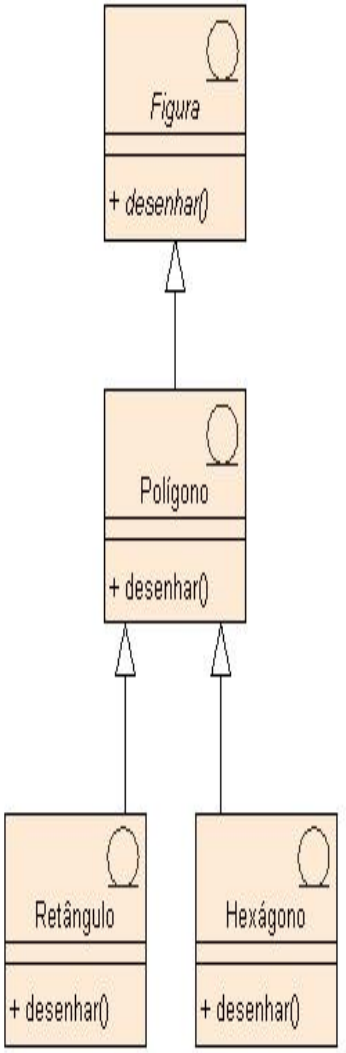
Através do **polimorfismo** instâncias de várias classes são tratadas de forma **única** em um sistema.

Polimorfismo

Cada tipo reimplementa alguma parte da interface em comum;

Outros objetos do sistema acessam a interface em comum de forma única;

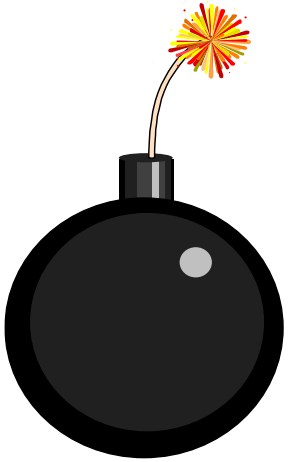
O comportamento do objeto será definido pela reimplementação contida no objeto.

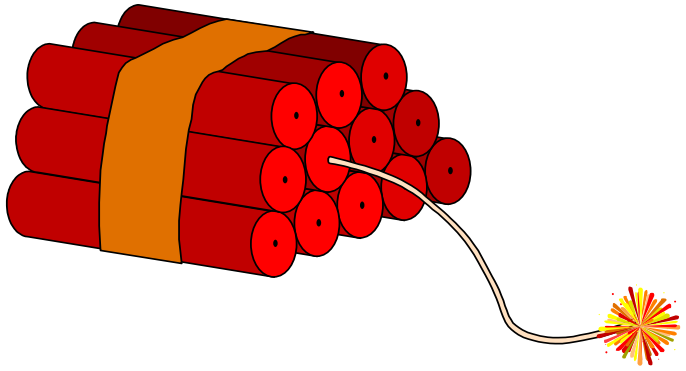


Polimorfismo

Através do **polimorfismo** instâncias de várias classes são tratadas de forma **única** em um sistema.

O Polimorfismo ocorre quando uma mesma mensagem chegando a objetos diferentes provoca respostas diferentes.





Polimorfismo

Polimorfismo de Inclusão:

- Modela **subtipos** e herança;
- O subtipo está incluído no tipo;
- Onde o objeto de um **tipo** for esperado, um objeto do **subtipo** deve ser aceito;

Exemplo:

- desenhar(Figura umaFigura).

Main.cpp

```cpp
#include <iostream>
#include "Pessoa.hpp"
#include "Aluno.hpp"
#include "Funcionario.hpp"

int main(int argc, char** argv) {
    Pessoa z; // chama o construtor-padrão
    float peso,altura, salario;
    char sexo;
    string nome,lot;
    int numCadastro;

    cout << "-------------FUNCIONÁRIO-----------";
    cout << "\nDigite o nome: ";
    cin.ignore();
    getline(cin,nome);
    cout << "\nDigite o sexo: ";
    cin >> sexo;
    cout<< "\nDigite a altura: ";
    cin >> altura;
    cout << "\nDigite o peso: ";
    cin >> peso;
    cout << "\nDigite o número do cadastro: ";
    cin >> numCadastro;
    cout << "\nDigite a lotação: ";
    cin >> lot;
    cout << "\nDigite o salário: ";
    cin >> salario;

    Funcionario f1 (numCadastro,lot,salario, nome,peso,altura,sexo);
    f1.print(); //enviar o resultado para void print()

    Pessoa p1=f1;
    cout<<"\n\n\n----Objeto pessoa----\n\n\n\n";
    p1.print();

    cout << "\n\n\n\n-------------ALUNO---------";
    cout << "\nDigite o nome: ";
    cin.ignore();
    getline(cin,nome);
    cout << "\nDigite o curso: ";
    string curso;
    cin.ignore();
    getline(cin,curso);
    cout << "\nDigite o sexo: ";
    cin >> sexo;
    cout<< "\nDigite a altura: ";
    cin >> altura;
    cout << "\nDigite o peso: ";
    cin >> peso;
    cout << "\nDigite a matricula: ";
    int mat=0;
    cin >> mat;

    Aluno al(mat,curso,nome,peso,altura,sexo);
    al.print();
    return 0;
}
```