UNIVERSIDADE ESTADUAL DO OESTE DO PARANÁ
CENTRO DE ENGENHARIA E CIÊNCIAS EXATAS
UNIOESTE - CAMPUS DE FOZ DO IGUAÇU
CURSO: CIÊNCIA DA COMPUTAÇÃO

COMPUTAÇÃO II
Orientação a Objetos

Professora: TERESINHA ARNAUTS HACHISUCA

Foz do Iguaçu, 4 de abril de 2016.

PTI Parque Tecnológico Itaipu

ITAIPU BINACIONAL

Integração que gera energia e desenvolvimento

O que é orientação a objetos

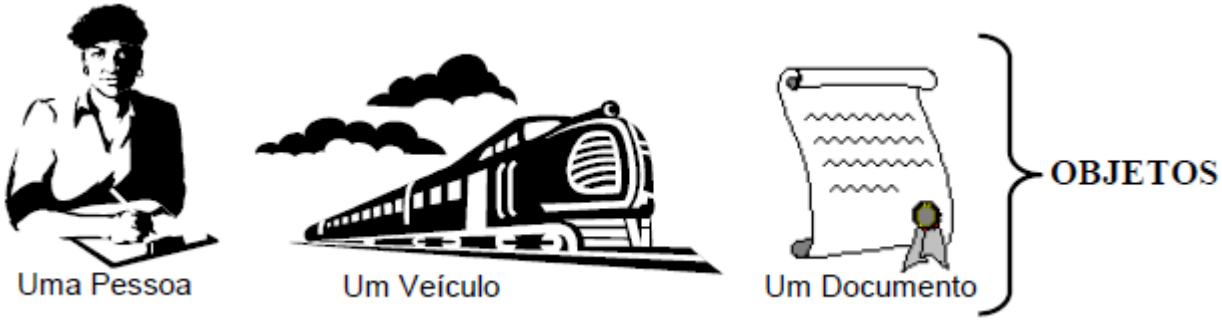
- É um paradigma para o desenvolvimento de software que baseia-se na utilização de componentes individuais (objetos) que colaboram para construir sistemas mais complexos. A colaboração entre os objetos é feita através do envio de mensagens.

- Um paradigma é um conjunto de regras que estabelecem fronteiras e descrevem como resolver problemas dentro desta fronteira. Um paradigma ajuda-nos a organizar a e coordenar a maneira como olhamos o mundo.

Objetos

- Um objeto consiste de um conjunto de operações encapsuladas (métodos) e um "estado" (determinado pelo valor dos atributos) que grava e recupera os efeitos destas operações.
- Em outras palavra um objeto possui tudo o que é necessário para conhecer a si próprio.

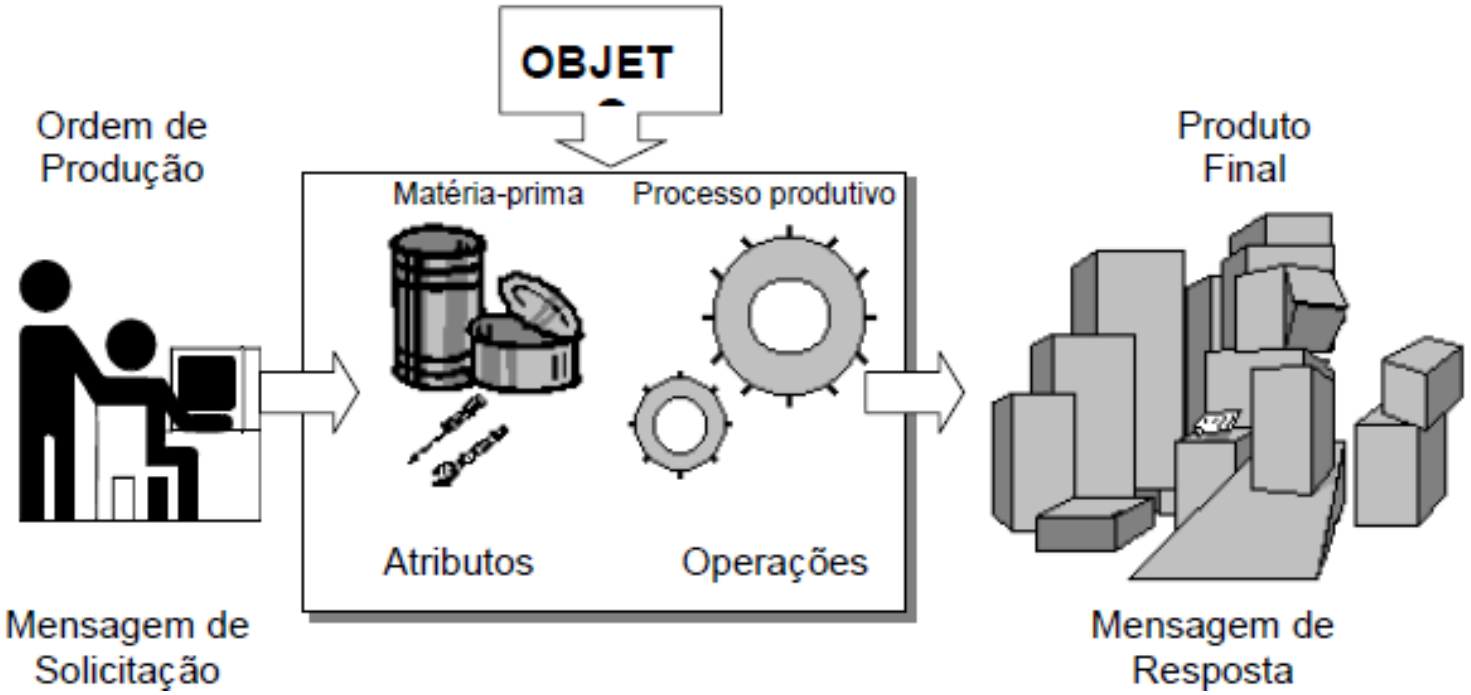
Exemplos de Objetos



Mensagem

- Mensagens são requisições enviadas de um objeto para outro, para que o objeto "receptor" forneça algum resultado desejado através da execução de uma operação.
- A natureza das operações realizadas para alcançar o resultado requerido é determinada pelo objeto "receptor".
- Mensagens podem também ser acompanhadas de parâmetros.

Comportamento e comunicação entre objetos



Métodos

- Métodos são similares a procedimentos e funções e consistem nas descrições das operações que um objeto executa quando recebe uma mensagem.
- Há, portanto, uma correspondência um para um entre mensagens e métodos que são executados quando a mensagem é recebida através de um determinado objeto.
- Uma mesma mensagem pode resultar em diferentes métodos quando for enviada para classes de objetos diferentes.

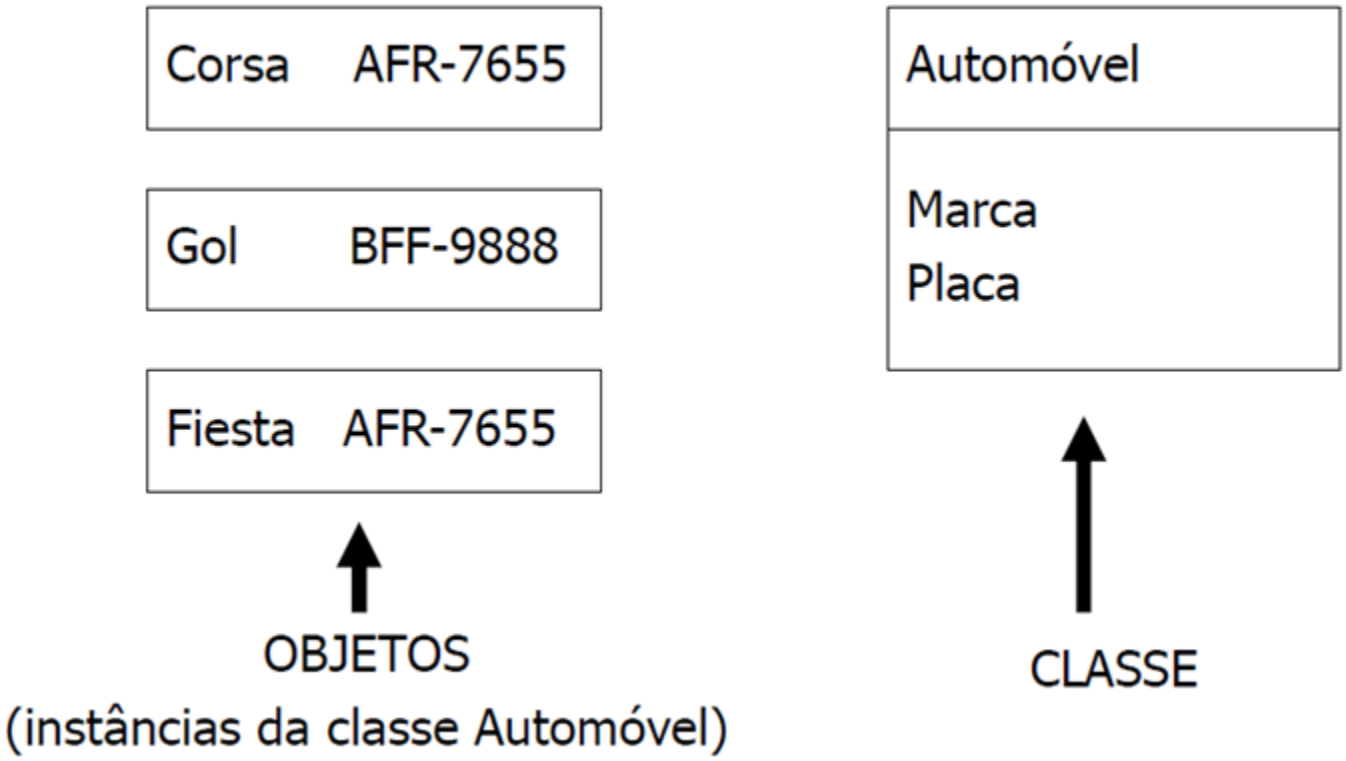
Atributos

- Um atributo consiste em um dado ou uma informação de estado, para o qual cada objeto de uma classe tem seu próprio valor.
- Existem dois tipos de atributos em um sistema orientado a objetos: os atributos das classes e os atributos dos objetos.
- Os atributos das classes representam informação cujo valor todos os seus objetos devem compartilhar.
- Os atributos dos objetos descrevem valores mantidos em um objeto.

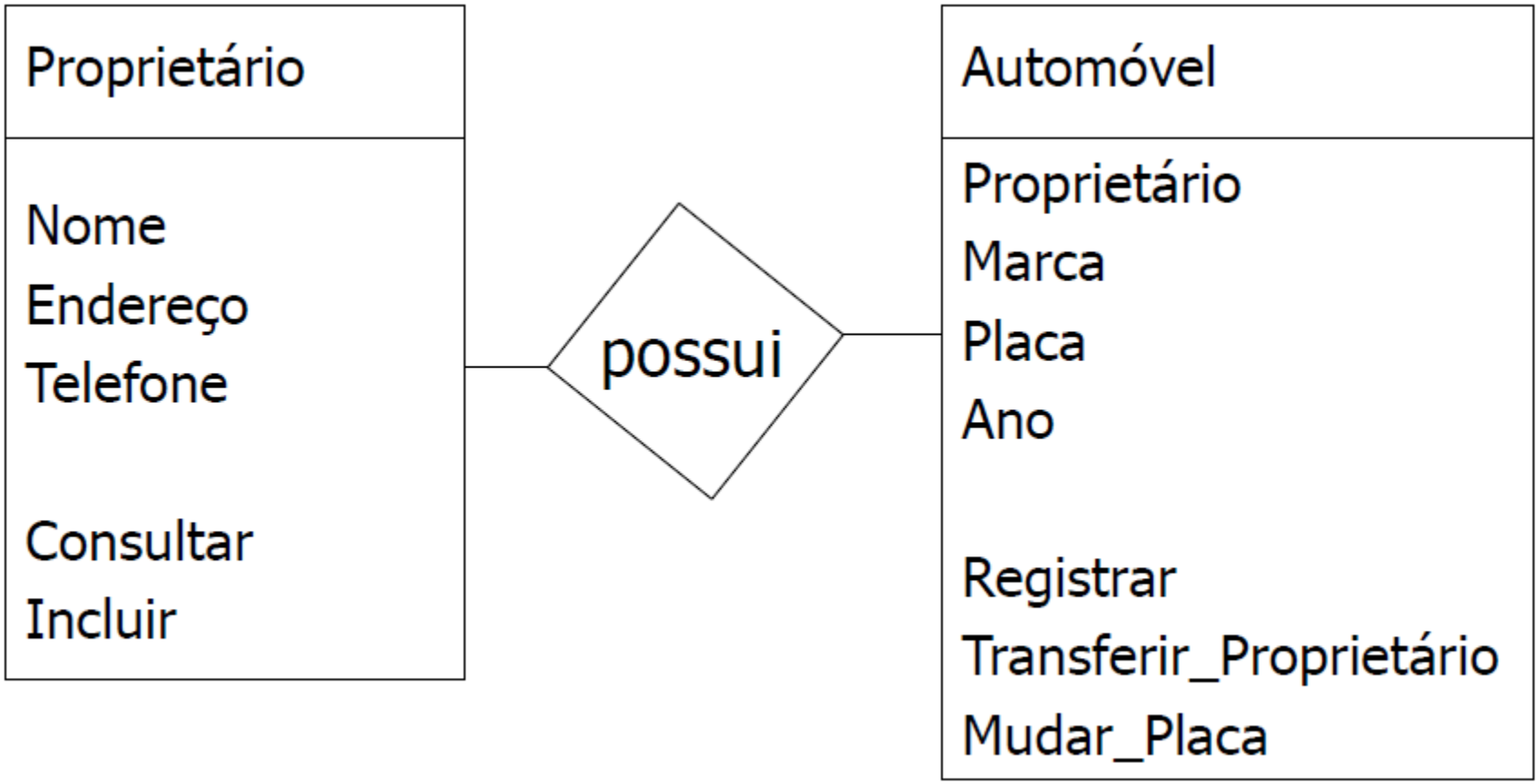
Classes

- Uma classe define as características de uma coleção de objetos.
- Classes consistem em descrições de métodos e atributos que objetos que pertencem à classe irão possuir.
- Uma classe é similar a um tipo abstrato de dado, no sentido que ela define uma estrutura interna e um conjunto de operações que todos os objetos que são instâncias daquela classe irão possuir.

Objetos e Classes

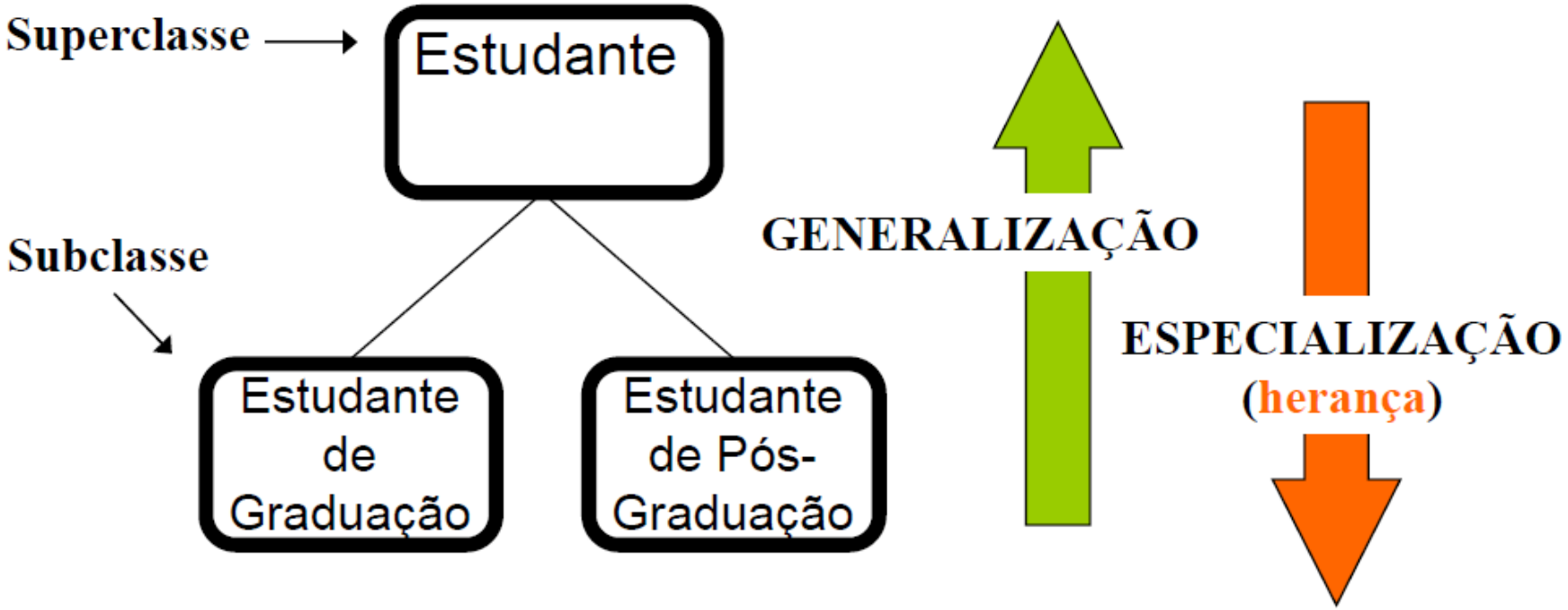


Relacionamento entre Classes



Generalização/Especialização

Generalização/Especialização

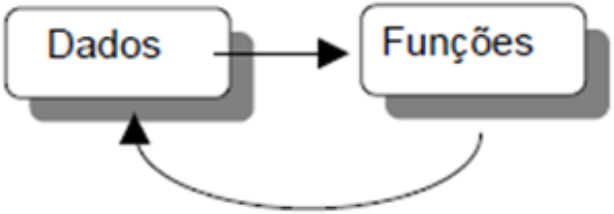


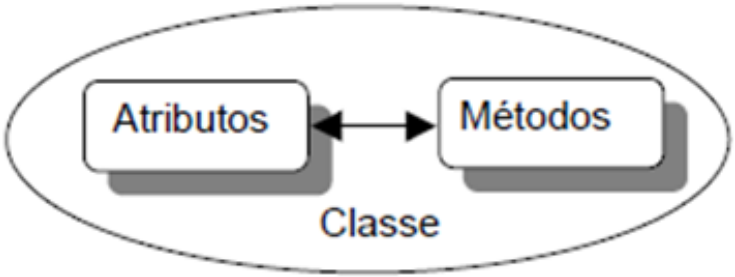
Construtores e Destrutores

- Construtores são os métodos que são chamados automaticamente na criação de uma instância de um objeto. Recebem, por padrão, o nome da classe a que pertencem.

- Destrutores são os métodos chamados quando o objeto é destruído (desalocado), em c++ recebem o nome da classe antecedido pelo caracter ~.

Programação Convencional x Programação OO





Implementação

Classe Pessoa
Nome Endereço
Setar() Mostrar() ~Pessoa()

Notação gráfica

```
class Pessoa {                  // Definição de uma superclasse
   private:                     // Mecanismo de restrição de acesso
      char nome[31];            //  \
      char endereco[40];        //  / Atributos da classe
   public :                     // Mecanismo de restrição de acesso
      void setar(void);         //  \
      void mostrar(void);       //   > Métodos da classe
      ~Pessoa(void);            //  /
};
```

Código C++ (pessoa.h)

Implementação

```
void Pessoa :: setar (void) {
    char newline;
    cout << "\n Digite o Nome: ";
    cin.get(nome,30,'\n');
    cin.get(newline);
    cout << "\n Digite o Endereco: ";
    cin.get(endereco,40,'\n');
    cin.get(newline);
}
void Pessoa :: mostrar(void) {
        cout << "\n Nome: " << nome  << "\n";
        cout << "\n Endereco: " << endereco  << "\n";
}
Pessoa::~Pessoa(void) {       // Destrutor
        cout << "\n Liberando a Memoria Alocada para Pessoa";
}
```

Código C++ (pessoa.cpp)

Exercício

1) Escrever um programa em c++ para uma classe ponto, que encapsule os valores das coordenadas do ponto. A classe deverá oferecer os seguintes métodos:
   a) Setar o valor da coordenada x e y (***setx (double x), sety (double y)***)
   b) Retornar o valor da coordenada x e y (***double getx(), double gety ()***)
   c) Apresentar o ponto (***show()***) ➔ (x,y)
   d) Construtor e destrutor (***Ponto(double x, double y) e ~Ponto()***).

Encapsulamento

- Cada objeto é visto como o encapsulamento de seu estado interno, suas mensagens e seus métodos. A estrutura do estado e dos pares “mensagem-método” são todas definidas através da classe à qual o objeto pertence.
- O valor atual do estado interno é determinado através dos métodos que o objeto executa em resposta às mensagens recebidas.
- Encapsulamento constitui-se no “coração” da programação orientada a objetos.

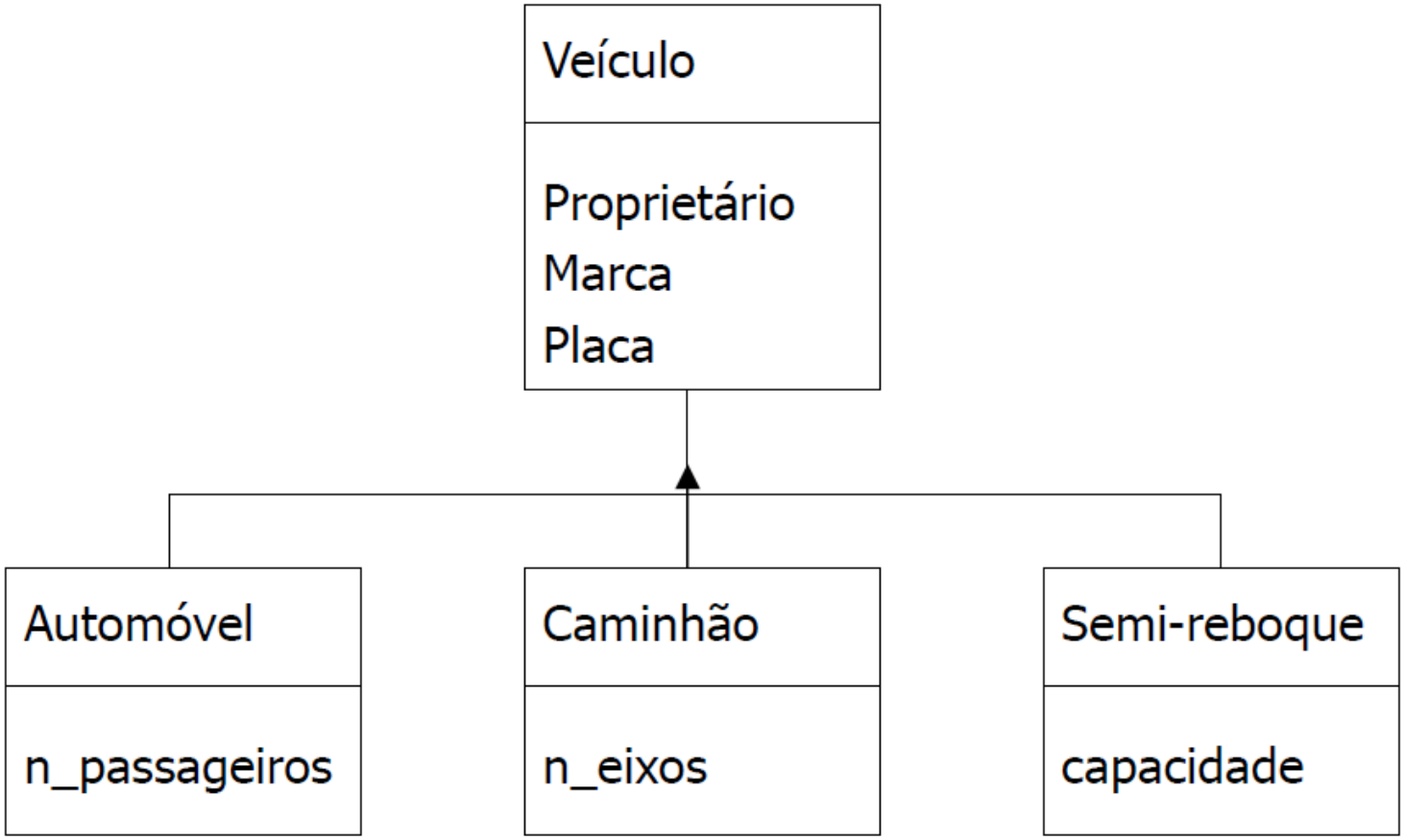
Encapsulamento

- Nós podemos determinar as características de encapsulamento alterando os chamados “especificadores de acesso”.
- Os especificadores de acesso mais conhecidos são: “public”, “private” e “protected”.
- Os membros privados (private) são acessíveis apenas pelos membros da própria classe;
- Os membros protegidos (protected), além dos direitos dados pelos membros privados, também estende esse direito para as classes derivadas. Quando esquecemos de informar o especificador, por padrão os membros serão protegidos.
- Os membros publicos (public) são acessíveis através de qualquer função ou classe que interage com os dados dessa classe.

Herança

- É uma das propriedades mais importantes do modelo de orientação a objetos. A herança é permitida através da definição de uma hierarquia de classes, isto é, uma “árvore” de classes onde cada uma possui zero ou mais subclasses.
- Herança refere-se ao fato de que uma subclasse herda todos os componentes da classe pai, incluindo uma estrutura interna e pares “mensagem-método”.
- Quaisquer propriedades herdadas podem ser redefinidas na definição da subclasse, substituindo-se assim a definição herdada.
- Propriedades que permanecem iguais não requerem redefinição.

Herança


Veículo
Proprietário
Marca
Placa
Automóvel
n_passageiros
Caminhão
n_eixos
Semi-reboque
capacidade

Herança (funcionario.h)

```
class Funcionario : public Pessoa {     // Definição de uma subclasse
    private:                            // Mecanismo de restrição de acesso
        int faltas;                     //  \
        date data_admissao;             //   > Atributos da classe
        float salario_base;             //  /
    public :                            // Mecanismo de restrição de acesso
        Funcionario(void);              //  \
        void setar(void);               //   \
        void mostrar(void);             //   / Métodos da classe
        ~Funcionario(void);             //  /
};
```

Herança (funcionario.cpp)

```
void Funcionario :: setar (void) {
        char newline;
        cout << "ENTRADA COM OS DADOS DO FUNCIONARIO \n";
        Pessoa::setar();
        cout << "\n Digite o Salario Base : ";
        cin  >> salario_base;
}
void Funcionario :: mostrar(void) {
        cout << "\n\n SAIDA DE DADOS DO FUNCIONARIO \n";
        Pessoa::mostrar();
        cout << "\n Salario Base: " << salario_base  << "\n";
        cout << "\n Faltas no Periodo: " << faltas << "\n";
        cout << "\n Data de Admissao: " << (int) data_admissao.da_day << '\\' << (int) data_admissao.da_mon
             << '\\' << data_admissao.da_year << "\n";
        getch();
}
Funcionario::~Funcionario(void) {     // Destrutor
        cout << "\nLiberando a Memoria Alocada para Funcionário";
}
Funcionario::Funcionario(void) {      // Construtor
        faltas = 0; getdate(&data_admissao);
}
```

main.cpp

```
main () {
    clrscr();
    Funcionario func;                    // Declaração de um objeto
    func.setar();
    func.mostrar();
}
```

Exercício

1) Implementar a classe círculo que herda as propriedades da classe ponto:
   a) Mover círculo, com sobrecarga:
      a) Mover (x, y)
      b) Mover (Ponto p)
   b) Aumentar; // Raio++
   c) Diminuir; // Raio—
   d) Retornar o valor do raio.
   e) Mostrar ➔ show() ➔ (x,y,raio).
   f) Construtor e destrutor (***Circulo(double x, double y, double r) e ~Circulo()***).

Polimorfismo

- É a propriedade que permite que a mesma mensagem seja enviada a diferentes objetos e que cada objeto execute a operação que é apropriada à sua classe.

- O objeto “emissor” não precisa conhecer a classe do objeto “receptor” e como este objeto irá responder à mensagem.

- Em c++ usa-se a palavra ***virtual*** na classe base para indicar um método que pode ser polimórfico.

Exercício

1) Implementar o seguinte programa:

```
void  main () {
  Ponto *p1, *p2;
  p1 = new Ponto(100,100);
  p2 = new Circulo(200,200);

  p1->show();
  p2->show(); // polimorfismo

  out << “x circulo: “ << p2->getx();
  out << “y circulo: “ << p2->gety();  // herança de método

  // imprima outros valores.
  delete p1;
  delete p2;
}
```

Exercícios Fixação

1) Escrever um programa em c++ para uma classe contador, que encapsule um valor utilizado para contagem de itens. A classe deverá oferecer os seguintes métodos:
   a) Zerar;
   b) Incrementar;
   c) Retornar o valor do contador;
   d) Construtor e destrutor.

Exercícios

2) Escreva uma classe Ponto2D que represente um ponto no plano cartesiano. Além dos atributos por você identificados, a classe deve oferecer os seguintes membros:

a) Construtores sobrecarregados que permitam a inicialização do ponto:
   - i) Por default (sem parâmetros) na origem do espaço 2D;
   - ii) Num local indicado por dois parâmetros do tipo double (indicando o valor de abcissa e ordenada do ponto que está sendo criado);
   - iii) Em um local indicado por outro ponto.

b) Métodos de acesso (getter/setter) dos atributos do ponto;

c) Métodos sobrecarregados de movimentação do ponto com os mesmos parâmetros indicados para os construtores;

d) Método de comparação semântica do ponto (equals);

e) Método que permita calcular a distância do ponto que recebe a mensagem, para outro ponto;

f) Método que permita a criação de um novo ponto no mesmo local do ponto que recebeu a mensagem (clone);

g) Destrutor da classe

Exercícios

3) Escreva uma classe que represente uma reta (y=ax+b). Forneça os seguintes membros de classe:

a) Construtores sobrecarregados que criem uma reta a partir de:
   i) Dois valores, representando o coeficiente angular e o coeficiente linear da reta;
   ii) Dois pontos;
b) Métodos de acesso para o coeficiente angular e para o coeficiente linear da reta;
c) Um método que verifique se um ponto dado pertence a reta;
d) Um método que dada uma outra reta, retorne o ponto de interseção da reta dada ou null se as retas forem paralelas.
f) Destrutor da classe

Exercícios

4) Escreva uma classe que represente um circulo no plano cartesiano. Forneça os seguintes membros de classe:

a) Um construtor que receba o raio e um ponto (o centro do círculo);

b) Um construtor que receba o raio e posicione o círculo na origem do espaço cartesiano;

c) Métodos de acesso ao atributo raio do círculo;

d) Métodos inflar e desinflar, que, respectivamente, aumentam e diminuem o raio do círculo de um dado valor;

e) Métodos sobrecarregados, inflar e desinflar, que, respectivamente, aumentam e diminuem o raio do círculo de uma unidade;

f) Métodos sobrecarregados mover, que:

   i) por default (sem parâmetros) levam o círculo para a origem do espaço 2D;

   ii) movem o círculo para um local indicado por dois parâmetros do tipo double (indicando o valor de abcissa e ordenada do ponto para onde o círculo se move);

   iii) movem o círculo para o local indicado por outro ponto.

g) Método que retorna a área do círculo

h) Destrutor da classe

Exercícios

5) Escreva uma classe que represente um país. Um país é representado através dos atributos: código ISO 3166-1 (ex.: BRA), nome (ex.: Brasil), população (ex.: 193.946.886) e a sua dimensão em Km2 (ex.: 8.515.767,049). Além disso, cada país mantém uma lista de outros países com os quais ele faz fronteira. Escreva a classe forneça os seus membros a seguir:

a) Construtor que inicialize o código ISO, o nome e a dimensão do país;

b) Métodos de acesso (getter/setter) para as propriedades código ISSO, nome, população e dimensão do país;

c) Um método que permita verificar se dois objetos representam o mesmo país (igualdade semântica). Dois países são iguais se tiverem o mesmo código ISO;

d) Um método que informe se outro país é limítrofe do país que recebeu a mensagem;

e) Um método que retorne a densidade populacional do país;

f) Um método que receba um país como parâmetro e retorne a lista de vizinhos comuns aos dois países.

Considere que um país tem no máximo 40 outros países com os quais ele faz fronteira.

Exercícios

6) Escreva uma classe Pessoa que representa uma pessoa numa árvore genealógica. A pessoa possui um nome, um pai e uma mãe (que também são pessoas). Forneça os seguintes membros para a classe:

a) Construtores sobrecarregados que:

i) inicialize o nome da pessoa, bem como seus antecessores (pai e mãe);

ii) inicialize o nome da pessoa, e coloque seus antecessores para null;

b) Um método que verifique a igualdade semântica entre duas pessoas (as pessoas são iguais se possuem o mesmo nome e a mesma mãe);

c) Um método que verifique se duas pessoas são irmãs;

d) Um método que verifique se uma pessoa é antecessora da pessoa que recebeu a mensagem (é seu pai ou sua mãe, ou antecessor do pai ou antecessor da mãe).

Exercícios

7) Crie uma classe Matriz que represente uma matriz matemática. Forneça um construtor que permita a inicialização das dimensões da Matriz Forneça métodos para acesso (leitura/escrita) de cada elemento da matriz.
Forneça os métodos adequados para as seguintes operações com matriz:
a) Comparação semântica da matriz;
b) Retornar a transposta (é aquela onde as linhas se transformam em colunas e as colunas em linhas) da matriz.
c) Retornar a oposta (é aquela onde todos os elementos possuem sinais trocados) da matriz;
d) Gere uma matriz nula (é aqueles onde todos os elementos são iguais a 0);
e) Informe se a matriz é identidade (matriz quadrada onde os elementos da diagonal principal são todos iguais a 1 e os demais 0);
f) Informe se a matriz é diagonal (matriz quadrada onde os elementos fora da diagonal principal são todos iguais a 0).
g) Informe se a matriz é singular (matriz diagonal onde os elementos da diagonal principal são todos iguais);
h) Informe se a matriz é simétrica (uma matriz quadrada é dita simétrica se ela é igual a sua transposta);
i) Informe se a matriz é anti-simétrica (uma matriz quadrada é dita anti-simétrica se sua oposta é igual a sua transposta)
j) Adicionar duas matrizes (alterando o valor da que recebeu a mensagem);
k) Subtrair duas matrizes(alterando o valor da que recebeu a mensagem);
l) Multiplicar duas matrizes(alterando o valor da que recebeu a mensagem);
m) Gere uma cópia da matriz.