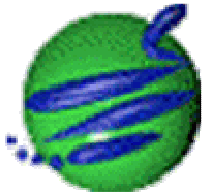
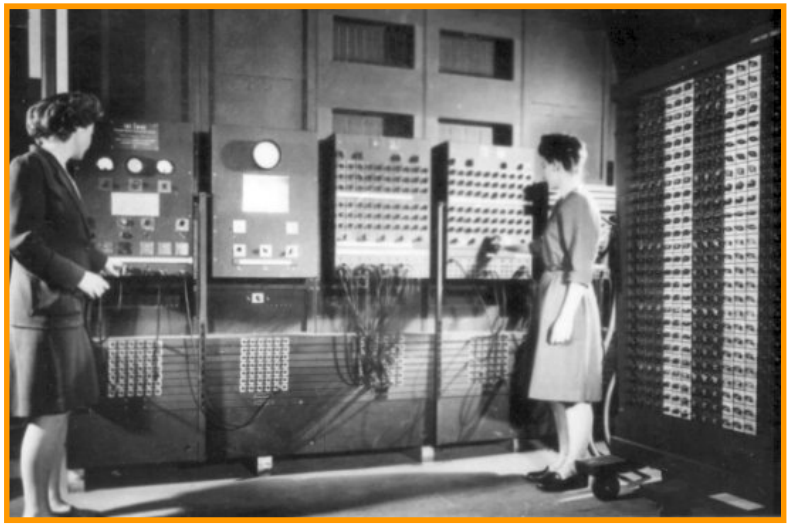


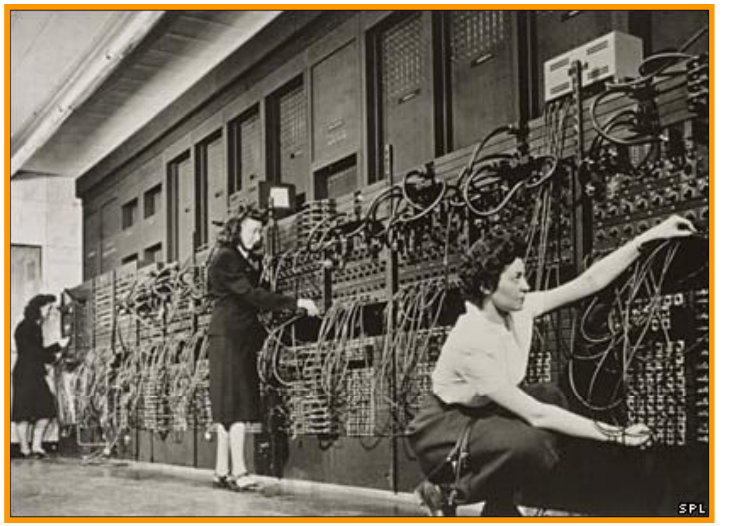
Introdução à Programação Orientada a Objetos

Adair Santa Catarina
Curso de Ciência da Computação
Unioeste – Campus de Cascavel – PR

Fev/2019

Histórico das linguagens de programação

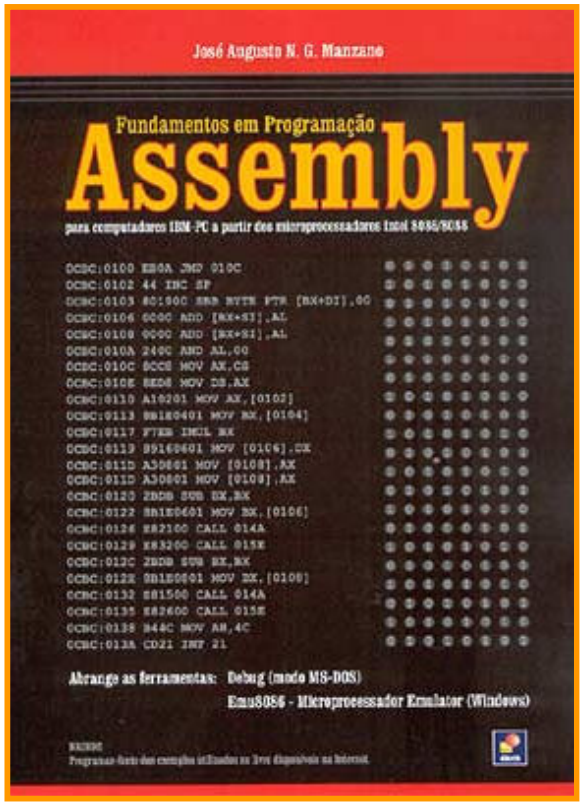




ENIAC (1944)

- Programação através de chaveamento de circuitos (seleção das instruções binárias).

Histórico das linguagens de programação

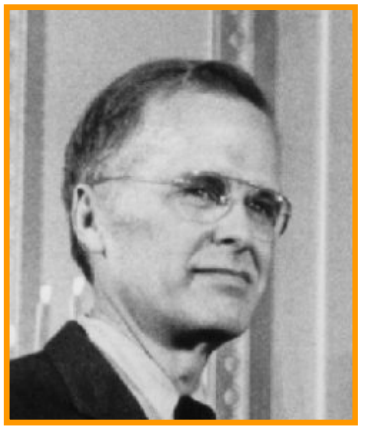


- Programas passaram a ter centenas de instruções;
- Programação em linguagem de montagem:
  - É uma notação legível por humanos para o código de máquina que uma arquitetura de computador específica usa.
  - Usa mnemônicos: ADD, SUB, MOV, PUSH, CALL, JMP, etc.

Histórico das linguagens de programação

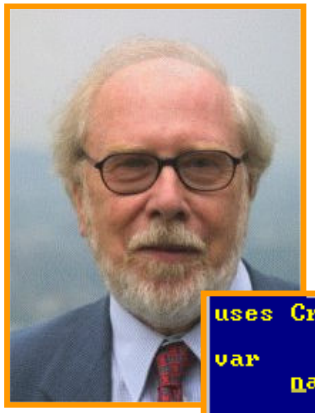
```
c  IF (X) 1,2,3 betekent:
c  spring naar label 1 als x < 0,
c  spring naar label 2 als x = 0
c  spring naar label 3 als x > 0

      IF (ISS)10,10,20
   20 IF (IX-IXO)12,21,12
   21 IF (IY-IYO)12,22,12
   22 IF (IS-ISO)12,23,12
   23 CALL PLALC(AM,NMAX,IRC)
      GO TO 73
   10 IF (IX2)13,50,13
   13 IF (IX2-MT)19,19,50
   19 IF (IY2)11,50,11
   11 IF (IY2-NT)12,12,50
   12 IF(TIND(IX2,IY2,1))     GO TO 73
      IF (CV-AM(IX2,IY2))206,206,5
  206 IF (IDX**2+IDY**2-1)213,6,213
  213 DCP=(AM(IX,IY)+AM(IX2,IY)+
     + AM(IX,IY2)+AM(IX2,IY2))/4.0
      IF (DCP-CV)5,217,217
  217 IF (INX(IS-1))214,215,214
  214 IX=IX+IDX

C .......
```

- FORTRAN : IBM Mathematical FORmula TRANslation System.
- 1954 – 1957 → Equipe liderada por John W. Backus;
- 1ª linguagem de programação de alto nível.

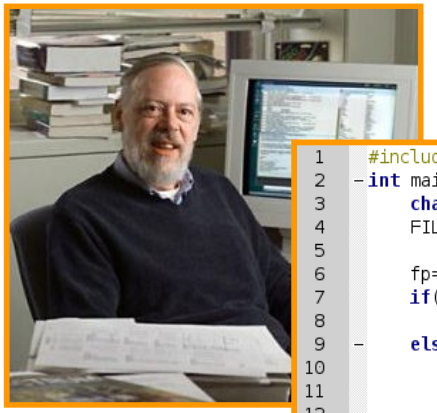


Histórico das linguagens de programação

- Linguagens Estruturadas:
  - Início com ALGOL 60;
  - Niklaus Wirth: ALGOL 68 → Pascal (1970);
  - Dennis Ritchie: C → 1972.



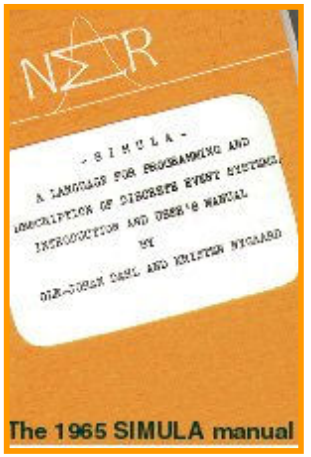
```
uses Crt;

var
     name: string[100];

begin
     clrscr;
     write ('NAME: ');
     readln(name);
     writeln ('Hello ', name);
     while not keypressed do;
end.
```



```c
#include <stdio.h>
int main(int argc,char *argv[]){
    char letra;
    FILE *fp;

    fp=fopen(argv[1],"r");
    if(fp==NULL)
        printf("o ficheiro nao existe\n");
    else{
        letra=fgetc(fp);
        fclose(fp);
        if(letra!='z')
            letra++;
        else
            letra='a';
        fp=fopen(argv[1],"w");
        fputc(letra, fp);
        fclose(fp);
    }
    return 0;
}
```

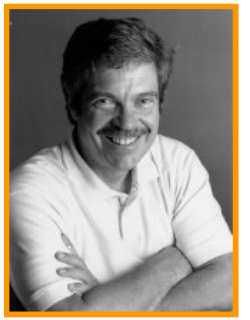
Histórico das linguagens de programação

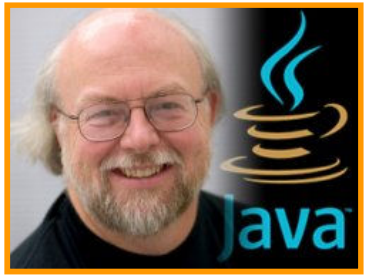
- Programação Orientada a Objetos:
  - Simula (1966) → extensão do ALGOL;
    - 1ª linguagem com facilidades para definição de classes de objetos;
    - Uma classe em Simula é um módulo que engloba as estruturas e o comportamento comuns aos objetos;
    - Linguagem para programação de simulações: os sistemas são modelados pela interação de muitos objetos distintos.



Histórico das linguagens de programação

- Programação Orientada a Objetos:
  - Smalltalk (1976):
    - Allan Kay;
    - Objetos ativos que reagem a mensagens ativando comportamentos específicos.
  - C++ (1983):
    - Bjarne Stroustrup;
    - Extensão da linguagem C para suportar classes.
  - Java (1995):
    - James Gosling → Sun Microsystems
    - Portabilidade → Bytecode → Máquina Virtual.





Origens da Orientação a Objetos

- Décadas de 1960-1970: Crise do Software;
  - Ao contrário do hardware, que teve um rápido desenvolvimento, o desenvolvimento de software continuava "empacado".
  - Por que a indústria de hardware prosperou?

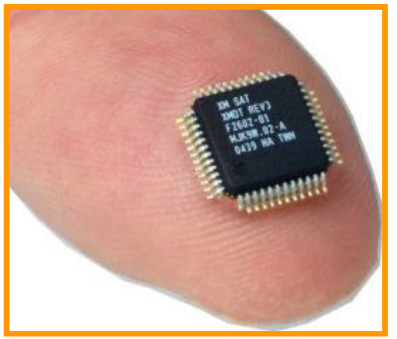
Indústria software

Indústria hardware

  - Reaproveitamento de esforços, através do uso de componentes padronizados.
  - Uma indústria de hardware não precisa produzir todos os componentes utilizados.

Origens da Orientação a Objetos

- Grande responsável pela alta produtividade da indústria de hardware:



- Objetivo da indústria de software:
  - Inventar o "Chip de software";
    - Permitir a montagem de sistemas complexos a partir de componentes de software, como acontece com os equipamentos de hardware.

O que são os Objetos?

- São entidades de software que encapsulam em si toda a funcionalidade necessário ao seu uso;
- Cumprem o papel de “Chip de software”;
- Foram os responsáveis por alavancar a indústria de software.

Programação Estruturada x POO

- Programação Estruturada (PE):
  - Na PE os programas são vistos como uma série de ações realizadas sobre um conjunto de dados;
    - Há separação entre os dados e as subrotinas que os manipulam.
  - Criada para proporcionar uma abordagem sistemática à organização de procedimentos e ao gerenciamento de grandes quantidades de dados;
  - Baseada na abordagem "Dividir e Conquistar": divide-se um programa complexo em uma série de procedimentos mais simples.

Programação Estruturada x POO

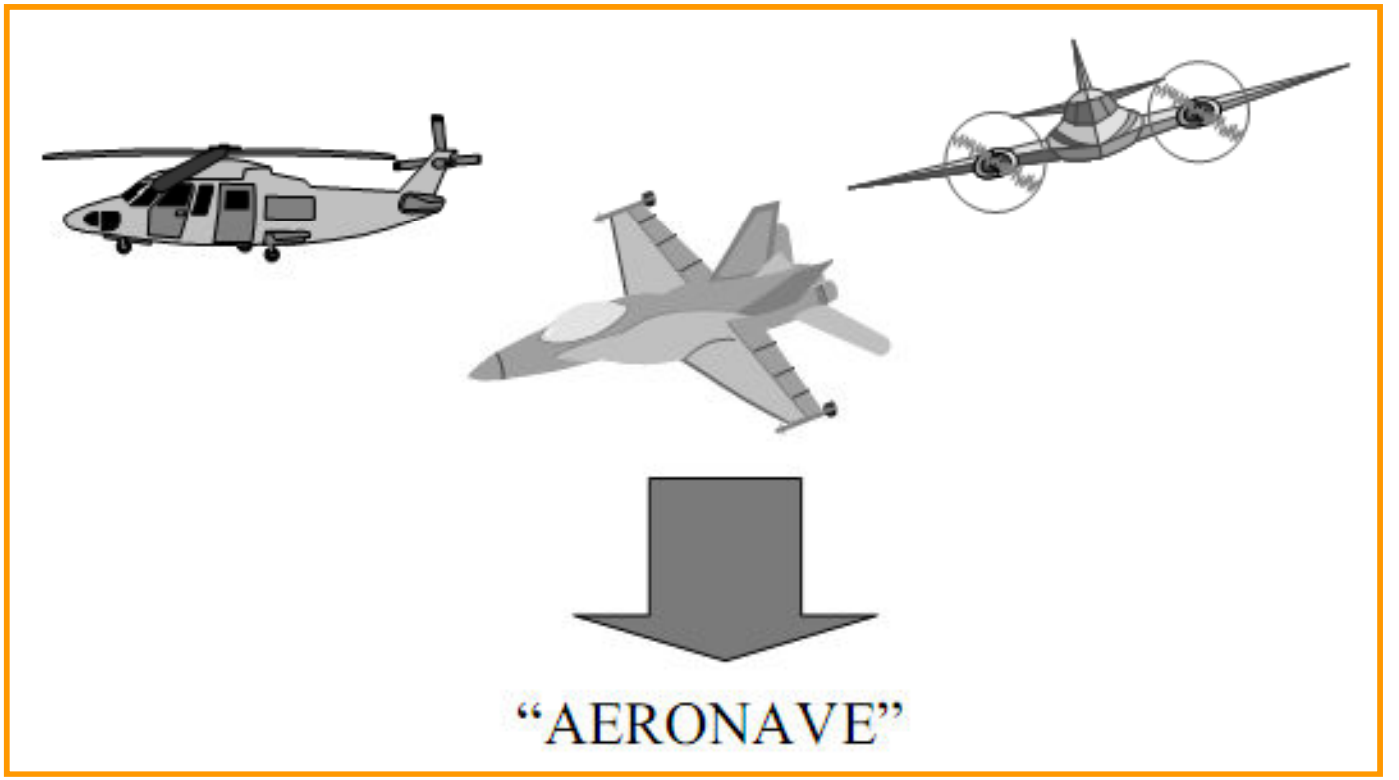
- Programação Orientada a Objetos (POO):
  - Metodologia de programação que trata os dados e os procedimentos que atuam sobre estes dados como um único "objeto" – uma entidade independente com uma identidade e características próprias;
  - Modela os objetos do mundo real em elementos de software;
  - Permite a criação de **Tipos Abstratos de Dados**.

Tipos Abstratos de Dados

- São semelhantes aos tipos concretos, porém:
  - Permitem que não apenas os dados, mas também as funções e procedimentos que atuam sobre estes dados sejam incluídas na definição do tipo, protegendo os dados contra acessos indevidos;
  - Os dados podem ser acessados apenas pelas funções incluídas na definição do tipo;
  - O mecanismo que permite a definição de tipos abstratos é a **Classe**.

Abstração

- Focalizar o essencial ignorando propriedades acidentais.



Classes

- Uma classe é como um record ou uma struct que pode conter, além da definição das variáveis, a definição das subrotinas que atuam sobre estas variáveis;
- Uma classe é uma estrutura que define um conjunto de objetos semelhantes;
- As variáveis definidas na classe são chamadas de **atributos da classe** e as subrotinas de **métodos da classe**;
- A classe pode ser entendida como um molde para criar objetos.

Objetos

- Um programa orientado a objetos é estruturado como uma comunidade de agentes que interagem entre si, denominados **objetos**;
- Cada objeto tem um papel a cumprir; objetos oferecem serviços ou realizam ações que são usadas por outros membros da comunidade;
- Um objeto é a concretização de uma classe, como uma variável é a concretização de um tipo.
- Instância ou instância de classe são sinônimos para o termo objeto.

Exemplo: montagem de um computador

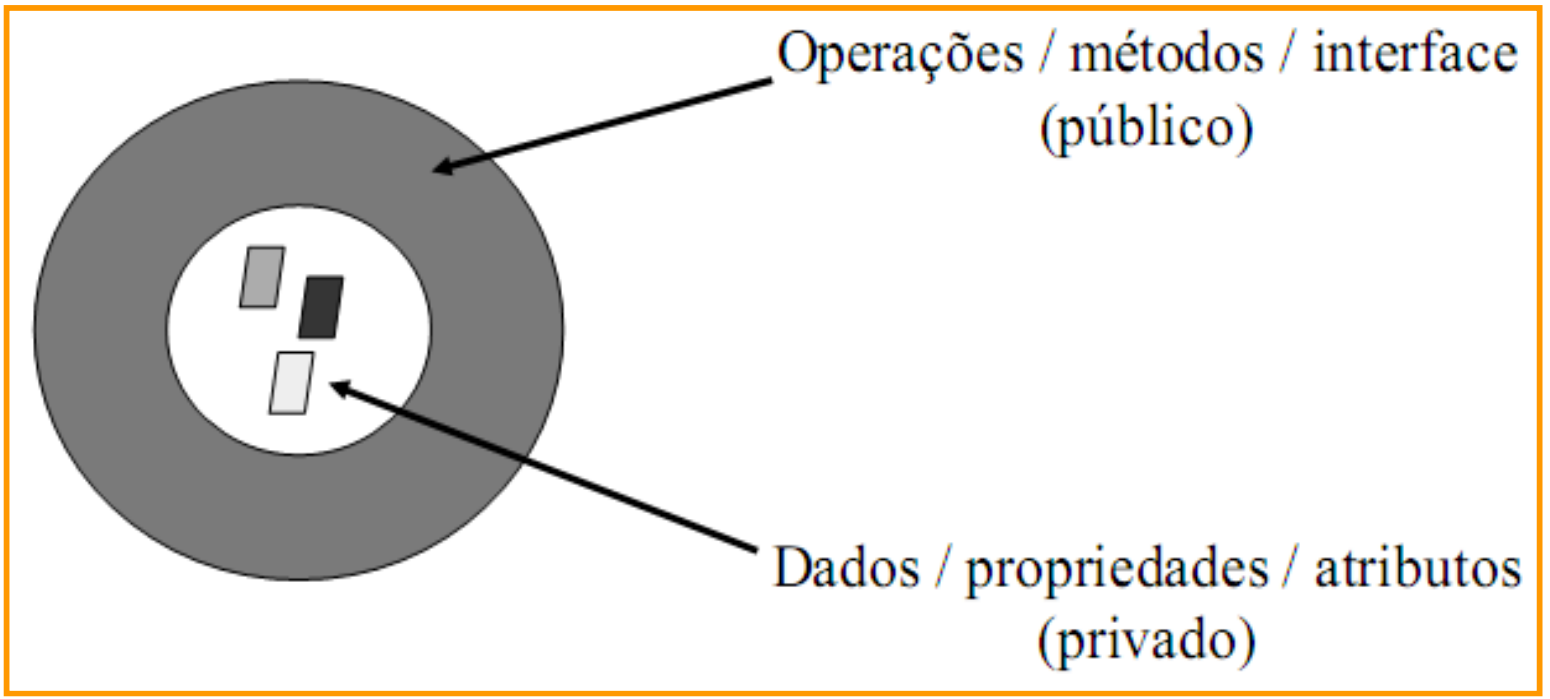
- Composto por vários componentes:
  - Placa-mãe;
  - CPU;
  - Placa de vídeo;
  - Disco rígido;
  - Teclado, etc.
- Cada componente é bastante sofisticado, mas o usuário não precisa saber como funciona internamente;
- Cada componente é independente dos demais;
- Para quem está montando, interessa apenas como os componentes interagem entre si:
  - A placa de vídeo encaixa no slot?
  - O monitor funciona com essa placa?
  - A CPU é compatível com a placa-mãe?

Membros de Classe

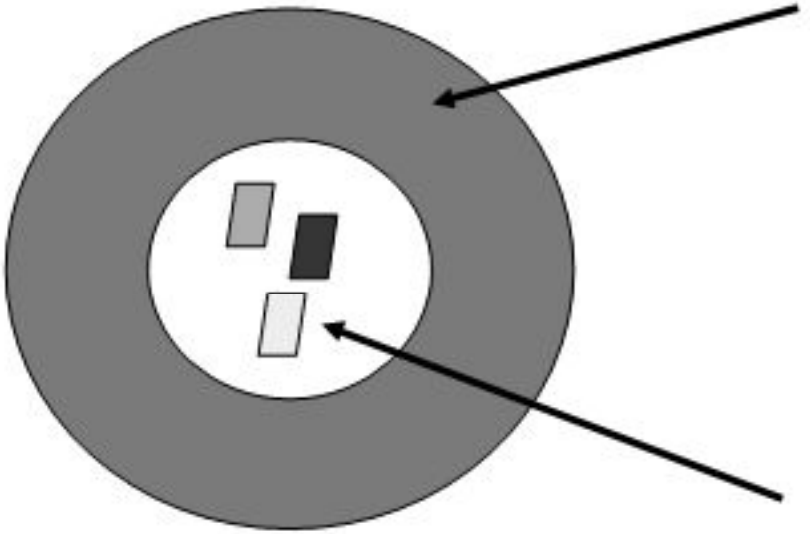
- Atributos:
  - São as variáveis internas ao objeto, as quais só podem ser acessadas pelos **métodos** da classe;
  - São também chamados de dados-membro.

- Métodos:
  - São as funções definidas dentro da classe, e que são as únicas a poder atuar sobre os **atributos** da classe.
  - São também chamados de funções-membro.

Representação de um Objeto



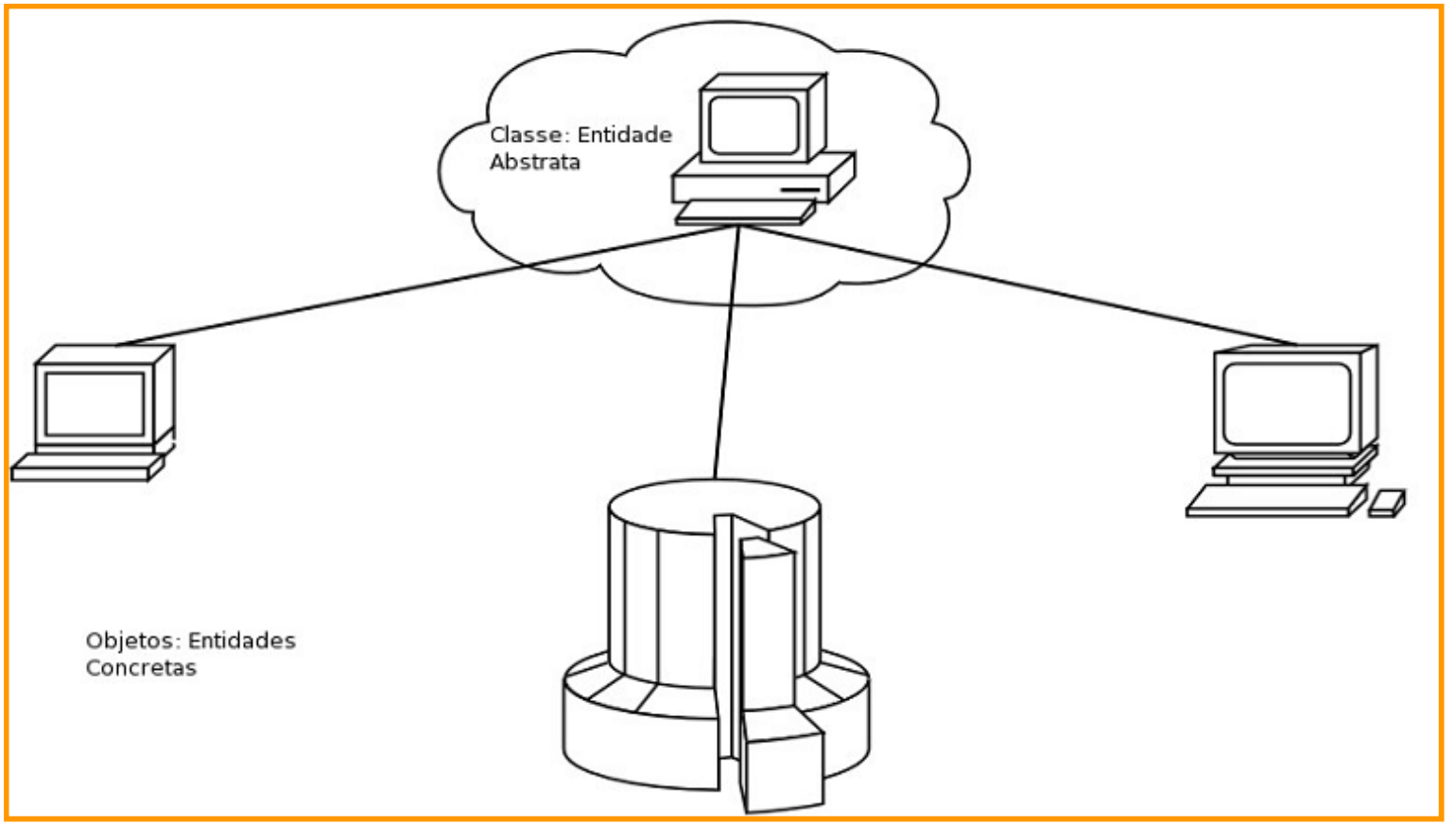
Exemplo: Lâmpada



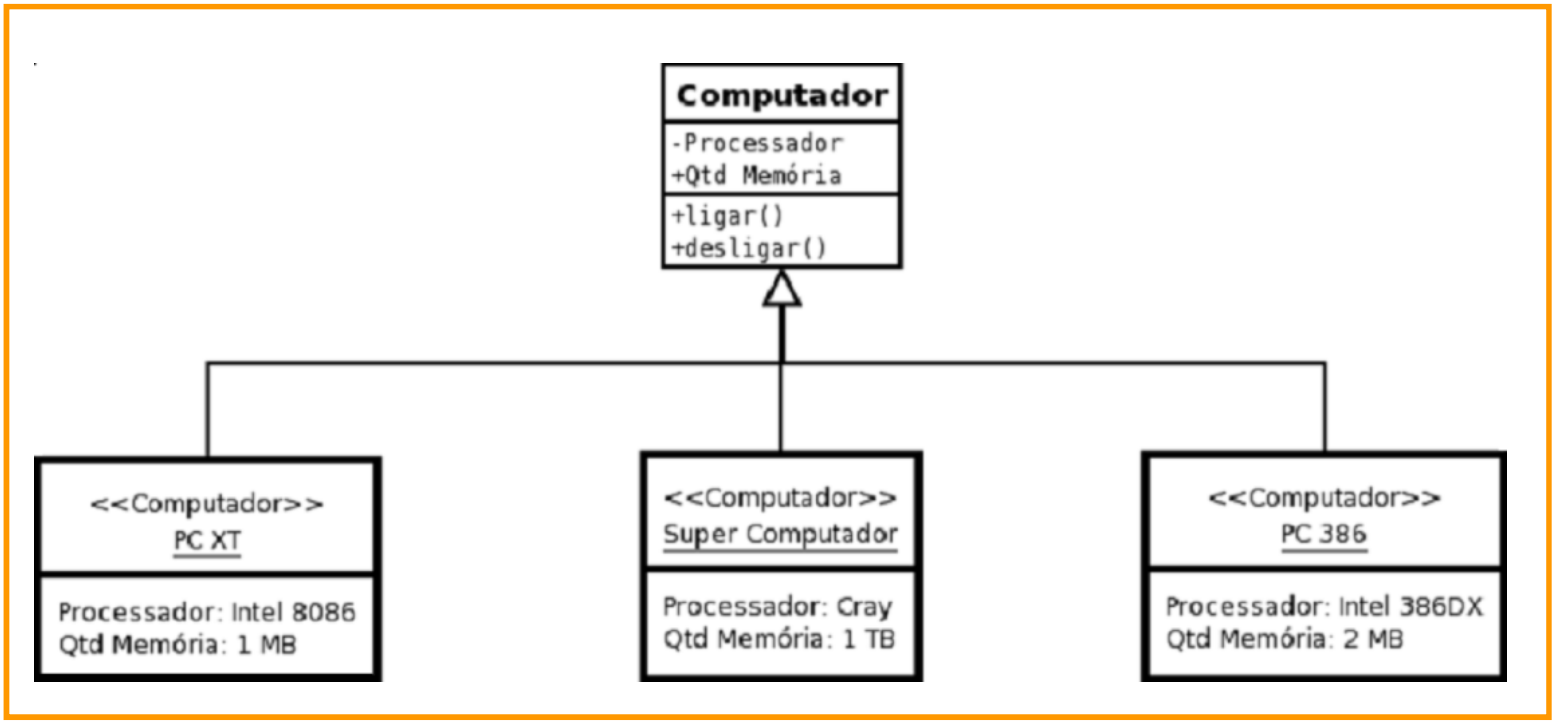
Operações:
- ligar
- desligar

Dados:
- ligada (s/n)
- potência
- voltagem

Exemplo: Computador

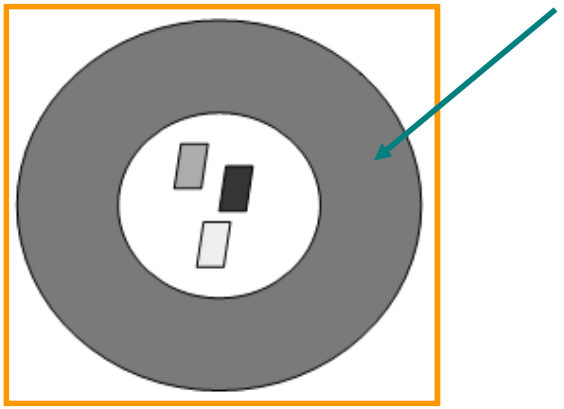


Exemplo: Diagrama de Classes



Encapsulamento

- Técnica para minimizar as interdependências entre módulos, através da definição de interfaces externas;
- "Caixa preta" – não é necessário saber como funciona internamente, mas sim como utilizar.



A interface (pública) de um objeto declara todas as operações permitidas (métodos)

O acesso aos dados através da chamada de um método definido pelo objeto.

Vantagens do Encapsulamento

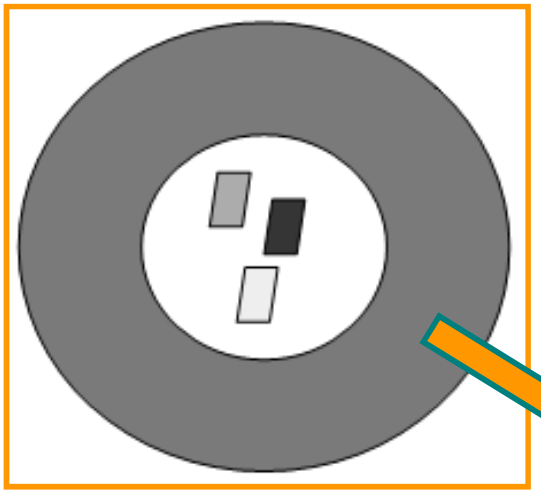
- Mudanças na implementação interna do objeto (que preservam sua interface externa) não afetam o resto do sistema;
- **Segurança**: protege os objetos de terem seus atributos corrompidos por outros objetos;
- **Independência**: "escondendo" seus atributos, um objeto protege outros de complicações de dependência de sua estrutura interna.

Mensagens e Protocolo

- Mensagens:
  - São solicitações enviadas a um objeto, para que este execute seus métodos;
  - Quando um objeto recebe uma mensagem, ele assume o controle do programa e executa um de seus métodos;
  - Após a execução do método, o controle é devolvido a quem enviou a mensagem.

- Protocolo:
  - É o conjunto de mensagens de um objeto.

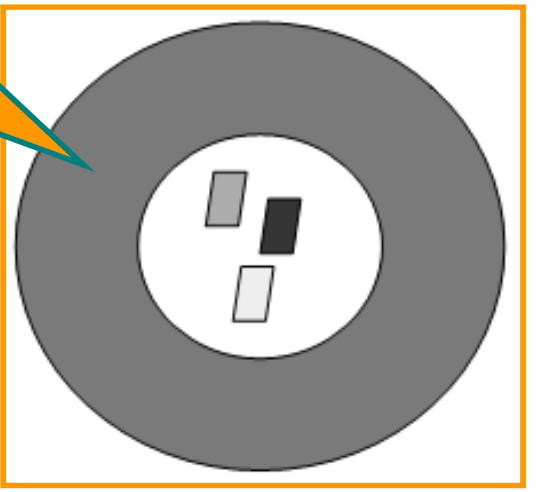
Mensagens



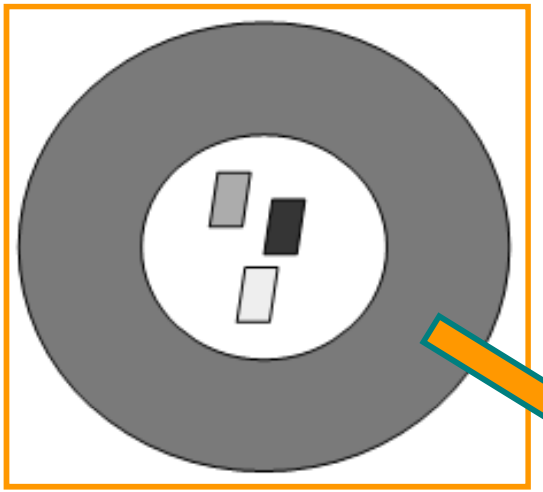
Emissor (cliente)

Objetos interagem e se comunicam através de **mensagens**...

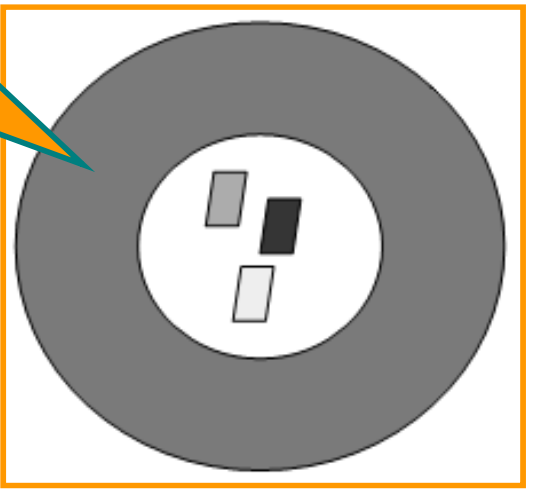
Receptor (servidor)



Métodos



... as mensagens identificam os **métodos** a serem executados no objeto receptor.



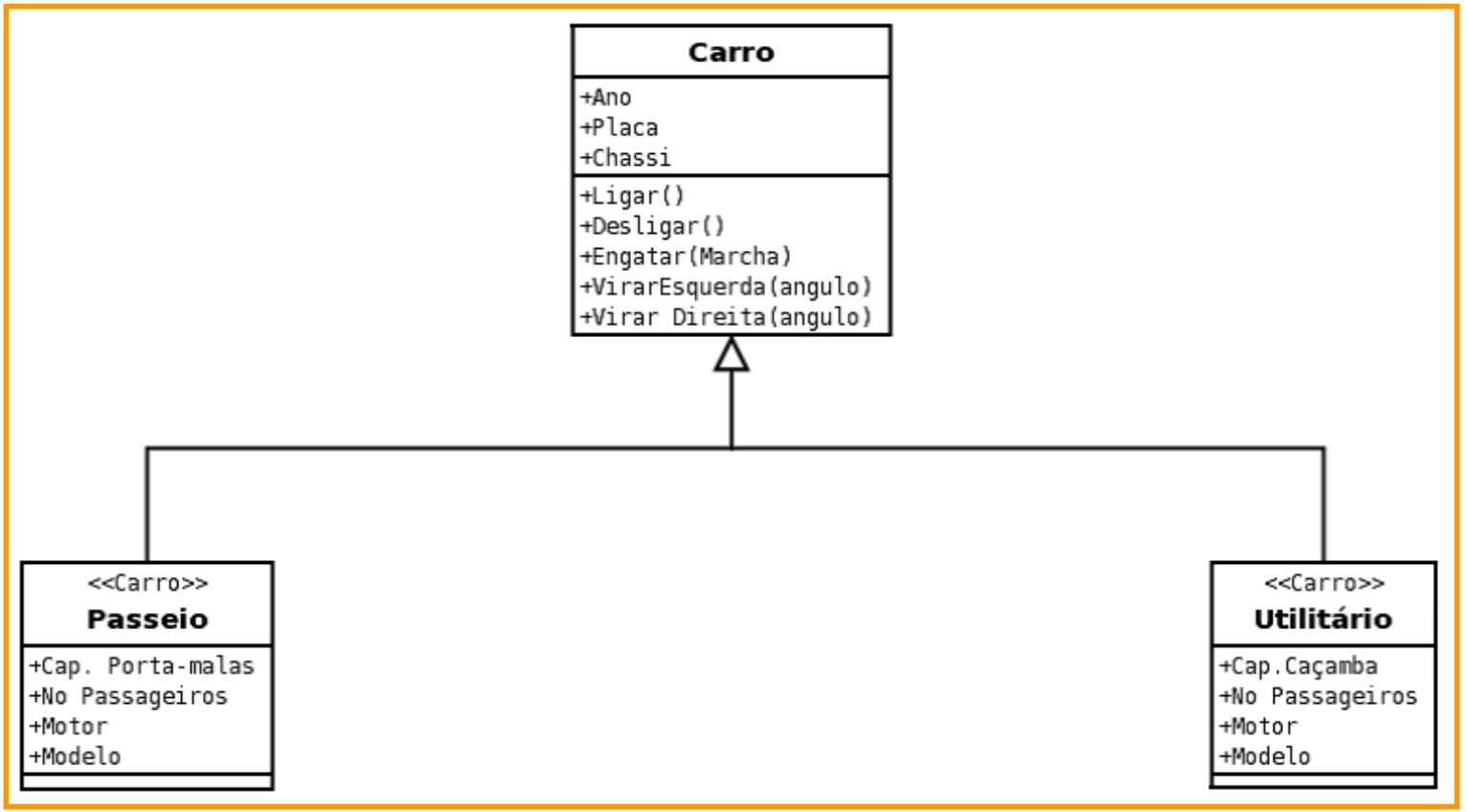
Mensagens e Métodos

- Para invocar um método, deve-se enviar uma mensagem ao objeto desejado;
- Para enviar uma mensagem, deve-se:
  - Identificar o **objeto** que receberá a mensagem;
  - Identificar o **método** que o objeto deverá executar;
  - Passar os **argumentos** requeridos pelo método.

Herança

- É a capacidade de se definir novas classes com base em classes já existentes;
- Permite ao programador criar uma nova classe programando apenas as diferenças entre a nova classe e a classe anterior (classe-pai);
- A herança pode ser:
  - Simples: atributos e métodos herdados de uma única classe;
  - Múltipla: atributos e métodos herdados de várias classes.

Herança


Carro
+Ano
+Placa
+Chassi
+Ligar()
+Desligar()
+Engatar(Marcha)
+VirarEsquerda(angulo)
+Virar Direita(angulo)
<<Carro>>
Passeio
+Cap. Porta-malas
+No Passageiros
+Motor
+Modelo
<<Carro>>
Utilitário
+Cap.Caçamba
+No Passageiros
+Motor
+Modelo

Polimorfismo

- É a propriedade de usar o mesmo nome para métodos diferentes, implementados em níveis diferentes de uma hierarquia de classes;
- Permite que métodos herdados por uma classe sejam alterados (sobrescritos) para adaptar-se a peculiaridades próprias da classe derivada.
- Tipos clássicos de polimorfismo:
  - De operadores: distinção pelo tipo de operandos, por exemplo soma de inteiros ou soma de reais;
  - Redefinição de operadores: criar operações não embutidas na linguagem, por exemplo a soma de matrizes;
  - Dois métodos iguais na mesma classe: distinção pelo número e/ou tipo dos parâmetros.