INE5603 Introdução à POO

Prof. A. G. Silva

31 de julho de 2017

Disciplina

INE5603 – Introdução à Programação Orientada a Objetos

- **Turma:** 01238C
- **Professor:** Alexandre Gonçalves Silva
  - https://www.inf.ufsc.br/~alexandre.goncalves.silva/
  - `alexandre.goncalves.silva@ufsc.br`
  - Sala INE-506
- **Carga horária:** 108 horas-aula • Teóricas: 30 • Práticas: 78
- **Curso:** Sistemas de Informação (238)
- **Requisitos:** *Não há*
- **Período:** 2º semestre de 2017
- **Materiais:** https://moodle.ufsc.br/course/view.php?id=79103
- **Horários:**
  - 2ª 18h30 (4 aulas) – CTC203 / LIICT1
  - 4ª 18h30 (2 aulas) – CTC101 / LIICT1

Ementa e objetivos

Ementa: Modelagem conceitual: Abstração × Representação. O Modelo de Objetos: Classes e Objetos, Comunicação por troca de mensagens. Herança e Polimorfismo.

Objetivos:

- **Geral:** Apresentar as noções básicas de programação de computadores, capacitando os alunos a analisar problemas de complexidade básica e projetar/desenvolver soluções de software sob a perspectiva de orientação a objetos.
- **Específicos:** • Apresentar os conceitos fundamentais da programação orientada a objetos. • Capacitar o aluno a analisar problemas de complexidade básica, abstraindo e modelando e implementando soluções sob o enfoque da programação orientada a objetos. • Desenvolver fluência em uma linguagem de programação orientada a objetos.

Conteúdo programático

- **Contextualização [8 horas-aula]:** • Modelo conceitual. • Processos de abstração e representação. • Histórico sobre linguagens de programação.
- **Conceitos básicos da POO [10 horas-aula]:** • Classes e objetos. • Atributos. • Métodos, argumentos e parâmetros.
- **Conceitos básicos de programação imperativa [40 horas-aula]:** • Algoritmos e programas. • Processo de edição, compilação e execução. • Variáveis e Tipos de dados. • Comando de atribuição. • Operadores aritméticos e lógicos. • Estruturas de controle: de seqüenciação, de decisão, de repetição.
- **Coleções [30 horas-aula]:** • Cadeias de caracteres (String). • Coleções unidimensionais. • Coleções bidimensionais.
- **Modelo de objetos [20 horas-aula]:** • Comunicação por troca de mensagens. • Encapsulamento e ocultamento de informações. • Hierarquia de agregação/decomposição. • Hierarquia de especialização/generalização. • Herança e Polimorfismo.

Metodologia e avaliação

Metodologia:

- Os aspectos teóricos da disciplina serão abordados ao longo do semestre em aulas expositivas, assim como através de leitura e discussão de textos pertinentes. A prática de programação será desenvolvida por meio de implementação computacional de soluções para problemas propostos utilizando a linguagem de programação JAVA, em sessões conduzidas pelo professor em sala de aula utilizando recursos multimídia ou em ambiente de laboratório.

Avaliação:

- A avaliação da aprendizagem será feita por meio de quatro provas escritas, P_1 a P_4, e a média E dos exercícios práticos (aprox. um por semana). A média final (MF) será calculada com as seguintes ponderações:

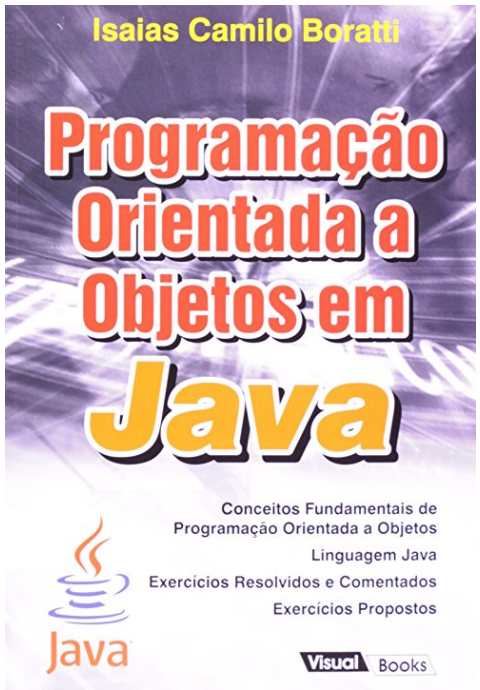
$$MF = \frac{2 \times P_1 + 2 \times P_2 + 2 \times P_3 + E}{7}$$

- As provas estão previstas para as seguintes datas:
  - P_1 - 11set • P_2 - 23out • P_3 - 20nov • Sub - 27nov

Bibliografia I

Básica:

- BORATTI, Isaias C. Programação Orientada a Objetos em Java. Florianópolis: VisualBooks. 2007.



- CAMARÃO, C. e FIGUEIREDO, L. Programação de Computadores em Java. Rio de Janeiro: LTC. 2003.
- DEITEL, Harvey M.; DEITEL, Paul J. . Java como programar. 6. ed. São Paulo (SP): Pearson Prentice Hall, 2005.

Bibliografia II

Complementar:

- BORATTI, Isaias C. e OLIVEIRA, A. B. Introdução a Programação – Algoritmos. Visual Books, 3 Ed. 2007
- SANTOS, R. Introdução à Programação Orientada a Objetos usando Java. São Paulo: Campus, 2003.
- SIERRA, Kathy; BATES, Bert . Use a cabeça!: Java. [tradução Aldir José Coelho] Rio de Janeiro: Alta Books, 2007.
- ASCENCIO, Ana Fernanda Gomes; CAMPOS, Edilene Aparecida Veneruchi de. Fundamentos da programação de computadores: algoritmos, Pascal, C/C++ e Java. 2. ed. São Paulo (SP): Pearson Prentice Hall, 2008.
- MANZANO, José Augusto N. G.; OLIVEIRA, Jayr Figueiredo de. Algoritmos: lógica para desenvolvimento de programação de computadores. 23.ed. São Paulo (SP): Érica, 2010.
- PUGA, Sandra; RISSETTI, Gerson. Lógica de programação e estruturas de dados: com aplicações em Java. 2. ed. São Paulo: Prentice Hall, 2009.

Tópicos da aula

1. Modelo conceitual
2. Processos de abstração e representação
   - Introdução
   - O processo de abstração
   - Operações de abstração
     - ★ Classificação/Instanciação
     - ★ Generalização/Especialização
     - ★ Agregação/Decomposição
     - ★ Associação
3. Histórico sobre linguagens de programação

Sobre a aula

- Objetivos
  - Apresentar alguns conceitos básicos sobre orientação a objetos.
  - Pensar a programação de computadores com base neste paradigma.
- As anotações desta apresentação são baseadas no livro texto [Boratti, 2007]

Introdução I

- Computador presente no cotidiano como máquina programável.
- Resolução dos mais variados problemas por meio de execução de programas (software).
- A construção de um programa implica na definição de um modelo de resolução, baseada na análise do problema associada ao paradigma de programação utilizado.
- Paradigmas de programação:
  - Programação estruturada/imperativa (C, Pascal, ...)
  - Programação orientada a objetos (Smalltalk, C++, Java, ...)
  - Programação funcional (Lisp, Haskell, ...)
  - Programação lógica/declarativa (Prolog, ...)

Introdução II

- Paradigma orientado a objetos procura abordar a resolução de um problema por meio de uma representação semelhante a do mundo real.

- O mundo real é constituído por entidades que interagem entre si. Uma entidade pode ser definida como um objeto com alguma função no mundo real.

- Exemplo de interação entre objetos:
  - **Problema:** "Determinar como deslocar um pessoa até o centro"
  - **Uma solução possível:**
    1. O objeto pessoa deve caminhar até um ponto de ônibus e tomar o objeto ônibus para o centro da cidade.
    2. O objeto ônibus transportará o objeto pessoa e, ao chegar no centro, o objeto pessoa deverá acionar o objeto campainha para que o objeto ônibus pare e o objeto pessoa desça.
  - Quais os objetos e as interações envolvidas?

Processo de abstração I

- Um programa de computador pode ser visto como um código (textual), em uma dada linguagem, que especifica os objetos e suas interações para a resolução de um problema.

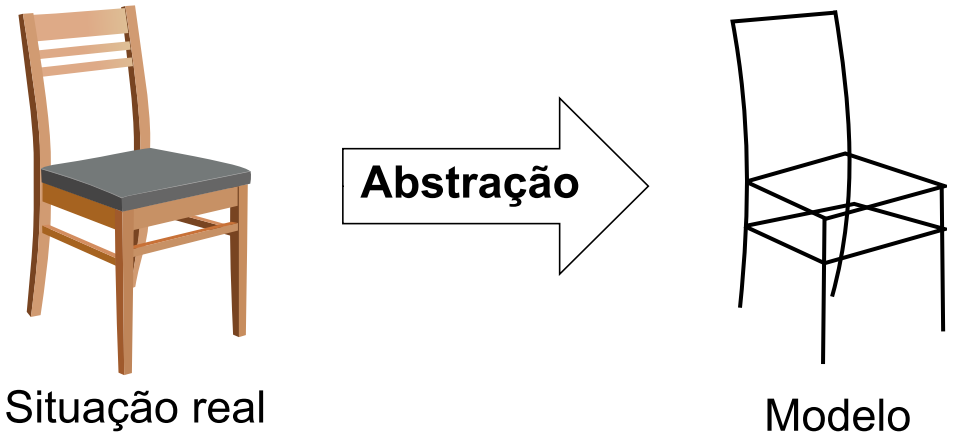
- Um problema normalmente envolve vários aspectos e sua solução implica em um processo de análise (identificação das características com relação direta à obtenção da solução).

- Um modelo de resolução de um problema deve espelhar adequadamente a situação real e sua construção deve considerar aspectos relevantes e irrelevantes num processo de **abstração**.

- Exemplo:
  - **Problema:** "Determinação da média final do aluno"
  - **Aspectos relevantes:** notas, participação em aula.
  - **Aspectos irrelevantes:** transporte, procedência escolar, caligrafia.

Processo de abstração II

A **abstração** constitui-se em um processo mental, no qual o ser humano modela uma entidade, isolando as características importantes, tendo como objetivo a redução de sua complexidade.



- O objetivo da abstração é a modelagem de determinada entidade ou a resolução de um problema.
- A abstração depende do contexto do problema.

Processo de abstração III

- O projetista deve considerar as características essenciais sob o ponto de vista de quem necessita da solução do problema.

- Um mesmo objeto pode ser modelado de várias formas, dependendo do contexto do problema e da respectiva abstração. Exemplo:
  - Do ponto de vista de um técnico em eletrônica, um aparelho de TV contém transistores, resistores, capacitores, circuitos integrados, etc.
  - Do ponto de vista do telespectador, uma aparelho de TV é algo que se pode ligar, desligar, trocar de canal, assistir programas, etc.

- Queremos definir modelos que possam resolver problemas por meio de um computador.
  - A abstração é representada principalmente por um programa de computador escrito em uma determinada linguagem.
  - Porém, há outras formas de representação: diagramas, textos descritivos, etc.

Operações de abstração

- Um programa é uma representação do modelo de resolução de um problema.
- Pode ser visto como um texto especificador de objetos, representando o mundo real, que executam determinadas interações.
- Antes do primeiro programa, é importante ter em mente as operações frequentemente utilizadas no processo de abstração de objetos.
- As operações de abstração mostram como o ser humano mentaliza, organiza e modela o mundo ao seu redor. As seguintes operações básicas podem ser aplicadas no mundo real:
  - Classificação/Instanciação
  - Generalização/Especialização
  - Agregação/Decomposição
  - Associação

Classificação/Instanciação I

- Entidades do mundo com sua existência e utilidade, ou seja, podendo interagir com o meio e prestar algum tipo de serviço.

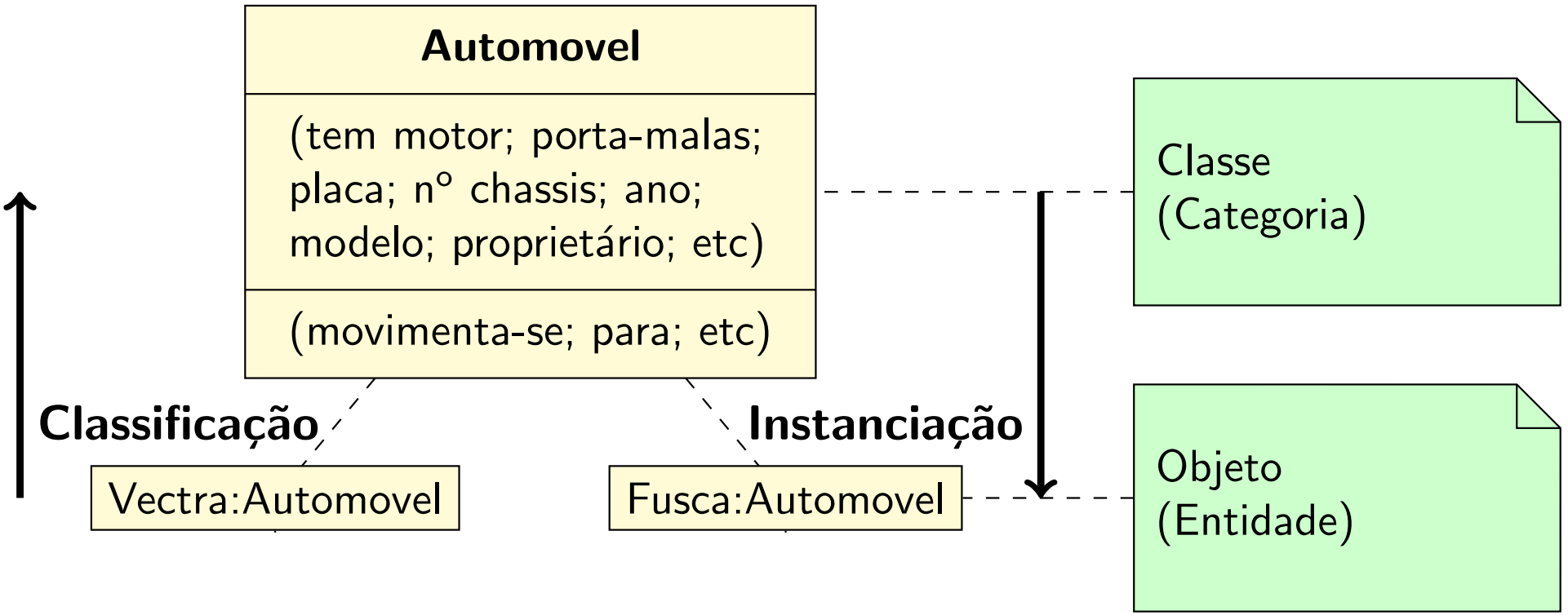
- Cada entidade tem determinadas características que as identificam.
  - Seu carro se caracteriza como um automóvel por ter itens comuns aos mesmos: motor, porta-malas, placas, sistema de câmbio, volante, além de se locomover, estacionar, etc.
  - O veículo de seu vizinho tambem é identificado com automóvel por apresentar as mesmas características.
  - Porém, seu carro é uma entidade distinta do carro de seu vizinho.

- Quando em um grupo de objetos, identificamos um conjunto de características comuns, definimos uma classe (a qual pertencem todos esses objetos) e estamos efetuando uma **operação de classificação**.

- Por outro lado, quando construímos um objeto contendo todas as características de determinada classe, estamos efetuando uma **operação de instanciação**.

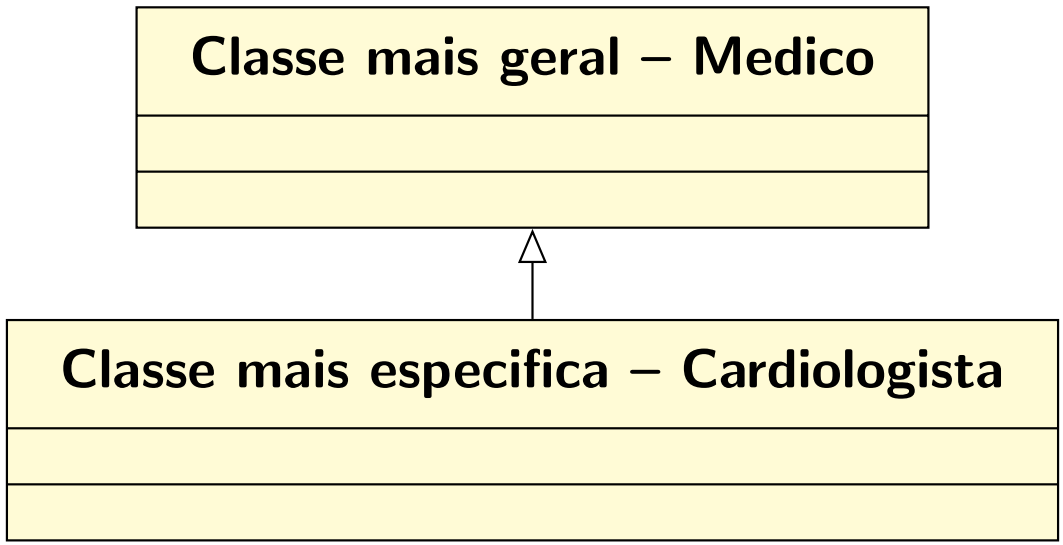
Classificação/Instanciação II

- **Classificação:** operação que tem por objetivo, da análise das características de um objeto, definir a que classe este pertence.
- **Instanciação:** operação que, dada uma determinada classe, define (ou constroi) um objeto pertencente a esta classe.
- Exemplo: vectra é um automóvel; fusca é um automóvel.



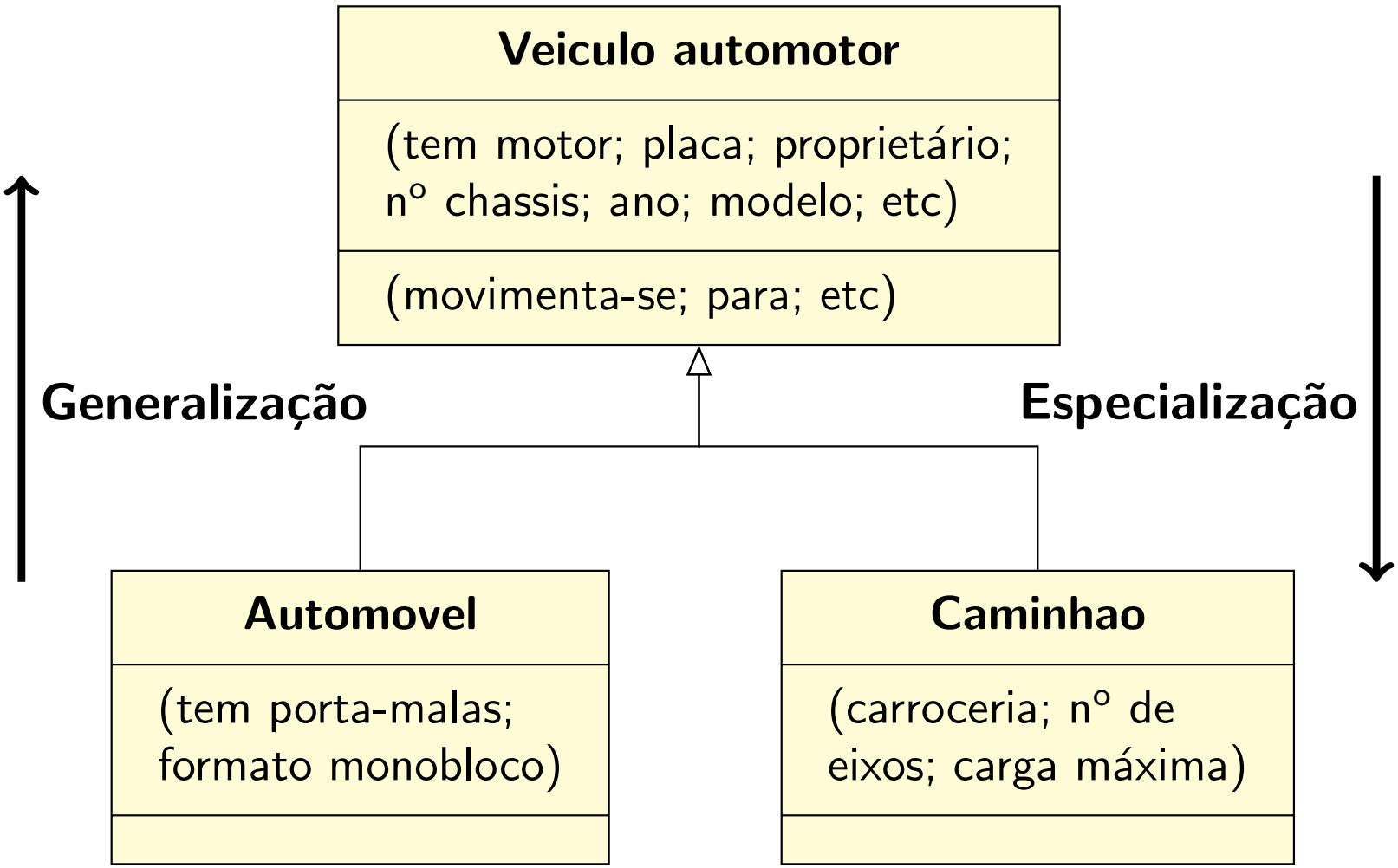
Generalização/Especialização I

- Um profissional formado em Medicina pode ser considerado um entidade (objeto) que pertence à classe Médico.
- Um médico com curso de especialização cardiovascular apresenta características adicionais que o habilita à classe Cardiologista.
- A classe Cardiologista constitui-se em uma especialização da classe Médico. Esta mantém (herda) todas as características da classe mais geral e adiciona características específicas.



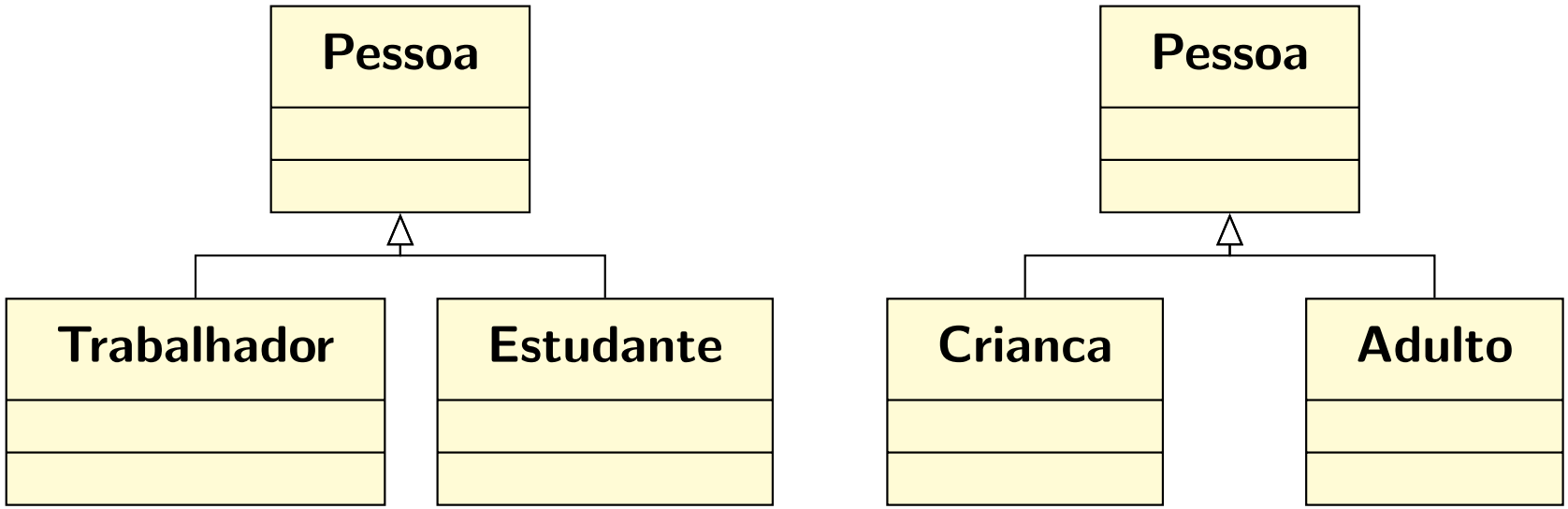
Generalização/Especialização II

- Exemplo:
  - Um automóvel é um tipo de veículo automotor;
  - Um caminhão é um tipo de veículo automotor.



Generalização/Especialização III

- **Generalização:** operação de análise de um conjunto de classes que identifica características comuns, para a definição de uma classe mais genérica, a qual especificará essas características.
- **Especialização:** operação em que, a partir de uma classe, identifica-se uma ou mais subclasses, cada uma especificando características adicionais em relação à classe mais geral.
- A abstração de generalização e especialização é dependente do contexto do problema. Exemplo:

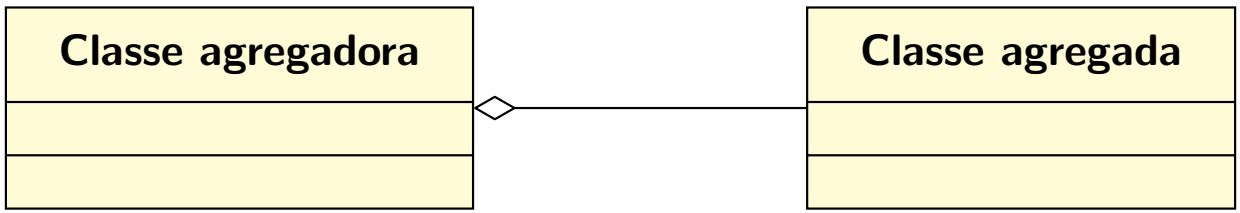


Agregação/Decomposição I

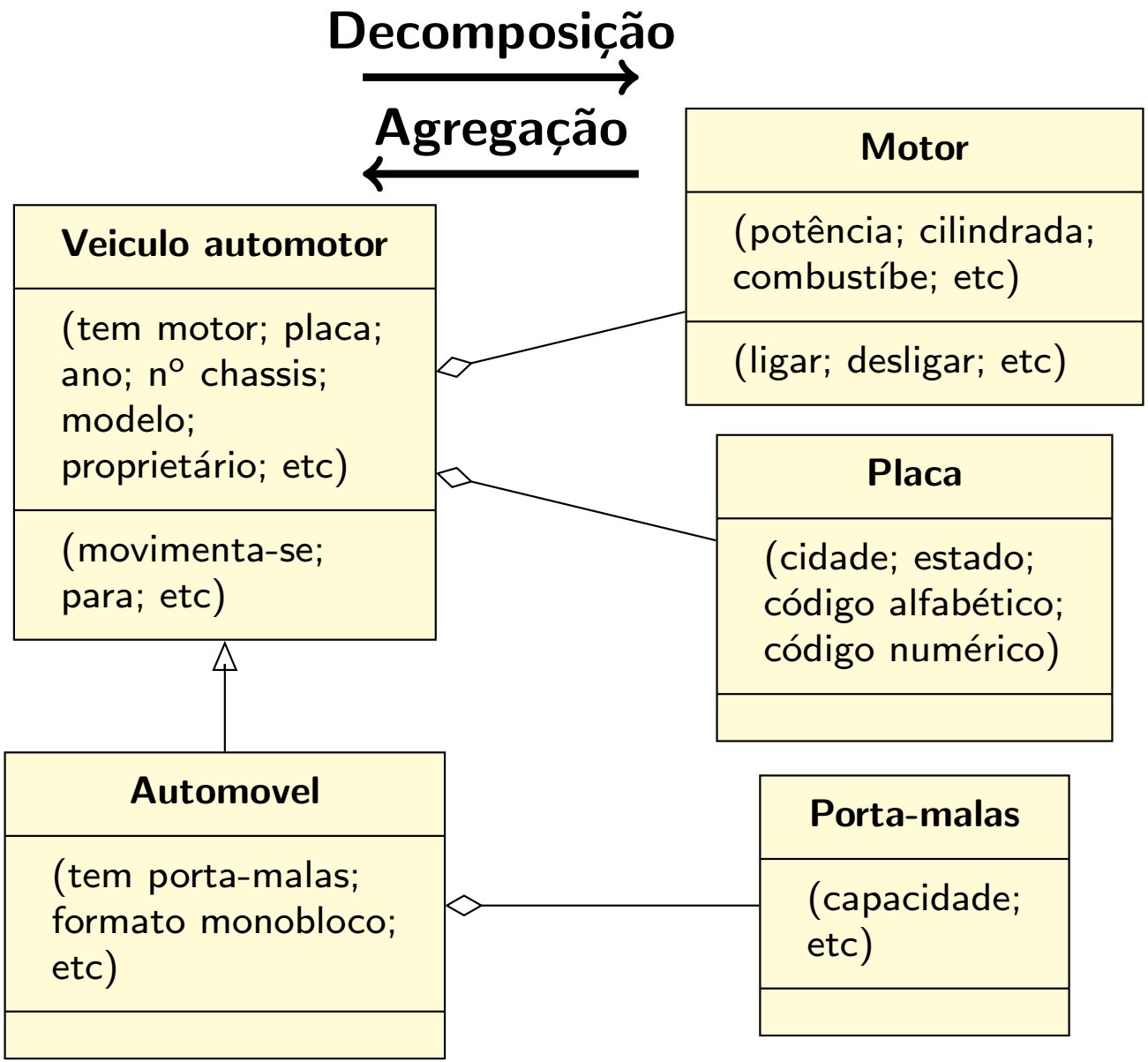
- As características de um objeto podem ser divididas em dois grupos: características de **composição** e características de **ação**.
- **Ação** diz respeito aos serviços que o objeto pode executar (ex.: Vectra da classe Automóvel, especialização da classe de Veículo automotor, pode executar o serviço movimenta-se (por herança).
- **Composição** diz respeito à constituição do objeto (ex.: se Vectra pertence à classe Automóvel, então é composto por elementos tais como: motor, porta-malas, placa, etc)
- Alguns elementos podem ser expressos como **simples valores** (no nosso caso, quantidades), tais como chassis ou ano de fabricação.
- Outros são objetos distintos pertencentes a outras classes (ex.: motor do Vectra constitui-se em um objeto, pertencente à classe Motor; o mesmo para porta-malas do Vectra que também constitui-se em um objeto).

Agregação/Decomposição II

- Um objeto, portanto, pode ser composto por outros objetos (no mundo real, isto é comum).
- Ao unir um conjunto de objetos com o objetivo de formar um novo, realiza-se uma **operação de agregação**.
- A agregação é caracterizada pela existência da relação "é composto por" ou "é um agregado de". Exemplo:
  - Automovel é composto por motor, placa, porta-malas.
  - Um motor é parte de um automóvel.
- Os objetos componentes são também denominados **Partes**, enquanto o objeto maior é denominado **Todo**. O diagrama que expressa operações de agregação/decomposição também pode ser chamado de **Diagrama Todo-Parte**.



Agregação/Decomposição III

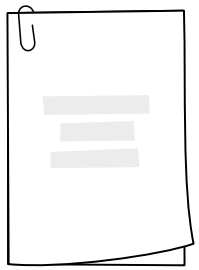


Associação

- Pode-se dizer que existe uma ligação entre as entidades professor e aluno, ou seja, a entidade professor ministra aulas para a entidade aluno, ou a entidade aluno assiste aulas da entidade professor.

- Uma **associação** consiste na descrição genérica de uma ou mais ligações entre as entidades, sendo que uma existe independentemente da outra (a agregação vista anteriormente implica em acoplamento forte, não fazendo sentido a existência do todo sem a existência da parte).

Exercícios sobre o capítulo 1



- Reflita sobre as questões de 1 a 9 do livro texto [Boratti, 2007].

Tópicos da aula

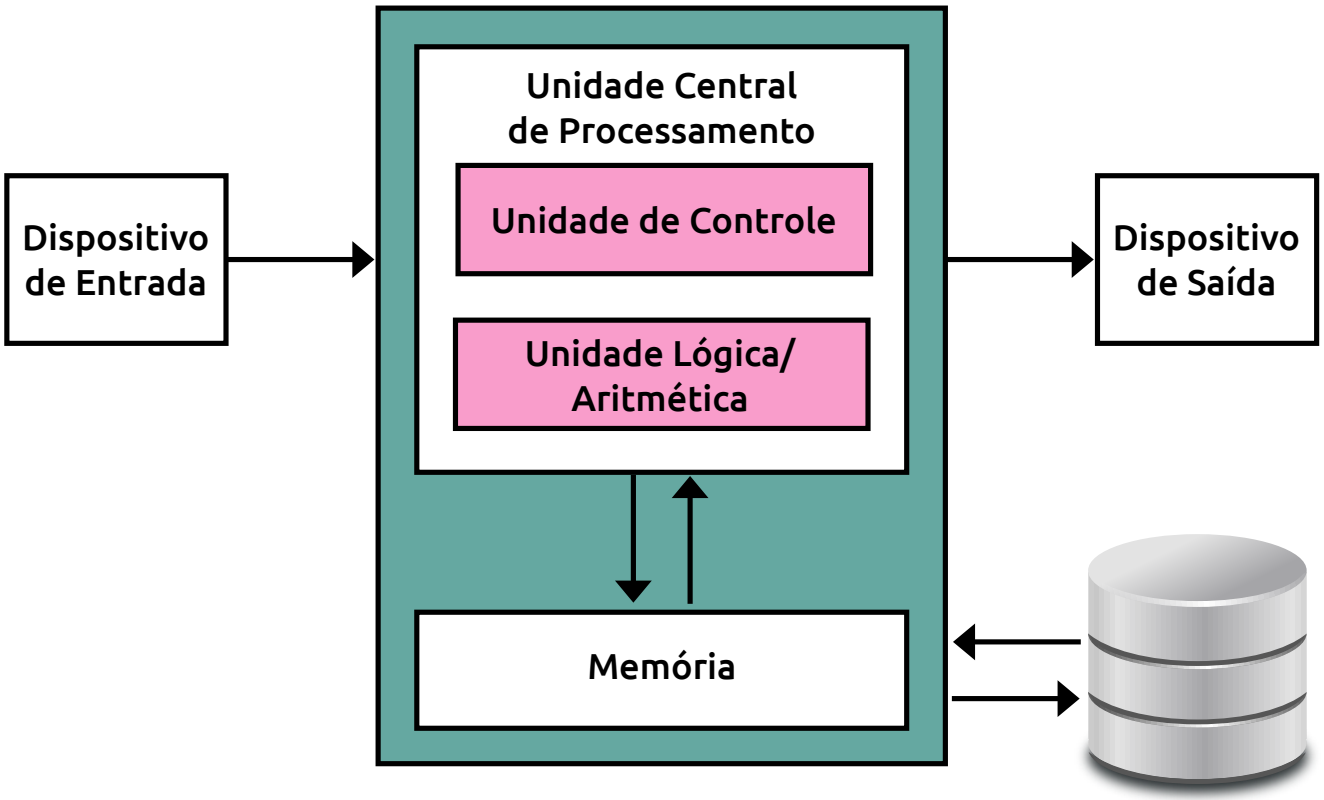
1. Modelo conceitual

2. Processos de abstração e representação
   - Introdução
   - O processo de abstração
   - Operações de abstração
     - ★ Classificação/Instanciação
     - ★ Generalização/Especialização
     - ★ Agregação/Decomposição
     - ★ Associação

3. Histórico sobre linguagens de programação

Computador

- Arquitetura de máquina de estado com memória e endereço (Modelo Von Neumann)
- Histórico... [Sobral, 2015]



Linguagens

- Dificuldade em montar programa diretamente no conjunto de instruções do processador.
- Assembly foi criado para facilitar a montagem do programa (assembler = montador).
- Linguagem de alto nível $\rightarrow$ código objeto $\rightarrow$ montador.
- Solução: Compiladores!

Assembly

rótulo: mnemônico argumento1, argumento2, argumento3 ...

- O rótulo é um identificador seguido de dois pontos.
- O mnemônico é uma palavra reservada para o código da instrução do processador.
- Os operadores "argumento" são opcionais e podem ser em número de 0 a 3, dependendo do código do processador.
- Exemplo:
  - `carr_reg:  MOV  AL, 61h  ⇒  10110000 01100001`

FORTRAN

- FORmula TRANslator – 1954-1958
- Procedural e imperativa
- Criada pela IBM (John Backus)
- Dedicada à resolução de equações e fórmulas matemáticas
- FORTRAN II: laços, funções, sub-rotinas e a primitiva do comando FOR
- Sugestão de filme: Estrelas Além do Tempo (*Hidden Figures*), 2016

FORTRAN (exemplos)

- FORTRAN90

```
nfatorial = PRODUCT((/(i, i=1,n)/))
```

- FORTRAN77

```
FUNCTION FAT(N)
      INTEGER N,I,FAT
      FACT=1
      DO 10 I=1,N
   10 FAT=FAT*I
      END
```

LISP

- LISt Processor – 1958-1960
- Funcional
- Criada por McCarthy
- Desenvolvida para processamento de listas
- Puramente recursiva e não iterativa
- Não diferencia código e dados

LISP (exemplo)

```
;; Programa fatorial em Lisp
(defun fatorial (n)
    (if (<= n 1)
        1 (* n (fatorial (- n 1)))
    )
)
```

ALGOL

- ALGOrithmic Language – 1958-1968
- Procedural, criada em 58 como IAL (International Algorithmic Language)
- Criada por comitê de especialistas em computação
- Primeira linguagem autônoma, independente de arquitetura (portável)
- Introduziu a declaração em blocos e variáveis locais, arrays dinâmicos, := para atribuição, laços, IF...THEN...ELSE, FOR, SWITCH, WHILE ALGOL 68 define cast de tipos e UNION.

ALGOL (exemplo)

```
integer procedure Fatorial(m); integer m;
begin
    integer F;
    F := if m=1 then 1 else m*Fatorial(m-1);
    Fatorial := F
end
```

COBOL

- COmmon Business Oriented Language – 1959
- Linguagem orientada para negócios e processamento de banco de dados comerciais
- Criado pelo Departamento de Defesa Norte-Americano sob direção de Grace Murray Hopper

COBOL (exemplo)

```
IDENTIFICATION DIVISION.
FUNCTION-ID. fatorial.

DATA DIVISION.
LOCAL-STORAGE SECTION.
01  i       PIC 9(10).

LINKAGE SECTION.
01  n       PIC 9(10).
01  ret     PIC 9(10).

PROCEDURE DIVISION USING BY VALUE n RETURNING ret.
    MOVE 1 TO ret

    PERFORM VARYING i FROM 2 BY 1 UNTIL n < i
        MULTIPLY i BY ret
    END-PERFORM

    GOBACK.
```

BASIC

- Beginers All-purposes Symbolic Instruction Code – 1963-1964
- Procedural
- Criada por John Kemeny e Thomas Kurtz (não pelo Bill Gates como citam algumas fontes).
- Originalmente, código de linhas numeradas e subrotinas chamadas por linha (GOTO e GOSUB)

BASIC (exemplo)

```
10 LET x=5: GO SUB 1000: PRINT "5! = ";r
999 STOP
1000 REM ************
1001 REM * FATORIAL *
1002 REM ************
1010 LET r=1
1020 IF x<2 THEN RETURN
1030 FOR i=2 TO x: LET r=r*i: NEXT i
1040 RETURN
```

C

- Imperativa, procedural, de propósito geral – 1969-1973
- Da linhagem do ALGOL, desenvolvida originalmente por Denis Ritchie.
- Destinada à programação de sistemas Unix, a partir do BCPL (*Basic Combined Programming Language*, 1965) e B (contração de BCPL, 1967) desenvolvidas pela Bell Labs.
- Padronizada em 1973 (ANSI C).
- Conceito de blocos, bibliotecas (headers) de funções, array, pointers e casting de tipos.
- É a linguagem mais utilizada até hoje.

C (exemplo)

```
int fatorial(int n) {
    int result = 1;
    for (int i = 1; i <= n; ++i)
        result *= i;
    return result;
}
```

HASKELL

- Nome em homenagem a Haskell Curry – 1990-1998
- Funcional (diretamente derivada da ML)
- Criada pela Universidade de Glasgow
- Linguagem puramente funcional e baseada em cálculo lambda tipado

HASKELL (exemplo)

```
-- Fatorial em Haskell
module Main () where

main = print (fatorial 20)

fatorial 0 = 1
fatorial n = n * fatorial (n-1)
```

Python

- Multiparadigma (orientada a objetos, imperativa, funcional) – 1991
- Nome em homenagem ao grupo humorístico e não à cobra
- Criada por Guido van Rossum
- Interpretada
- Tipagem dinâmica

Python (exemplo)

```
def factorial(n):
    result = 1
    for i in range(1, n+1):
        result *= i
    return result
```

Java

- Green Project (convergência entre computadores e eletrodomésticos) da Sun Microsystems – 1991
- *7 (StarSeven) - controle remoto com interface toutchscreen (mascote Duke) – 1992
- Oak (carvalho) como nova especificação por James Gosling – vídeo por demanda e programas interativos (muito cedo para época)
- Java consistiu numa adaptação do Oak para a internet – 1995
  - Projetada para se mover por redes de dispositivos heterogêneos
  - Aplicações executadas nos navegadores (Applets Java)
- Orientada a objetos, portabilidade, recursos de rede, segurança
- Sintaxe similar a C/C++, Unicode nativo
- Vasto conjunto de bibliotecas (API)
- Facilidade para programas distribuídos e multitarefas
- Desalocação de memória automática por processo de coletor de lixo

Java (exemplo)

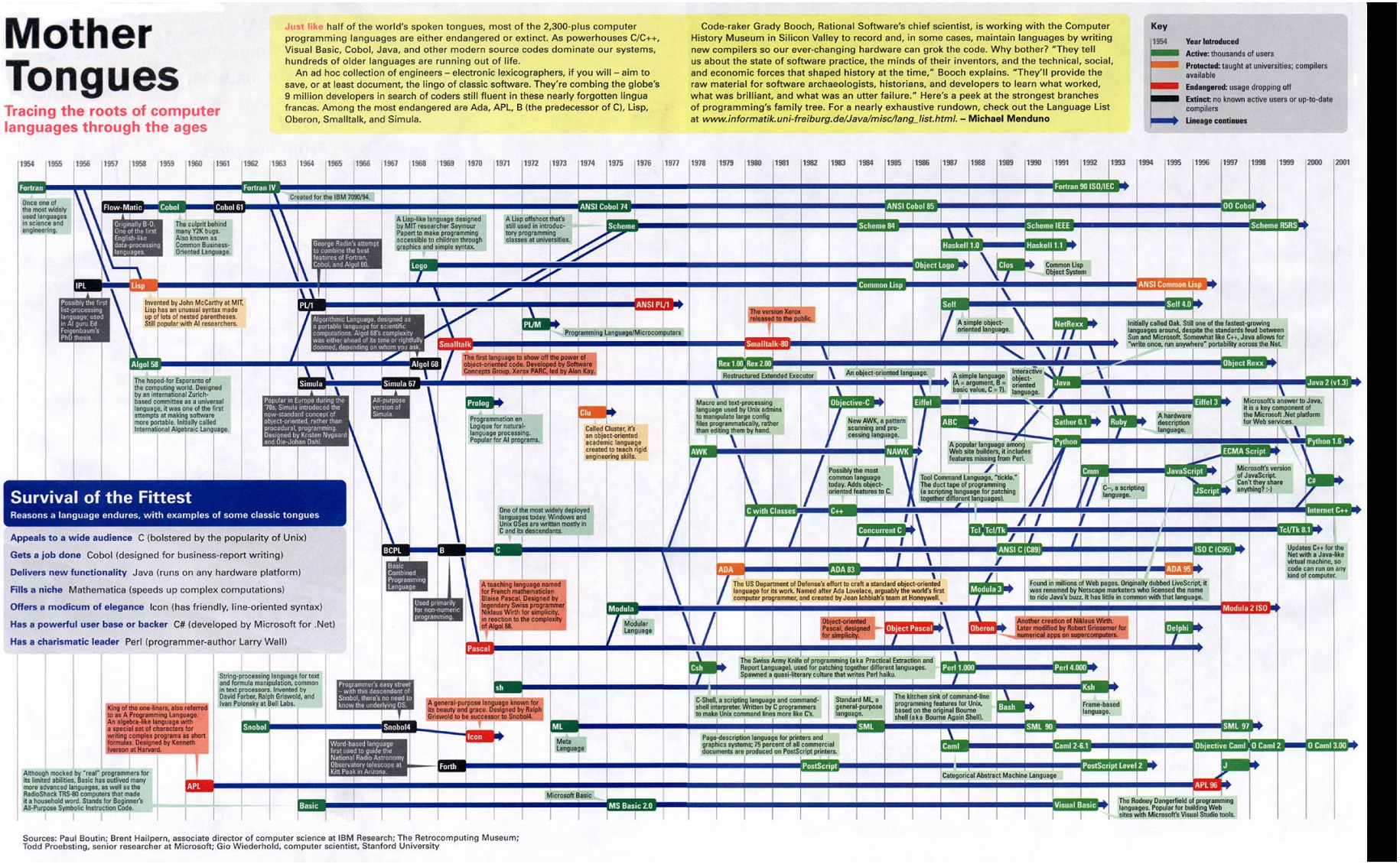
```
public static long fatorial(final int n) {
    if (n < 0) {
        System.err.println("No negative numbers");
        return 0;
    }
    long ans = 1;
    for (int i = 1; i <= n; i++) {
        ans *= i;
    }
    return ans;
}
```

Décadas 1970-2000

- Várias linguagens derivadas das anteriores
  - **Imperativas:** Forth - Modula 2 - Perl - PHP - Javascript - C# ...
  - **Orientada a objetos:** Smalltalk - ADA - C++ - Eiffel - Ruby ...
  - **Declarativas:** Prolog - SQL - HTML - UML - XML - Scheme - ML - Miranda - Haskell - O'Haskell (OOP) - Clean - CAML - OCAML (OOP) - UML ...

- Objetivo de todas: gerar código em mais baixo nível para a instrução da máquina.

Genealogia das linguagens



Exemplos iniciais em Java

- Método `main`

```
public class OlaMundo {
    /*
     * Metodo que executa o programa
     * public = e' visto em qualquer lugar da aplicacao
     * static = e' iniciado automaticamente pela JVM, sem
         precisar de uma instancia, sendo comum a todos
        os objetos da mesma classe
     * void = metodo sem retorno (retorno vazio)
     * main = nome do metodo (este e' obrigatorio),
        recebe como parametro um array de String.
     * String[] args = array de argumentos que podem ser
        repassados na chamada do programa.
     */
    public static void main(String[] args) {
        System.out.println("Ola, Mundo!"); //Imprime na
            tela a frase
    }
}
```

Exemplos iniciais em Java

- Classe

```
public abstract class Animal {
    public abstract void fazerBarulho();
}

public class Cachorro extends Animal {
    public void fazerBarulho() {
        System.out.println("AuAu!");
    }
}

public class Gato extends Animal {
    public void fazerBarulho() {
        System.out.println("Miau!");
    }
}
```

Ferramentas

- Compiladores
  - Javac, incluido na Java Development Kit (JDK) da Oracle Corporation
  - GCJ (GNU Compiler for Java) é um compilador estático parte do GCC
- Ambientes de desenvolvimento: BlueJ, JCreator, jEdit, Eclipse
- IDE completas (programas profissionais): Eclipse (IBM), JBuilder (Borland), JDeveloper (Oracle), NetBeans (Sun Microsystems)

Referências Bibliográficas I

Boratti, I. C. (2007).

Programação Orientada a Objetos em Java.

Visual Books, 1 edition.

Sobral, J. B. M. (2015).

Volume II: Da Computabilidade Formal às Máquinas Programáveis.

Edição do Autor, 1 edition.