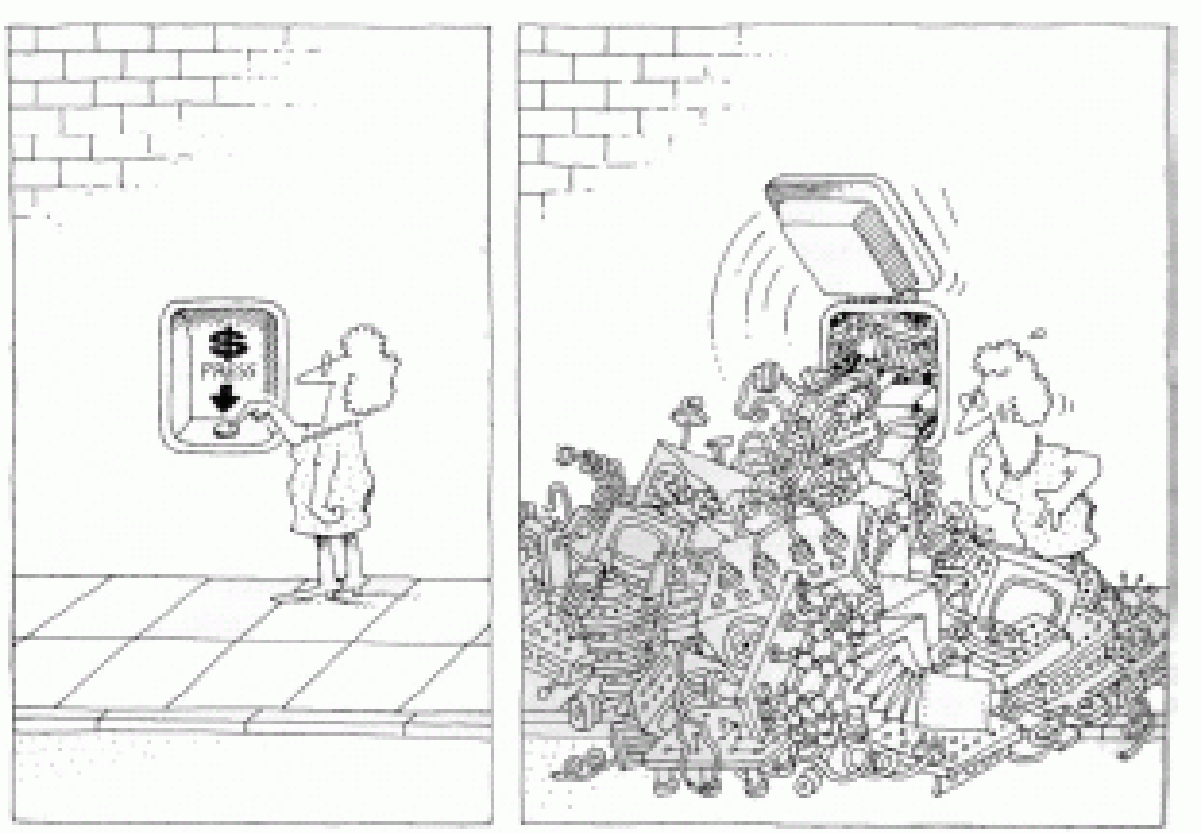
Programação Orientada a Objetos

Profa Andréa Schwertner Charão

DLSC/CT/UFSM

Motivação



The task of the software development team
is to engineer the illusion of simplicity.

Booch et al., 2007

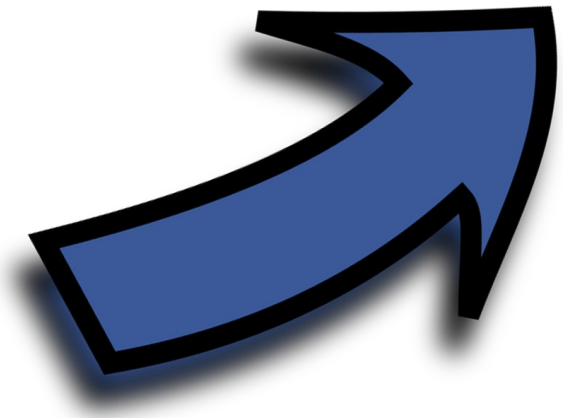
Motivação

Evolução da programação:
programas melhores, mas...

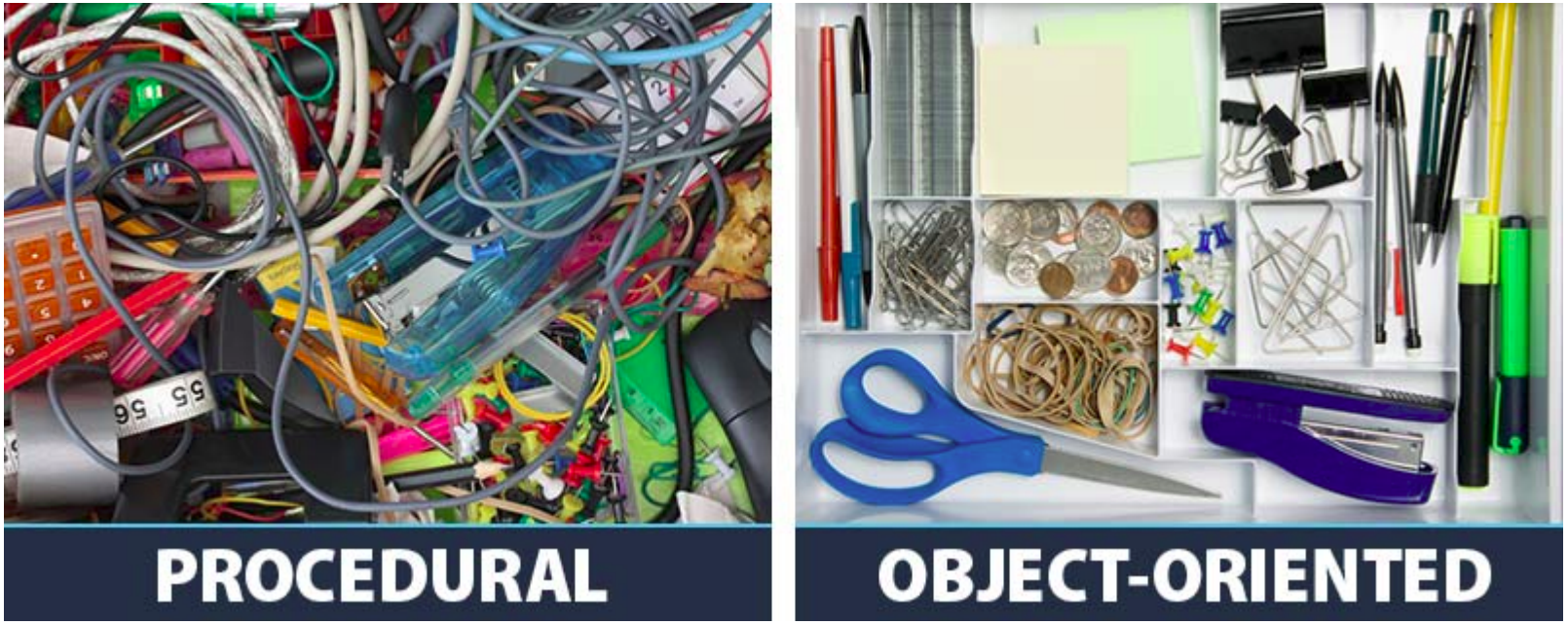
complexos, grandes
difíceis de ler e alterar
pouco reuso de código

Evolução da programação:
programas
REALMENTE
melhores

programas:
mais organizados
mais reusáveis



Motivação



https://www.zionandzion.com

Evolução da POO

Mother Tongues

Tracing the roots of computer languages through the ages

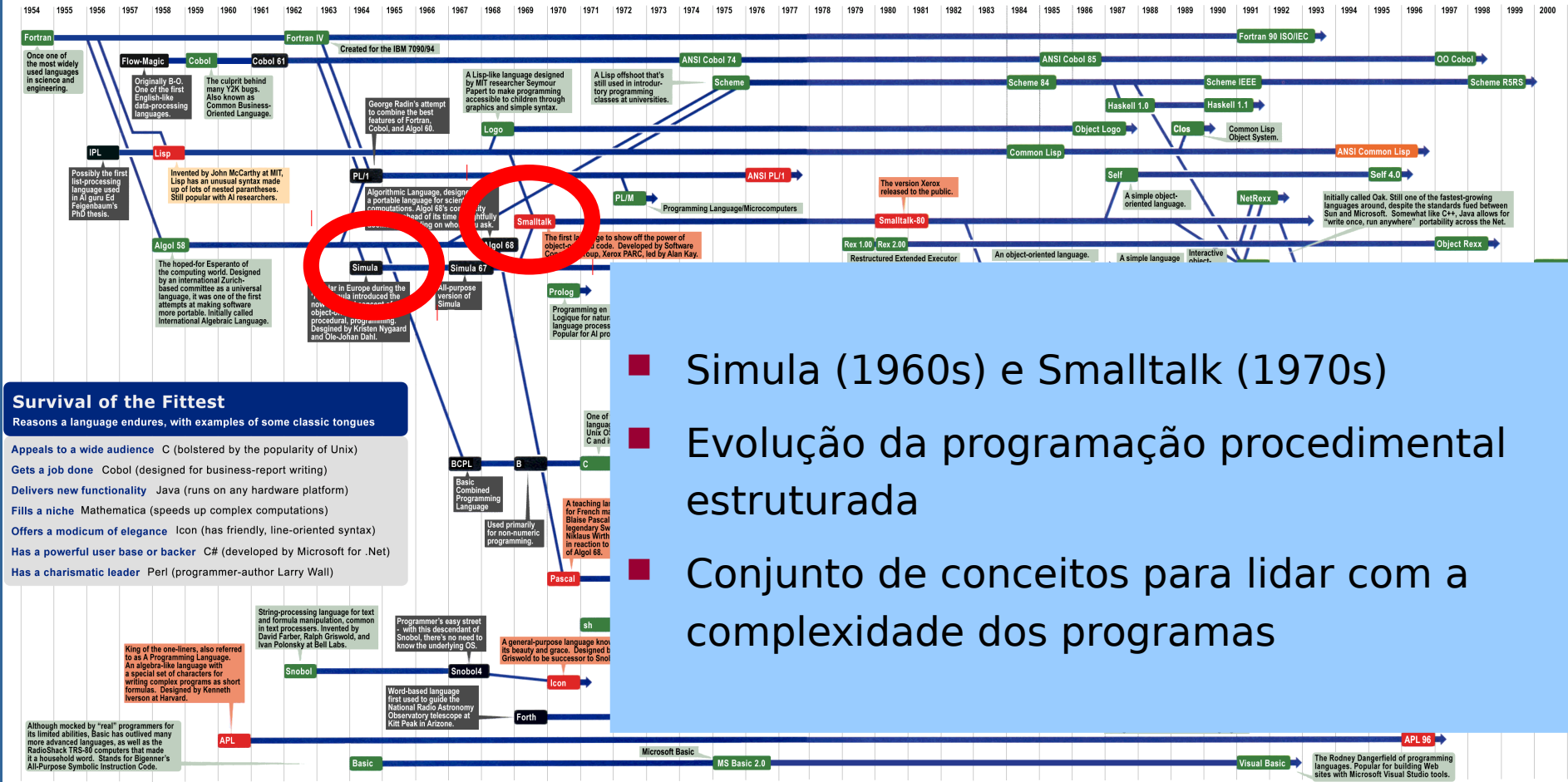
Just like half of the world's spoken tongues, most of the 2,300-plus computer programming languages are either endangered or extinct. As powerhouses C/C++, Visual Basic, Cobol, Java and other modern source codes dominate our systems, hundreds of older languages are running out of life.

An ad hoc collection of engineers-electronic lexicographers, if you will-aim to save, or at least document the lingo of classic software. They're combing the globe's 9 million developers in search of coders still fluent in these nearly forgotten lingua frangas. Among the most endangered are Ada, APL, B (the predecessor of C), Lsp, Oberon, Smalltalk, and Simula.

Code-raker Grady Booch, Rational Software's chief scientist, is working with the Computer History Musuem in Silicon Valley to record and, in some cases, maintain languages by writing new compilers so our ever-changing hardware can grok the code. Why bother? "They tell us about the state of software practice, the minds of their inventors, and the technical, social, and economic forces that shaped history at the time," Booch explains. "They'll provide the raw material for software archaeologists, historians, and developers to learn what worked, what was brilliant, and what was an utter failure." Here's a peek at the strongest branches of programming's family tree. For a nearly exhaustive rundown, check out the Language List at *HTTP://www.informatik.uni-freiburg.de/Java/misc/lang_list.html.* - Michael Mendeno

Key

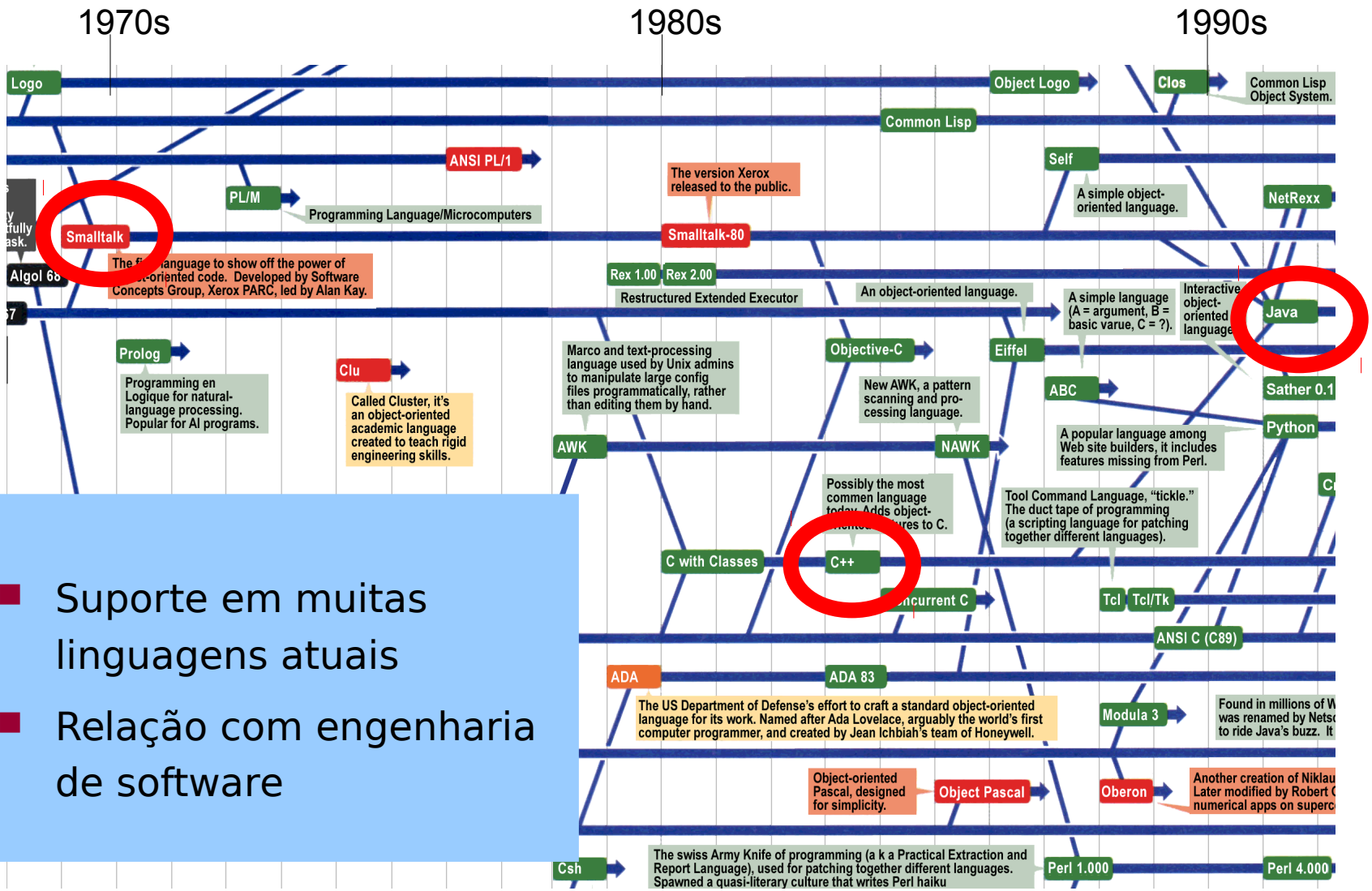
- 1954 Year Introduced
- **Active:** thousands of users
- **Protected:** taught at universities; compilers available
- **Endangered:** usage dropping off
- **Extinct:** no known active users or up-to-date compilers
- **Lineage continues**



- Simula (1960s) e Smalltalk (1970s)
- Evolução da programação procedimental estruturada
- Conjunto de conceitos para lidar com a complexidade dos programas

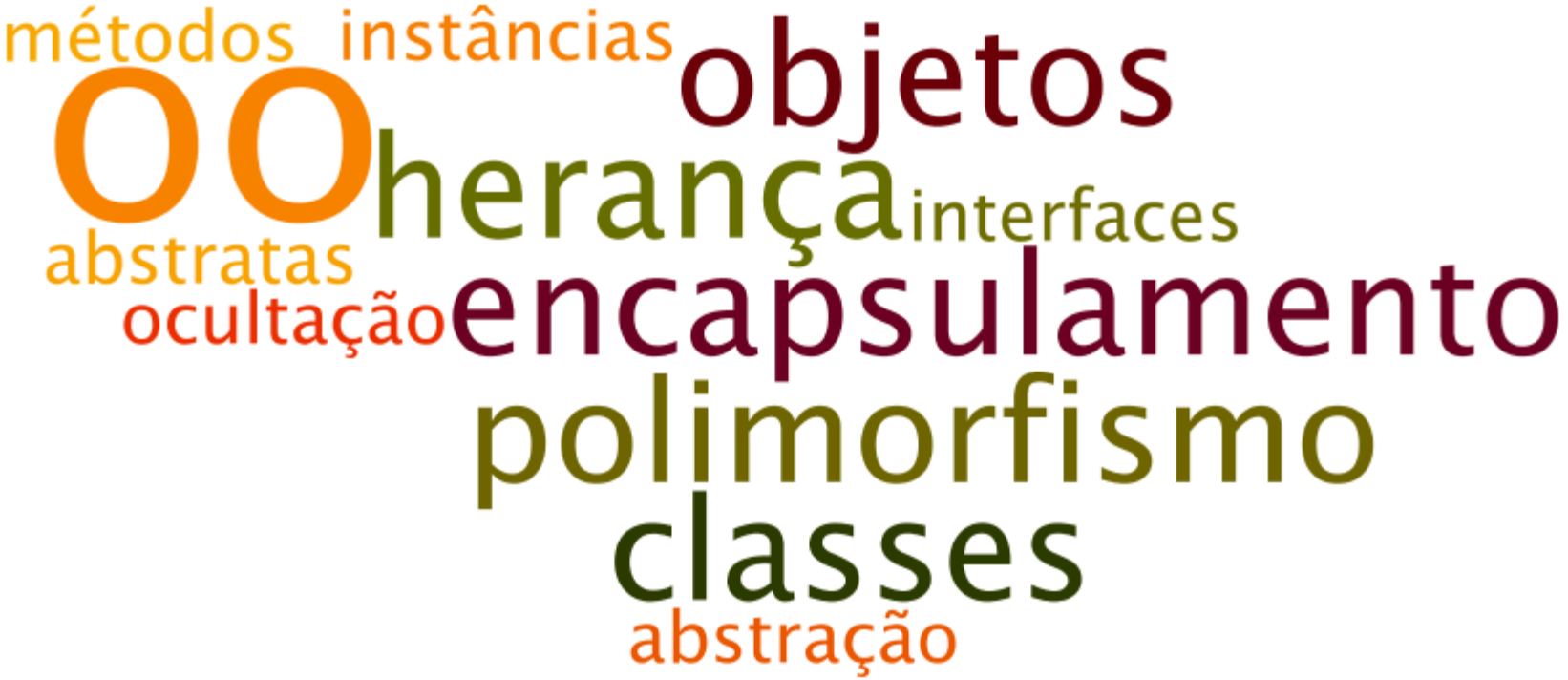
Sources: Paul Boutin; Brent Hailpern, associate director of computer science at IBM Research; The Retrocomputing Museum; Todd Proebsting, senior researcher at Microsoft; Gio Wiederhold, computer scientist, Stanford University

Evolução da POO



- Suporte em muitas linguagens atuais
- Relação com engenharia de software

POO: conceitos fundamentais



http://www.wordle.net/show/wrdl/3702615/OO

POO: conceitos fundamentais

- **Abstração**
- Encapsulamento
- Herança
- Composição (EngSW)
- Polimorfismo

http://www.metropoledigital.ufrn.br

Abstração

- Software formado por objetos do mundo real

Exemplos

Características: (dados, atributos)
tipo: Ferrari
placa: KZE1018
cor: vermelha
número de portas: 2

Comportamentos: (operações, métodos)
ligar
desligar
acelerar
frear

Características:
nome: Camila
cor do cabelo: negro
biotipo: magro

Comportamentos:
andar
correr
dirigir Carro

Figura 5 –Exemplos de objetos

http://www.metropoledigital.ufrn.br

Abstração

- Considerar apenas os atributos importantes no contexto em questão
- Ex.:

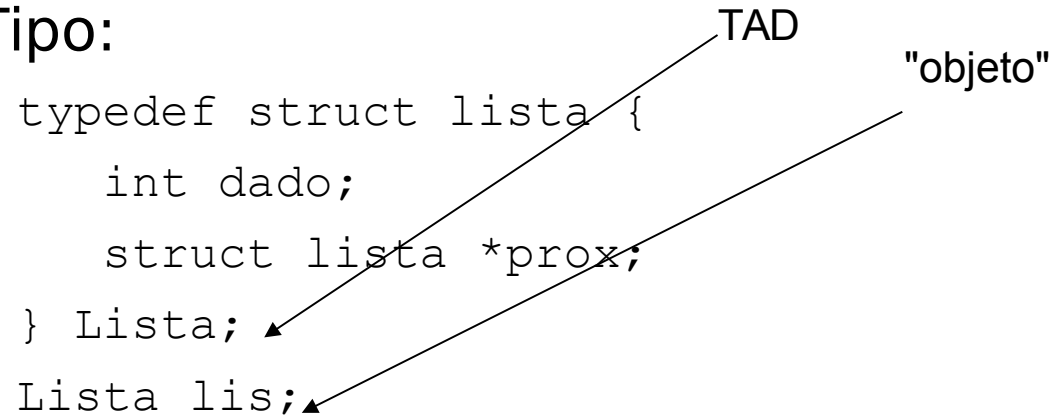
```
strcmp(str1, str2);
quicksort(myarray, 0, n);
```

para usar estas funções, não é necessário saber como elas foram implementadas !

- Simplifica o desenvolvimento de programas
- Facilita reutilização

Tipos abstratos de dados

- Código composto por
  - representação de um tipo de dado
  - subprogramas que implementam operações com esse tipo
- **"Objeto"**: instância (concreta) de um TAD
- Exemplo: Tipo:

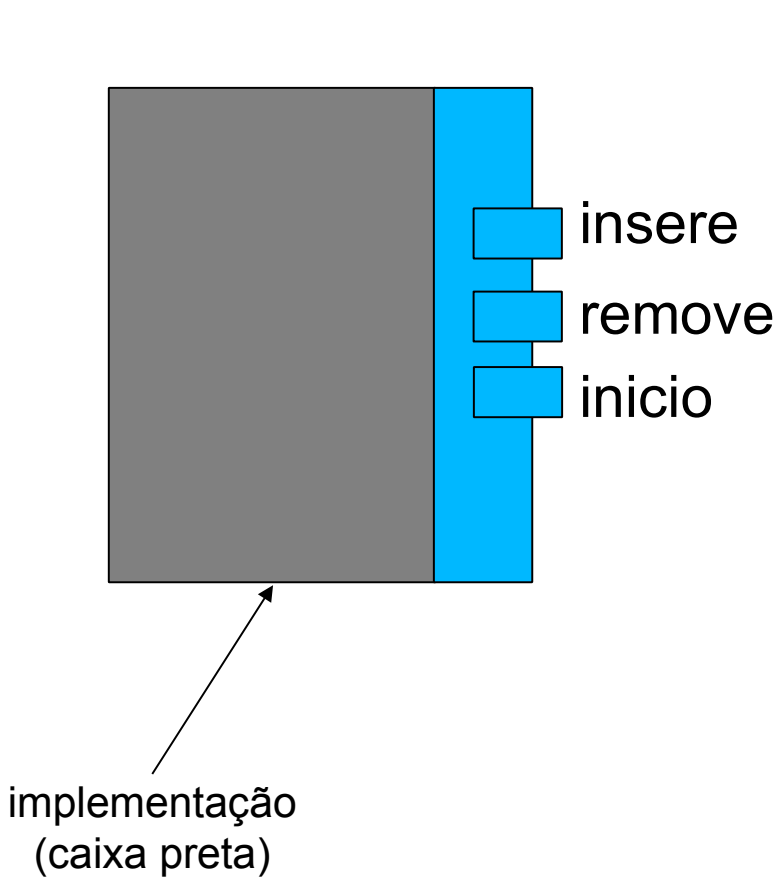



```
typedef struct lista {
   int dado;
   struct lista *prox;
} Lista;
Lista lis;
```

Operação:

```
void insere(Lista* lis, int i)
```

Encapsulamento



- Interface do TAD acessível ao código usuário
- Implementação do TAD oculta do código usuário

Encapsulamento

implementação

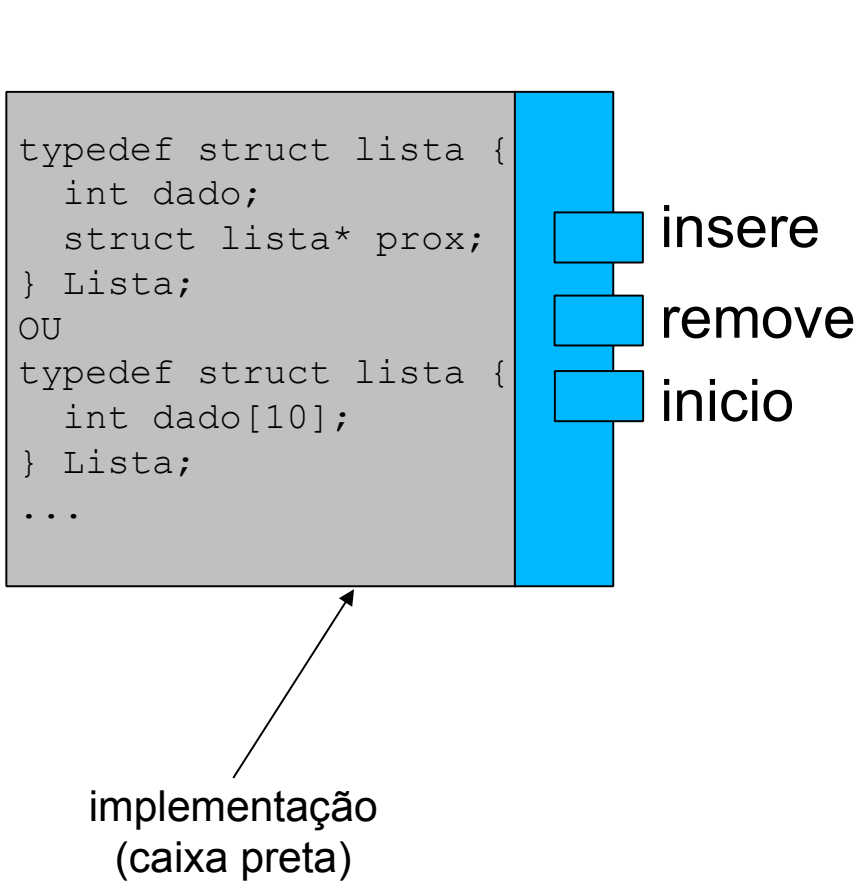
interface

implementação

interface

- Interface do TAD acessível ao código usuário
- Implementação do TAD oculta do código usuário

Encapsulamento



- Interface do TAD acessível ao código usuário
- Implementação do TAD oculta do código usuário

Encapsulamento

- Facilita reuso e manutenção de código

```
typedef struct lista {
   int dado;
   struct lista* prox;
} Lista;
typedef struct lista {
   int dado[10];
} Lista;
void insere(Lista* lis, int dado);
```

alternativas de implementação:
alocação encadeada
OU
alocação contígua (array)
OU
etc.

```
Lista lis;
insere(&lis, 250);
lis.prox = ...; ?????
```

código usuário não deveria fazer isso!

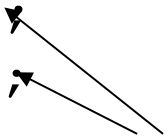
Encapsulamento

- Outro exemplo em linguagem C

```
typedef struct {
    int dia;
    int mes;
    int ano;
} data_t;

int data_inicializa(data_t* d,
                    int dia, int mes, int ano);
bool data_igual(data_t* d1, data_t* d2);
void data_imprime(data_t* d);
int data_diferenca(data_t* d1, data_t* d2);

int main()
{
    data_t data_nasc, data_formatura;
}
```

Encapsulamento

```
int data_inicializa(data_t* d,
                    int dia, int mes, int ano)
{
   /* verifica se dia, mes e ano sao validos */
}

int main()
{
   data_t data_nasc, data_formatura;
   data_nasc.dia = 20;
   data_nasc.mes = 20;
}
```



violação do encapsulamento!

Encapsulamento

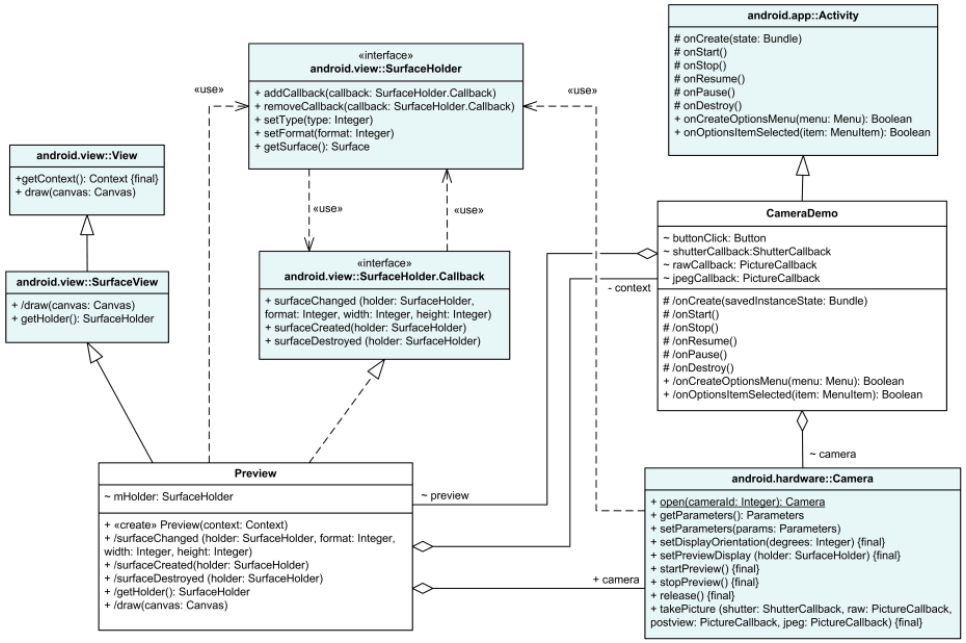
- Em C, encapsulamento não faz parte da linguagem
- Linguagens orientadas a objetos fornecem mecanismos de encapsulamento
- TAD + encapsulamento = classe

Classe "Lista"

Notação UML (Engenharia de Software)

Desafios da POO

- Decompor a solução de um problema em objetos de classes que se relacionam
- Definir classes reutilizáveis
- Escolha da linguagem



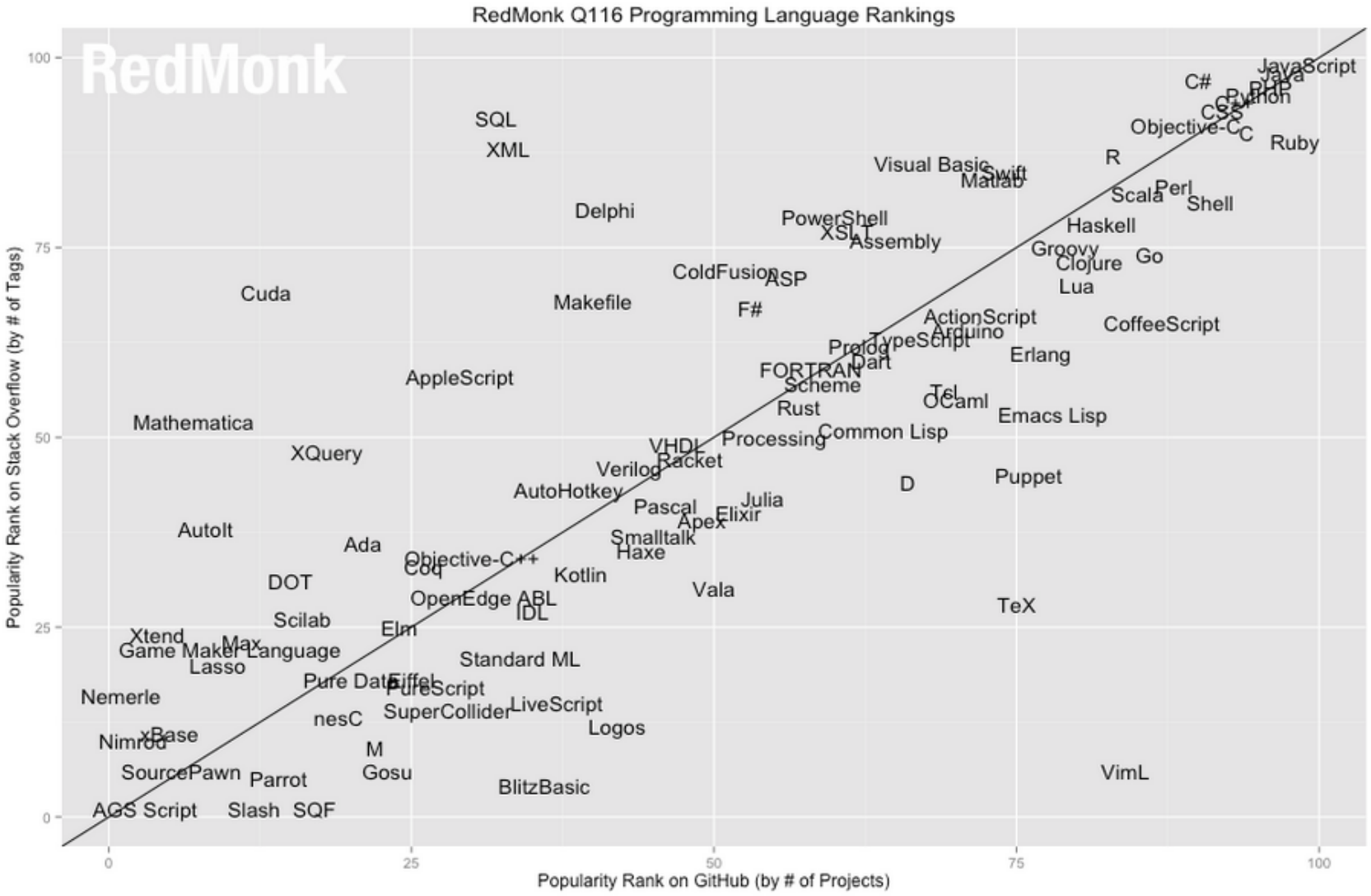
C++, Java, Python, C#, PHP, Objective-C, etc.

Linguagens de POO (ou não)

Programming Language	2016	2011	2006	2001	1996	1991	1986
Java	1	1	1	3	25	-	-
C	2	2	2	1	1	1	1
C++	3	3	3	2	2	2	7
C#	4	5	6	12	-	-	-
Python	5	6	7	26	18	-	-
PHP	6	4	4	10	-	-	-
Visual Basic .NET	7	188	-	-	-	-	-
JavaScript	8	9	8	9	31	-	-
Perl	9	8	5	4	3	-	-
Ruby	10	10	24	31	-	-	-
Lisp	27	12	13	16	6	3	2
Ada	28	16	15	19	7	4	3

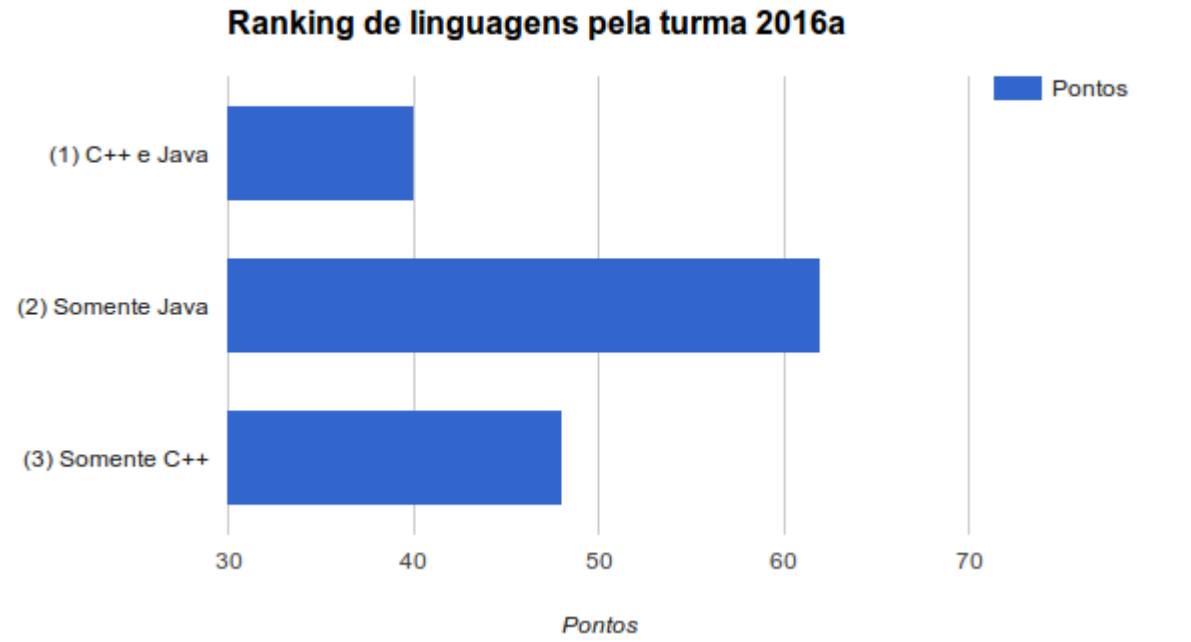
http://www.tiobe.com

Linguagens de POO (ou não)



Linguagens de POO

Qual linguagem usaremos para estudar orientação a objetos?



Mais sobre isso em...

- [E-book] Sebesta, R. **Conceitos de Linguagens de Programação.** Bookman, 2011. Capítulo 12: Suporte para a Programação Orientada a Objetos