Programação Orientada a Objetos

Profa Andréa Schwertner Charão

DELC/CT/UFSM

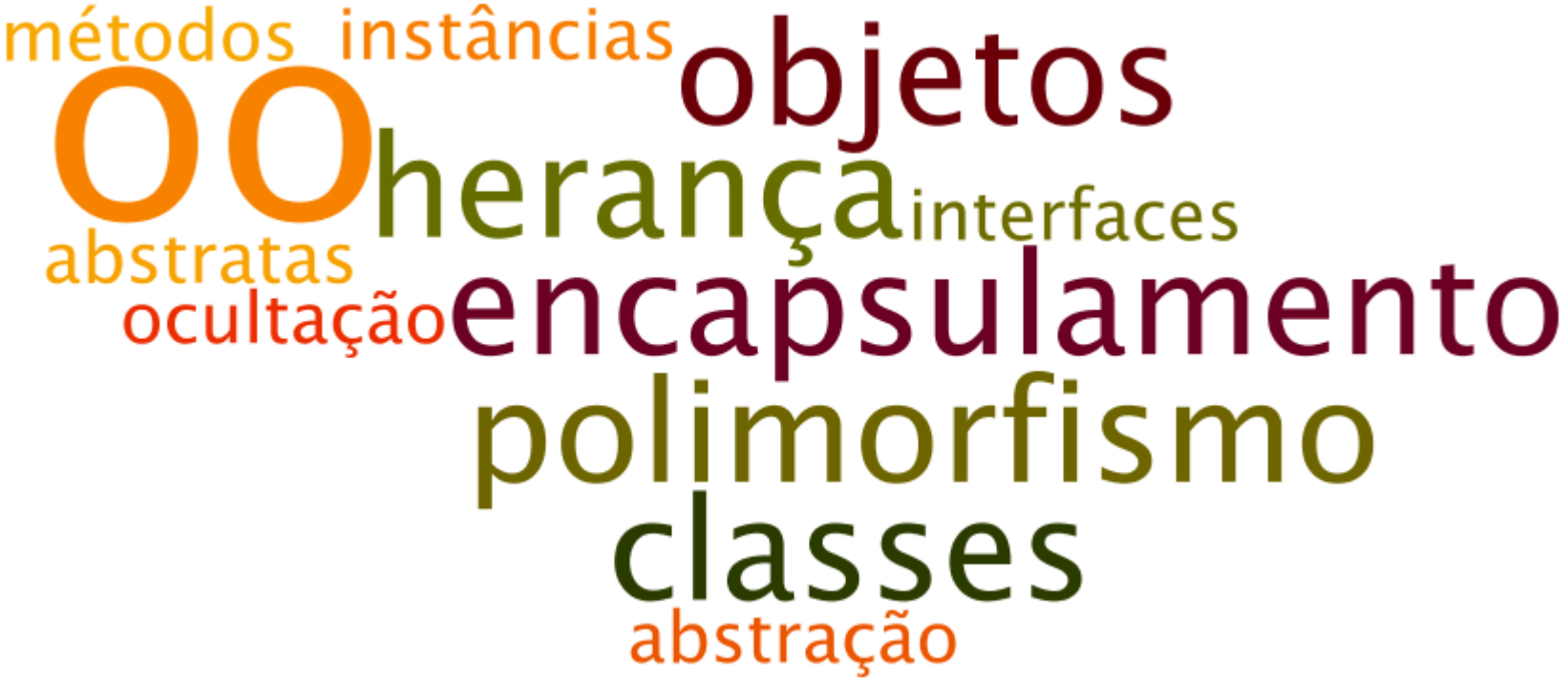
Sumário

- Introdução à POO
  - abstração
  - encapsulamento, ocultação
  - tipos abstratos de dados
- Linguagem Java
  - Comparação com C/C++
  - Tipos abstratos de dados
  - Plataforma Java

Evolução da POO

- Smalltalk (1970s)
- Evolução da programação procedimental estruturada
- Conjunto de conceitos para lidar com a complexidade dos programas
- **Reutilização** e legibilidade
- Análise, projeto e programação orientados a objetos
- Suporte em muitas linguagens atuais

POO: conceitos fundamentais



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

POO: conceitos fundamentais

- Abstração
- **Encapsulamento**
- Ocultação da informação
- Classes, objetos, métodos, instâncias
- **Polimorfismo**
- **Herança**
- Classes abstratas
- Interfaces

Abstração

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

```
strcmp(str1, str2)
ordena(lista, tam_lista)
```

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

- Abordagem para lidar com a complexidade da programação
- Simplifica o desenvolvimento de programas
- Facilita reutilização

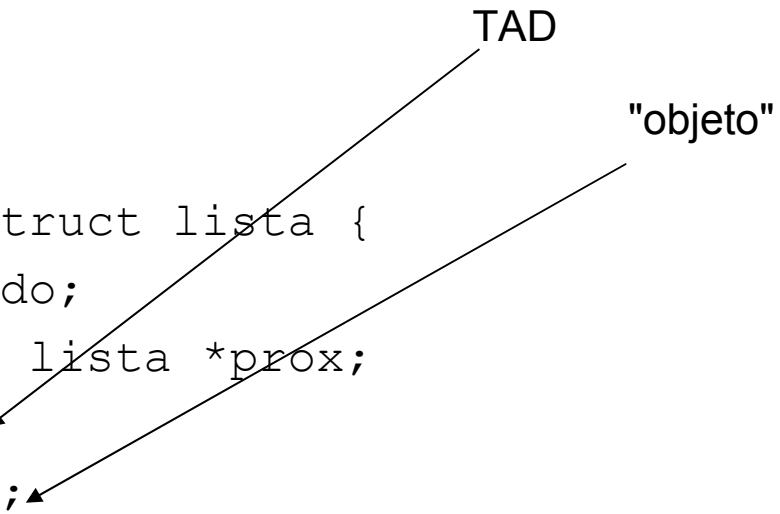
Encapsulamento

- Agrupamento de subprogramas e de dados que estes manipulam
- Diz respeito à organização dos programas: **modularização**
- Modularização
  - essencial no caso de programas de grande porte
  - evita recompilação de código
- Encapsulamentos são frequentemente colocados em bibliotecas **reutilizáveis**

Tipos abstratos de dados

- Encapsulamento que contém
  - representação de um tipo de dado
  - subprogramas que implementam operações com esse tipo
- **Objeto**: instância (concreta) de um TAD
- Exemplo:
  - Tipo:

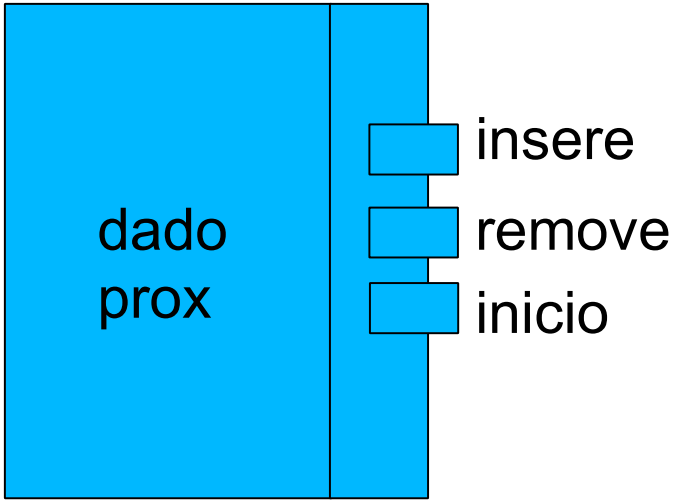
```
typedef struct lista {
   int dado;
   struct lista *prox;
} Lista;
Lista lis;
```



  - Operação:

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

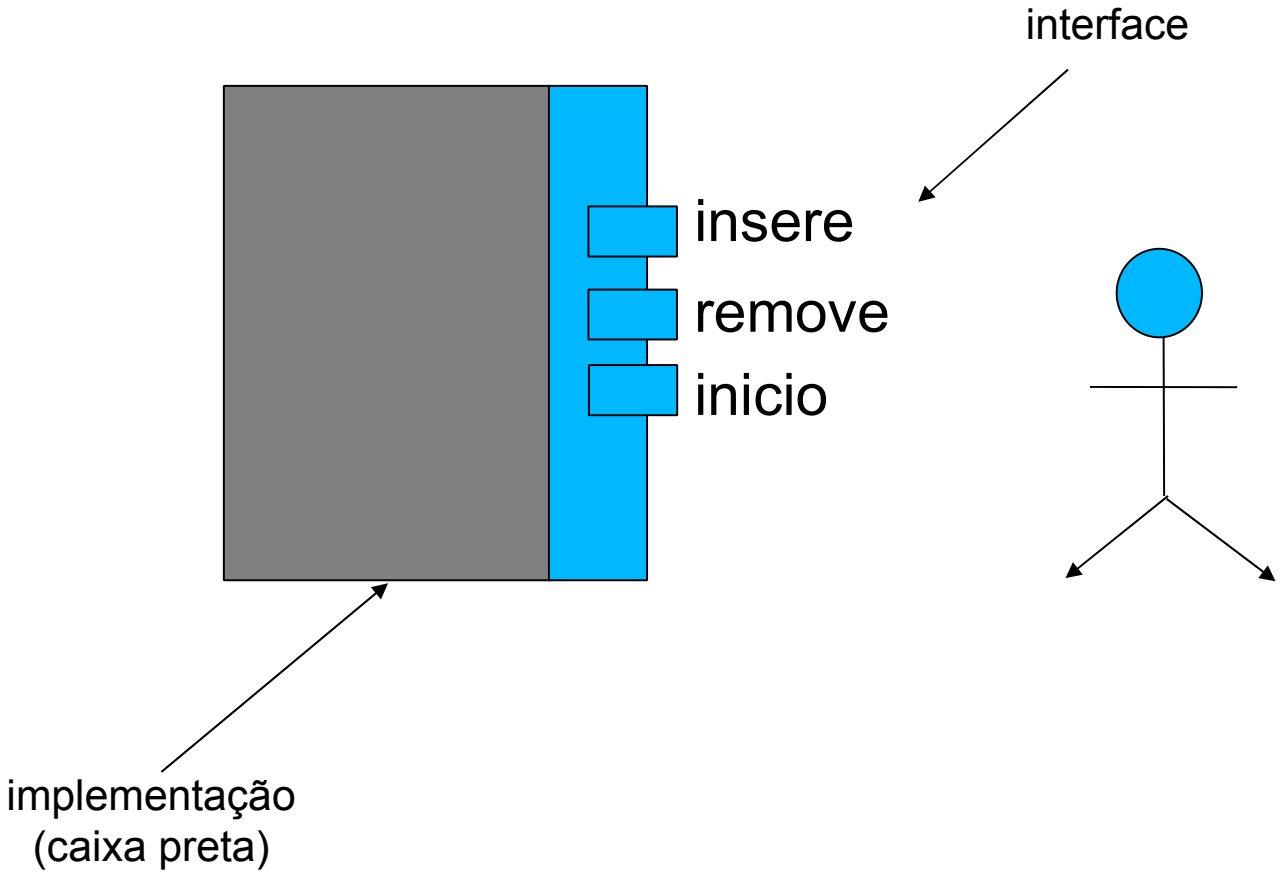
Exemplo: tipo abstrato Lista



Ocultação da informação

- Restrição da visibilidade externa dos detalhes de implementação
- Nome do tipo e operações de acesso ao tipo ficam visíveis aos usuários do TAD
- Interface X implementação
- Vantagens
  - Simplifica a programação
  - Mudanças na implementação não exigem alteração nos programas

Exemplo: ocultação



Tipos abstratos de dados

- TADs definidos pelo usuário
- Exemplo em linguagem C:

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

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

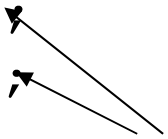
alternativas de implementação:
alocação encadeada
OU
alocação contígua (array)
OU
etc.

Tipos abstratos de dados: exemplo em 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;
}
```

Tipos abstratos de dados

```
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 confiável

- Programas não devem mudar a representação/estado dos objetos diretamente
- Integridade dos objetos
- Em C, isso não é garantido pela linguagem, mas sim pelo programador
- Linguagens orientadas a objetos fornecem mecanismos confiáveis de encapsulamento e ocultação de informação

Desafios da POO

- Decompor a solução de um problema em termos de tipos abstratos de dados que se relacionam entre si
- Definir TADs que possam ser reutilizados em diferentes programas
- Escolha da linguagem

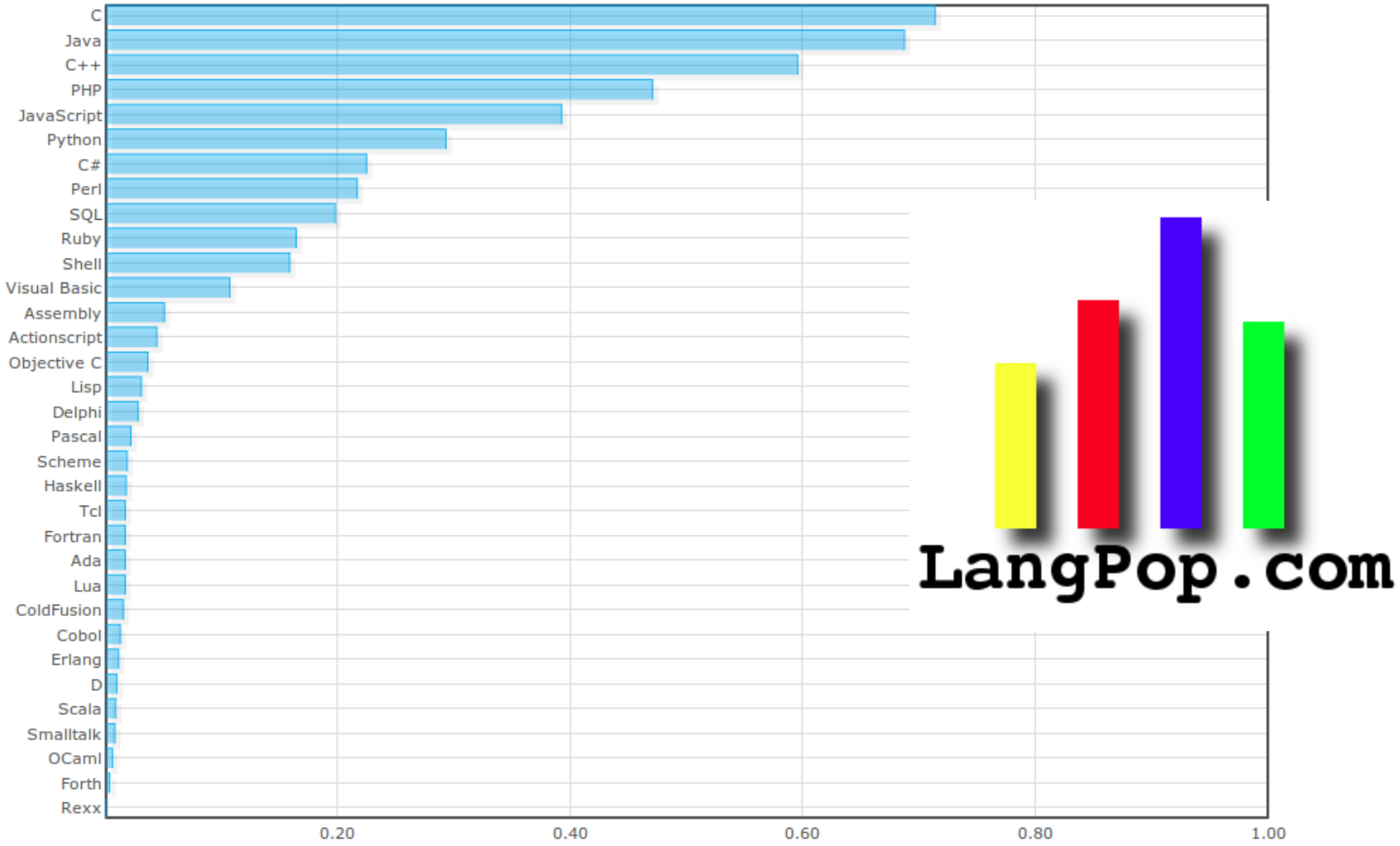
Projeto de TADs

- Nome do tipo visível externamente
- Definição do tipo e implementação das operações são ocultos
- Operações gerais
  - construtores (inicialização) e destrutores
  - atribuição
  - comparação (igualdade/desigualdade)

Linguagens de POO

- Muitas linguagens têm suporte à POO mas também permitem a programação procedimental (C++, Python, etc.)

Linguagens de POO (ou não)



Introdução a Java

- Linguagem semelhante a C++, mais recente e mais simples
- Portabilidade (“*write once, run anywhere*”) obtida através de código objeto intermediário (*bytecode*) interpretado por máquinas virtuais (JVMs)
- Amplo conjunto de APIs (classes reutilizáveis) para várias áreas de aplicação (p.ex.: programação para Web, interfaces gráficas, etc.)
- Suporte a concorrência

Semelhanças e diferenças básicas entre Java e C/C++

- Estruturas de repetição, seleção e seqüência são praticamente idênticas
- Tipos de dados primitivos são basicamente os mesmos (short, int, long, float, double, char), com algumas diferenças:
  - não há números unsigned
  - os tamanhos são sempre os mesmos em qualquer arquitetura
  - tipos boolean e byte
- Arrays e strings são sempre objetos

Semelhanças e diferenças básicas entre Java e C/C++

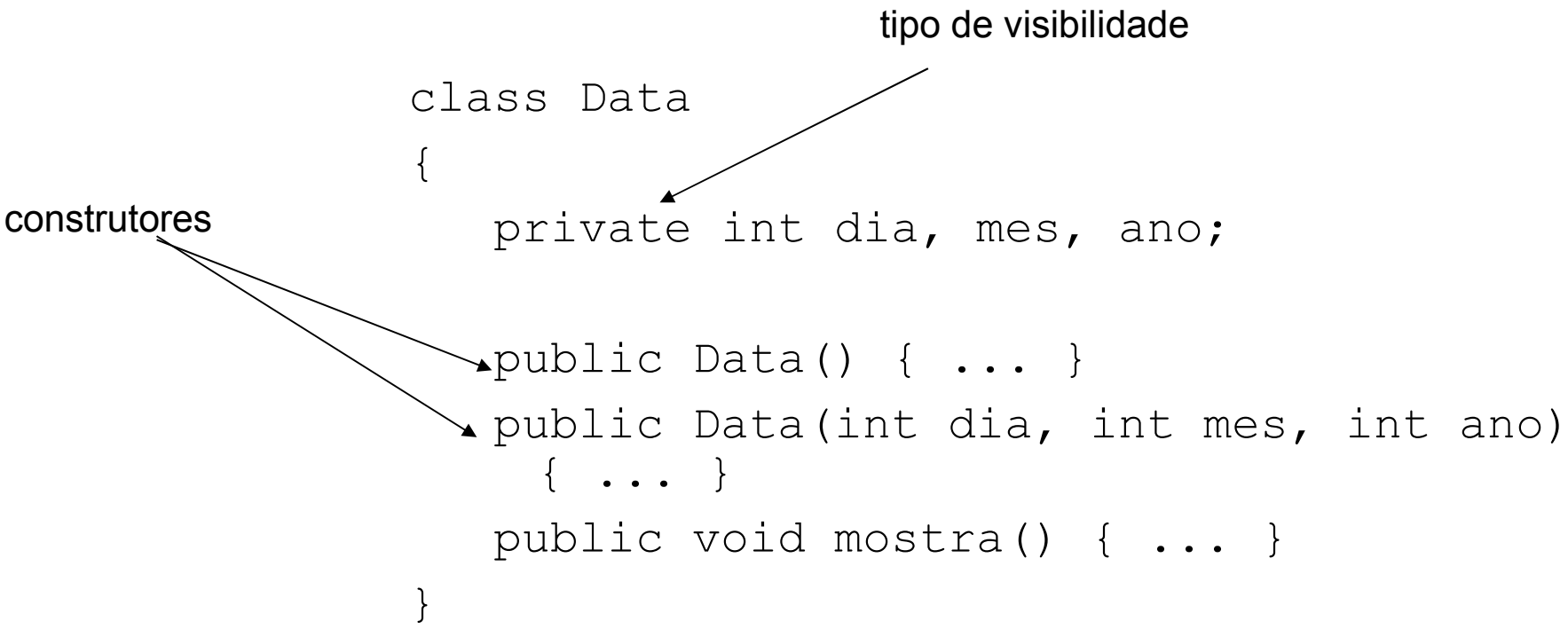
- Expressões aritméticas/relacionais são formadas como em C/C++
- Operadores aritméticos/relacionais são praticamente idênticos
- Não há ponteiros em Java!
- Passagem de argumentos em Java é sempre por valor
- Sintaxe dos comentários é idêntica

Tipos abstratos em Java

- Suporte ao encapsulamento/TAD: **classes**
- Classes
  - “Parente” do struct em C
  - definição de **atributos/campos**
  - definição de **métodos**
- Suporte à ocultação
  - cláusula private: atributos/métodos
  - cláusula public: atributos/métodos visíveis

Tipos abstratos de dados

- Classes em Java

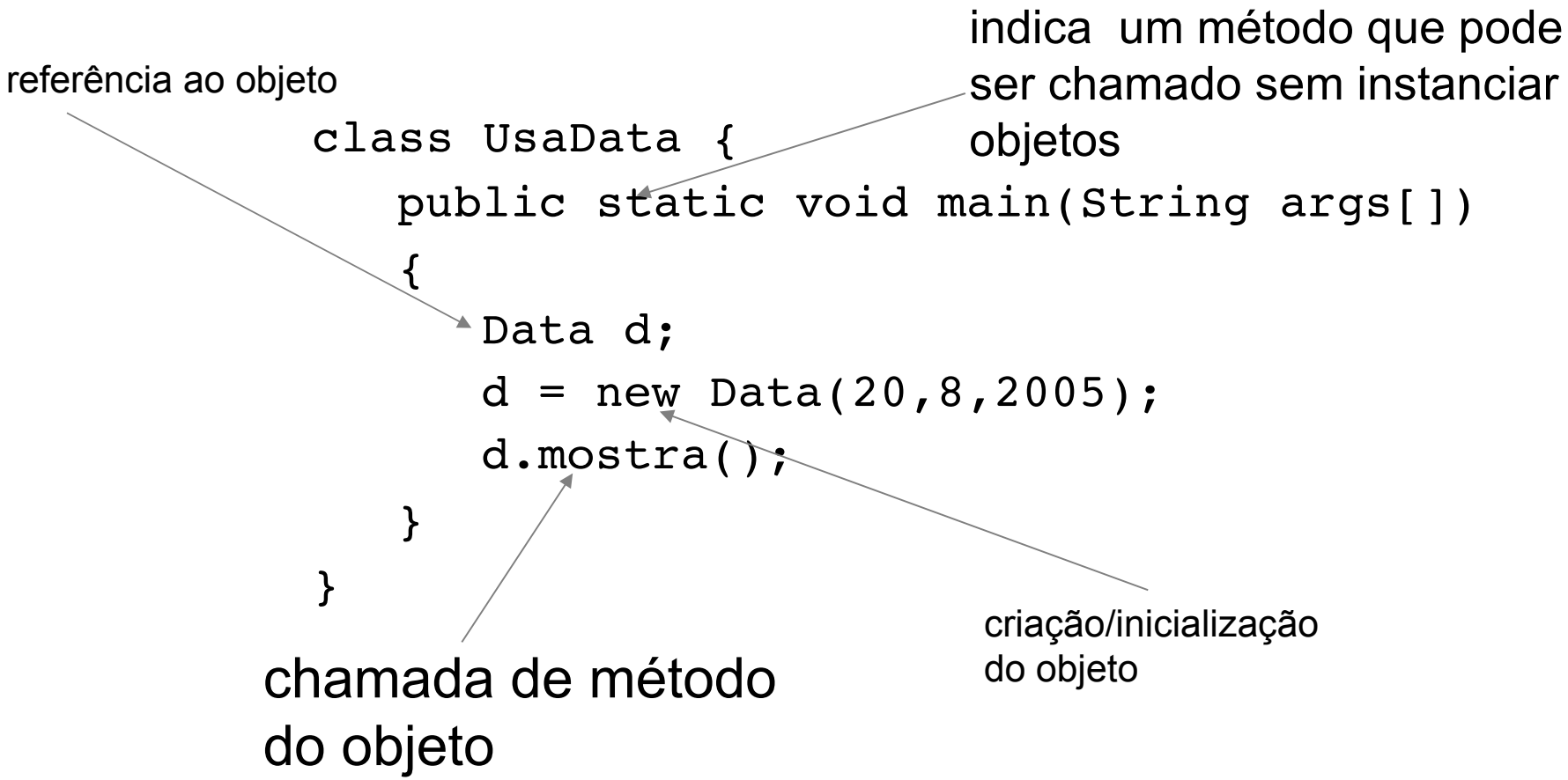


Tipos abstratos de dados

- Construtores
  - servem para inicialização de objetos
  - possuem o mesmo nome da classe
  - não possuem tipo de retorno
  - uma classe pode ter tantos construtores quantos forem necessários
  - a diferenciação entre os construtores se dá pelos seus argumentos (tipo e número)

Tipos abstratos de dados

- Objetos e referências a objetos

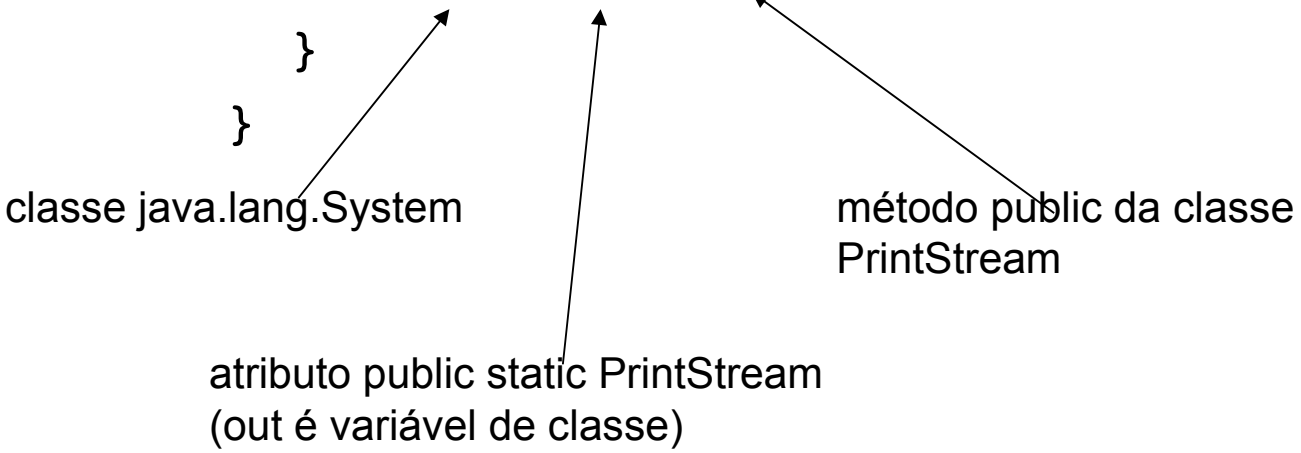


Em Java...

- Ao contrário de C++, tudo deve estar dentro de classes (incluindo a função principal do programa)
- Não há funções ou dados globais (só static)
- Todas as definições de métodos são feitas no corpo da classe (não ficam num .h)
- Não há **destrutores** (o programador não se preocupa em desalocar objetos na memória)
- **Coletor de lixo** desaloca memória para objetos que não são mais referenciados

Programa "Olá, mundo"

```
class HelloApp
{
  public static void main(String[] args)
  {
    System.out.println("Hello World!");
  }
}
```

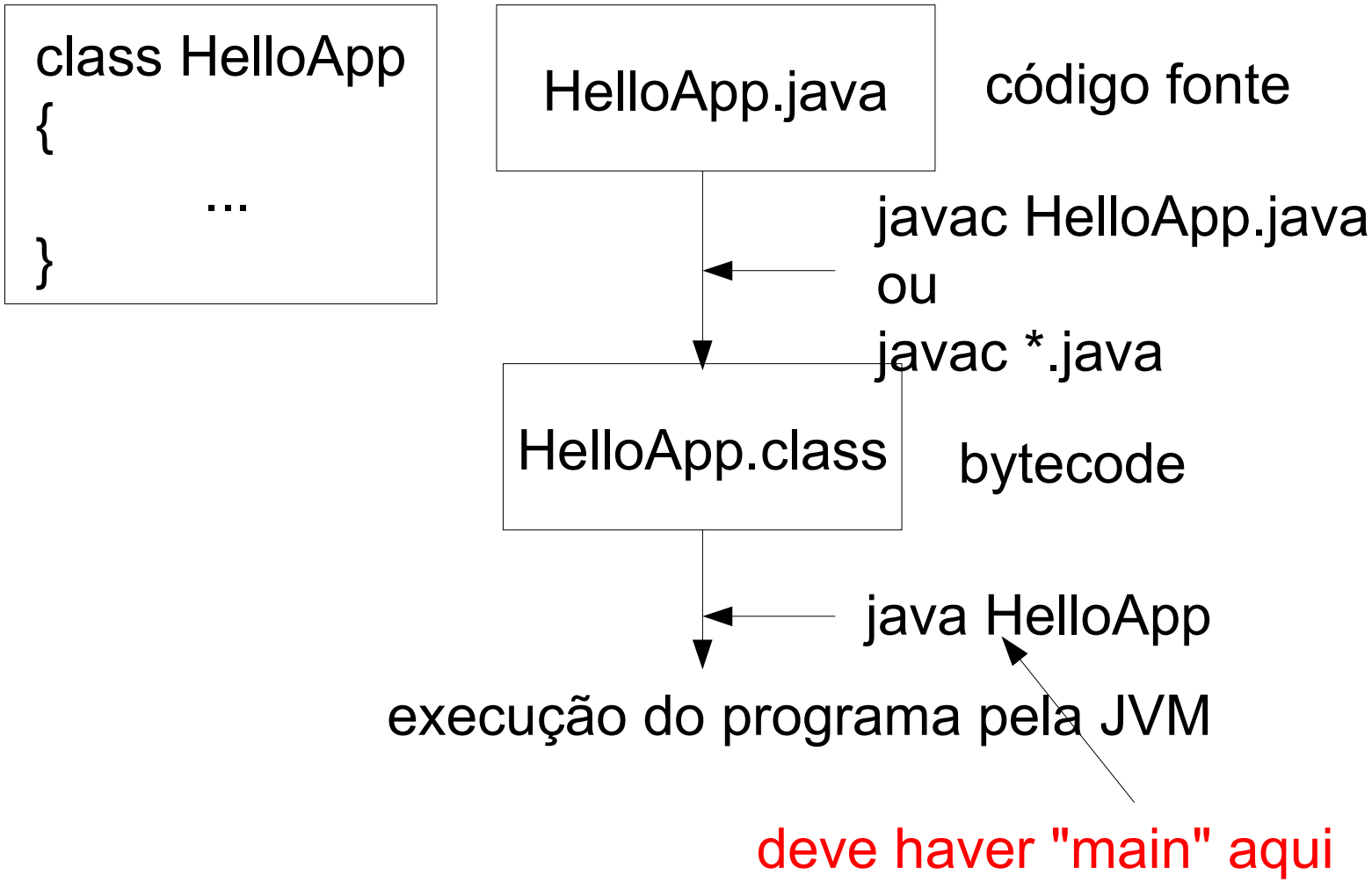


Veja: http://download.oracle.com/javase/tutorial/getStarted/application/index.html

Resumindo: encapsulamento em Java

- Tipos de visibilidade/ocultação
  - **visibilidade interna (private)**
  - **visibilidade externa (public)**
  - visibilidade para classes derivadas (protected)
  - visibilidade no pacote (classes dentro de um diretório)
    - ✓ Pacote java.lang está visível automaticamente

Plataforma Java

- Java SE: Standard Edition
  - JDK (desenvolvimento, inclui JRE)
  - JRE (só p/execução)
- Java EE: Enterprise Edition
- Java ME: Micro Edition

Plataforma Java: execução