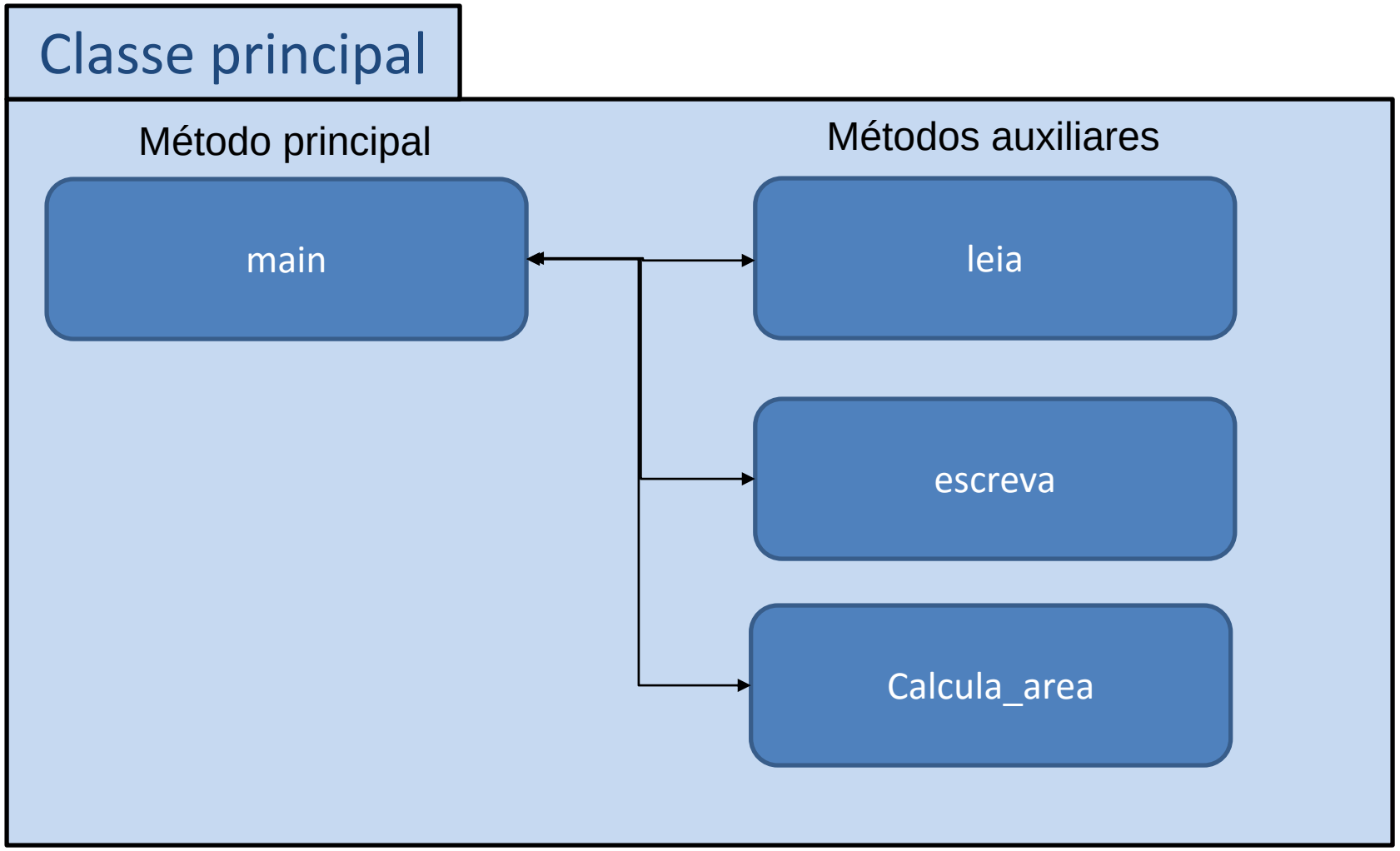
Modularização (funções)

Módulos

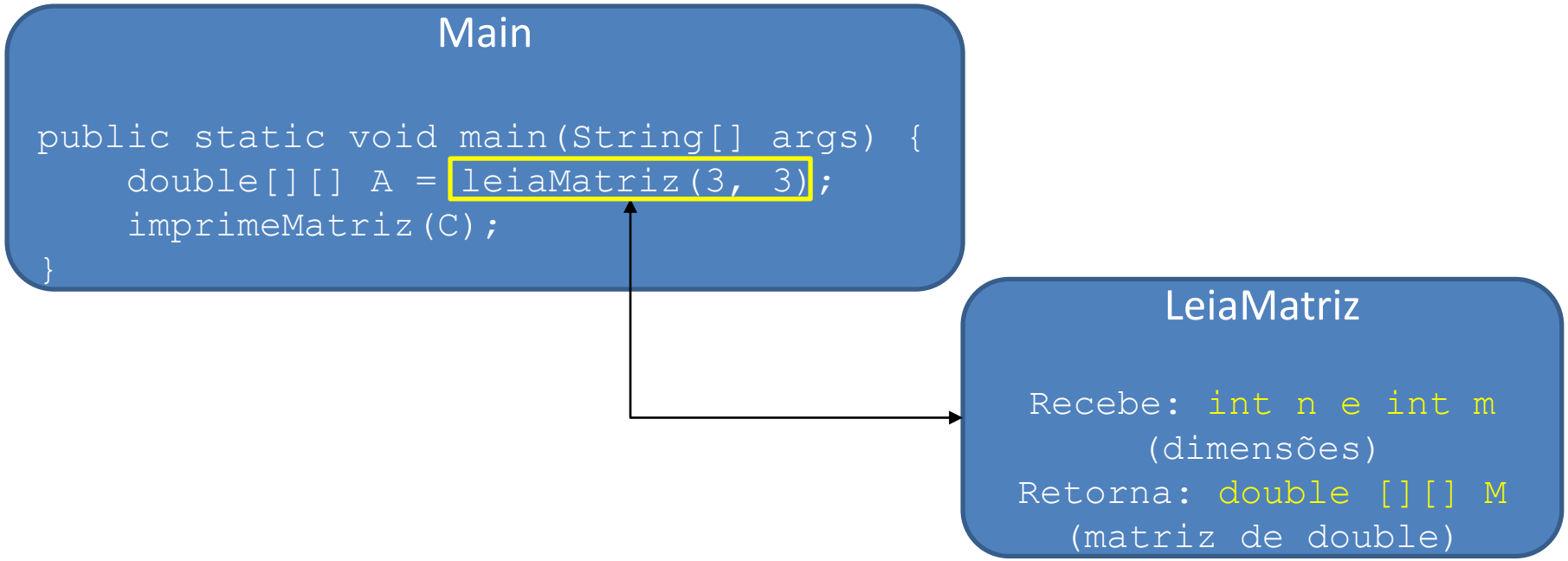

Classe principal
Método principal
Métodos auxiliares
main
leia
escreva
Calcula_area

Método principal

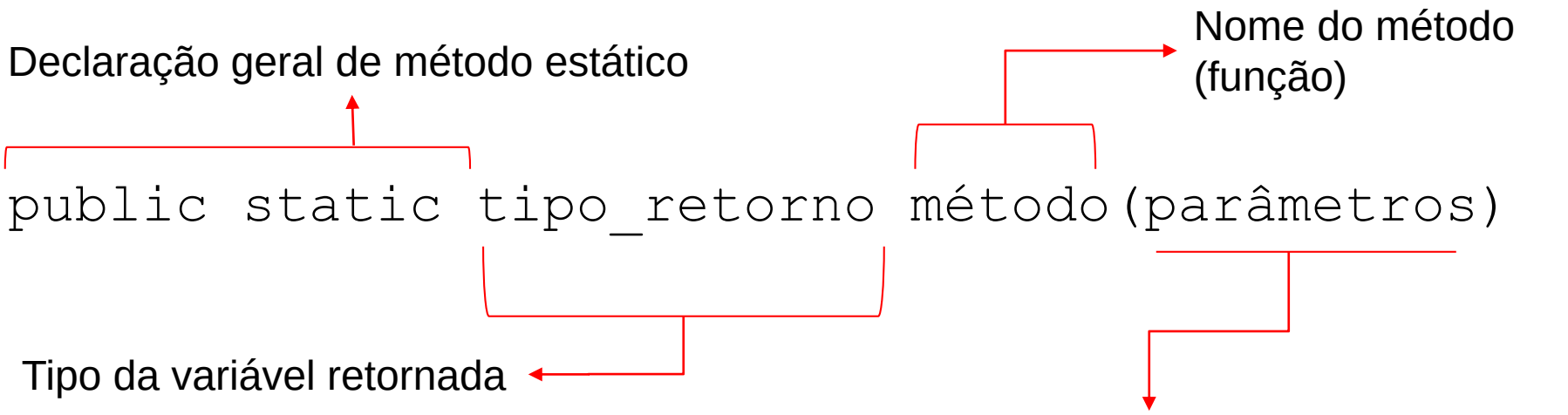
- Primeiro a ser executado na classe principal

Métodos auxiliáres

-Precisam ser invocados



Sintaxe: declaração de métodos



Lista de parâmetros passados ao método:

tipos e **nomes** das variáveis **locais** que recebem valores no **escopo** do método, separados por virgula

Sintaxe: chamada (invocação)

`nome_do_método();`	Sem parâmetro/sem retorno
`nome_do_método(10);`	Com parâmetro/sem retorno
`A=nome_do_método();`	Sem parâmetro/com retorno
`Y=nome_f(16);`	Com parâmetros e retorno
`Z=nome_f(4f,-1f);`	Com 2 parâmetros e retorno

Sintaxe: Exemplos

```
pula_linha();
pula_linhas(10);
long[] F=fibonacci();
Y=grausEmKelvin(16.0);
Z=f_de_X_Y(4f,-1f);
```

Sem parâmetro/sem retorno
Com parâmetro/sem retorno
Sem parâmetro/com retorno
Com parâmetros e retorno
Com 2 parâmetros e retorno

`pula_linha();`	Sem parâmetro/sem retorno
`pula_linhas(10);`	Com parâmetro/sem retorno
`long[] F=fibonacci();`	Sem parâmetro/com retorno
`Y=grausEmKelvin(16.0);`	Com parâmetros e retorno
`Z=f_de_X_Y(4f,-1f);`	Com 2 parâmetros e retorno

Exercício:
Implemente os métodos acima, dados:

- Pula uma única linha
- Pula n linhas
- 100 primeiros números da sequencia
- double f(x) = grausEmKelvin(double)
- double f(x,y) = seno(x)*tan(y/x)

Programação Orientada a Objetos

Os objetos emcapsulam os **dados** e os **métodos**

Exemplo:

```
import Complexo
...
// Instanciar
double real = 2, imag = -3
Complexo a = new Complexo(real, imag);

// Métodos
System.out.print(a.real);
a.print();
```

Acessando métodos de outras classes

Com import: (exemplo: classe Scanner)

```
import java.util.Scanner;
… main:
Scanner objeto = new Scanner(System.in)
A = objeto.[método];
```

Sem import:

```
java.util.Scanner.[método];
Math.[método];
```

Classes java nativas (não precisam de import):

```
Math, Double, String, etc.;
```

Convenção:

NomeClasse.nomeMétodo

*Repare que classes tem nomes iniciando em maiúsculas, métodos em minúsculas

Exemplos:

```
Scanner.nextInt();
Math.sin(x);
String.format();
String a =  "rogerio".toUpperCase();
Char b = a.charAt(3);
```

Prática

1. Funções escalares

Fazer utilizando console

i. Faça uma função que retorne o fatorial em `long` do número `int` passado como parâmetro

ii. Faça uma função que retorne a distância em `double` de um ponto (x_1, y_1) até outro ponto (x_2, y_2)

iii. Faça uma função que receba uma temperatura e uma string, que pode ser “ck”, “cf”, “fk”, “fc”, “kf”, “kc”, indicando o sentido da conversão (ex: “kf” → de Kelvin para Fahrenheit)

```
public static double temperatura(double t, String s)
```

2. Classe “Complexo”

- Construa um projeto com nome “NumerosComplexos”
- Na criação do projeto, crie também uma classe principal
  - Chame a classe principal de “Principal”
  - Ela deverá conter o método `main` (como de costume)
- No mesmo projeto, crie outra classe auxiliar *
  - Chame a classe de “Complexo”
  - Não contem método principal
  - Contem apenas duas variáveis globais re (real) e im (imaginário)
  - Contem os métodos para operação de números complexos
    - Soma, Subtração, Divisão, Multiplicação, etc.
- Importe e utilize a classe Complexo na classe Principal para fazer o exercício a seguir

* Para criar outra classe dentro do mesmo projeto, clique com o botão direito sobre o ícone do projeto (xícara) e selecione “Nova classe”.

Método principal da classe principal

```
public static void main(String[] args) {
  Complexo a = new Complexo(1, 2), // Construtor com números
            b = new Complexo(), c; // Construtor padrão (0, 0)
  a.leia("Entre o complexo a");
  b = ortogonal(a);
  // b é ortogonal, i.e. b.re. a.im e b.im = a.re

  // testa métodos de operação de complexos
  c = soma(a, b); // retorna a soma
  a = multiplica(b, c); // retorna a multiplicação
  b = subtrai(a, c); // retorna a subtração
  // exibe a, b, e c com o método escreva
  escreva("Valores finais:\na = " + a +
          "\nb = " + b + "\nc = " + c); m
  // conversão automática com o toString() da classe complexo
}
```