

# **Unidade I: Introdução**

## **Estrutura dos nossos códigos em Java e C**




**PUC Minas**

Instituto de Ciências Exatas e Informática  
Departamento de Ciência da Computação

# Agenda

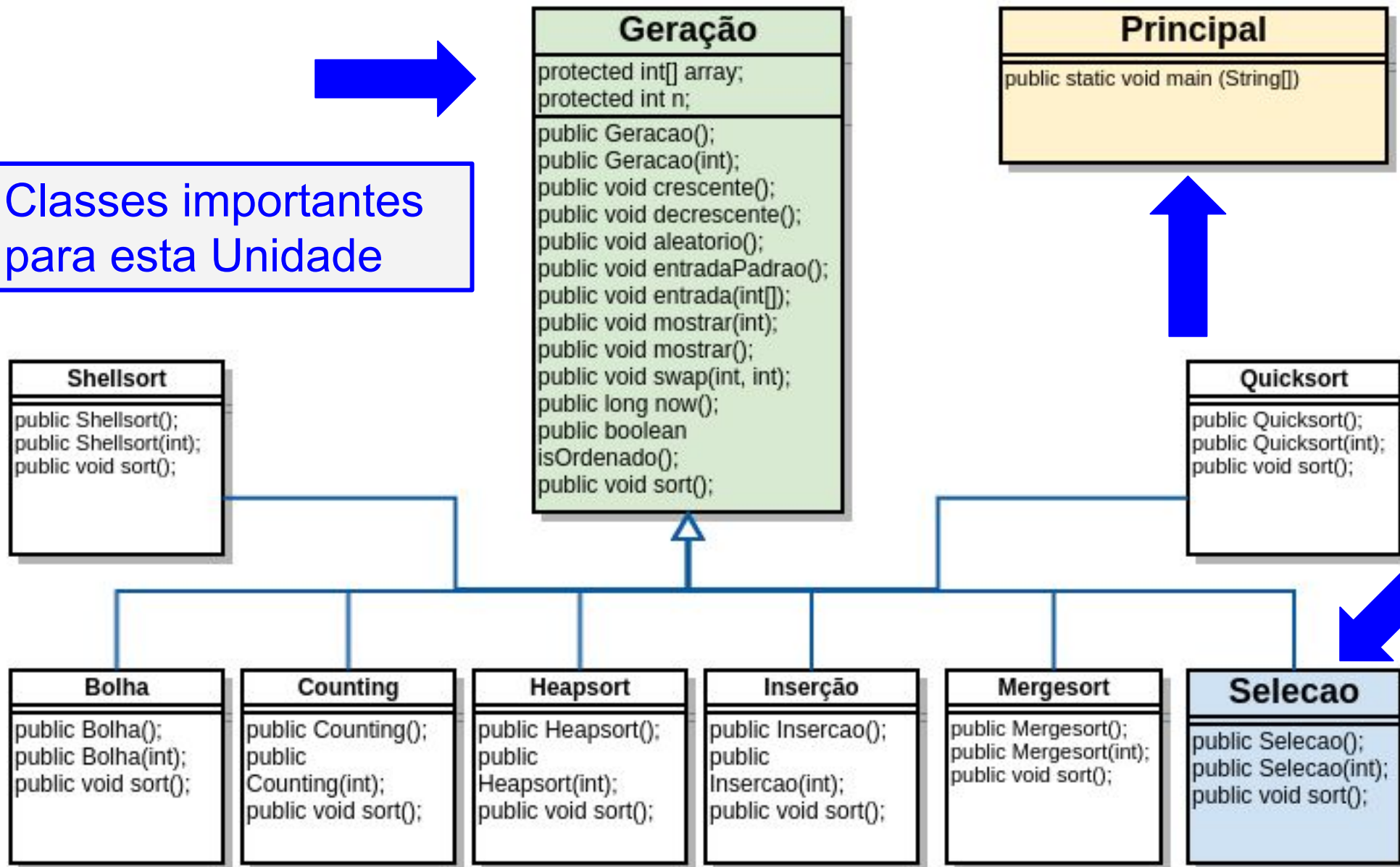
- Estrutura do Código em Java
- Estrutura do Código em C
- makefile para C/C++

# Agenda

- **Estrutura do Código em Java** 
- Estrutura do Código em C
- makefile para C/C++

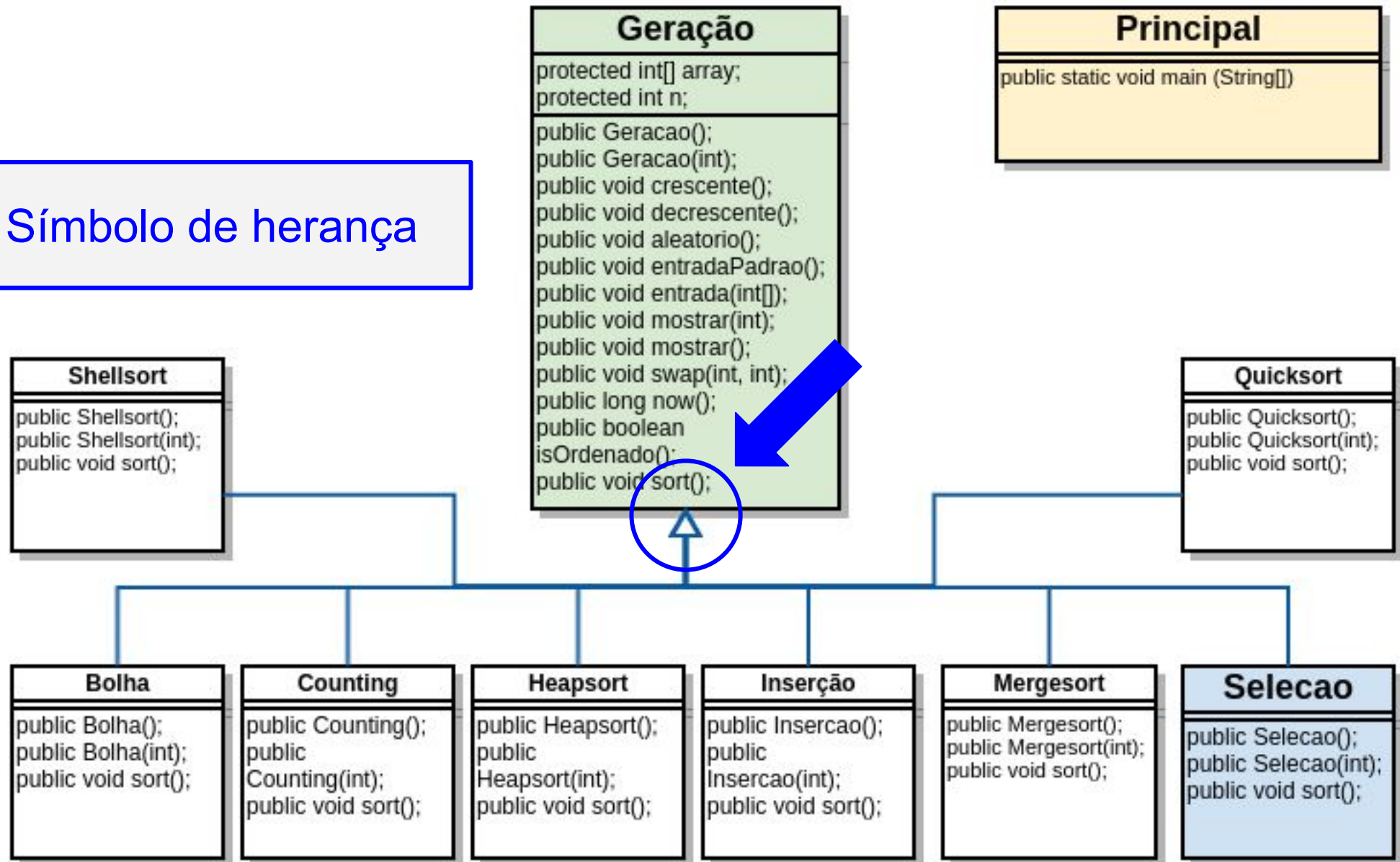
# Estrutura do Código em Java

Classes importantes  
para esta Unidade



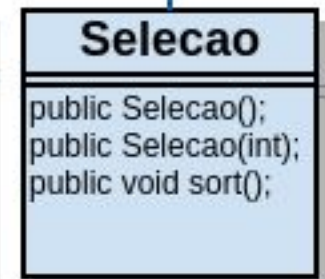
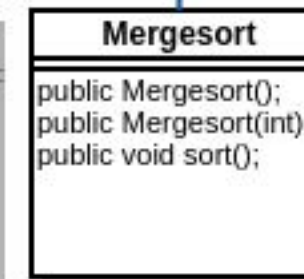
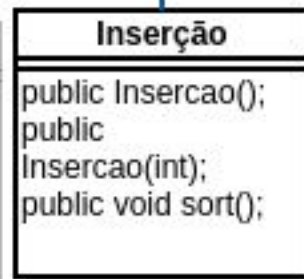
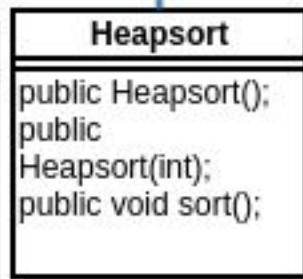
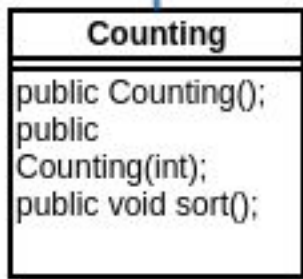
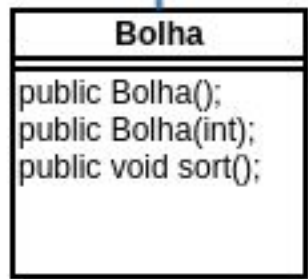
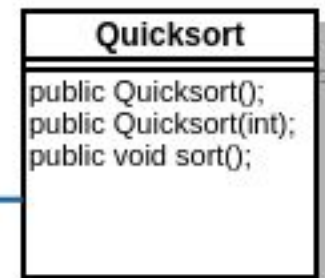
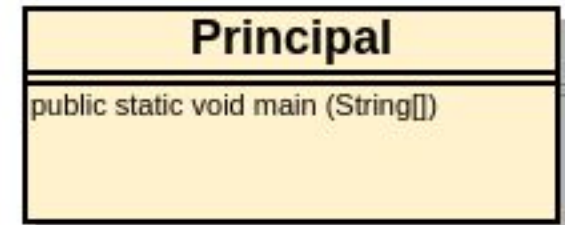
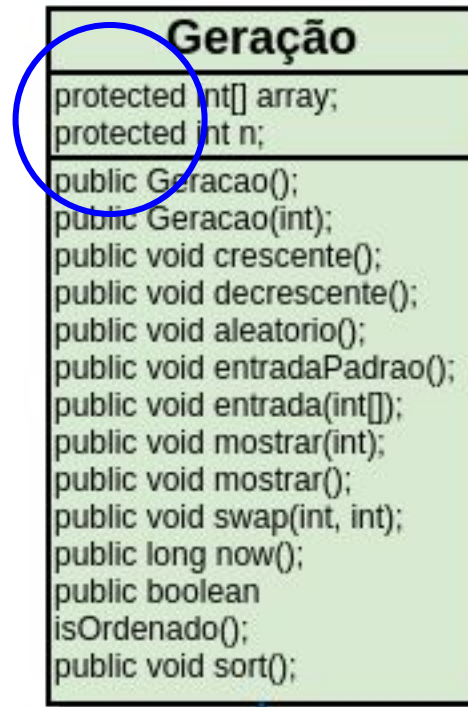
# Estrutura do Código em Java

Símbolo de herança



# Estrutura do Código em Java

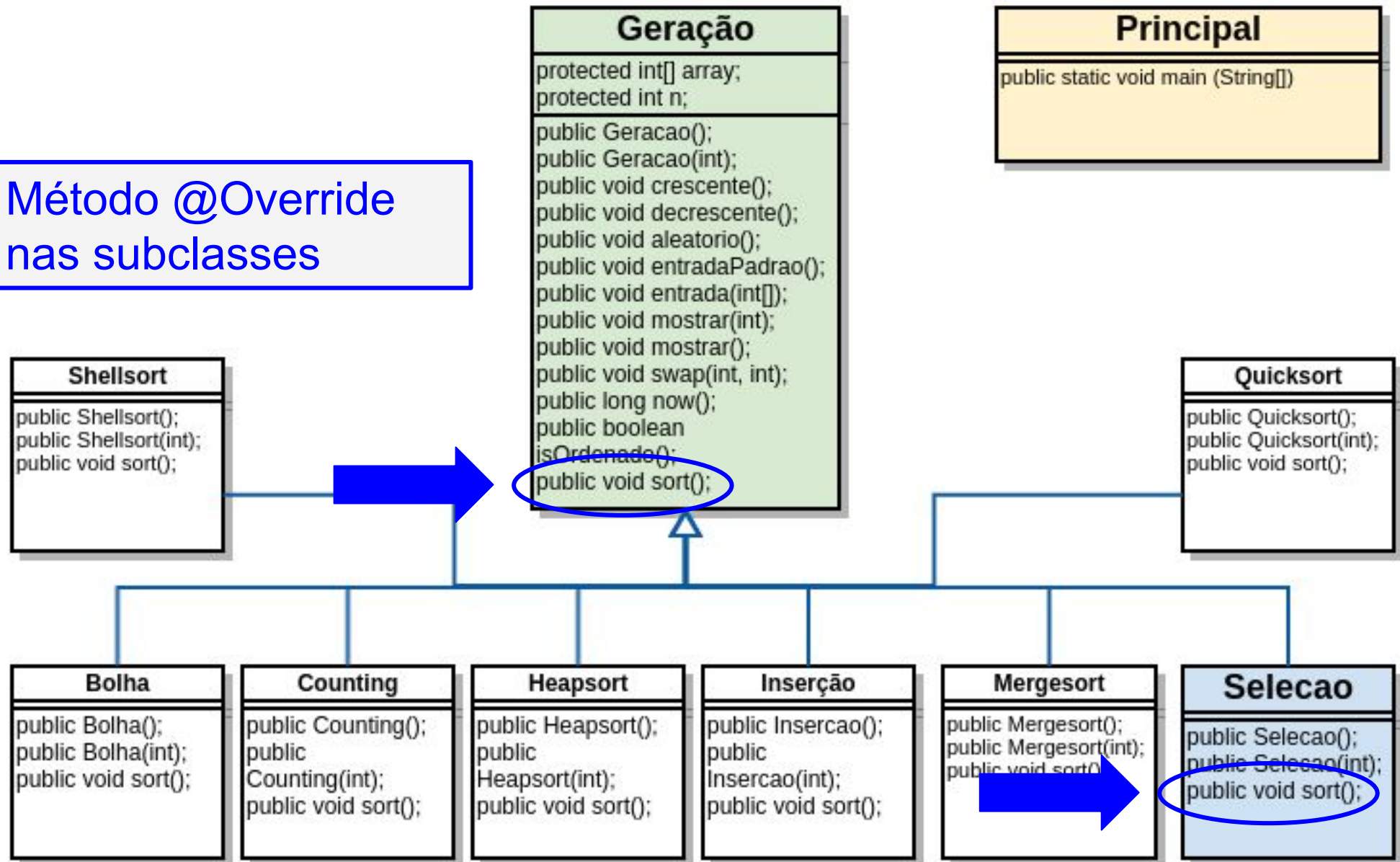
Atributos com visibilidade protegida





# Estrutura do Código em Java

Método @Override nas subclasses



# Classe Geração





# Classe Geração

```
class Geracao {  
    protected int[] array;  
    protected int n;  
  
    public Geracao(){  
        array = new int[100];  
        n = array.length;  
    }  
  
    public Geracao(int tamanho){  
        array = new int[tamanho];  
        n = array.length;  
    }  
  
    public void crescente() {  
        for (int i = 0; i < n; i++) array[i] = i;  
    }  
  
    public void decrescente() {  
        for (int i = 0; i < n; i++) array[i] = n - 1 - i;  
    }  
}
```

```
    public void aleatorio() {  
        Random rand = new Random();  
        crescente();  
        for (int i = 0; i < n; i++)  
            swap(i, Math.abs(rand.nextInt()) % n);  
    }  
  
    public void entradaPadrao() {  
        n = MyIO.readInt();  
        array = new int[n];  
        for (int i = 0; i < n; i++)  
            array[i] = MyIO.readInt();  
    }  
  
    public void entrada(int[] vet){  
        n = vet.length;  
        array = new int[n];  
        for (int i = 0; i < n; i++) array[i] = vet[i];  
    }  
}
```

# Classe Geração

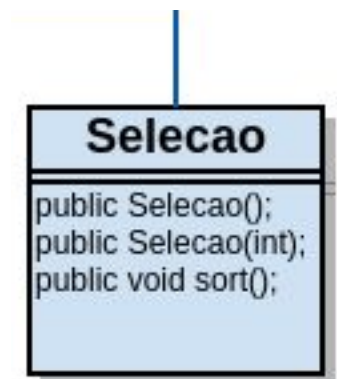
```
public void mostrar() {  
    System.out.print("[");  
    for (int i = 0; i < n; i++)  
        System.out.print(" (" + i + ") " + array[i]);  
    System.out.println("]");  
}
```

```
public void swap(int i, int j) {  
    int temp = array[i];  
    array[i] = array[j];  
    array[j] = temp;  
}
```

```
public long now(){  
    return new Date().getTime();  
}
```

```
public boolean isOrdenado(){  
    boolean resp = true;  
    for (int i = 1; i < n; i++) {  
        if (array[i] < array[i-1]){  
            i = n;  
            resp = false;  
        }  
    }  
    return resp;  
}  
  
public void sort(){  
    System.out.println("Método a ser  
                        implementado nas subclasses.");  
}
```

# Classe Seleção



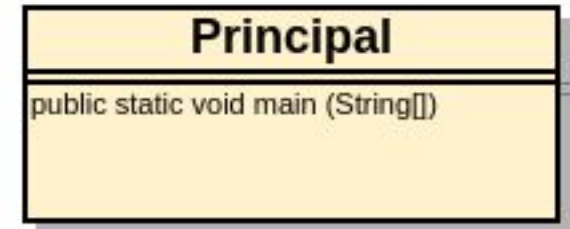
# Classe Seleção

```
class Selecao extends Geracao {  
    public Selecao(){  
        super();  
    }  
    public Selecao(int tamanho){  
        super(tamanho);  
    }  
    @Override  
    public void sort() {  
        for (int i = 0; i < (n - 1); i++) {  
            int menor = i;  
            for (int j = (i + 1); j < n; j++){  
                if (array[menor] > array[j]){  
                    menor = j;  
                }  
            }  
            swap(menor, i);  
        }  
    }  
}
```

**super()** é o construtor da superclasse (pai)

**@Override** é facultativo, contudo, boa prática

# Classe Principal



# Classe Principal

```
class Principal {  
    public static void main(String[] args){  
        Geracao algoritmo;  
        int n = (args.length < 1) ? 1000 :  
                Integer.parseInt(args[0]);  
        double inicio, fim;  
  
        //Inicialização do algoritmo de ordenação  
        //algoritmo = new Bolha(n);  
        //algoritmo = new Countingsort(n);  
        //algoritmo = new Heapsort(n);  
        //algoritmo = new Insercao(n);  
        //algoritmo = new Mergesort(n);  
        //algoritmo = new Quicksort(n);  
        algoritmo = new Selecao(n);  
        //algoritmo = new Shellsort(n);  
  
        //Geração do conjunto a ser ordenado  
        algoritmo.aleatorio();  
        //algoritmo.crescente();
```

```
        //algoritmo.decrecente();  
  
        //Mostrar o conjunto a ser ordenado  
        //algoritmo.mostrar();  
  
        //Execução do algoritmo de ordenação  
        inicio = algoritmo.now();  
        algoritmo.sort();  
        fim = algoritmo.now();  
  
        //Mostrar o conjunto ordenado, tempo de  
        //execução e status da ordenação  
        //algoritmo.mostrar();  
        System.out.println("Tempo para ordenar:  
" + (fim-inicio)/1000.0 + " s.");  
        System.out.println("isOrdenado: " +  
        algoritmo.isOrdenado());  
    }  
}
```



# Classe Principal

```
class Principal {
    public static void main(String[] args){
        Geracao algoritmo;
1      int n = (args.length < 1) ? 1000 :
            Integer.parseInt(args[0]);
        double inicio, fim;

        //Inicialização do algoritmo de ordenação
        //algoritmo = new Bolha(n);
        //algoritmo = new Countingsort(n);
2      //algoritmo = new Heapsort(n);
        //algoritmo = new Insercao(n);
        //algoritmo = new Mergesort(n);
        //algoritmo = new Quicksort(n);
        algoritmo = new Selecao(n);
        //algoritmo = new Shellsort(n);

        //Geração do conjunto a ser ordenado
3      algoritmo.aleatorio();
        //algoritmo.crescente();
```

```
        //algoritmo.decrecente();

        //Mostrar o conjunto a ser ordenado
4      //algoritmo.mostrar();

        //Execução do algoritmo de ordenação
        inicio = algoritmo.now();
        algoritmo.sort();
        fim = algoritmo.now();
5      //Mostrar o conjunto ordenado, tempo de
        //execução e status da ordenação
        //algoritmo.mostrar();
        System.out.println("Tempo para ordenar:
" + (fim-inicio)/1000.0 + " s.");
        System.out.println("isOrdenado: " +
        algoritmo.isOrdenado());
    }
}
```

# Classe Principal

```
class Principal {
    public static void main(String[] args){
        Geracao algoritmo;
        1 int n = (args.length < 1) ? 1000 :
            Integer.parseInt(args[0]);
```

Operador ternário e uso  
do *array* de argumentos

```
//algoritmo = new Boina(n);
//algoritmo = new Countingsort(n);
//algoritmo = new Heapsort(n);
//algoritmo = new Insercao(n);
//algoritmo = new Mergesort(n);
//algoritmo = new Quicksort(n);
algoritmo = new Selecao(n);
//algoritmo = new Shellsort(n);

//Geração do conjunto a ser ordenado
algoritmo.aleatorio();
//algoritmo.crescente();
```

```
//algoritmo.decrecente();

//Mostrar o conjunto a ser ordenado
//algoritmo.mostrar();

//Execução do algoritmo de ordenação
inicio = algoritmo.now();
algoritmo.sort();
fim = algoritmo.now();

//Mostrar o conjunto ordenado, tempo de
execução e status da ordenação
//algoritmo.mostrar();
System.out.println("Tempo para ordenar:
" + (fim-inicio)/1000.0 + " s.");
System.out.println("isOrdenado: " +
algoritmo.isOrdenado());
}
```

# Classe Principal

```
class Principal {
    public static void main(String[] args){
        Geracao algoritmo;
        int n = (args.length < 1) ? 1000 :
                Integer.parseInt(args[0]);
        double inicio, fim;

        //Inicialização do algoritmo de ordenação
        //algoritmo = new Bolha(n);
        //algoritmo = new Countingsort(n);
        2 //algoritmo = new Heapsort(n);
        //algoritmo = new Insercao(n);
        //algoritmo = new Mergesort(n);
        //algoritmo = new Quicksort(n);
        algoritmo = new Selecao(n);
        //algoritmo = new Shellsort(n);
```

Mudando comentários, trocamos de algoritmo

```
//algoritmo.decrecente();

//Mostrar o conjunto a ser ordenado
//algoritmo.mostrar();

//Execução do algoritmo de ordenação
inicio = algoritmo.now();
algoritmo.sort();
fim = algoritmo.now();

//Mostrar o conjunto ordenado, tempo de
execução e status da ordenação
//algoritmo.mostrar();
System.out.println("Tempo para ordenar:
" + (fim-inicio)/1000.0 + " s.");
System.out.println("isOrdenado: " +
algoritmo.isOrdenado());
}
```

# Classe Principal

```
class Principal {
    public static void main(String[] args){
        Geracao algoritmo;
        int n = (args.length < 1) ? 1000 :
                Integer.parseInt(args[0]);
        double inicio, fim;

        //Inicialização do algoritmo de ordenação
        //algoritmo = new Bolha(n);
        //algoritmo = new Countingsort(n);
        //algoritmo = new Heapsort(n);
        //algoritmo = new Insercao(n);
        //algoritmo = new Mergesort(n);
        //algoritmo = new Quicksort(n);
        algoritmo = new Selecao(n);
```

Mudando comentários,  
trocamos a ordem inicial

3

```
algoritmo.aleatorio();
//algoritmo.crescente();
```

```
//algoritmo.decrecente();

//Mostrar o conjunto a ser ordenado
//algoritmo.mostrar();

//Execução do algoritmo de ordenação
inicio = algoritmo.now();
algoritmo.sort();
fim = algoritmo.now();

//Mostrar o conjunto ordenado, tempo de
execução e status da ordenação
//algoritmo.mostrar();
System.out.println("Tempo para ordenar:
" + (fim-inicio)/1000.0 + " s.");
System.out.println("isOrdenado: " +
algoritmo.isOrdenado());
}
}
```

# Classe Principal

```
class Principal {
    public static void main(String[] args){
        Geracao algoritmo;
        int n = (args.length < 1) ? 1000 :
                Integer.parseInt(args[0]);
        double inicio, fim;

        //Inicialização do algoritmo de ordenação
        //algoritmo = new Bolha(n);
        //algoritmo = new Countingsort(n);
        //algoritmo = new Heapsort(n);
        //algoritmo = new Insercao(n);
        //algoritmo = new Mergesort(n);
        //algoritmo = new Quicksort(n);
        algoritmo = new Selecao(n);
        //algoritmo = new Shellsort(n);

        //Geração do conjunto a ser ordenado
        algoritmo.aleatorio();
        //algoritmo.crescente();
```

```
        //algoritmo.decrecente();

        //Mostrar o conjunto a ser ordenado
        4 → //algoritmo.mostrar();

        Mudando comentários, mostramos o array
        fim = algoritmo.now();

        //Mostrar o conjunto ordenado, tempo de
        execução e status da ordenação
        //algoritmo.mostrar();
        System.out.println("Tempo para ordenar:
        " + (fim-inicio)/1000.0 + " s.");
        System.out.println("isOrdenado: " +
        algoritmo.isOrdenado());
    }
}
```

# Classe Principal

```
class Principal {
    public static void main(String[] args){
        Geracao algoritmo;
        int n = (args.length < 1) ? 1000 :
                Integer.parseInt(args[0]);
```

```
double inicio = 0;
```

```
//Inicialização
```

```
//algoritmo = new Boina(n);
//algoritmo = new Countingsort(n);
//algoritmo = new Heapsort(n);
//algoritmo = new Insercao(n);
//algoritmo = new Mergesort(n);
//algoritmo = new Quicksort(n);
algoritmo = new Selecao(n);
//algoritmo = new Shellsort(n);
```

```
//Geração do conjunto a ser ordenado
algoritmo.aleatorio();
//algoritmo.crescente();
```

Mostrar tempo de  
execução e se ordenado

```
//algoritmo.decrecente();
```

```
//Mostrar o conjunto a ser ordenado
//algoritmo.mostrar();
```

```
//Execução do algoritmo de ordenação
inicio = algoritmo.now();
algoritmo.sort();
fim = algoritmo.now();
```

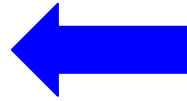
5

```
//Mostrar o conjunto ordenado, tempo de
//execução e status da ordenação
//algoritmo.mostrar();
System.out.println("Tempo para ordenar:
" + (fim-inicio)/1000.0 + " s.");
System.out.println("isOrdenado: " +
algoritmo.isOrdenado());
}
```

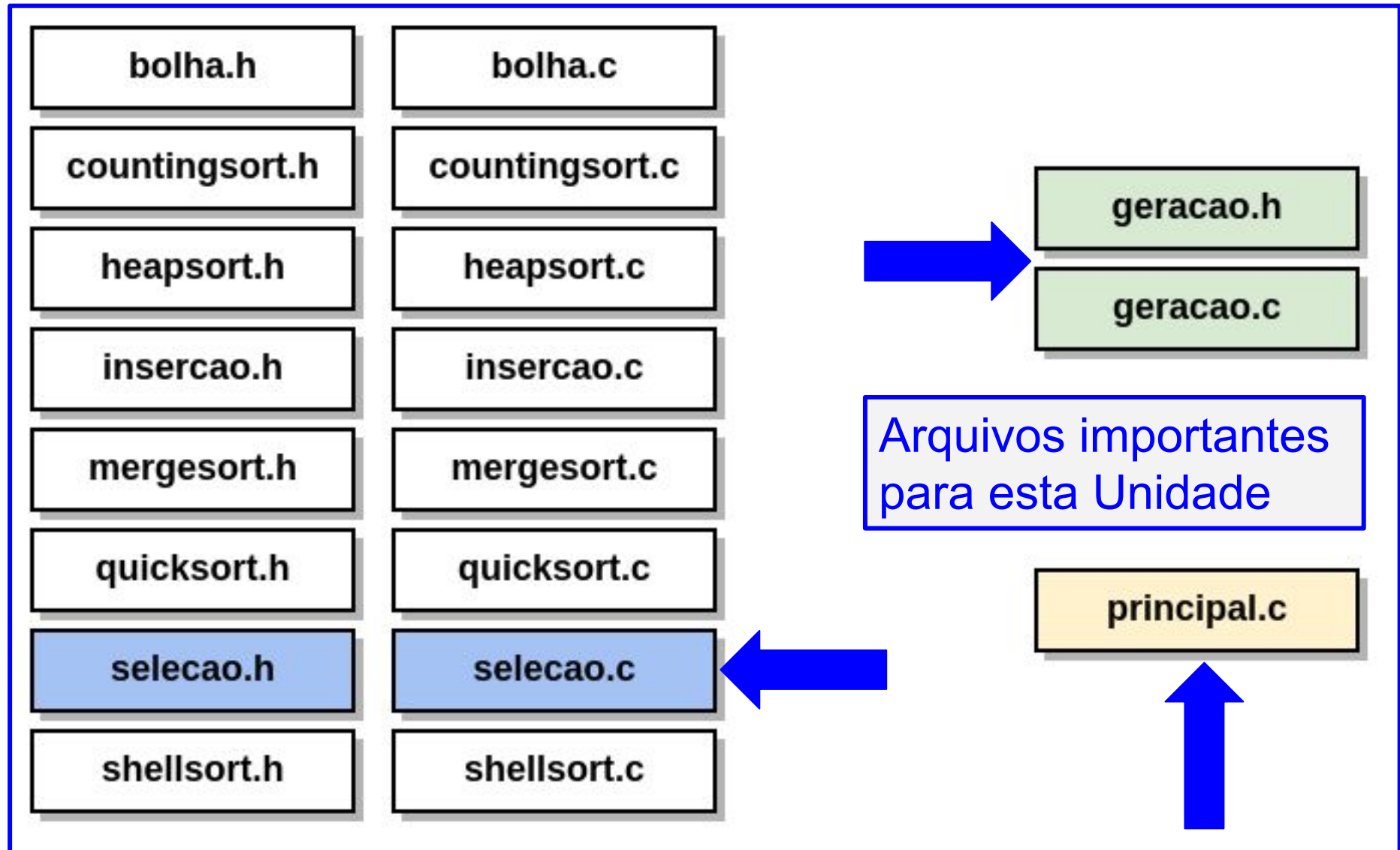


# Agenda

- Estrutura do Código em Java
- **Estrutura do Código em C**
- makefile para C/C++



# Estrutura do Código em C



# Estrutura do Código em C

**selecao.h**

**selecao.c**

# Arquivos selecao.h e selecao.c

//selecao.h

```
#ifndef SELECAO_H
#define SELECAO_H
//=====
#include "geracao.h"
//=====
void selecao(int *array, int n);
//=====
#endif
```

//selecao.c

```
#include "selecao.h"
//=====
void selecao(int *array, int n){
    for (int i = 0; i < (n - 1); i++) {
        int menor = i;
        for (int j = (i + 1); j < n; j++){
            if (array[menor] > array[j]){
                menor = j;
            }
        }
        swap(&array[menor], &array[i]);
    }
}
//=====
```

# Estrutura do Código em C

geracao.h

geracao.c

# Arquivos geracao.h e geracao.c

/geracao.h

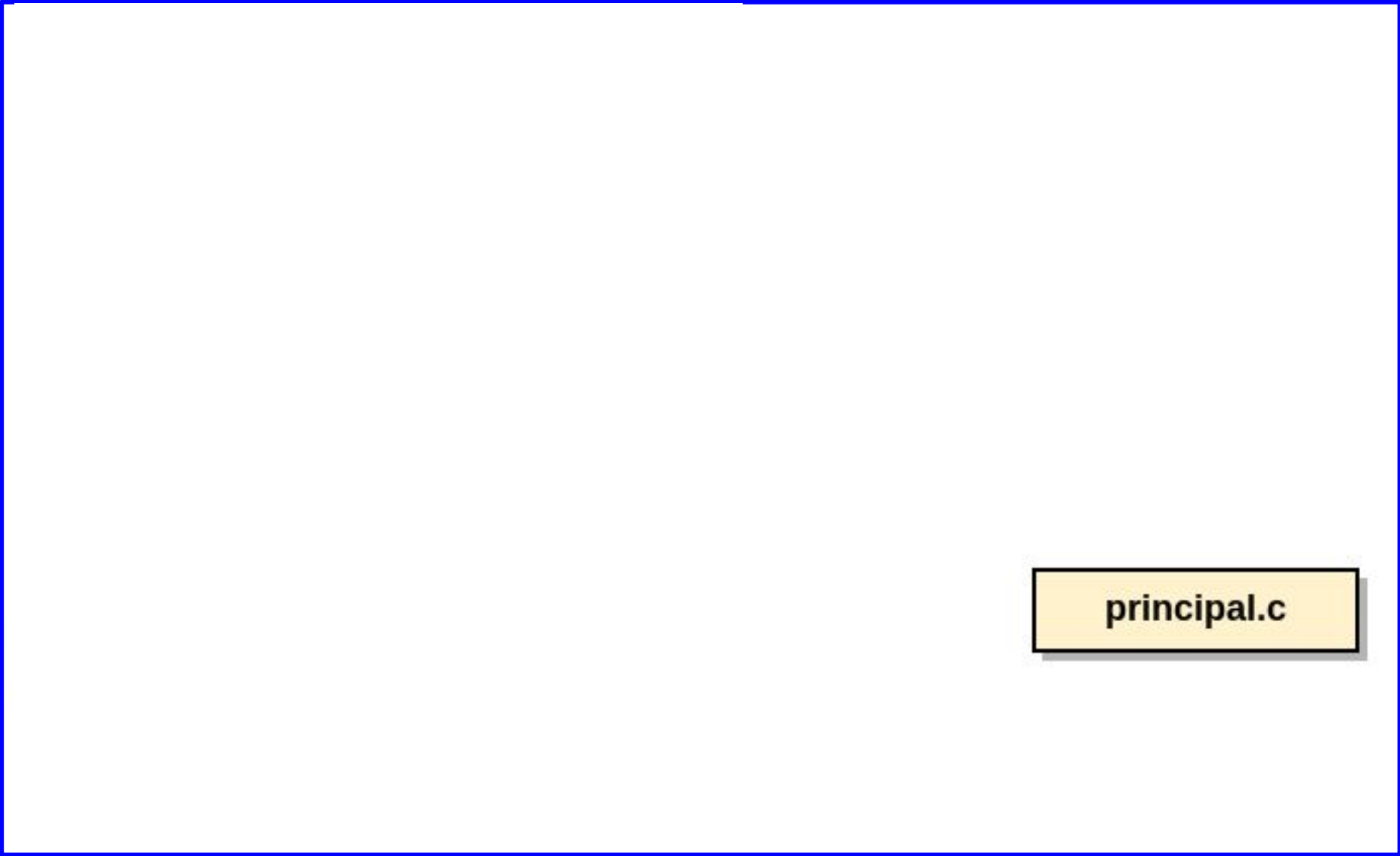
```
#ifndef GERACAO_H
#define GERACAO_H
//=====
#include <stdbool.h>
//=====
void swap(int *i, int *j);
//=====
void crescente(int *array, int n);
//=====
void decrescente(int *array, int n);
//=====
void aleatorio(int *array, int n);
//=====
void mostrar(int *array, int n);
//=====
bool isOrdenado(int *array, int n);
//=====
#endif
```

//geracao.c

■ ■ ■



# Estrutura do Código em C



principal.c

# Arquivo principal.c

```
#include "bolha.h"
#include "countingsort.h"
#include "heapsort.h"
#include "insercao.h"
#include "mergesort.h"
#include "quicksort.h"
#include "selecao.h"
#include "shellsort.h"

#include <stdio.h>
#include <stdlib.h>
#include <time.h>

//=====
int main(int argc, char **argv) {

    //Declaração de variáveis
    int n = (argc < 2) ? 1000 : atoi(argv[1]);
    int *array = (int*) malloc(n*sizeof(int));
    clock_t inicio, fim;
    double total;
```

```
//Geração do conjunto a ser ordenado
aleatorio(array, n);
//crescente(array, n);
//decrescente(array, n);

//Mostrar o conjunto a ser ordenado
//mostrar(array, n);

//Execução do algoritmo de ordenação
inicio = clock();
//bolha(array, n);
//countingsort(array, n);
//heapsort(array, n);
//insercao(array, n);
//mergesort(array, n);
//quicksort(array, n);
selecao(array, n);
//shellsort(array, n);
fim = clock();
```

# Arquivo principal.c

```
total = ((fim - inicio) / (double)CLOCKS_PER_SEC);

//Mostrar o conjunto ordenado, tempo de execução e status da ordenação
//algoritmo.mostrar(array, n);
printf("Tempo para ordenar: %f s.\n", total);
printf("isOrdenado: %s\n", isOrdenado(array, n) ? "true" : "false");

//Desalocar o espaço de memória do array
free(array);

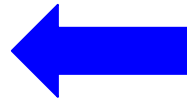
return 0;
}
```



Como alocamos o vetor,  
devemos desalocá-lo

# Agenda

- Estrutura do Código em Java
- Estrutura do Código em C
- **makefile para C/C++**



- Arquivo contendo um conjunto de diretivas usadas pela ferramenta de automação de compilação *make* para gerar um alvo / meta
- Nesse caso, os arquivos serão compilados digitando ***make***

**all:** exec

**exec:** principal.o geracao.o bolha.o countingsort.o ... shellsort.o  
gcc -o exec principal.o geracao.o bolha.o countingsort.o ... shellsort.o

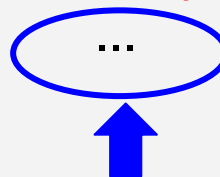
**principal.o:** principal.c  
gcc -o principal.o principal.c -c -W -Wall -pedantic

**bolha.o:** bolha.c  
gcc -o bolha.o bolha.c -c -W -Wall -pedantic

**geracao.o:** geracao.c  
gcc -o geracao.o geracao.c -c -W -Wall -pedantic

**clean:**  
rm -rf \*.o \*~ exec

**limpa:**  
rm -rf \*.o



Criamos um **alvo** / **comando** para cada arquivo .c (**pré-requisito**)



**all:** exec

**exec:** principal.o geracao.o bolha.o countingsort.o ... shellsort.o  
gcc -o exec principal.o geracao.o bolha.o countingsort.o ... shellsort.o

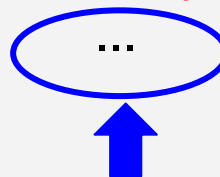
**principal.o:** principal.c  
gcc -o principal.o principal.c -c -W -Wall -pedantic

**bolha.o:** bolha.c  
gcc -o bolha.o bolha.c -c -W -Wall -pedantic

**geracao.o:** geracao.c  
gcc -o geracao.o geracao.c -c -W -Wall -pedantic

**clean:**  
rm -rf \*.o \*~ exec

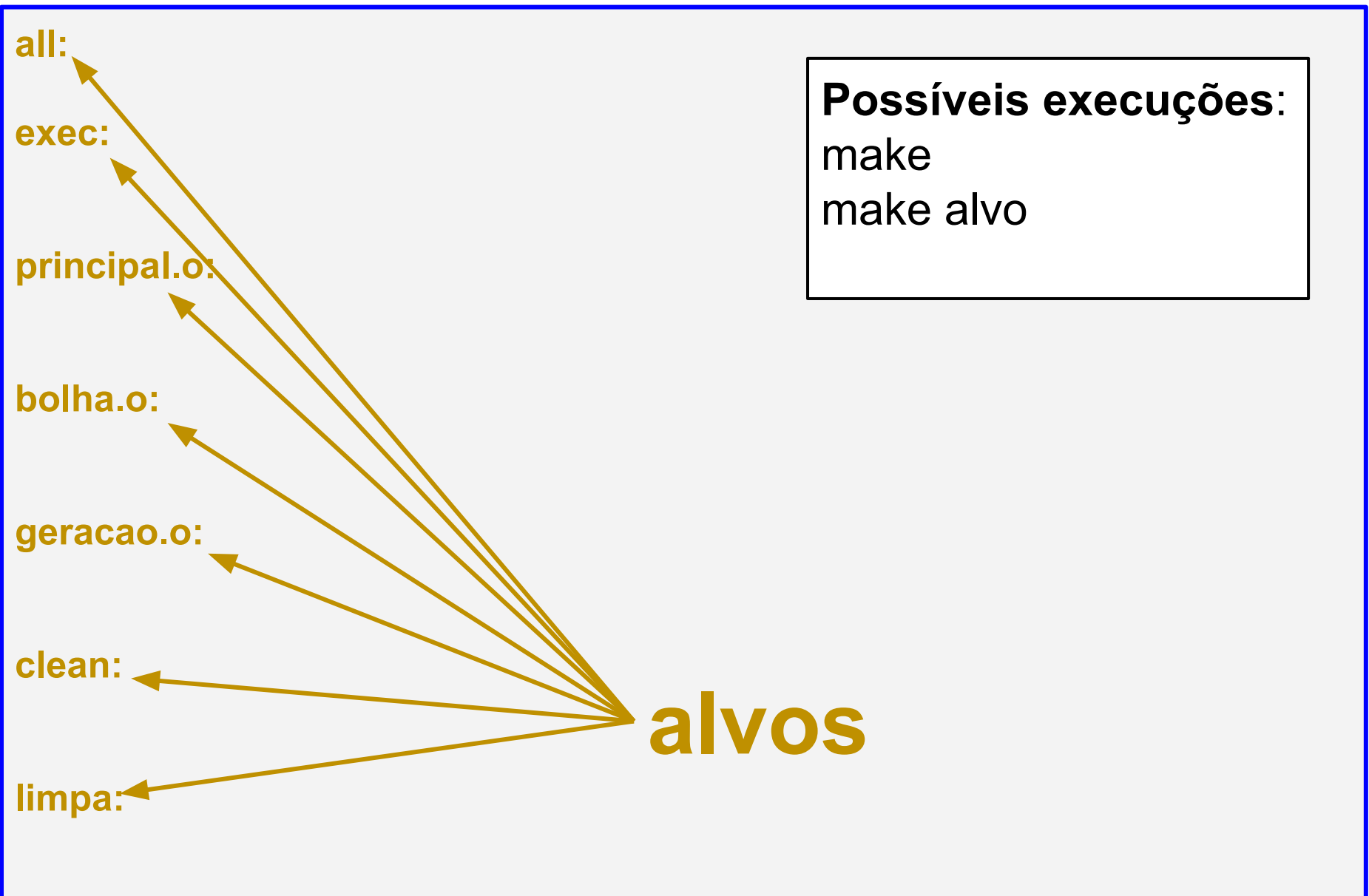
**limpa:**  
rm -rf \*.o



### Arquivos

countingsort.c  
heapsort.c  
insercao.c  
mergesort.c  
quicksort.c  
selecao.c  
shellsort.c

Criamos um **alvo** / **comando** para cada arquivo .c (**pré-requisito**)



**all:**

**exec:**

gcc -o exec principal.o geracao.o bolha.o countingsort.o ... shellsort.o

**principal.o:**

gcc -o principal.o principal.c -c -W -Wall -pedantic

**bolha.o:**

gcc -o bolha.o bolha.c -c -W -Wall -pedantic

**geracao.o:**

gcc -o geracao.o geracao.c -c -W -Wall -pedantic

**clean:**

rm -rf \*.o \*~ exec

**limpa:**

rm -rf \*.o

**comandos**



**all:** exec

**exec:** principal.o geracao.o bolha.o countingsort.o ... shellsort.o

**principal.o:** principal.c

**bolha.o:** bolha.c

**geracao.o:** geracao.c

**clean:**

**limpa:**

**pré-requisitos**

```
graph LR;
    pre[pré-requisitos] --> geracao.o;
    pre --> bolha.o;
    pre --> principal.o;
    pre --> exec;
    pre --> all;
```

**all:** exec

**exec:** principal.o geracao.o bolha.o countingsort.o ... shellsort.o  
gcc -o exec principal.o geracao.o bolha.o countingsort.o ... shellsort.o

**principal.o:** principal.c  
gcc -o principal.o principal.c -c -W -Wall -pedantic

**bolha.o:** bolha.c  
gcc -o bolha.o bolha.c -c -W -Wall -pedantic

**geracao.o:** geracao.c  
gcc -o geracao.o geracao.c -c -W -Wall -pedantic

...

**clean:**  
rm -rf \*.o \*~ exec

**limpa:**  
rm -rf \*.o

# Exercício Resolvido (1)

- Na pasta da “Unidade 4/c/”, digite a sequência de comandos abaixo e explique a saída na tela

- 1) `make all ; ls`
- 2) `make clean ; ls`
- 3) `make ; ls`
- 4) `make clean ; ls`
- 5) `make exec ; ls`
- 6) `make limpa ; ls`
- 7) `make bolha.o ; ls`

# Exercício Resolvido (1)

- Na pasta da “Unidade 4/c/”, digite a sequência de comandos abaixo e explique a saída na tela

- 1) **make all ; ls**
- 2) **make clean ; ls**
- 3) **make ; ls**
- 4) **make clean ; ls**
- 5) **make exec ; ls**
- 6) **make limpa ; ls**
- 7) **make bolha.o ; ls**

```
:$ make all ; ls
```

```
gcc -o principal.o principal.c -c -W -Wall -pedantic  
gcc -o geracao.o geracao.c -c -W -Wall -pedantic  
gcc -o bolha.o bolha.c -c -W -Wall -pedantic  
gcc -o countingsort.o countingsort.c -c -W -Wall -pedantic  
gcc -o heapsort.o heapsort.c -c -W -Wall -pedantic  
gcc -o insercao.o insercao.c -c -W -Wall -pedantic  
gcc -o mergesort.o mergesort.c -c -W -Wall -pedantic  
gcc -o quicksort.o quicksort.c -c -W -Wall -pedantic  
gcc -o selecao.o selecao.c -c -W -Wall -pedantic  
gcc -o shellsort.o shellsort.c -c -W -Wall -pedantic  
gcc -o exec principal.o geracao.o bolha.o countingsort.o heapsort.o insercao.o  
mergesort.o quicksort.o selecao.o shellsort.o
```

```
bolha.c bolha.o countingsort.h exec geracao.h heapsort.c heapsort.o insercao.h  
makefile mergesort.h principal.c quicksort.c quicksort.o selecao.h shellsort.c  
shellsort.o bolha.h countingsort.c countingsort.o geracao.c geracao.o heapsort.h  
insercao.c insercao.o mergesort.c mergesort.o principal.o quicksort.h selecao.c  
selecao.o shellsort.h
```

# Exercício Resolvido (1)

- Na pasta da “Unidade 4/c/”, digite a sequência de comandos abaixo e explique a saída na tela

- 1) make all ; ls
- 2) **make clean ; ls**
- 3) make ; ls
- 4) make clean ; ls
- 5) make exec ; ls
- 6) make limpa ; ls
- 7) make bolha.o ; ls

```
:$ make clean ; ls
```

```
rm -rf *.o *~ exec
```

```
bolha.c bolha.h countingsort.c countingsort.h geracao.c geracao.h heapsort.c  
heapsort.h insercao.c insercao.h makefile mergesort.c mergesort.h principal.c  
quicksort.c quicksort.h selecao.c selecao.h shellsort.c shellsort.h
```



# Exercício Resolvido (1)

- Na pasta da “Unidade 4/c/”, digite a sequência de comandos abaixo e explique a saída na tela

- 1) make all ; ls
- 2) make clean ; ls
- 3) **make ; ls**
- 4) make clean ; ls
- 5) make exec ; ls
- 6) make limpa ; ls
- 7) make bolha.o ; ls

```
:$ make ; ls
```

```
gcc -o principal.o principal.c -c -W -Wall -pedantic  
gcc -o geracao.o geracao.c -c -W -Wall -pedantic  
gcc -o bolha.o bolha.c -c -W -Wall -pedantic  
gcc -o countingsort.o countingsort.c -c -W -Wall -pedantic  
gcc -o heapsort.o heapsort.c -c -W -Wall -pedantic  
gcc -o insercao.o insercao.c -c -W -Wall -pedantic  
gcc -o mergesort.o mergesort.c -c -W -Wall -pedantic  
gcc -o quicksort.o quicksort.c -c -W -Wall -pedantic  
gcc -o selecao.o selecao.c -c -W -Wall -pedantic  
gcc -o shellsort.o shellsort.c -c -W -Wall -pedantic  
gcc -o exec principal.o geracao.o bolha.o countingsort.o heapsort.o insercao.o  
mergesort.o quicksort.o selecao.o shellsort.o
```

```
bolha.c bolha.o countingsort.h exec geracao.h heapsort.c heapsort.o insercao.h  
makefile mergesort.h principal.c quicksort.c quicksort.o selecao.h shellsort.c  
shellsort.o bolha.h countingsort.c countingsort.o geracao.c geracao.o heapsort.h  
insercao.c insercao.o mergesort.c mergesort.o principal.o quicksort.h selecao.c  
selecao.o shellsort.h
```

# Exercício Resolvido (1)

- Na pasta da “Unidade 4/c/”, digite a sequência de comandos abaixo e explique a saída na tela

- 1) make all ; ls
- 2) make clean ; ls
- 3) make ; ls
- 4) **make clean ; ls**
- 5) make exec ; ls
- 6) make limpa ; ls
- 7) make bolha.o ; ls

```
:$ make clean ; ls
```

```
rm -rf *.o *~ exec
```

```
bolha.c bolha.h countingsort.c countingsort.h geracao.c geracao.h heapsort.c  
heapsort.h insercao.c insercao.h makefile mergesort.c mergesort.h principal.c  
quicksort.c quicksort.h selecao.c selecao.h shellsort.c shellsort.h
```

# Exercício Resolvido (1)

- Na pasta da “Unidade 4/c/”, digite a sequência de comandos abaixo e explique a saída na tela

- 1) make all ; ls
- 2) make clean ; ls
- 3) make ; ls
- 4) make clean ; ls
- 5) **make exec ; ls**
- 6) make limpa ; ls
- 7) make bolha.o ; ls

```
:$ make ; ls
```

```
gcc -o principal.o principal.c -c -W -Wall -pedantic
gcc -o geracao.o geracao.c -c -W -Wall -pedantic
gcc -o bolha.o bolha.c -c -W -Wall -pedantic
gcc -o countingsort.o countingsort.c -c -W -Wall -pedantic
gcc -o heapsort.o heapsort.c -c -W -Wall -pedantic
gcc -o insercao.o insercao.c -c -W -Wall -pedantic
gcc -o mergesort.o mergesort.c -c -W -Wall -pedantic
gcc -o quicksort.o quicksort.c -c -W -Wall -pedantic
gcc -o selecao.o selecao.c -c -W -Wall -pedantic
gcc -o shellsort.o shellsort.c -c -W -Wall -pedantic
gcc -o exec principal.o geracao.o bolha.o countingsort.o heapsort.o insercao.o
mergesort.o quicksort.o selecao.o shellsort.o
```

```
bolha.c bolha.o countingsort.h exec geracao.h heapsort.c heapsort.o insercao.h
makefile mergesort.h principal.c quicksort.c quicksort.o selecao.h shellsort.c
shellsort.o bolha.h countingsort.c countingsort.o geracao.c geracao.o heapsort.h
insercao.c insercao.o mergesort.c mergesort.o principal.o quicksort.h selecao.c
selecao.o shellsort.h
```

# Exercício Resolvido (1)

- Na pasta da “Unidade 4/c/”, digite a sequência de comandos abaixo e explique a saída na tela

- 1) make all ; ls
- 2) make clean ; ls
- 3) make ; ls
- 4) make clean ; ls
- 5) make exec ; ls
- 6) **make limpa ; ls**
- 7) make bolha.o ; ls

```
:$ make limpa ; ls
```

```
rm -rf *.o
```

```
bolha.c bolha.h countingsort.c countingsort.h exec geracao.c geracao.h heapsort.c  
heapsort.h insercao.c insercao.h makefile mergesort.c mergesort.h principal.c  
quicksort.c quicksort.h selecao.c selecao.h shellsort.c shellsort.h
```

# Exercício Resolvido (1)

- Na pasta da “Unidade 4/c/”, digite a sequência de comandos abaixo e explique a saída na tela

- 1) make all ; ls
- 2) make clean ; ls
- 3) make ; ls
- 4) make clean ; ls
- 5) make exec ; ls
- 6) make limpa ; ls
- 7) **make bolha.o ; ls**

```
:$ make bolha.o ; ls
```

```
gcc -o bolha.o bolha.c -c -W -Wall -pedantic
```

```
bolha.c bolha.h bolha.o countingsort.c countingsort.h exec geracao.c geracao.h  
heapsort.c heapsort.h insercao.c insercao.h makefile mergesort.c mergesort.h  
principal.c quicksort.c quicksort.h selecao.c selecao.h shellsort.c shellsort.h
```

**all:** exec

**exec:** principal.o geracao.o bolha.o countingsort.o ... shellsort.o  
gcc -o exec principal.o geracao.o bolha.o countingsort.o ... shellsort.o

**principal.o:** principal.c  
gcc -o principal.o principal.c -c -W -Wall -pedantic

**bolha.o:** bolha.c  
gcc -o bolha.o bolha.c -c -W -Wall -pedantic

**geracao.o:** geracao.c  
gcc -o geracao.o geracao.c -c -W -Wall -pedantic

...

**clean:**  
rm -rf \*.o \*~ exec

**limpa:**  
rm -rf \*.o

Podemos otimizar o makefile, contudo,  
isso é assunto para outra aula!!!