Aula 16 – Ordenação

Norton Trevisan Roman

16 de maio de 2013

Mistério

- O que este código faz? Tente com $v = \{4,1,3,5\}$

```
static void ???(int[] v) {
    for (int ult = v.length-1; ult>0; ult--) {
        for (int i=0; i<ult; i++) {
            if (v[i] > v[i+1]) {
                int aux = v[i];
                v[i] = v[i+1];
                v[i+1] = aux;
            }
        }
    }
}
```

Ordenação – Método da Bolha

- E como ordenamos?

Ordenação – Método da Bolha

- E como ordenamos?
- Primeiro método: **Bubble sort**

Ordenação – Método da Bolha

- E como ordenamos?
- Primeiro método: **Bubble sort**
  - Percorra todo o arranjo tomando seus elementos adjacentes para a par

Ordenação – Método da Bolha

- E como ordenamos?
- Primeiro método: **Bubble sort**
  - Percorra todo o arranjo tomando seus elementos adjacentes para a par
  - Se os elemento no par estiverem ordenados, siga ao próximo par

Ordenação – Método da Bolha

- E como ordenamos?
- Primeiro método: **Bubble sort**
  - Percorra todo o arranjo tomando seus elementos adjacentes para a par
  - Se os elemento no par estiverem ordenados, siga ao próximo par
  - Senão, troque-os de lugar

Ordenação – Método da Bolha

- E como ordenamos?
- Primeiro método: **Bubble sort**
  - Percorra todo o arranjo tomando seus elementos adjacentes para a par
  - Se os elemento no par estiverem ordenados, siga ao próximo par
  - Senão, troque-os de lugar
  - Repita a operação até que nenhuma troca possa ser feita no arranjo inteiro

Ordenação – Método da Bolha

- E como ordenamos?
- Primeiro método: **Bubble sort**
  - Percorra todo o arranjo tomando seus elementos adjacentes para a par
  - Se os elemento no par estiverem ordenados, siga ao próximo par
  - Senão, troque-os de lugar
  - Repita a operação até que nenhuma troca possa ser feita no arranjo inteiro
- Ex: ordene em ordem crescente

9 8 4 6 3

Ordenação – Método da Bolha

- E como ordenamos?
- Primeiro método: **Bubble sort**
  - Percorra todo o arranjo tomando seus elementos adjacentes para a par
  - Se os elemento no par estiverem ordenados, siga ao próximo par
  - Senão, troque-os de lugar
  - Repita a operação até que nenhuma troca possa ser feita no arranjo inteiro
- Ex: ordene em ordem crescente
  - Executando a primeira passada:

9 8 4 6 3

Ordenação – Método da Bolha

- E como ordenamos?
- Primeiro método: **Bubble sort**
  - Percorra todo o arranjo tomando seus elementos adjacentes para a par
  - Se os elemento no par estiverem ordenados, siga ao próximo par
  - Senão, troque-os de lugar
  - Repita a operação até que nenhuma troca possa ser feita no arranjo inteiro
- Ex: ordene em ordem crescente
  - Executando a primeira passada:

$$\underline{9 \quad 8} \quad 4 \quad 6 \quad 3$$

  - Comparando os dois primeiros números

Ordenação – Método da Bolha

- E como ordenamos?
- Primeiro método: **Bubble sort**
  - Percorra todo o arranjo tomando seus elementos adjacentes para a par
  - Se os elemento no par estiverem ordenados, siga ao próximo par
  - Senão, troque-os de lugar
  - Repita a operação até que nenhuma troca possa ser feita no arranjo inteiro
- Ex: ordene em ordem crescente
  - Executando a primeira passada:

$$\underline{8 \quad 9} \quad 4 \quad 6 \quad 3$$

  - Trocando porque $8<9$

Ordenação – Método da Bolha

- E como ordenamos?
- Primeiro método: **Bubble sort**
  - Percorra todo o arranjo tomando seus elementos adjacentes para a par
  - Se os elemento no par estiverem ordenados, siga ao próximo par
  - Senão, troque-os de lugar
  - Repita a operação até que nenhuma troca possa ser feita no arranjo inteiro
- Ex: ordene em ordem crescente
  - Executando a primeira passada:

$$8 \quad \underline{9 \quad 4} \quad 6 \quad 3$$

  - Comparando segundo e terceiro números

Ordenação – Método da Bolha

- E como ordenamos?
- Primeiro método: **Bubble sort**
  - Percorra todo o arranjo tomando seus elementos adjacentes para a par
  - Se os elemento no par estiverem ordenados, siga ao próximo par
  - Senão, troque-os de lugar
  - Repita a operação até que nenhuma troca possa ser feita no arranjo inteiro
- Ex: ordene em ordem crescente
  - Executando a primeira passada:

$$8 \quad \underline{4 \quad 9} \quad 6 \quad 3$$

  - Trocando porque $4<9$

Ordenação – Método da Bolha

- E como ordenamos?
- Primeiro método: **Bubble sort**
  - Percorra todo o arranjo tomando seus elementos adjacentes para a par
  - Se os elemento no par estiverem ordenados, siga ao próximo par
  - Senão, troque-os de lugar
  - Repita a operação até que nenhuma troca possa ser feita no arranjo inteiro
- Ex: ordene em ordem crescente
  - Executando a primeira passada:

$$8 \quad 4 \quad \underline{9 \quad 6} \quad 3$$

  - Comparando terceiro e quarto números

Ordenação – Método da Bolha

- E como ordenamos?
- Primeiro método: **Bubble sort**
  - Percorra todo o arranjo tomando seus elementos adjacentes para a par
  - Se os elemento no par estiverem ordenados, siga ao próximo par
  - Senão, troque-os de lugar
  - Repita a operação até que nenhuma troca possa ser feita no arranjo inteiro
- Ex: ordene em ordem crescente
  - Executando a primeira passada:

$$8 \quad 4 \quad \underline{6 \quad 9} \quad 3$$

  - Trocando porque $6<9$

Ordenação – Método da Bolha

- E como ordenamos?
- Primeiro método: **Bubble sort**
  - Percorra todo o arranjo tomando seus elementos adjacentes para a par
  - Se os elemento no par estiverem ordenados, siga ao próximo par
  - Senão, troque-os de lugar
  - Repita a operação até que nenhuma troca possa ser feita no arranjo inteiro
- Ex: ordene em ordem crescente
  - Executando a primeira passada:

$$8 \quad 4 \quad 6 \quad \underline{9 \quad 3}$$

  - Comparando quarto e quinto números

Ordenação – Método da Bolha

- E como ordenamos?
- Primeiro método: **Bubble sort**
  - Percorra todo o arranjo tomando seus elementos adjacentes para a par
  - Se os elemento no par estiverem ordenados, siga ao próximo par
  - Senão, troque-os de lugar
  - Repita a operação até que nenhuma troca possa ser feita no arranjo inteiro
- Ex: ordene em ordem crescente
  - Executando a primeira passada:

$$8 \quad 4 \quad 6 \quad \underline{3 \quad 9}$$

  - Trocando porque $3<9$

Ordenação – Método da Bolha

- E como ordenamos?
- Primeiro método: **Bubble sort**
  - Percorra todo o arranjo tomando seus elementos adjacentes para a par
  - Se os elemento no par estiverem ordenados, siga ao próximo par
  - Senão, troque-os de lugar
  - Repita a operação até que nenhuma troca possa ser feita no arranjo inteiro
- Ex: ordene em ordem crescente
  - Primeira passada completa. Último elemento fixado:

$$8 \quad 4 \quad 6 \quad \underline{3 \quad \mathbf{9}}$$

Ordenação – Método da Bolha

- E como ordenamos?
- Primeiro método: **Bubble sort**
  - Percorra todo o arranjo tomando seus elementos adjacentes para a par
  - Se os elemento no par estiverem ordenados, siga ao próximo par
  - Senão, troque-os de lugar
  - Repita a operação até que nenhuma troca possa ser feita no arranjo inteiro
- Ex: ordene em ordem crescente
  - Executando a segunda passada:

8	4	6	3	**9**
8	4	6	3	**9**

Ordenação – Método da Bolha

- E como ordenamos?
- Primeiro método: **Bubble sort**
  - Percorra todo o arranjo tomando seus elementos adjacentes para a par
  - Se os elemento no par estiverem ordenados, siga ao próximo par
  - Senão, troque-os de lugar
  - Repita a operação até que nenhuma troca possa ser feita no arranjo inteiro
- Ex: ordene em ordem crescente
  - Executando a segunda passada:

8	4	6	3	**9**
8	4	6	3	**9**

  - Comparando os dois primeiros números

Ordenação – Método da Bolha

- E como ordenamos?
- Primeiro método: **Bubble sort**
  - Percorra todo o arranjo tomando seus elementos adjacentes para a par
  - Se os elemento no par estiverem ordenados, siga ao próximo par
  - Senão, troque-os de lugar
  - Repita a operação até que nenhuma troca possa ser feita no arranjo inteiro
- Ex: ordene em ordem crescente
  - Executando a segunda passada:

$$\begin{array}{ccccc} 8 & 4 & 6 & 3 & \mathbf{9} \\ \underline{4} & \underline{8} & 6 & 3 & \mathbf{9} \end{array}$$

  - Trocando porque $4<8$

Ordenação – Método da Bolha

- E como ordenamos?
- Primeiro método: **Bubble sort**
  - Percorra todo o arranjo tomando seus elementos adjacentes para a par
  - Se os elemento no par estiverem ordenados, siga ao próximo par
  - Senão, troque-os de lugar
  - Repita a operação até que nenhuma troca possa ser feita no arranjo inteiro
- Ex: ordene em ordem crescente
  - Executando a segunda passada:

$$\begin{array}{ccccc} 8 & 4 & 6 & 3 & \mathbf{9} \\ 4 & 8 & 6 & 3 & \mathbf{9} \end{array}$$

  - Comparando segundo e terceiro números

Ordenação – Método da Bolha

- E como ordenamos?
- Primeiro método: **Bubble sort**
  - Percorra todo o arranjo tomando seus elementos adjacentes para a par
  - Se os elemento no par estiverem ordenados, siga ao próximo par
  - Senão, troque-os de lugar
  - Repita a operação até que nenhuma troca possa ser feita no arranjo inteiro
- Ex: ordene em ordem crescente
  - Executando a segunda passada:

$$\begin{array}{ccccc} 8 & 4 & 6 & 3 & \mathbf{9} \\ 4 & 6 & 8 & 3 & \mathbf{9} \end{array}$$

  - Trocando porque $6<8$

Ordenação – Método da Bolha

- E como ordenamos?
- Primeiro método: **Bubble sort**
  - Percorra todo o arranjo tomando seus elementos adjacentes para a par
  - Se os elemento no par estiverem ordenados, siga ao próximo par
  - Senão, troque-os de lugar
  - Repita a operação até que nenhuma troca possa ser feita no arranjo inteiro
- Ex: ordene em ordem crescente
  - Executando a segunda passada:

8	4	6	3	**9**
4	6	8	3	**9**

  - Comparando terceiro e quarto números

Ordenação – Método da Bolha

- E como ordenamos?
- Primeiro método: **Bubble sort**
  - Percorra todo o arranjo tomando seus elementos adjacentes para a par
  - Se os elemento no par estiverem ordenados, siga ao próximo par
  - Senão, troque-os de lugar
  - Repita a operação até que nenhuma troca possa ser feita no arranjo inteiro
- Ex: ordene em ordem crescente
  - Executando a segunda passada:

8	4	6	3	**9**
4	6	3	8	**9**

  - Trocando porque 3<8

Ordenação – Método da Bolha

- E como ordenamos?
- Primeiro método: **Bubble sort**
  - Percorra todo o arranjo tomando seus elementos adjacentes para a par
  - Se os elemento no par estiverem ordenados, siga ao próximo par
  - Senão, troque-os de lugar
  - Repita a operação até que nenhuma troca possa ser feita no arranjo inteiro
- Ex: ordene em ordem crescente
  - Segunda passada completa. Último elemento fixado:

8	4	6	3	**9**
4	6	3	**8**	**9**

Ordenação – Método da Bolha

- E como ordenamos?
- Primeiro método: **Bubble sort**
  - Percorra todo o arranjo tomando seus elementos adjacentes para a par
  - Se os elemento no par estiverem ordenados, siga ao próximo par
  - Senão, troque-os de lugar
  - Repita a operação até que nenhuma troca possa ser feita no arranjo inteiro
- Ex: ordene em ordem crescente
  - Executando a terceira passada:

8	4	6	3	**9**
4	6	3	**8**	**9**
4	6	3	**8**	**9**

Ordenação – Método da Bolha

- E como ordenamos?
- Primeiro método: **Bubble sort**
  - Percorra todo o arranjo tomando seus elementos adjacentes para a par
  - Se os elemento no par estiverem ordenados, siga ao próximo par
  - Senão, troque-os de lugar
  - Repita a operação até que nenhuma troca possa ser feita no arranjo inteiro
- Ex: ordene em ordem crescente
  - Executando a terceira passada:

8	4	6	3	**9**
4	6	3	**8**	**9**
4	6	3	**8**	**9**

  - Comparando os dois primeiros números

Ordenação – Método da Bolha

- E como ordenamos?
- Primeiro método: **Bubble sort**
  - Percorra todo o arranjo tomando seus elementos adjacentes para a par
  - Se os elemento no par estiverem ordenados, siga ao próximo par
  - Senão, troque-os de lugar
  - Repita a operação até que nenhuma troca possa ser feita no arranjo inteiro
- Ex: ordene em ordem crescente
  - Executando a terceira passada:

8	4	6	3	**9**
4	6	3	**8**	**9**
4	6	3	**8**	**9**

  - $6>4$, logo não há necessidade de troca

Ordenação – Método da Bolha

- E como ordenamos?
- Primeiro método: **Bubble sort**
  - Percorra todo o arranjo tomando seus elementos adjacentes para a par
  - Se os elemento no par estiverem ordenados, siga ao próximo par
  - Senão, troque-os de lugar
  - Repita a operação até que nenhuma troca possa ser feita no arranjo inteiro
- Ex: ordene em ordem crescente
  - Executando a terceira passada:

8	4	6	3	**9**
4	6	3	**8**	**9**
4	6	3	**8**	**9**

  - Comparando os segundo e terceiro números

Ordenação – Método da Bolha

- E como ordenamos?
- Primeiro método: **Bubble sort**
  - Percorra todo o arranjo tomando seus elementos adjacentes para a par
  - Se os elemento no par estiverem ordenados, siga ao próximo par
  - Senão, troque-os de lugar
  - Repita a operação até que nenhuma troca possa ser feita no arranjo inteiro
- Ex: ordene em ordem crescente
  - Executando a terceira passada:

8	4	6	3	**9**
4	6	3	**8**	**9**
4	3	6	**8**	**9**

  - Trocando porque 3<6

Ordenação – Método da Bolha

- E como ordenamos?
- Primeiro método: **Bubble sort**
  - Percorra todo o arranjo tomando seus elementos adjacentes para a par
  - Se os elemento no par estiverem ordenados, siga ao próximo par
  - Senão, troque-os de lugar
  - Repita a operação até que nenhuma troca possa ser feita no arranjo inteiro
- Ex: ordene em ordem crescente
  - Terceira passada completa. Último elemento fixado:

8	4	6	3	**9**
4	6	3	**8**	**9**
4	3	**6**	**8**	**9**

Ordenação – Método da Bolha

- E como ordenamos?
- Primeiro método: **Bubble sort**
  - Percorra todo o arranjo tomando seus elementos adjacentes para a par
  - Se os elemento no par estiverem ordenados, siga ao próximo par
  - Senão, troque-os de lugar
  - Repita a operação até que nenhuma troca possa ser feita no arranjo inteiro
- Ex: ordene em ordem crescente
  - Executando a quarta passada:

8	4	6	3	**9**
4	6	3	**8**	**9**
4	3	**6**	**8**	**9**
4	3	**6**	**8**	**9**

Ordenação – Método da Bolha

- E como ordenamos?
- Primeiro método: **Bubble sort**
  - Percorra todo o arranjo tomando seus elementos adjacentes para a par
  - Se os elemento no par estiverem ordenados, siga ao próximo par
  - Senão, troque-os de lugar
  - Repita a operação até que nenhuma troca possa ser feita no arranjo inteiro
- Ex: ordene em ordem crescente
  - Executando a quarta passada:

8	4	6	3	**9**
4	6	3	**8**	**9**
4	3	**6**	**8**	**9**
4	3	**6**	**8**	**9**

  - Comparando os dois primeiros números

Ordenação – Método da Bolha

- E como ordenamos?
- Primeiro método: **Bubble sort**
  - Percorra todo o arranjo tomando seus elementos adjacentes para a par
  - Se os elemento no par estiverem ordenados, siga ao próximo par
  - Senão, troque-os de lugar
  - Repita a operação até que nenhuma troca possa ser feita no arranjo inteiro
- Ex: ordene em ordem crescente
  - Executando a quarta passada:

8	4	6	3	**9**
4	6	3	**8**	**9**
4	3	**6**	**8**	**9**
3	4	**6**	**8**	**9**

  - Trocando porque 3<4

Ordenação – Método da Bolha

- E como ordenamos?
- Primeiro método: **Bubble sort**
  - Percorra todo o arranjo tomando seus elementos adjacentes para a par
  - Se os elemento no par estiverem ordenados, siga ao próximo par
  - Senão, troque-os de lugar
  - Repita a operação até que nenhuma troca possa ser feita no arranjo inteiro
- Ex: ordene em ordem crescente
  - Quarta passada completa. Último elemento fixado:

8	4	6	3	**9**
4	6	3	**8**	**9**
4	3	**6**	**8**	**9**
3	**4**	**6**	**8**	**9**

Ordenação – Método da Bolha

- E como ordenamos?
- Primeiro método: **Bubble sort**
  - Percorra todo o arranjo tomando seus elementos adjacentes para a par
  - Se os elemento no par estiverem ordenados, siga ao próximo par
  - Senão, troque-os de lugar
  - Repita a operação até que nenhuma troca possa ser feita no arranjo inteiro
- Ex: ordene em ordem crescente
  - Quarta passada completa. Último elemento fixado:

8	4	6	3	**9**
4	6	3	**8**	**9**
4	3	**6**	**8**	**9**
3	**4**	**6**	**8**	**9**

  - Há somente um elemento no arranjo. O algoritmo pára. O arranjo está ordenado.

Ordenação – Método da Bolha

- Observe que não precisamos correr sempre o arranjo até o final

Ordenação – Método da Bolha

- Observe que não precisamos correr sempre o arranjo até o final
  - Basta irmos até onde garantimos estar ordenado...

Ordenação – Método da Bolha

- Observe que não precisamos correr sempre o arranjo até o final
  - Basta irmos até onde garantimos estar ordenado...
  - Como?

Ordenação – Método da Bolha

- Observe que não precisamos correr sempre o arranjo até o final
  - Basta irmos até onde garantimos estar ordenado...
  - Como?
    - ★ Marcando esse final

Ordenação – Método da Bolha

- Observe que não precisamos correr sempre o arranjo até o final
  - Basta irmos até onde garantimos estar ordenado...
  - Como?
    - ★ Marcando esse final

```
static void bolha(int[] v) {
    for (int ult = v.length-1; ult>0; ult--) {
        for (int i=0; i<ult; i++) {
            if (v[i] > v[i+1]) {
                int aux = v[i];
                v[i] = v[i+1];
                v[i+1] = aux;
            }
        }
    }
}

public static void main(String[] args) {
    int[] v = {55,0,-78,-4,32,200,52,63,69,125};

    bolha(v);

    for (int el : v)
        System.out.print(el+" ");
    System.out.println();
}
```

Ordenação – Método da Bolha

- Observe que não precisamos correr sempre o arranjo até o final
  - Basta irmos até onde garantimos estar ordenado...
  - Como?
    - ★ Marcando esse final
    - ★ *ult* é esse marcador

```
static void bolha(int[] v) {
    for (int ult = v.length-1; ult>0; ult--) {
        for (int i=0; i<ult; i++) {
            if (v[i] > v[i+1]) {
                int aux = v[i];
                v[i] = v[i+1];
                v[i+1] = aux;
            }
        }
    }
}

public static void main(String[] args) {
    int[] v = {55,0,-78,-4,32,200,52,63,69,125};

    bolha(v);

    for (int el : v)
        System.out.print(el+" ");
    System.out.println();
}
```

Ordenação – Método da Bolha

- Observe que não precisamos correr sempre o arranjo até o final
  - Basta irmos até onde garantimos estar ordenado...
  - Como?
    - ★ Marcando esse final
    - ★ *ult* é esse marcador
    - ★ Sempre corremos de 0 a *ult*

```
static void bolha(int[] v) {
    for (int ult = v.length-1; ult>0; ult--) {
        for (int i=0; i<ult; i++) {
            if (v[i] > v[i+1]) {
                int aux = v[i];
                v[i] = v[i+1];
                v[i+1] = aux;
            }
        }
    }
}

public static void main(String[] args) {
    int[] v = {55,0,-78,-4,32,200,52,63,69,125};

    bolha(v);

    for (int el : v)
        System.out.print(el+" ");
    System.out.println();
}
```

Ordenação – Método da Bolha

- Observe que não precisamos correr sempre o arranjo até o final
  - Basta irmos até onde garantimos estar ordenado...
  - Como?
    - ★ Marcando esse final
    - ★ *ult* é esse marcador
    - ★ Sempre corremos de 0 a *ult*
    - ★ E a cada passada, decrementamos *ult*

```
static void bolha(int[] v) {
    for (int ult = v.length-1; ult>0; ult--) {
        for (int i=0; i<ult; i++) {
            if (v[i] > v[i+1]) {
                int aux = v[i];
                v[i] = v[i+1];
                v[i+1] = aux;
            }
        }
    }
}

public static void main(String[] args) {
    int[] v = {55,0,-78,-4,32,200,52,63,69,125};

    bolha(v);

    for (int el : v)
        System.out.print(el+" ");
    System.out.println();
}
```

Ordenação – Método da Bolha

- Observe que não precisamos correr sempre o arranjo até o final
  - Basta irmos até onde garantimos estar ordenado...
  - Como?
    - ★ Marcando esse final
    - ★ *ult* é esse marcador
    - ★ Sempre corremos de 0 a *ult*
    - ★ E a cada passada, decrementamos *ult*

```
static void bolha(int[] v) {
    for (int ult = v.length-1; ult>0; ult--) {
        for (int i=0; i<ult; i++) {
            if (v[i] > v[i+1]) {
                int aux = v[i];
                v[i] = v[i+1];
                v[i+1] = aux;
            }
        }
    }
}

public static void main(String[] args) {
    int[] v = {55,0,-78,-4,32,200,52,63,69,125};

    bolha(v);

    for (int el : v)
        System.out.print(el+" ");
    System.out.println();
}
```

Saída

```
$ java Projeto
-78 -4 0 32 52 55 63 69 125 200
```

Ordenação – Método da Bolha

- Observe que não precisamos correr sempre o arranjo até o final
  - Basta irmos até onde garantimos estar ordenado...
  - Como?
    - ★ Marcando esse final
    - ★ *ult* é esse marcador
    - ★ Sempre corremos de 0 a *ult*
    - ★ E a cada passada, decrementamos *ult*

```
static void bolha(int[] v) {
    for (int ult = v.length-1; ult>0; ult--) {
        for (int i=0; i<ult; i++) {
            if (v[i] > v[i+1]) {
                int aux = v[i];
                v[i] = v[i+1];
                v[i+1] = aux;
            }
        }
    }
}

public static void main(String[] args) {
    int[] v = {55,0,-78,-4,32,200,52,63,69,125};

    bolha(v);

    for (int el : v)
        System.out.print(el+" ");
    System.out.println();
}
```

Saída

```
$ java Projeto
-78 -4 0 32 52 55 63 69 125 200
```

Cuidado! Esse método modifica o arranjo original!

Ordenação – Método da Bolha

- E se em vez de inteiros, quisermos usar objetos?

Ordenação – Método da Bolha

- E se em vez de inteiros, quisermos usar objetos?
  - Como nossa *Residencia*

Ordenação – Método da Bolha

- E se em vez de inteiros, quisermos usar objetos?
  - Como nossa *Residencia*
- O problema está em comparar os objetos

Ordenação – Método da Bolha

- E se em vez de inteiros, quisermos usar objetos?
  - Como nossa *Residencia*
- O problema está em comparar os objetos
  - Não podemos fazer *obj1 == obj2*

Ordenação – Método da Bolha

- E se em vez de inteiros, quisermos usar objetos?
  - Como nossa *Residencia*
- O problema está em comparar os objetos
  - Não podemos fazer *obj1 == obj2*
  - Resta apenas definirmos um método que compare objetos com base em seus atributos

Ordenação – Método da Bolha

- E se em vez de inteiros, quisermos usar objetos?
  - Como nossa *Residencia*
- O problema está em comparar os objetos
  - Não podemos fazer *obj1 == obj2*
  - Resta apenas definirmos um método que compare objetos com base em seus atributos
  - Por exemplo, a área total

```
class Residencia {
    ...
    int comparaRes(Residencia outra) {
        if (outra == null) return(1); // esta é maior
        return((int)(this.area() - outra.area()));
    }

    static void bolha(Residencia[] v) {
        for (int ult = v.length-1; ult>0; ult--) {
            for (int i=0; i<ult; i++) {
                if (v[i].comparaRes(v[i+1]) > 0) {
                    Residencia aux = v[i];
                    v[i] = v[i+1];
                    v[i+1] = aux;
                }
            }
        }
    }

    public static void main(String[] args) {
        AreaCasa c = new AreaCasa();
        AreaPiscina p = new AreaPiscina();
        Residencia r1 = new Residencia(c,p);

        c = new AreaCasa();
        p = new AreaPiscina(11);
        Residencia r2 = new Residencia(c,p);

        System.out.println(r1.comparaRes(r2));
    }
}
```

Ordenação – Método da Bolha

- Então...

```
public static void main(String[] args) {
    Projeto pr = new Projeto(5);

    for (int i=0; i<5; i++) {
        AreaCasa c = new AreaCasa(Math.random()*100,
                                  Math.random()*30);
        AreaPiscina p = new AreaPiscina(
                                  Math.random()*10);
        Residencia r = new Residencia(c,p);
        pr.adicionaRes(r);
    }

    for (Residencia r : pr.condominio)
        System.out.println(r.area());
    System.out.println();

    bolha(pr.condominio);

    for (Residencia r : pr.condominio)
        System.out.println(r.area());
}
```

Ordenação – Método da Bolha

- Então...
- *Math.random()*?

```
public static void main(String[] args) {
    Projeto pr = new Projeto(5);

    for (int i=0; i<5; i++) {
        AreaCasa c = new AreaCasa(Math.random()*100,
                                  Math.random()*30);
        AreaPiscina p = new AreaPiscina(
                                  Math.random()*10);
        Residencia r = new Residencia(c,p);
        pr.adicionaRes(r);
    }

    for (Residencia r : pr.condominio)
        System.out.println(r.area());
    System.out.println();

    bolha(pr.condominio);

    for (Residencia r : pr.condominio)
        System.out.println(r.area());
}
```

Ordenação – Método da Bolha

- Então...
- *Math.random()*?
  - Gera um número pseudo-aleatório $0 \leq n < 1$

```
public static void main(String[] args) {
    Projeto pr = new Projeto(5);

    for (int i=0; i<5; i++) {
        AreaCasa c = new AreaCasa(Math.random()*100,
                                  Math.random()*30);
        AreaPiscina p = new AreaPiscina(
                                  Math.random()*10);
        Residencia r = new Residencia(c,p);
        pr.adicionaRes(r);
    }

    for (Residencia r : pr.condominio)
        System.out.println(r.area());
    System.out.println();

    bolha(pr.condominio);

    for (Residencia r : pr.condominio)
        System.out.println(r.area());
}
```

Ordenação – Método da Bolha

- Então...
- *Math.random()*?
  - Gera um número pseudo-aleatório $0 \leq n < 1$

Saída

```
$ java Projeto
2055.600048644749
949.1972834436008
3316.7903140566305
6698.682789640099
584.1255507074843

584.1255507074843
949.1972834436008
2055.600048644749
3316.7903140566305
6698.682789640099
```

```
public static void main(String[] args) {
    Projeto pr = new Projeto(5);

    for (int i=0; i<5; i++) {
        AreaCasa c = new AreaCasa(Math.random()*100,
                                  Math.random()*30);
        AreaPiscina p = new AreaPiscina(
                                  Math.random()*10);
        Residencia r = new Residencia(c,p);
        pr.adicionaRes(r);
    }

    for (Residencia r : pr.condominio)
        System.out.println(r.area());
    System.out.println();

    bolha(pr.condominio);

    for (Residencia r : pr.condominio)
        System.out.println(r.area());
}
```

Ordenação – Seleção

- Segundo método: **Selection sort**

Ordenação – Seleção

- Segundo método: **Selection sort**
  - Primeiro encontre o menor elemento do arranjo

Ordenação – Seleção

- Segundo método: **Selection sort**
  - Primeiro encontre o menor elemento do arranjo
  - Então troque esse elemento de lugar com o que está na primeira posição

Ordenação – Seleção

- Segundo método: **Selection sort**
  - Primeiro encontre o menor elemento do arranjo
  - Então troque esse elemento de lugar com o que está na primeira posição
  - Encontre o segundo menor do arranjo

Ordenação – Seleção

- Segundo método: **Selection sort**
  - Primeiro encontre o menor elemento do arranjo
  - Então troque esse elemento de lugar com o que está na primeira posição
  - Encontre o segundo menor do arranjo
  - Troque com o da segunda posição

Ordenação – Seleção

- Segundo método: **Selection sort**
  - Primeiro encontre o menor elemento do arranjo
  - Então troque esse elemento de lugar com o que está na primeira posição
  - Encontre o segundo menor do arranjo
  - Troque com o da segunda posição
  - E assim por diante, até chegar ao fim do arranjo

Ordenação – Seleção

- Ex: ordene em ordem crescente

9 8 4 6 3

Ordenação – Seleção

- Ex: ordene em ordem crescente
  - Executando a primeira passada:

$$\begin{array}{ccccc} 9 & 8 & 4 & 6 & 3 \\ \underline{9} & 8 & 4 & 6 & 3 \end{array}$$

Ordenação – Seleção

- Ex: ordene em ordem crescente
  - Executando a primeira passada:

9	8	4	6	3
9	8	4	6	3

  - Encontrando o menor elemento

Ordenação – Seleção

- Ex: ordene em ordem crescente
  - Executando a primeira passada:

9	8	4	6	3
3	8	4	6	9

  - Trocando com o primeiro elemento, pois $3 < 9$

Ordenação – Seleção

- Ex: ordene em ordem crescente
  - Primeira passada completa. Primeiro elemento fixado:

9	8	4	6	3
3	8	4	6	9

Ordenação – Seleção

- Ex: ordene em ordem crescente
  - Executando a segunda passada:

9	8	4	6	3
3	8	4	6	9
3	8	4	6	9

Ordenação – Seleção

- Ex: ordene em ordem crescente
  - Executando a segunda passada:

9	8	4	6	3
3	8	4	6	9
3	8	4	6	9

  - Encontrando o segundo menor elemento

Ordenação – Seleção

- Ex: ordene em ordem crescente
  - Executando a segunda passada:

9	8	4	6	3
3	8	4	6	9
3	4	8	6	9

  - Trocando com o segundo elemento, pois $4 < 8$

Ordenação – Seleção

- Ex: ordene em ordem crescente
  - Segunda passada completa. Segundo elemento fixado:

9	8	4	6	3
3	8	4	6	9
3	**4**	8	6	9

Ordenação – Seleção

- Ex: ordene em ordem crescente
  - Executando a terceira passada:

9	8	4	6	3
3	8	4	6	9
3	**4**	8	6	9
3	**4**	8	6	9

Ordenação – Seleção

- Ex: ordene em ordem crescente
  - Executando a terceira passada:

9	8	4	6	3
3	8	4	6	9
3	**4**	8	6	9
3	**4**	8	6	9

  - Encontrando o terceiro menor elemento

Ordenação – Seleção

- Ex: ordene em ordem crescente
  - Executando a terceira passada:

9	8	4	6	3
3	8	4	6	9
3	**4**	8	6	9
3	**4**	6	8	9

  - Trocando com o terceiro elemento, pois $6 < 8$

Ordenação – Seleção

- Ex: ordene em ordem crescente
  - Terceira passada completa. Terceiro elemento fixado:

9	8	4	6	3
3	8	4	6	9
3	**4**	8	6	9
3	**4**	**6**	8	9

Ordenação – Seleção

- Ex: ordene em ordem crescente
  - Executando a quarta passada:

9	8	4	6	3
3	8	4	6	9
3	**4**	8	6	9
3	**4**	**6**	8	9
3	**4**	**6**	8	9

Ordenação – Seleção

- Ex: ordene em ordem crescente
  - Executando a quarta passada:

9	8	4	6	3
3	8	4	6	9
3	**4**	8	6	9
3	**4**	**6**	8	9
3	**4**	**6**	8	9

  - Encontrando o quarto menor elemento

Ordenação – Seleção

- Ex: ordene em ordem crescente
  - Executando a quarta passada:

9	8	4	6	3
3	8	4	6	9
3	**4**	8	6	9
3	**4**	**6**	8	9
3	**4**	**6**	8	9

  - Não há troca, pois já está na quarta posição

Ordenação – Seleção

- Ex: ordene em ordem crescente
  - Quarta passada completa. Quarto elemento fixado:

9	8	4	6	3
3	8	4	6	9
3	**4**	8	6	9
3	**4**	**6**	8	9
3	**4**	**6**	**8**	9

Ordenação – Seleção

- Ex: ordene em ordem crescente
  - Executando a quinta passada:

9	8	4	6	3
3	8	4	6	9
3	**4**	8	6	9
3	**4**	**6**	8	9
3	**4**	**6**	**8**	9
3	**4**	**6**	**8**	9

Ordenação – Seleção

- Ex: ordene em ordem crescente
  - Executando a quinta passada:

9	8	4	6	3
3	8	4	6	9
3	**4**	8	6	9
3	**4**	**6**	8	9
3	**4**	**6**	**8**	9
3	**4**	**6**	**8**	9

  - Última posição do arranjo. O algoritmo pára

Ordenação – Seleção

- Ex: ordene em ordem crescente
  - Quinta passada completa. Quinto elemento fixado:

9	8	4	6	3
3	8	4	6	9
3	**4**	8	6	9
3	**4**	**6**	8	9
3	**4**	**6**	**8**	9
3	**4**	**6**	**8**	**9**

Ordenação – Seleção

- Então...

```
static int posMenorEl(int[] v, int inicio) {
    int posMenor = -1;
    if ((v!=null) && (inicio>=0)
                          && (inicio < v.length)) {
        posMenor = inicio;
        for (int i=inicio+1; i<v.length; i++)
            if (v[i] < v[posMenor]) posMenor = i;
    }
    return(posMenor);
}

static void selecao(int[] v) {
    for (int i=0; i<v.length-1; i++) {
        int posMenor = posMenorEl(v,i);
        if (v[posMenor] < v[i]) {
            int aux = v[i];
            v[i] = v[posMenor];
            v[posMenor] = aux;
        }
    }
}

public static void main(String[] args) {
    int[] v = {9,8,4,6,3};
    selecao(v);
    for (int el : v) System.out.print(el+" ");
    System.out.println();
}
```

Ordenação – Seleção

- Então...
  - Método que diz a posição do menor elemento em subvetor $inicio \leq i < fim$

```
static int posMenorEl(int[] v, int inicio) {
    int posMenor = -1;
    if ((v!=null) && (inicio>=0)
                          && (inicio < v.length)) {
        posMenor = inicio;
        for (int i=inicio+1; i<v.length; i++)
            if (v[i] < v[posMenor]) posMenor = i;
    }
    return(posMenor);
}

static void selecao(int[] v) {
    for (int i=0; i<v.length-1; i++) {
        int posMenor = posMenorEl(v,i);
        if (v[posMenor] < v[i]) {
            int aux = v[i];
            v[i] = v[posMenor];
            v[posMenor] = aux;
        }
    }
}

public static void main(String[] args) {
    int[] v = {9,8,4,6,3};
    selecao(v);
    for (int el : v) System.out.print(el+" ");
    System.out.println();
}
```

Ordenação – Seleção

- Então...
  - Método que diz a posição do menor elemento em subvetor $inicio \leq i < fim$
    - ★ Sempre é bom testar a entrada

```
static int posMenorEl(int[] v, int inicio) {
    int posMenor = -1;
    if ((v!=null) && (inicio>=0)
                          && (inicio < v.length)) {
        posMenor = inicio;
        for (int i=inicio+1; i<v.length; i++)
            if (v[i] < v[posMenor]) posMenor = i;
    }
    return(posMenor);
}

static void selecao(int[] v) {
    for (int i=0; i<v.length-1; i++) {
        int posMenor = posMenorEl(v,i);
        if (v[posMenor] < v[i]) {
            int aux = v[i];
            v[i] = v[posMenor];
            v[posMenor] = aux;
        }
    }
}

public static void main(String[] args) {
    int[] v = {9,8,4,6,3};
    selecao(v);
    for (int el : v) System.out.print(el+" ");
    System.out.println();
}
```

Ordenação – Seleção

- Então...
  - Método que diz a posição do menor elemento em subvetor $inicio \leq i < fim$
    - ★ Sempre é bom testar a entrada
  - Para cada elemento do arranjo (exceto o último, que sobra já ordenado)

```
static int posMenorEl(int[] v, int inicio) {
    int posMenor = -1;
    if ((v!=null) && (inicio>=0)
                          && (inicio < v.length)) {
        posMenor = inicio;
        for (int i=inicio+1; i<v.length; i++)
            if (v[i] < v[posMenor]) posMenor = i;
    }
    return(posMenor);
}

static void selecao(int[] v) {
    for (int i=0; i<v.length-1; i++) {
        int posMenor = posMenorEl(v,i);
        if (v[posMenor] < v[i]) {
            int aux = v[i];
            v[i] = v[posMenor];
            v[posMenor] = aux;
        }
    }
}

public static void main(String[] args) {
    int[] v = {9,8,4,6,3};
    selecao(v);
    for (int el : v) System.out.print(el+" ");
    System.out.println();
}
```

Ordenação – Seleção

- Então...
  - Método que diz a posição do menor elemento em subvetor $inicio \leq i < fim$
    - ★ Sempre é bom testar a entrada
  - Para cada elemento do arranjo (exceto o último, que sobra já ordenado)
    - ★ Busca o menor elemento a partir desse

```
static int posMenorEl(int[] v, int inicio) {
    int posMenor = -1;
    if ((v!=null) && (inicio>=0)
                          && (inicio < v.length)) {
        posMenor = inicio;
        for (int i=inicio+1; i<v.length; i++)
            if (v[i] < v[posMenor]) posMenor = i;
    }
    return(posMenor);
}

static void selecao(int[] v) {
    for (int i=0; i<v.length-1; i++) {
        int posMenor = posMenorEl(v,i);
        if (v[posMenor] < v[i]) {
            int aux = v[i];
            v[i] = v[posMenor];
            v[posMenor] = aux;
        }
    }
}

public static void main(String[] args) {
    int[] v = {9,8,4,6,3};
    selecao(v);
    for (int el : v) System.out.print(el+" ");
    System.out.println();
}
```

Ordenação – Seleção

- Então...
  - Método que diz a posição do menor elemento em subvetor $inicio \leq i < fim$
    - ★ Sempre é bom testar a entrada
  - Para cada elemento do arranjo (exceto o último, que sobra já ordenado)
    - ★ Busca o menor elemento a partir desse
    - ★ Troca com a posição desse elemento, se for o caso

```
static int posMenorEl(int[] v, int inicio) {
    int posMenor = -1;
    if ((v!=null) && (inicio>=0)
                          && (inicio < v.length)) {
        posMenor = inicio;
        for (int i=inicio+1; i<v.length; i++)
            if (v[i] < v[posMenor]) posMenor = i;
    }
    return(posMenor);
}

static void selecao(int[] v) {
    for (int i=0; i<v.length-1; i++) {
        int posMenor = posMenorEl(v,i);
        if (v[posMenor] < v[i]) {
            int aux = v[i];
            v[i] = v[posMenor];
            v[posMenor] = aux;
        }
    }
}

public static void main(String[] args) {
    int[] v = {9,8,4,6,3};
    selecao(v);
    for (int el : v) System.out.print(el+" ");
    System.out.println();
}
```

Ordenação – Seleção

- E como fica com objetos?

```
static int posMenorEl(Residencia[] v, int inicio) {
    int posMenor = -1;
    if ((v!=null) && (inicio>=0) &&
                            (inicio < v.length)) {
        posMenor = inicio;
        for (int i=inicio+1; i<v.length; i++) {
            if (v[i].comparaRes(v[posMenor]) < 0)
                posMenor = i;
        }
    }
    return(posMenor);
}

static void selecao(Residencia[] v) {
    for (int i=0; i<v.length-1; i++) {
        int posMenor = posMenorEl(v,i);
        if (v[posMenor].comparaRes(v[i]) < 0) {
            Residencia aux = v[i];
            v[i] = v[posMenor];
            v[posMenor] = aux;
        }
    }
}
```

```
public static void main(String[] args) {
    Projeto pr = new Projeto(5);

    for (int i=0; i<5; i++) {
        AreaCasa c = new AreaCasa(Math.random()*100,
                                  Math.random()*30);
        AreaPiscina p = new AreaPiscina(
                                  Math.random()*10);
        Residencia r = new Residencia(c,p);
        pr.adicionaRes(r);
    }

    for (Residencia r : pr.condominio)
        System.out.println(r.area());
    System.out.println();

    selecao(pr.condominio);

    for (Residencia r : pr.condominio)
        System.out.println(r.area());
}
```

Ordenação – Inserção

- Terceiro método: **Insertion sort**

Ordenação – Inserção

- Terceiro método: **Insertion sort**
  - Percorremos o arranjo e, a cada novo elemento:

Ordenação – Inserção

- Terceiro método: **Insertion sort**
  - Percorremos o arranjo e, a cada novo elemento:
    - ★ Procuramos onde, à esquerda desse elemento, ele se encaixa

Ordenação – Inserção

- Terceiro método: **Insertion sort**
  - Percorremos o arranjo e, a cada novo elemento:
    - ★ Procuramos onde, à esquerda desse elemento, ele se encaixa
    - ★ Abrimos espaço para o elemento lá, deslocando para a direita todos os elementos que estão entre essa posição e a original do elemento

Ordenação – Inserção

- Terceiro método: **Insertion sort**
  - Percorremos o arranjo e, a cada novo elemento:
    - ★ Procuramos onde, à esquerda desse elemento, ele se encaixa
    - ★ Abrimos espaço para o elemento lá, deslocando para a direita todos os elementos que estão entre essa posição e a original do elemento
    - ★ Inserimos o elemento nesse espaço assim aberto

Ordenação – Inserção

- Terceiro método: **Insertion sort**
  - Percorremos o arranjo e, a cada novo elemento:
    - ★ Procuramos onde, à esquerda desse elemento, ele se encaixa
    - ★ Abrimos espaço para o elemento lá, deslocando para a direita todos os elementos que estão entre essa posição e a original do elemento
    - ★ Inserimos o elemento nesse espaço assim aberto
  - Ou seja, aumentamos a parte ordenada do array em uma posição, inserindo um novo elemento na posição correta e deslocando os demais para a direita

Ordenação – Inserção

- Terceiro método: **Insertion sort**
  - Percorremos o arranjo e, a cada novo elemento:
    - ★ Procuramos onde, à esquerda desse elemento, ele se encaixa
    - ★ Abrimos espaço para o elemento lá, deslocando para a direita todos os elementos que estão entre essa posição e a original do elemento
    - ★ Inserimos o elemento nesse espaço assim aberto
  - Ou seja, aumentamos a parte ordenada do array em uma posição, inserindo um novo elemento na posição correta e deslocando os demais para a direita
- Semelhante ao modo como ordenamos cartas de baralho

Ordenação – Inserção

- Terceiro método: **Insertion sort**
  - Percorremos o arranjo e, a cada novo elemento:
    - ★ Procuramos onde, à esquerda desse elemento, ele se encaixa
    - ★ Abrimos espaço para o elemento lá, deslocando para a direita todos os elementos que estão entre essa posição e a original do elemento
    - ★ Inserimos o elemento nesse espaço assim aberto
  - Ou seja, aumentamos a parte ordenada do array em uma posição, inserindo um novo elemento na posição correta e deslocando os demais para a direita
- Semelhante ao modo como ordenamos cartas de baralho
  - Percorremos da esquerda para a direita e, à medida que avançamos vamos deixando as cartas mais à esquerda ordenadas

Ordenação – Inserção

- Ex: ordene em ordem crescente

9 8 4 10 6

Ordenação – Inserção

- Ex: ordene em ordem crescente
  - Em azul está o sub-arranjo já ordenado

9 8 4 10 6

Ordenação – Inserção

- Ex: ordene em ordem crescente
  - Em azul está o sub-arranjo já ordenado
  - Analisando o primeiro elemento a ser inserido:

9	8	4	10	6
9	8	4	10	6

Ordenação – Inserção

- Ex: ordene em ordem crescente
  - Em azul está o sub-arranjo já ordenado
  - Analisando o primeiro elemento a ser inserido:

$$\begin{array}{ccccc} 9 & 8 & 4 & 10 & 6 \\ 9 & \underline{8} & 4 & 10 & 6 \\ \uparrow & & & & \end{array}$$

  - Identificamos onde ele deve estar, na parte ordenada

Ordenação – Inserção

- Ex: ordene em ordem crescente
  - Em azul está o sub-arranjo já ordenado
  - Analisando o primeiro elemento a ser inserido:

9	8	4	10	6
9	9	4	10	6
↑	→			

  - Deslocamos a parte ordenada a partir dessa posição

Ordenação – Inserção

- Ex: ordene em ordem crescente
  - Em azul está o sub-arranjo já ordenado
  - Analisando o primeiro elemento a ser inserido:

9	8	4	10	6
8	9	4	10	6
↑	———			

  - Inserimos o 8 na posição correta

Ordenação – Inserção

- Ex: ordene em ordem crescente
  - Em azul está o sub-arranjo já ordenado
  - Analisando o segundo elemento a ser inserido:

9	8	4	10	6
8	9	4	10	6
8	9	4	10	6

Ordenação – Inserção

- Ex: ordene em ordem crescente
  - Em azul está o sub-arranjo já ordenado
  - Analisando o segundo elemento a ser inserido:

$$\begin{array}{ccccc} 9 & 8 & 4 & 10 & 6 \\ 8 & 9 & 4 & 10 & 6 \\ 8 & 9 & \underline{\quad 4 \quad} & 10 & 6 \\ \uparrow & & & & \end{array}$$

  - Identificamos onde ele deve estar, na parte ordenada

Ordenação – Inserção

- Ex: ordene em ordem crescente
  - Em azul está o sub-arranjo já ordenado
  - Analisando o segundo elemento a ser inserido:

$$
\begin{array}{ccccc}
9 & 8 & 4 & 10 & 6 \\
8 & 9 & 4 & 10 & 6 \\
8 & 8 & 9 & 10 & 6 \\
\uparrow & \rightarrow & \underline{\quad} & &
\end{array}
$$

  - Deslocamos a parte ordenada a partir dessa posição

Ordenação – Inserção

- Ex: ordene em ordem crescente
  - Em azul está o sub-arranjo já ordenado
  - Analisando o segundo elemento a ser inserido:

$$
\begin{array}{ccccc}
9 & 8 & 4 & 10 & 6 \\
8 & 9 & 4 & 10 & 6 \\
4 & 8 & \underline{9} & 10 & 6 \\
\uparrow & & & &
\end{array}
$$

  - Inserimos o 4 na posição correta

Ordenação – Inserção

- Ex: ordene em ordem crescente
  - Em azul está o sub-arranjo já ordenado
  - Analisando o terceiro elemento a ser inserido:

9	8	4	10	6
8	9	4	10	6
4	8	9	10	6
4	8	9	10	6

Ordenação – Inserção

- Ex: ordene em ordem crescente
  - Em azul está o sub-arranjo já ordenado
  - Analisando o terceiro elemento a ser inserido:

9	8	4	10	6
8	9	4	10	6
4	8	9	10	6
4	8	9	10	6
			↑	

  - Identificamos onde ele deve estar, na parte ordenada

Ordenação – Inserção

- Ex: ordene em ordem crescente
  - Em azul está o sub-arranjo já ordenado
  - Analisando o terceiro elemento a ser inserido:

9	8	4	10	6
8	9	4	10	6
4	8	9	10	6
4	8	9	10	6

  - Não há necessidade de deslocamento e inserção. Já está na posição correta

Ordenação – Inserção

- Ex: ordene em ordem crescente
  - Em azul está o sub-arranjo já ordenado
  - Analisando o quarto elemento a ser inserido:

9	8	4	10	6
8	9	4	10	6
4	8	9	10	6
4	8	9	10	6
4	8	9	10	6

Ordenação – Inserção

- Ex: ordene em ordem crescente
  - Em azul está o sub-arranjo já ordenado
  - Analisando o quarto elemento a ser inserido:

9	8	4	10	6
8	9	4	10	6
4	8	9	10	6
4	8	9	10	6
4	8	9	10	6
	↑			

  - Identificamos onde ele deve estar, na parte ordenada

Ordenação – Inserção

- Ex: ordene em ordem crescente
  - Em azul está o sub-arranjo já ordenado
  - Analisando o quarto elemento a ser inserido:

9	8	4	10	6
8	9	4	10	6
4	8	9	10	6
4	8	9	10	6
4	8	8	9	10
	↑	→		

  - Deslocamos a parte ordenada a partir dessa posição

Ordenação – Inserção

- Ex: ordene em ordem crescente
  - Em azul está o sub-arranjo já ordenado
  - Analisando o quarto elemento a ser inserido:

9	8	4	10	6
8	9	4	10	6
4	8	9	10	6
4	8	9	10	6
4	6	8	9	10
	↑			

  - Inserimos o 6 na posição correta

Ordenação – Inserção

- Note que, a cada passo:

Ordenação – Inserção

- Note que, a cada passo:
  - Encontrávamos a posição em que o valor deveria estar

Ordenação – Inserção

- Note que, a cada passo:
  - Encontrávamos a posição em que o valor deveria estar
  - Deslocávamos os elementos necessários

Ordenação – Inserção

- Note que, a cada passo:
  - Encontrávamos a posição em que o valor deveria estar
  - Deslocávamos os elementos necessários
  - Inseríamos o valor nessa posição

Ordenação – Inserção

- Note que, a cada passo:
  - Encontrávamos a posição em que o valor deveria estar
  - Deslocávamos os elementos necessários
  - Inseríamos o valor nessa posição
- Uma melhoria direta seria:

Ordenação – Inserção

- Note que, a cada passo:
  - Encontrávamos a posição em que o valor deveria estar
  - Deslocávamos os elementos necessários
  - Inseríamos o valor nessa posição
- Uma melhoria direta seria:
  - Para cada passo do vetor original, a partir da posição 1 até o final

Ordenação – Inserção

- Note que, a cada passo:
  - Encontrávamos a posição em que o valor deveria estar
  - Deslocávamos os elementos necessários
  - Inseríamos o valor nessa posição
- Uma melhoria direta seria:
  - Para cada passo do vetor original, a partir da posição 1 até o final
    - ★ Deslocar o arranjo ordenado para direita (uma posição por vez) até encontrar local adequado para o valor a ser inserido

Ordenação – Inserção

- Note que, a cada passo:
  - Encontrávamos a posição em que o valor deveria estar
  - Deslocávamos os elementos necessários
  - Inseríamos o valor nessa posição
- Uma melhoria direta seria:
  - Para cada passo do vetor original, a partir da posição 1 até o final
    - ★ Deslocar o arranjo ordenado para direita (uma posição por vez) até encontrar local adequado para o valor a ser inserido
    - ★ Inserir esse valor na posição correta

Ordenação – Inserção

- Note que, a cada passo:
  - Encontrávamos a posição em que o valor deveria estar
  - Deslocávamos os elementos necessários
  - Inseríamos o valor nessa posição
- Uma melhoria direta seria:
  - Para cada passo do vetor original, a partir da posição 1 até o final
    - ★ Deslocar o arranjo ordenado para direita (uma posição por vez) até encontrar local adequado para o valor a ser inserido
    - ★ Inserir esse valor na posição correta
- Evita assim a necessidade de se buscar a posição de antemão

Ordenação – Inserção

- Então...

Ordenação – Inserção

- Então...

```
static void insercao(int[] v) {
    for (int i=1; i<v.length; i++) {
        int aux = v[i];
        int j = i;
        while ((j > 0) && (aux < v[j-1])) {
            v[j] = v[j-1];
            j--;
        }
        v[j] = aux;
    }
}

public static void main(String[] args) {
    int[] v = {9,8,4,10,6};

    insercao(v);

    for (int el : v)
        System.out.print(el+" ");
    System.out.println();
}
```

Ordenação – Inserção

- Então...
- Corremos todos os possíveis candidatos a inserção

```
static void insercao(int[] v) {
    for (int i=1; i<v.length; i++) {
        int aux = v[i];
        int j = i;
        while ((j > 0) && (aux < v[j-1])) {
            v[j] = v[j-1];
            j--;
        }
        v[j] = aux;
    }
}

public static void main(String[] args) {
    int[] v = {9,8,4,10,6};

    insercao(v);

    for (int el : v)
        System.out.print(el+" ");
    System.out.println();
}
```

Ordenação – Inserção

- Então...
- Corremos todos os possíveis candidatos a inserção
  - Enquanto o candidato estiver fora de lugar, deslocamos os anteriores a ele, até achar o lugar certo

```
static void insercao(int[] v) {
    for (int i=1; i<v.length; i++) {
        int aux = v[i];
        int j = i;
        while ((j > 0) && (aux < v[j-1])) {
            v[j] = v[j-1];
            j--;
        }
        v[j] = aux;
    }
}

public static void main(String[] args) {
    int[] v = {9,8,4,10,6};

    insercao(v);

    for (int el : v)
        System.out.print(el+" ");
    System.out.println();
}
```

Ordenação – Inserção

- Então...
- Corremos todos os possíveis candidatos a inserção
  - Enquanto o candidato estiver fora de lugar, deslocamos os anteriores a ele, até achar o lugar certo
  - E colocamos ele lá

```
static void insercao(int[] v) {
    for (int i=1; i<v.length; i++) {
        int aux = v[i];
        int j = i;
        while ((j > 0) && (aux < v[j-1])) {
            v[j] = v[j-1];
            j--;
        }
        v[j] = aux;
    }
}

public static void main(String[] args) {
    int[] v = {9,8,4,10,6};

    insercao(v);

    for (int el : v)
        System.out.print(el+" ");
    System.out.println();
}
```

Ordenação – Inserção

- Então...
- Corremos todos os possíveis candidatos a inserção
  - Enquanto o candidato estiver fora de lugar, deslocamos os anteriores a ele, até achar o lugar certo
  - E colocamos ele lá

Saída

```
$ java Projeto
4 6 8 9 10
```

```
static void insercao(int[] v) {
    for (int i=1; i<v.length; i++) {
        int aux = v[i];
        int j = i;
        while ((j > 0) && (aux < v[j-1])) {
            v[j] = v[j-1];
            j--;
        }
        v[j] = aux;
    }
}

public static void main(String[] args) {
    int[] v = {9,8,4,10,6};

    insercao(v);

    for (int el : v)
        System.out.print(el+" ");
    System.out.println();
}
```

Ordenação – Inserção

- E como ficaria a versão com objetos?

Ordenação – Inserção

- E como ficaria a versão com objetos?

```
static void insercao(Residencia[] v) {
    for (int i=1; i<v.length; i++) {
        Residencia aux = v[i];
        int j = i;
        while ((j > 0) &&
               (aux.comparaRes(v[j-1]) < 0)) {
            v[j] = v[j-1];
            j--;
        }
        v[j] = aux;
    }
}
```

```
public static void main(String[] args) {
    Projeto pr = new Projeto(5);

    for (int i=0; i<5; i++) {
        AreaCasa c = new AreaCasa(
            Math.random()*100,Math.random()*30);
        AreaPiscina p = new AreaPiscina(
                                Math.random()*10);
        Residencia r = new Residencia(c,p);
        pr.adicionaRes(r);
    }

    for (Residencia r : pr.condominio)
        System.out.println(r.area());
    System.out.println();

    insercao(pr.condominio);

    for (Residencia r : pr.condominio)
        System.out.println(r.area());
}
```

Ordenação – Inserção

- E como ficaria a versão com objetos?

```
static void insercao(Residencia[] v) {
    for (int i=1; i<v.length; i++) {
        Residencia aux = v[i];
        int j = i;
        while ((j > 0) &&
               (aux.comparaRes(v[j-1]) < 0)) {
            v[j] = v[j-1];
            j--;
        }
        v[j] = aux;
    }
}
```

Saída

```
$ java Projeto
6583.183940438665
1130.4506182200782
7379.352930903931
3289.6719206296757
5739.294165717424

1130.4506182200782
3289.6719206296757
5739.294165717424
6583.183940438665
7379.352930903931
```

```
public static void main(String[] args) {
    Projeto pr = new Projeto(5);

    for (int i=0; i<5; i++) {
        AreaCasa c = new AreaCasa(
            Math.random()*100,Math.random()*30);
        AreaPiscina p = new AreaPiscina(
                                Math.random()*10);
        Residencia r = new Residencia(c,p);
        pr.adicionaRes(r);
    }

    for (Residencia r : pr.condominio)
        System.out.println(r.area());
    System.out.println();

    insercao(pr.condominio);

    for (Residencia r : pr.condominio)
        System.out.println(r.area());
}
```

Ordenação – Comparação

- Seleção e Inserção:

Ordenação – Comparação

- Seleção e Inserção:
  - Garantem que, no passo i, o subvetor de 0 a i está ordenado

Ordenação – Comparação

- Seleção e Inserção:
  - Garantem que, no passo i, o subvetor de 0 a i está ordenado
  - Diferem em como fazem isso:

Ordenação – Comparação

- Seleção e Inserção:
  - Garantem que, no passo i, o subvetor de 0 a i está ordenado
  - Diferem em como fazem isso:
    - ★ Seleção troca 2 elementos

Ordenação – Comparação

- Seleção e Inserção:
  - Garantem que, no passo i, o subvetor de 0 a i está ordenado
  - Diferem em como fazem isso:
    - ★ Seleção troca 2 elementos
    - ★ Inserção desloca à direita todo o sub-vetor

Ordenação – Comparação

- Seleção e Inserção:
  - Garantem que, no passo i, o subvetor de 0 a i está ordenado
  - Diferem em como fazem isso:
    - Seleção troca 2 elementos
    - Inserção desloca à direita todo o sub-vetor
- Bolha:

Ordenação – Comparação

- Seleção e Inserção:
  - Garantem que, no passo i, o subvetor de 0 a i está ordenado
  - Diferem em como fazem isso:
    - ★ Seleção troca 2 elementos
    - ★ Inserção desloca à direita todo o sub-vetor
- Bolha:
  - Garante que, no passo i, o subvetor de *tam-1-i* a *tam-1* está ordenado

Ordenação – Comparação

- Seleção e Inserção:
  - Garantem que, no passo i, o subvetor de 0 a i está ordenado
  - Diferem em como fazem isso:
    - Seleção troca 2 elementos
    - Inserção desloca à direita todo o sub-vetor
- Bolha:
  - Garante que, no passo i, o subvetor de *tam-1-i* a *tam-1* está ordenado
- Todos ordenam in loco (no próprio vetor, sem precisar de vetor auxiliar)

Curiosidades

- Bolha:
  - http://www.youtube.com/watch?v=IyZQPjUT5B4
  - http://www.youtube.com/watch?feature=fvwp&NR=1&v=_h3aMVBe8k8
- Seleção:
  - http://www.youtube.com/watch?v=Ns4TPTC8whw (versão levemente diferente do algoritmo)
- Inserção:
  - http://www.youtube.com/watch?v=ROaIU379l3U