# Insertion Sort:

## Class variables:

Comparisons: private (-). needs to be static to be used within static methods. type: int ⟶ `private static int Comparisons = 0;`

Swaps: private to this class (-), needs to be static to be used within static methods, type: int ⟶ `private static int Swaps = 0;`

## Methods:

**insertionSort ( ):** Public (+); needs to be static to be called without creating instance; no return type - void; Creating an array of numbers

`public static void insertionSort(int[] numbers) { }` // sorts numbers in array and prints after each iteration.

- `i;` within public static method (no need to specify public static), return type is int ⟶ `int i;` // for loop variable
- `j;` within public static method (no need to specify public static); return type is int ⟶ `int j;` // while loop variable
- `temp;` within public static method (no need to specify public static); return type is int ⟶ `int temp;` // temporary variable for swap

: for loop: Since this method check each number vs. the number before it, initialize i to index 1 rather than 0 because the element in the first position is assumed to be sorted; check that i is less than the length of the array; increment i if it is

`for (i = 1; i < numbers.length; ++i) { }`

- the nested while loop will compare and swap if necessary. it will start at the same index as the for loop, so initialize j to equal i

  `j = i;`
- while loop will run while j is greater than 0 AND while the value of the element in j position is less than the value of the element in the j-1 position

  `while (j > 0 && numbers[j] < numbers[j-1]) { }`
  - while conditions checked: need to increment Comparisons value ⟶ `Comparisons ++;`
  - while conditions satisfied: need to swap numbers:
    1) element in position j to temp variable ⟶ `temp = numbers[j];`
    2) element in position j-1 to position j ⟶ `numbers[j] = numbers[j-1];`
    3) element in temp to position j-1 ⟶ `numbers[j-1] = temp;`
  - Swap has been made: need to increment Swaps ⟶ `swaps ++;`
    4) Check if element that is now in position j-1 for next loop: decrement j ⟶ `--j;`
- to account for final comparison made, since the other comparison increment only happened when the comparison met both conditions for the while loop, the for loop will then increase i, increasing j, until the conditions are no longer met for the while loop. so j will be set to the value of i which is the length of the array. Check to see if j is greater than 0. If it is, increment comparisons one last time. ⟶ `if (j > 0) comparisons ++;`
- Print the array of numbers after each iteration of the for loop ⟶ `numPrint(numbers);`

**numRead ( ):** Public, Static to be called without creating an instance, return type is an array of integers

`public static int[] numRead ( ) { }`

- i: for loop variable; within public static method (no need to specify Public Static); type int ⟶ `int i;`
- Create Scanner object to take standard user input ⟶ `Scanner scnr = new Scanner (System.in);`
- Size: Create a variable to take integer input from user ⟶ `int size = scnr.nextInt();`
- numbers array: Create an array of integers of size specified by user ⟶ `int[] numbers = new int[size];` ← Enter prompt for user
- This needs to repeat the process of taking input values for array from user for the same amount of times as specified for size of array: for loop (i initialized to index zero; check i against size of array, if i is less than size of array; increment i)

```
for (i = 0; i < size; ++i) {
    numbers[i] = scnr.nextInt();
}
```

- This method returns the array of numbers ⟶ `return numbers;`

numPrint(): This method needs to print the array at each step of the process. public static void numPrint(int[] nums);

- To print the array with a space between each number, use for loop to print each number of a collection of number (for each)

```
for (int[] num : nums) {
```

- print each number followed by a space

```
    System.out.print(num + " ");
}
```

- newline after whole array is printed

```
System.out.println();
```

main(): In this method: create array taking input from numRead() method; insert newline to space from input; print initial array; insert another newline; call insertionSort() where the method has been built to sort the numbers in the array and print the array every time it is changed; print another newline, output number of comparisons and swaps:

```
Public static void main(String[] args) {
```

1) Create array and intialize to numRead return ⟶ int[] numbers = numRead();
2) insert newline to space from input ⟶ System.out.println();
3) Print initial array as formatted in numPrint ⟶ numPrint(numbers);
4) insert newline to separate initial array from sorted outputs ⟶ System.out.println();
5) Call insertionSort() method on numbers array ⟶ insertionSort(numbers);
6) insert newline to separate sorted outputs from comparison and sort output ⟶ System.out.println();
7) print comparisons ⟶ System.out.println("Comparisons: " + comparisons);
8) print swaps ⟶ System.out.println("Swaps: " + swaps);

}