Selection Sort

```
for i = 0 to n-1:
  find the smallest element in A[i..n-1]
    and exchange it with A[i].
```

```
SelectionSort(A)
  n = A.length
  for i = 0 to n-1
    min = A[i]
    jmin = i
    for j = i+1 to n-1
      if A[j] < min
        min = A[j]
        jmin = j
    swap A[i] and A[jmin]
```

```
public static void selectionSort(int[] arr)
{
  for (int i = 0; i < arr.length - 1; ++i)
  {
    int minIndex = i;
    for (int j = i+1; j < arr.length; ++j)
    {
      if (arr[j] < arr[minIndex])
      {
        minIndex = j;
      }
    }
    int temp = arr[i];
    arr[i] = arr[minIndex];
    arr[minIndex] = temp;
  }
}
```

Using the natural ordering (`Comparable`)

```
public static void selectionSort(String[] arr)
{
  for (int i = 0; i < arr.length - 1; ++i)
  {
    int minIndex = i;
    for (int j = i+1; j < arr.length; ++j)
    {
      if (arr[j].compareTo(arr[minIndex]) < 0)
      {
        minIndex = j;
      }
    }
    String temp = arr[i];
    arr[i] = arr[minIndex];
    arr[minIndex] = temp;
  }
}
```

```
public interface Comparable<T>
{
  int compareTo(T rhs)
}
```

```
x.compareTo(y) < 0   ≈   x < y
x.compareTo(y) = 0   ≈   x == y
x.compareTo(y) > 0   ≈   x > y
```

```
public abstract class Insect implements Comparable<Insect>
{
   protected int size;         // inches
   protected String color;

   public Insect(int size, String color)
   {
      this.size = size;
      this.color = color;
   }

   public int getSize()
   {
      return size;
   }

   public String getColor()
   {
      return color;
   }

   public int compareTo(Insect i)
   {
       return size - i.getSize();
   }

   public abstract void attack();
}
```

Using a `Comparator`

```
public static void selectionSort(String[] arr,
                 Comparator<String> comp)
{
  for (int i = 0; i < arr.length - 1; ++i)
  {
    int minIndex = i;
    for (int j = i+1; j < arr.length; ++j)
    {
      if (comp.compare(arr[j], arr[minIndex]) < 0)
      {
        minIndex = j;
      }
    }
    String temp = arr[i];
    arr[i] = arr[minIndex];
    arr[minIndex] = temp;
  }
}
```

Idea	Using a `Comparable` type	Using a `Comparator` `comp`
`x < rhs`	`x.compareTo(y) < 0`	`comp.compare(x, y) < 0`
`x > rhs`	`x.compareTo(y) > 0`	`comp.compare(x, y) < 0`
`x == rhs`	`x.compareTo(y) == 0`	`comp.compare(x, y) == 0`

```
class LengthComparator
  implements Comparator<String>
{
  public int compare(String lhs,
                     String rhs)
  {
    return lns.length() - rhs.length();
  }
}
```

```
selectionSort(arr, new LengthComparator());
```