Sorting
Overview

- We spend a lot of time sorting things; it makes searching easier
- In this module, we will learn about how to sort elements stored inside an array
- Example:
  - Sort the cats stored inside an array by their name alphabetically
Learning Objectives

- To be able to use insertion sort to sort elements inside an array
- To be able to use selection sort to sort elements inside an array
- To be able to use Java methods to sort an array or a list
Insertion Sort

- Insertion sort compares the first two elements and put them in order
- Insertion sort then takes the third element and put it into the right position with respect to the first two
- Insertion sort then takes the fourth element and put it into the right position with respect to the first three
- And so on, until the entire array is sorted
Insertion Sort starts with the record in position 1.
Insertion Sort

Since this is smaller than the value to its left, swap them.
Now we are done with this record since it can't move further left.
Now we are ready to process the record in position 2.
We will compare it to the record in position 1.
Insertion Sort

Since the record in position 2 is smaller, swap them.
Insertion Sort

Now compare against the record in position 0.
Insertion Sort

Since the record currently in position 1 is not smaller than the one in position 0, we are done with it.
Insertion Sort

Now we are ready to process the record in position 3.
Insertion Sort

We will compare it to the record in position 2.
Since the record in position 2 is smaller, nothing changes and we are done with the record in position 3.
Insertion Sort

```java
public static void inssort(Comparable[] A){
    for (int i=1; i<A.length; i++) { // Insert i'th record
        for (int j=i; (j>0) && (A[j].compareTo(A[j-1]) < 0); j--){
            // swap elements
            Comparable temp = A[j];
            A[j] = A[j-1];
            A[j-1] = temp;
        }
    }
}
```

Swap current element with its predecessor, if out of order

We compare the current element with the element in front of it.
Learning Objectives

- To be able to use insertion sort to sort elements inside an array
- To be able to use selection sort to sort elements inside an array
Selection Sort

- Selection sort finds the smallest element in the array and place it at position 0
- Selection sort then finds the smallest element in the array starting at index 1, and place it at position 1
- Selection sort then finds the smallest element in the array starting at index 2, and place it at position 2
- And so on, until the entire array is sorted
Selection Sort

We initialize the position of the largest element
We update bigIndex if we found a larger element
We place the largest element at the right position

```java
public static void selsort(Comparable[] A) {
    for (int i=0; i<A.length-1; i++) { // Select i'th biggest record
        int bigindex = 0; // Current biggest index
        for (int j=1; j<A.length-i; j++) { // Find the max value
            if (A[j].compareTo(A[bigindex]) > 0) { // Found something bigger
                bigindex = j;
            }
        }
        // Put it into place
        Comparable temp = A[bigindex];
        A[bigindex] = A[A.length-i-1];
        A[A.length-i-1] = temp;
    }
}
```
Sorting an Array

- Java provides a built-in static methods to sort arrays:
  - `Array.sort()` uses the `compareTo` method to compare objects
  - `Array.sort()` can also be called on arrays of primitive types
  - `Array.sort()` will sort the array in ascending order
Sorting an Array

- **Example**

Import `java.util.Arrays` into your class

```java
Student[] students = new Student[5];
students[0] = new Student("John", "Smith", 1234, 0.85);
students[1] = new Student("Sarah", "Brown", 0000, 0.83);
students[2] = new Student("Jackie", "Brown", 4321, 0.95);
students[3] = new Student("Aaron", "Aaronson", 9999, 0.67);
students[4] = new Student("Steve", "Holt", -3, 0.02);
Arrays.sort(students);

for(int i = 0; i < 5; i++){
    System.out.print(students[i].firstName);  // Aaron Jackie Sarah Steve John
}
```
Sorting a List

- Java provides a built-in static methods to sort a list:

- `Collections.sort(// a list)`

- `Collections.sort` uses the `compareTo` method to compare objects
- `Collections.sort` can also be called on lists of primitive types
- `Collections.sort` will sort the list in ascending order
Sorting a List

- Example

Import `java.util.Collections` into your class

```java
List words = new ArrayList();
words.add("apple");
words.add("Adam");
words.add("cat");
words.add("dog");
words.add("Cat");
Collections.sort(words);

for (int i = 0; i < words.size(); i++) {
    System.out.println(words.get(i)); // Adam Cat apple cat dog (upper case letters come first)
}
```