More on Searching and Sorting

- Will talk about using built-in sorting
  - How to use
  - How to make the sort do it your way

- Next time we’ll talk about programming a simple sort routine.

Classwork today

- Library of books
  - Title
  - Author
  - Date pub.

- Search for books
- Sort the books found

Want to Sort Query Results

- Several ways we could sort
  - By Title first
  - By Author first
  - By Counts (number of copies) first

- How can we tell the sort program what we want.
  - Can it read your mind?
  - We’ll use a class that specifies the ordering criteria
**Interface Types**

- Not a class – but similar syntax
- Makes code more general and more reusable
- Express common operations
- Also allows “inheritance”
- Maps and Sets are interfaces
  
  TreesMaps and TreeSets implement them

**Example for Today**

- **public interface Comparator<T>**
  
  - Look at API for Collections.sort
  - Look at the API for Comparator
  - Requires a compare method
    - returns negative integer, 0, positive integer for <, ==, >

- Implement comparison by Titles

  **Comparator Example:**

  ```java
  public class TitleComparator implements Comparator<Book> {
      public int compare(Book left, Book right) {
          return left.getTitle().compareTo(right.getTitle());
      }
  }
  ```

**More Interfaces**

- How are Strings sorted?
  - A bit of a different approach:
  - Strings implement comparable (another interface)
  - Requires compareTo
    - returns negative integer, 0, positive integer for <, ==, >

**You’ll implement**

- Comparator for Author first, etc...
- Comparator for Count of number of copies first, etc.
- See handout