



Polymorphism Java Interface

Part 2: Comparable and Comparator

Instructor: Dr. Noor Bajunaid

Department of Computer Science

College of Computer and Information Sciences

Arrays.sort()

Arrays.sort() is a built-in static method in the java.util.Arrays class.

- It is used to arrange array elements in a specific order.
- It modifies the original array **in-place**.

```
int[] integers = {42, 13, 7, 91, 24};  
  
Arrays.sort(integers);  
  
System.out.println("Sorted int[]: " + Arrays.toString(integers));  
  
// Output: [7, 13, 24, 42, 91]
```

Arrays.sort()

Arrays.sort() is a built-in static method in the java.util.Arrays class.

- It is used to arrange array elements in a specific order.
- It modifies the original array **in-place**.

```
String[] colors = {"Red", "Blue", "Green", "Yellow", "Black"};

Arrays.sort(colors);

System.out.println("Sorted String[]: " + Arrays.toString(colors));

// Output: [Black, Blue, Green, Red, Yellow]
```

Arrays.sort()

Arrays.sort() is a built-in static method in the java.util.Arrays class.

- It is used to arrange array elements in a specific order.
- It modifies the original array **in-place**.

```
double[] prices = {19.99, 5.50, 100.2, 5.25};  
  
Arrays.sort(prices);  
  
System.out.println("Sorted double[]: " + Arrays.toString(prices));  
  
// Output: [5.25, 5.5, 19.99, 100.2]
```

Arrays.sort()

```
public static void main(String[] args) {  
    // --- 1. Sorting Integers (Primitives) ---  
    int[] integers = {42, 13, 7, 91, 24};  
    Arrays.sort(integers);  
    System.out.println("Sorted int[]: " + Arrays.toString(integers));  
    // Output: [7, 13, 24, 42, 91]  
  
    // --- 2. Sorting Strings (Alphabetical Order) ---  
    String[] colors = {"Red", "Blue", "Green", "Yellow", "Black"};  
    Arrays.sort(colors);  
    System.out.println("Sorted String[]: " + Arrays.toString(colors));  
    // Output: [Black, Blue, Green, Red, Yellow]  
  
    // --- 3. Sorting Doubles ---  
    double[] prices = {19.99, 5.50, 100.2, 5.25};  
    Arrays.sort(prices);  
    System.out.println("Sorted double[]: " + Arrays.toString(prices));  
    // Output: [5.25, 5.5, 19.99, 100.2]  
}
```

Can Java sort books?

```
public class Book {
```

```
    private String title;  
    private String author;  
    private String genre;  
    private int year;
```

```
    public Book(String title, String author, String genre, int year) {  
        this.title = title;  
        this.author = author;  
        this.genre = genre;  
        this.year = year;  
    }
```



Can Java sort books?

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

```
    myLibrary[0] = new Book("Clean Code", "Robert Martin", "Programming", 2008);  
    myLibrary[1] = new Book("Effective Java", "Joshua Bloch", "Programming", 2018);  
    myLibrary[2] = new Book("The Hobbit", "J.R.R. Tolkien", "Fantasy", 1937);  
    myLibrary[3] = new Book("1984", "George Orwell", "Dystopian", 1949);  
    myLibrary[4] = new Book("To Kill a Mockingbird", "Harper Lee", "Classic", 1960);
```

```
    Arrays.sort(myLibrary);
```

```
}
```



Can Java sort books?

```
public static void main(String[] args) {  
    Book[] myLibrary = new Book[5];  
  
    myLibrary[0] = new Book("Clean Code", "Robert Martin", "Programming", 2008);  
    myLibrary[1] = new Book("Effective Java", "Joshua Bloch", "Programming", 2018);  
    myLibrary[2] = new Book("The Hobbit", "J.R.R. Tolkien", "Fantasy", 1937);  
    myLibrary[3] = new Book("1984", "George Orwell", "Dystopian", 1949);  
    myLibrary[4] = new Book("To Kill a Mockingbird", "Harper Lee", "Classic", 1960);  
  
    Arrays.sort(myLibrary);  
}
```



```
public static void main(String[] args) {  
    Book[] myLibrary = new Book[5];  
  
    myLibrary[0] = new Book("Clean Code", "Robert Martin", "Programming", 2008);  
    myLibrary[1] = new Book("Effective Java", "Joshua Bloch", "Programming", 2018);  
    myLibrary[2] = new Book("The Hobbit", "J.R.R. Tolkien", "Fantasy", 1937);  
    myLibrary[3] = new Book("1984", "George Orwell", "Dystopian", 1949);  
    myLibrary[4] = new Book("To Kill a Mockingbird", "Harper Lee", "Classic", 1960);  
  
    Arrays.sort(myLibrary);  
  
}
```

run:

```
] Exception in thread "main" java.lang.ClassCastException: class ii.Book cannot be cast to class java.lang.Comparable (ii.Book is in unnamed  
    at java.base/java.util.ComparableTimSort.countRunAndMakeAscending(ComparableTimSort.java:320)  
    at java.base/java.util.ComparableTimSort.sort(ComparableTimSort.java:188)  
    at java.base/java.util.Arrays.sort(Arrays.java:1038)  
    at ii.Book.main(Book.java:32)  
/Users/noorbajunaid/Library/Caches/NetBeans/28/executor-snippets/run.xml:111: The following error occurred while executing this line:  
/Users/noorbajunaid/Library/Caches/NetBeans/28/executor-snippets/run.xml:68: Java returned: 1  
BUILD FAILED (total time: 1 second)
```

```

public static void main(String[] args) {
    Book[] myLibrary = new Book[5];

    myLibrary[0] = new Book("Clean Code", "Robert Martin", "Programming", 2008);
    myLibrary[1] = new Book("Effective Java", "Joshua Bloch", "Programming", 2018);
    myLibrary[2] = new Book("The Hobbit", "J.R.R. Tolkien", "Fantasy", 1937);
    myLibrary[3] = new Book("1984", "George Orwell", "Dystopian", 1949);
    myLibrary[4] = new Book("To Kill a Mockingbird", "Harper Lee", "Classic", 1960);

    Arrays.sort(myLibrary);

}

```

```

run:
] Exception in thread "main" java.lang.ClassCastException: class ii.Book cannot be cast to class java.lang.Comparable
    at java.base/java.util.ComparableTimSort.countRunAndMakeAscending(ComparableTimSort.java:320)
    at java.base/java.util.ComparableTimSort.sort(ComparableTimSort.java:188)
    at java.base/java.util.Arrays.sort(Arrays.java:1038)
    at ii.Book.main(Book.java:32)
/Users/noorbajunaid/Library/Caches/NetBeans/28/executor-snippets/run.xml:111: The following error occurred while executing this line:
/Users/noorbajunaid/Library/Caches/NetBeans/28/executor-snippets/run.xml:68: Java returned: 1
BUILD FAILED (total time: 1 second)

```

Comparable

How do you **compare** two books?

Which book comes first?

```
public class Book {  
    private String title;  
    private String author;  
    private String genre;  
    private int year;  
}
```



Comparable Interface

```
interface Comparable
{
    public int compareTo(Object other);
}
```

Java provides the interface `Comparable`, which has one abstract method:

```
public int compareTo(Object other);
```

Make books comparable

```
public class Book implements Comparable{

    private String title;
    private String author;
    private String genre;
    private int year;

    public Book(String title, String author, String genre, int year) {
        this.title = title;
        this.author = author;
        this.genre = genre;
        this.year = year;
    }

    @Override
    public int compareTo(Object o) {

        Book other = (Book)o;
        return title.compareTo(other.title);
    }
}
```

Compare books by title

Make books comparable

```
public static void main(String[] args) {  
    Book[] myLibrary = new Book[5];  
  
    myLibrary[0] = new Book("Clean Code", "Robert Martin", "Programming", 2008);  
    myLibrary[1] = new Book("Effective Java", "Joshua Bloch", "Programming", 2018);  
    myLibrary[2] = new Book("The Hobbit", "J.R.R. Tolkien", "Fantasy", 1937);  
    myLibrary[3] = new Book("1984", "George Orwell", "Dystopian", 1949);  
    myLibrary[4] = new Book("To Kill a Mockingbird", "Harper Lee", "Classic", 1960);  
  
    Arrays.sort(myLibrary);  
    System.out.println(Arrays.toString(myLibrary));  
}
```

```
run:  
1984, George Orwell, Dystopian, 1949  
Clean Code, Robert Martin, Programming, 2008  
Effective Java, Joshua Bloch, Programming, 2018  
The Hobbit, J.R.R. Tolkien, Fantasy, 1937  
To Kill a Mockingbird, Harper Lee, Classic, 1960  
BUILD SUCCESSFUL (total time: 0 seconds)
```

Make books comparable

```
public static void main(String[] args) {  
    Book[] myLibrary = new Book[5];  
  
    myLibrary[0] = new Book("Clean Code", "Robert Martin", "Programming", 2008);  
    myLibrary[1] = new Book("Effective Java", "Joshua Bloch", "Programming", 2018);  
    myLibrary[2] = new Book("The Hobbit", "J.R.R. Tolkien", "Fantasy", 1937);  
    myLibrary[3] = new Book("1984", "George Orwell", "Dystopian", 1949);  
    myLibrary[4] = new Book("To Kill a Mockingbird", "Harper Lee", "Classic", 1960);  
  
    Arrays.sort(myLibrary);  
    System.out.println(Arrays.toString(myLibrary));  
}
```

```
run:  
1984, George Orwell, Dystopian, 1949  
Clean Code, Robert Martin, Programming, 2008  
Effective Java, Joshua Bloch, Programming, 2018  
The Hobbit, J.R.R. Tolkien, Fantasy, 1937  
To Kill a Mockingbird, Harper Lee, Classic, 1960  
BUILD SUCCESSFUL (total time: 0 seconds)
```

Compare books by author

```
public class Book implements Comparable{
```

```
    private String title;  
    private String author;  
    private String genre;  
    private int year;
```

```
@Override
```

```
public int compareTo(Object o) {
```

```
    Book other = (Book)o;
```

```
    return author.compareTo(other.author);
```

```
}
```

Compare books by
author

```
run:
```

```
1984, George Orwell, Dystopian, 1949
```

```
To Kill a Mockingbird, Harper Lee, Classic, 1960
```

```
The Hobbit, J.R.R. Tolkien, Fantasy, 1937
```

```
Effective Java, Joshua Bloch, Programming, 2018
```

```
Clean Code, Robert Martin, Programming, 2008
```

```
BUILD SUCCESSFUL (total time: 1 second)
```

Compare books by year

```
public class Book implements Comparable{  
  
    private String title;  
    private String author;  
    private String genre;  
    private int year;  
  
    @Override  
    public int compareTo(Object o) {  
  
        Book other = (Book)o;  
        return year - other.year;  
    }  
}
```

Compare books by
year

```
run:  
The Hobbit, J.R.R. Tolkien, Fantasy, 1937  
1984, George Orwell, Dystopian, 1949  
To Kill a Mockingbird, Harper Lee, Classic, 1960  
Clean Code, Robert Martin, Programming, 2008  
Effective Java, Joshua Bloch, Programming, 2018  
BUILD SUCCESSFUL (total time: 1 second)
```

Compare books to...?

```
public class Book implements Comparable{  
  
    private String title;  
    private String author;  
    private String genre;  
    private int year;  
  
    @Override  
    public int compareTo(Object o) {  
  
        Book other = (Book)o;  
        return year - other.year;  
    }  
}
```

Note that compareTo
accepts any Object.

This is NOT SAFE!

Compare books to...?

```
Book CleanCode = new Book("Clean Code", "Robert Martin", "Programming", 2008);  
Book EffectiveJava = new Book("Effective Java", "Joshua Bloch", "Programming", 2018);  
  
System.out.println(CleanCode.compareTo(EffectiveJava)); ✓  
  
System.out.println(CleanCode.compareTo("EffectiveJava"));
```

Note that compareTo
accepts any Object.

This is NOT SAFE!

Compare books to...?

```
Book CleanCode = new Book("Clean Code", "Robert Martin", "Programming", 2008);  
Book EffectiveJava = new Book("Effective Java", "Joshua Bloch", "Programming", 2018);
```

```
System.out.println(CleanCode.compareTo(EffectiveJava));
```

 ✓

```
System.out.println(CleanCode.compareTo("EffectiveJava"));
```

```
run:  
-10  
] Exception in thread "main" java.lang.ClassCastException: class java.lang.String cannot be cast to class ii.Book  
|   at ii.Book.compareTo(Book.java:28)  
|   at ii.Book.main(Book.java:52)  
|  
| /Users/noorbajunaid/Library/Caches/NetBeans/28/executor-snippets/run.xml:111: The following error occurred while  
| /Users/noorbajunaid/Library/Caches/NetBeans/28/executor-snippets/run.xml:68: Java returned: 1  
BUILD FAILED (total time: 1 second)
```

Books should be compared to Books!

Not to String

Not to int

Not to Student

Not to Object!

Make Sure to Compare Books to Books

```
public class Book implements Comparable{  
  
    private String title;  
    private String author;  
    private String genre;  
    private int year;  
  
    @Override  
    public int compareTo(Object o) {  
  
        Book other = (Book)o;  
        return year - other.year;  
    }  
}
```

Make Sure to Compare Books to Books

```
public class Book implements Comparable<Book>{  
  
    private String title;  
    private String author;  
    private String genre;  
    private int year;  
  
    @Override  
    public int compareTo(Book other) {  
        return year - other.year;  
    }  
}
```

↑
Generic

Make Sure to Compare Books to Books

```
@Override  
public int compareTo(Book other) {  
    return year - other.year;  
}
```

The compiler will warn you now if you compare a Book to anything else

```
System.out.println(CleanCode.compareTo(EffectiveJava));  
incompatible types: java.lang.String cannot be converted to ii.Book
```

(Alt-Enter shows hints)

```
System.out.println(CleanCode.compareTo("EffectiveJava"));
```

Interface Comparable

The `java.lang.Comparable` interface in Java is used to define the natural ordering for objects of a class.

By implementing this interface and its `compareTo` method, a class indicates that its instances can be compared and sorted automatically using built-in Java sorting methods like `Arrays.sort()`

Any class that needs to be stored in an array and sorted, must implement `Comparable`.