Iterators and Comparators

SoftUni Team

Technical Trainers

Software University

Software University

https://softuni.bg

Have a Question?

Software University

sli.do

#java-advanced

Table of Contents

`args...`

Variable Arguments

Variable Arguments (Varargs)

- Allows the method to accept **zero** or **multiple** arguments

```
static void display(String... values) {
  System.out.println("display method invoked");
}
static void main() {
  display();
  display("first");
  display("multiple", "Strings");
}
```

Ellipsis syntax

Variable Arguments Rules

- There can be **only one** variable argument **in the method**
- The variable argument **must** be the **last argument**

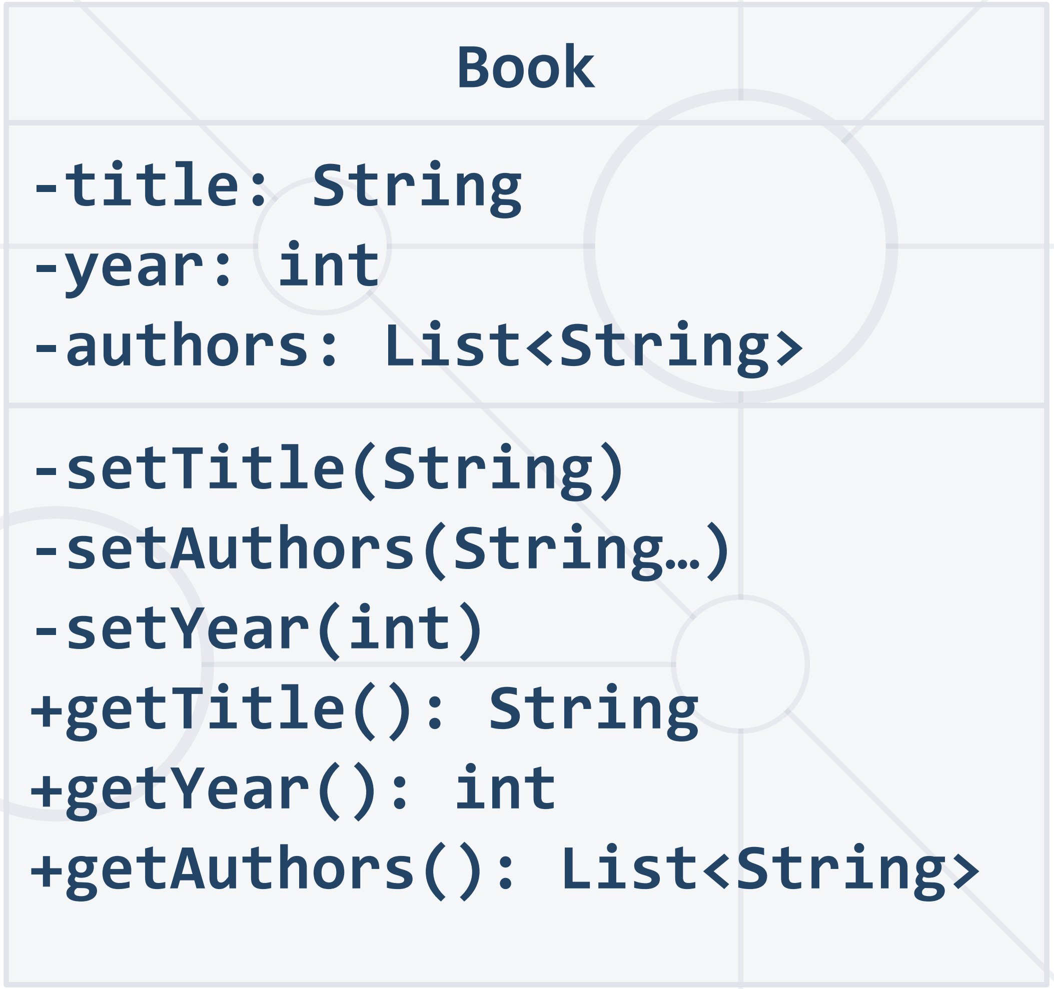
```
static void display(int num, String... values) {
  System.out.println("display method invoked");
}
```

```
void method(String... a, int... b){} //Compile time error

void method(int... a, String b){}    //Compile time error
```

Problem: Book

- Create a class Book, which has:
  - Title
  - Year
  - Authors
- Use **only one constructor** for Book
- There can be **no authors**, **one author** or **many authors**



```
Book
-title: String
-year: int
-authors: List<String>
-setTitle(String)
-setAuthors(String…)
-setYear(int)
+getTitle(): String
+getYear(): int
+getAuthors(): List<String>
```

Check your solution here: https://judge.softuni.bg/Contests/1542/Iterators-and-Comparators-Lab

Solution: Book (1)

```
//TODO: Add fields
public Book(String title, int year, String... authors) {
  this.setTitle(title);
  this.setYear(year);
  this.setAuthors(authors);
}
```

Check your solution here: https://judge.softuni.bg/Contests/1542/Iterators-and-Comparators-Lab

Solution: Book (2)

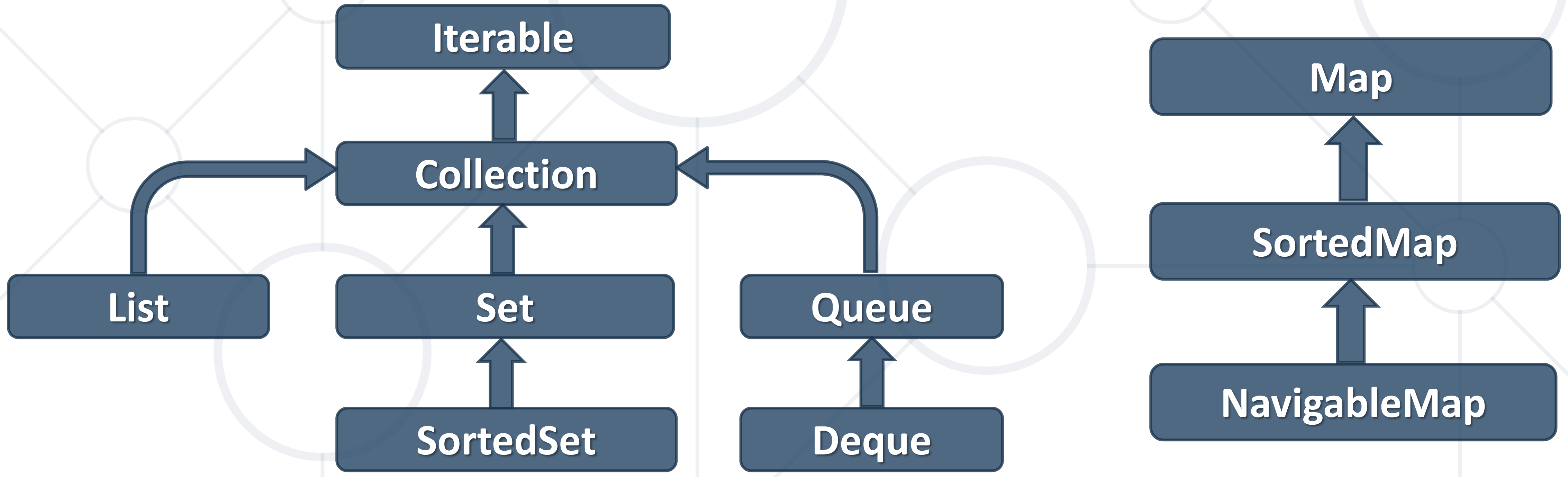
```
//TODO: Add all other getters and setters
private void setAuthors(String... authors) {
  if (authors.length == 0) {
    this.authors = new ArrayList<String>();
  } else {
    this.authors = new ArrayList<>(Arrays.asList(authors));
  }
}
```

Check your solution here: https://judge.softuni.bg/Contests/1542/Iterators-and-Comparators-Lab

Iterable<T> and Iterator<T>

Collections Hierarchy

- An **Inheritance** leads to **hierarchies** of classes and/or interfaces in an application:



Iterable<T>

- Root interface of the Java collection classes
- A class that implements the **Iterable<T>** can be used with the new **for loop**

```java
List list = new ArrayList();
for(Object o : list) {
  // do something o;
}
```

Iterable<T> Methods

- Abstract methods
  - iterator()

```
public interface Iterable<T> {
  public Iterator<T> iterator();
}
```

- Default methods
  - forEach(Consumer<? super T> action)
  - spliterator() - used for parallel programming

Iterator<T>

- Enables you to cycle through a collection
- Nested class for **Iterator<T>**

```
public class Library<T> implements Iterable<T> {
  private final class LibIterator implements Iterator<T> {}
}
```

✓

- Don't implement both **Iterable<T>** and **Iterator<T>**

```
class MyClass implements Iterable<T>, Iterator<T> {}
```

✗

Problem: Library

- Create a class Library, which implements **`Iterable<Book>`**
- Create nested class LibIterator, which implements **`Iterator<Book>`**

```
<<Iterable<Book>>>
      Library
-------------------------
-books: Book[]
-------------------------
+iterator(): Iterator<Book>
```

```
<<Iterator<Book>>>
    LibIterator
-------------------------
-counter: int
-------------------------
+hasNext(): Boolean
+next(): Book
```

Check your solution here: https://judge.softuni.bg/Contests/1542/Iterators-and-Comparators-Lab

Solution: Library (1)

```
public class Library<Book> implements Iterable<Book> {
  private Book[] books;
  public Library(Book... books) {
    this.books = books;
  }
  public Iterator<Book> iterator() {
    return new LibIterator();
  }
  //TODO: Add nested iterator, look for it on next slide
}
```

Check your solution here: https://judge.softuni.bg/Contests/1542/Iterators-and-Comparators-Lab

Solution: Library (2)

```
private final class LibIterator implements Iterator<Book> {
  private int counter = 0;
  public boolean hasNext() {
    if(this.counter < books.length) { return true; }
    return false;
  }
  public Book next() { return books[counter++]; }
}
```

Check your solution here: https://judge.softuni.bg/Contests/1542/Iterators-and-Comparators-Lab

`Comparable<T>` and `Comparator<T>`

Comparator<E>

- The comparator provides a way for you to **provide custom comparison logic** for types that you have no control over
  - **Multiple** sorting sequence
  - **Doesn't affect** the original class
  - **`compare()`** method

Comparable<E>

- Comparable allows you to specify how objects **that you are implementing** get compared
  - **Single** sorting sequence
  - **Affects** the original class
  - `compareTo()` method

Comparable<E>

- Allows you to specify how objects that **you are implementing** get compared – the student's grades **st** and the **otherStudent**

```
class Student implements Comparable<Student> {
  // same as before
  @Override
  public int compareTo(Student st) {
      return Integer.compare(st.getGrades(),
      otherStudent.getGrades());
  }
}
```

Provide data type of compared object

Comparator<E>

Software University

- Allows you to provide **custom comparison logic**. Compares the grades of a **st** with the grades of a **st1**:

```
class StudentGradesComparator implements Comparator<Student> {
  // same as before
  @Override
  public int compare(Student st, Student st1) {
      return Integer.compare(st.getGrades(), st1.getGrades());
  }
}
```

Problem: Comparable Book

- Expand Book by implementing **`Comparable<Book>`**
- Book has to be **compared by title**
  - When title is equal, **compare** them by **year**

```
<<Comparable<Book>>> Book
---------------------------------
-title: String
-year: int
-authors: List<String>
---------------------------------
-setTitle(String)
-setYear(String)
-setAuthors(String…)
+getTitle(): String
+getYear(): int
+getAuthors(): +List<String>
+compareTo(Book): int
```

Check your solution here: https://judge.softuni.bg/Contests/1542/Iterators-and-Comparators-Lab

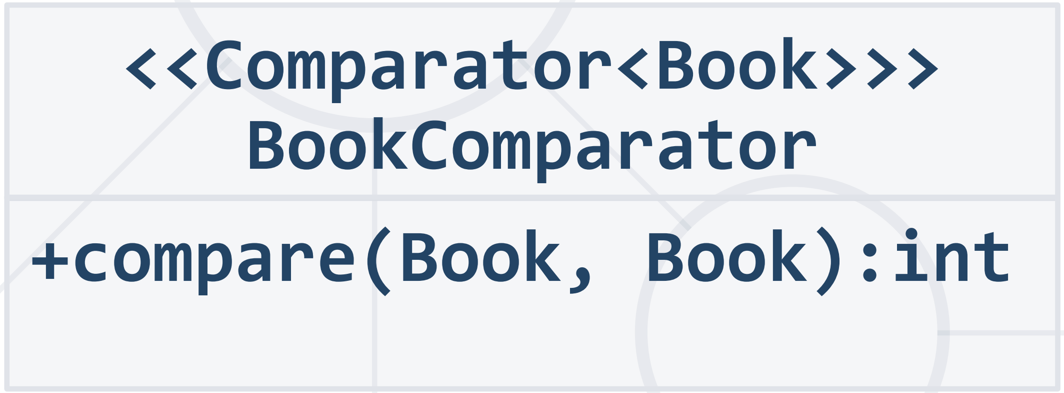
Solution: Comparable Book

```java
public int compareTo(Book book) {
  if (this.getTitle().compareTo(book.getTitle()) == 0) {
    if (this.getYear() > book.getYear()) { return 1;}
    else if (this.getYear() < book.getYear()) { return -1; }
    return 0;
  } else {
    return this.getTitle().compareTo(book.getTitle());
  }
}
```

Check your solution here: https://judge.softuni.bg/Contests/1542/Iterators-and-Comparators-Lab

Problem: Book Comparator

- Create a class, which can **compare** two books
- Use your **BookComparator** to sort list of Books



Check your solution here: https://judge.softuni.bg/Contests/1542/Iterators-and-Comparators-Lab

Solution: Book Comparator (1)

```
public class BookComparator implements Comparator<Book> {
  @Override
  public int compare(Book first, Book second) {
    if (first.getTitle().compareTo(second.getTitle()) == 0) {
      if (first.getYear() > second.getYear()) { return 1; }

      // Continues on the next slide
```

Check your solution here: https://judge.softuni.bg/Contests/1542/Iterators-and-Comparators-Lab

Solution: Book Comparator (2)

```
        // …
        else if (first.getYear() < second.getYear())
          return -1;
        return 0;
      } else {
        return first.getTitle().compareTo(second.getTitle());
      }
    }
}
```

Check your solution here: https://judge.softuni.bg/Contests/1542/Iterators-and-Comparators-Lab

Summary

- Variable arguments
- Iterable<T>
- Iterator<T>
- Comparable<T>
- Comparator<T>

Software University

Questions?

SoftUni

Software University

SoftUni Svetlina

SoftUni Creative

SoftUni Digital

SoftUni Foundation

SoftUni Kids

SoftUni Diamond Partners

Software University

DRAFT KINGS

POKERSTARS | CAREERS

SCHWARZ

LIDL

Kaufland

SCHWARZ Produktion

pre zero

Postbank

Решения за твоето утре

SOFTWARE GROUP

Coca-Cola HBC Bulgaria

SUPER HOSTING .BG

SmartIT

INDEAVR

Serving the high achievers

Bosch.IO

PHAR VISION

createX

taulia

Educational Partners

CODE IT UP

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg

License