Java

Collections

Tobias Hanf, Manik Khurana

6. Dezember 2021

Java-Course

Overview

- Generics
  - What is a generic
  - Wrapper Classes
- Collections
  - Overview
  - List
  - Iterating
- Hands-On
  - Comparable
  - Collections

Generics

Generics

```
1        Object myStringAsObject = "klaus";
2        String myStringAsString = (String) myStringAsObject;
3
```

Generics

```
1        Object myStringAsObject = Integer.valueOf("42");
2        String myStringAsString = (String) myStringAsObject;
3
```

Why it won’t work:

Integer can’t be casted to String.

The Code before will compile but still cause an Exception in the JVM.

Generics

```
1        public class Box {
2            private Object object;
3
4            public void set(Object object) { this.object = object; }
5            public Object get() { return object; }
6        }
7
8
```

Generics

```
1   public class Box<T> {
2       // T stands for "Type"
3       private T t;
4
5       public void set(T t) { this.t = t; }
6       public T get() { return t; }
7   }
8
9   Box<Integer> integerBox; = new Box<Integer>();
10
11
```

Wrapper Class

Primitive data types can not be elements in collections. Use wrapper classes like *Integer* instead.

boolean	Boolean
byte	Byte
char	Character
int	Integer
float	Float
double	Double
long	Long
short	Short

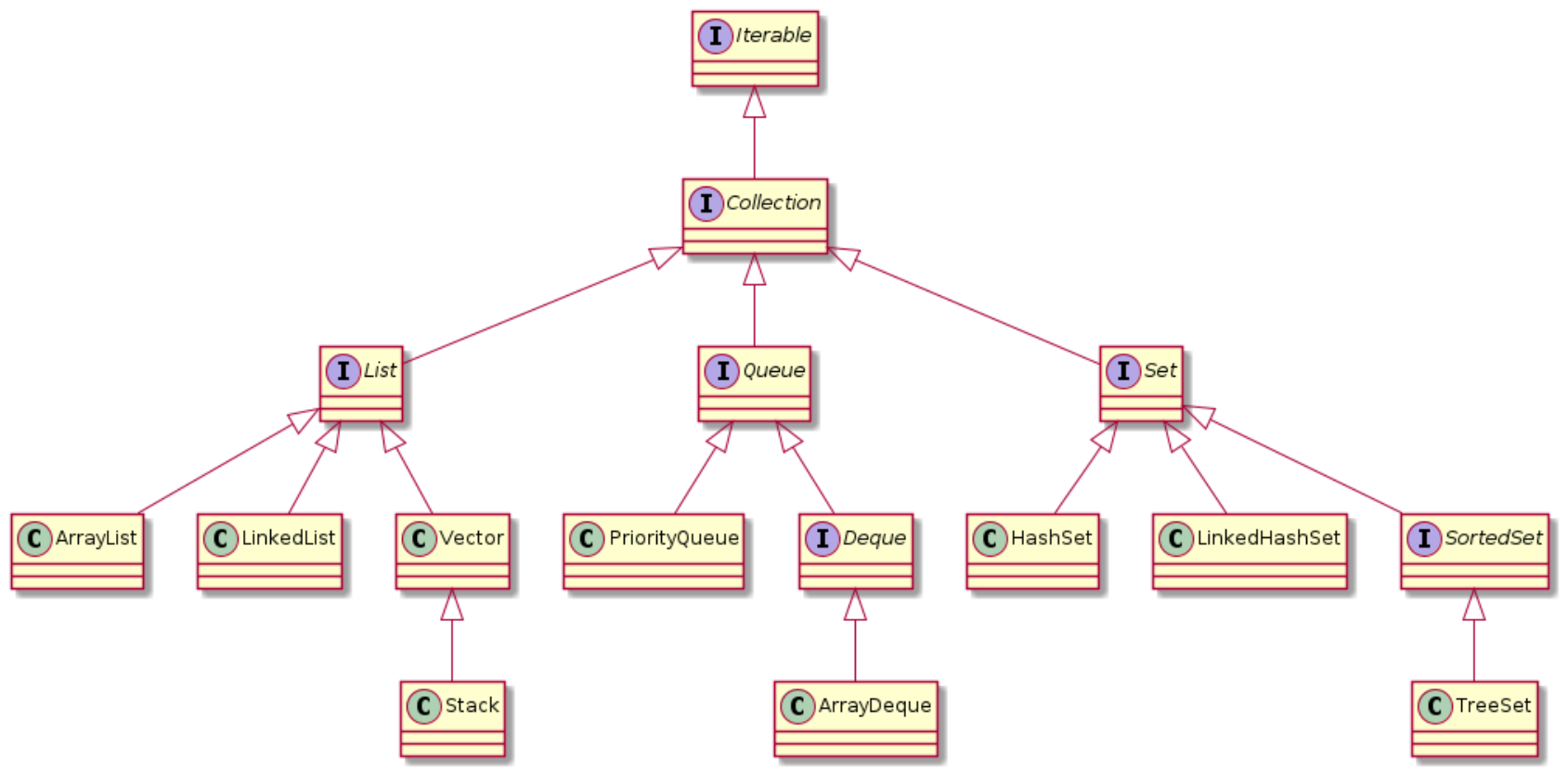
Collections

Collections Framework

Java offers various data structures like **Sets**, **Lists** and **Maps**. Those structures are part of the collections framework.

There are interfaces to access the data structures in an easy way. There are multiple implementations for various needs. Alternatively you can use your own implementations.

Documentation: `https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/Collection.html`


Iterable
Collection
List
Queue
Set
ArrayList
LinkedList
Vector
PriorityQueue
Deque
HashSet
LinkedHashSet
SortedSet
Stack
ArrayDeque
TreeSet

List

A list is an ordered collection.

The implementation `LinkedList` is a double-linked list.

```
1     public static void main(String[] args) {
2
3         List<String> list = new LinkedList<String>();
4
5         list.add("foo");
6         list.add("foo"); // insert "foo" at the end
7         list.add("bar");
8         list.add("foo");
9         list.remove("foo"); // removes the first "foo"
10
11        System.out.println(list); // prints: [foo, bar, foo]
12    }
13
```

https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/List.html

List Methods

some useful List methods:

void	`add(int index, E element)`	insert element at position index
E	`get(int index)`	get element at position index
E	`set(int index, E element)`	replace element at position index
E	`remove(int index)`	remove element at position index

some useful LinkedList methods:

void	`addFirst(E element)`	append element to the beginning
E	`getFirst()`	get first element
void	`addLast(E element)`	append element to the end
E	`getLast()`	get last element

LinkedList vs ArrayList i

ArrayList[1]:

- Resizeable-array implementation
- List has a specific capacity, may have to be resized (automatically)
- But `add()` runs in amortized constant time (O(n))
- `size`, `isEmpty`, `get`, `set`, `iterator`, and `listIterator` in constant time
- Any other method runs in linear time

LinkedList vs ArrayList ii

LinkedList[2]:

- Doubly-linked list implementation
- Can grow indefinitely without resizing (memory constraint)
- `add`, `get` and `remove` at the end/beginning fast
- Any other element, time depending on position

[1] https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/ArrayList.html
[2] https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/LinkedList.html

For Loop

The for loop can iterate over every element of a collection:

`for (E e : collection)`

```
1   public static void main(String[] args) {
2
3       List<Integer> list =
4           new LinkedList<Integer>();
5
6       list.add(1);
7       list.add(3);
8       list.add(3);
9       list.add(7);
10
11      for (Integer i : list) {
12          System.out.print(i + " "); // prints: 1 3 3 7
13      }
14  }
15
```

Iterator

An iterator iterates step by step over a collection.

```
1     public static void main(String[] args) {
2
3         List<Integer> list = new LinkedList<Integer>();
4
5         list.add(1);
6         list.add(3);
7         list.add(3);
8         list.add(7);
9
10        Iterator<Integer> iter = list.iterator();
11
12        while (iter.hasNext()) {
13            System.out.print(iter.next());
14        }
15        // prints: 1337
16    }
17
```

Iterator

A standard iterator has only three methods:

- `boolean hasNext()` - indicates if therer are more elements
- `E next()` - returns the next element
- `void remove()` - returns the current element

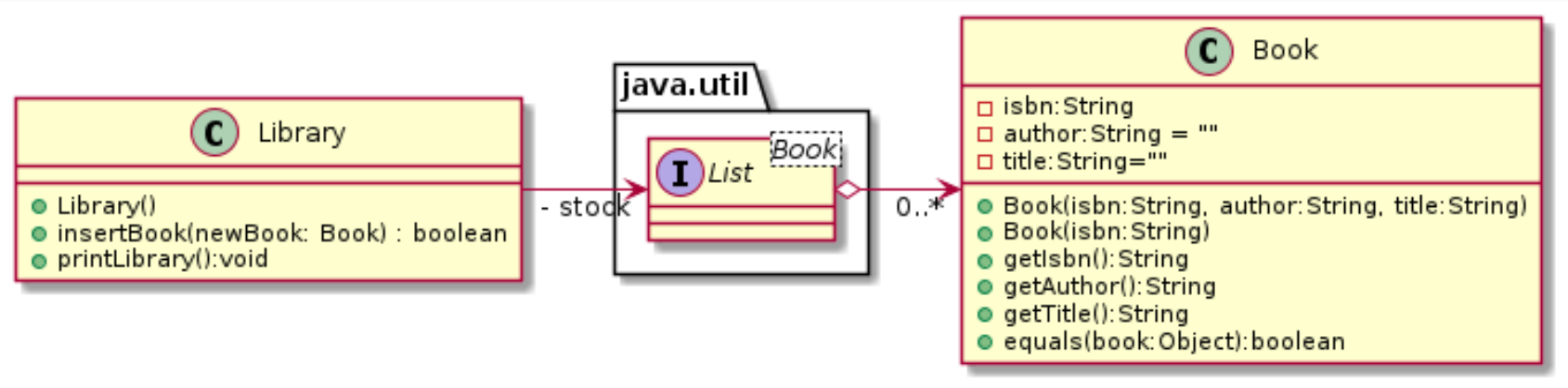
The iterator is instanced via `collection.iterator()` :

```
1         Collection<E> collection = new Implementation<E>;
2         Iterator<E> iter = collection.iterator();
3
```

Special iterators like *ListIterator* are more sophisticated.

Hands-On

Library (Part 1)

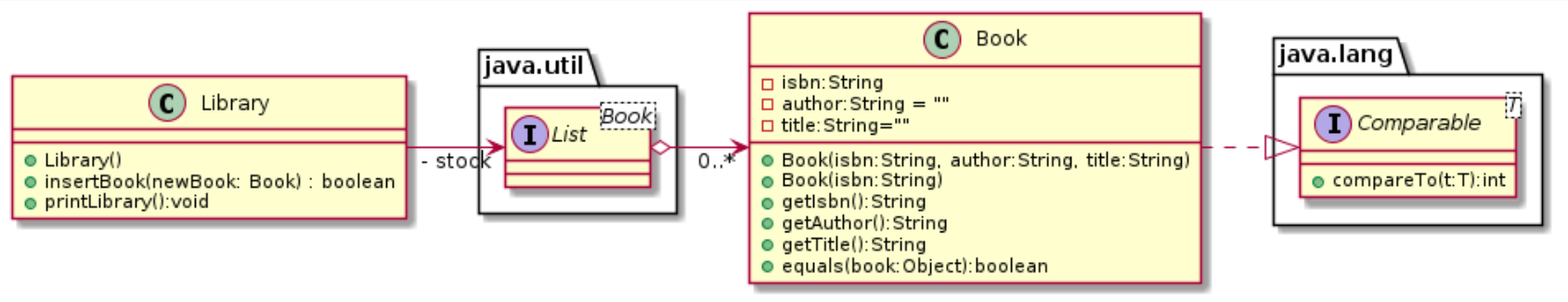


Comparable

- `public interface Comparable<T>`[3]
- Interface to create a natural ordering of objects
- One method `int compareTo(T o)`
- should return
  - 0 if equal
  - -1 if smaller
  - 1 if greater

[3] `https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/lang/Comparable.html`

Part 2


Library
Library()
insertBook(newBook: Book) : boolean
printLibrary():void
- stock
java.util
List
Book
0..*
Book
isbn:String
author:String = ""
title:String=""
Book(isbn:String, author:String, title:String)
Book(isbn:String)
getIsbn():String
getAuthor():String
getTitle():String
equals(book:Object):boolean
java.lang
Comparable
T
compareTo(t:T):int

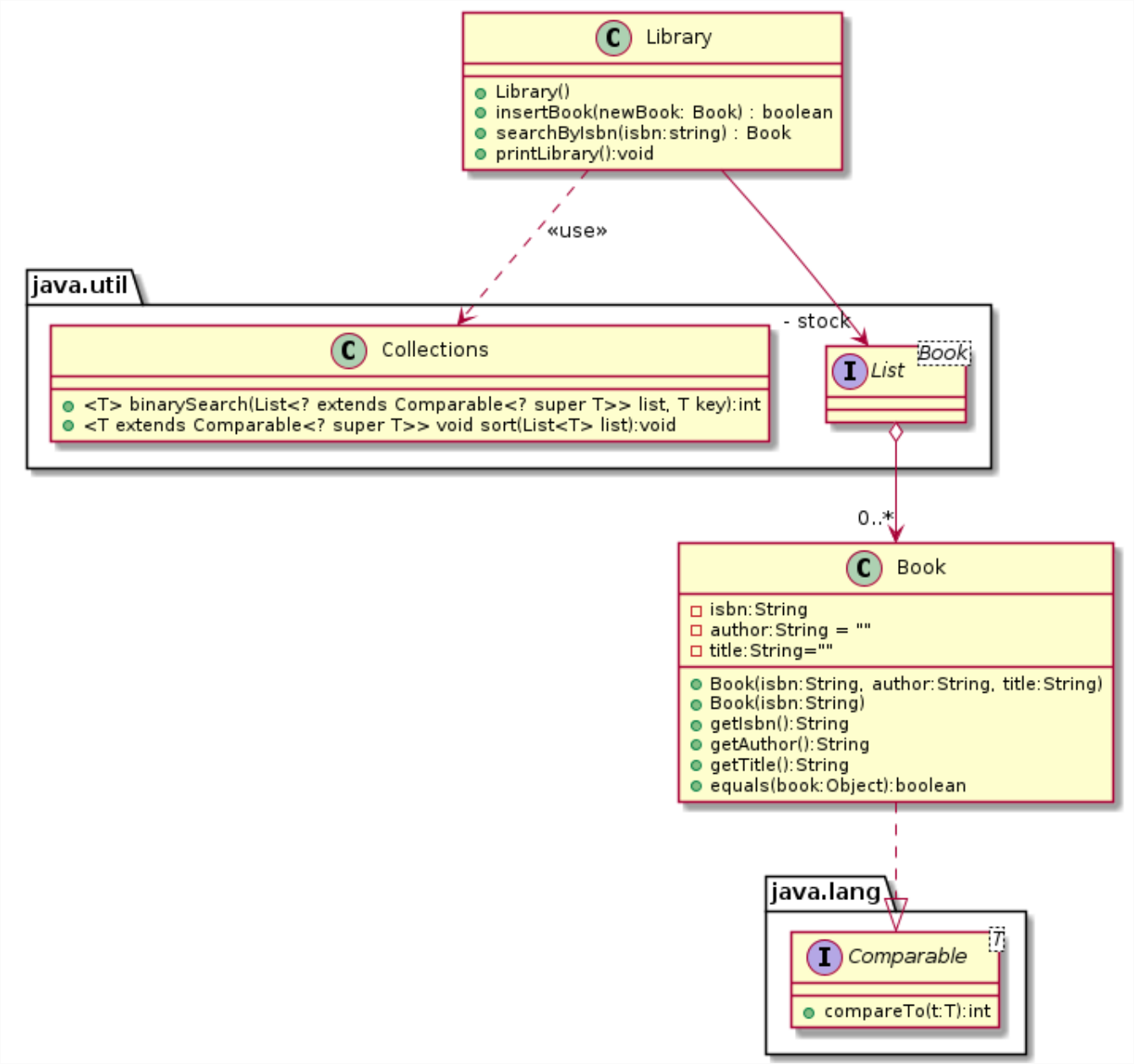
Collections

”This class consists exclusively of static methods that operate on or return collections”[4]

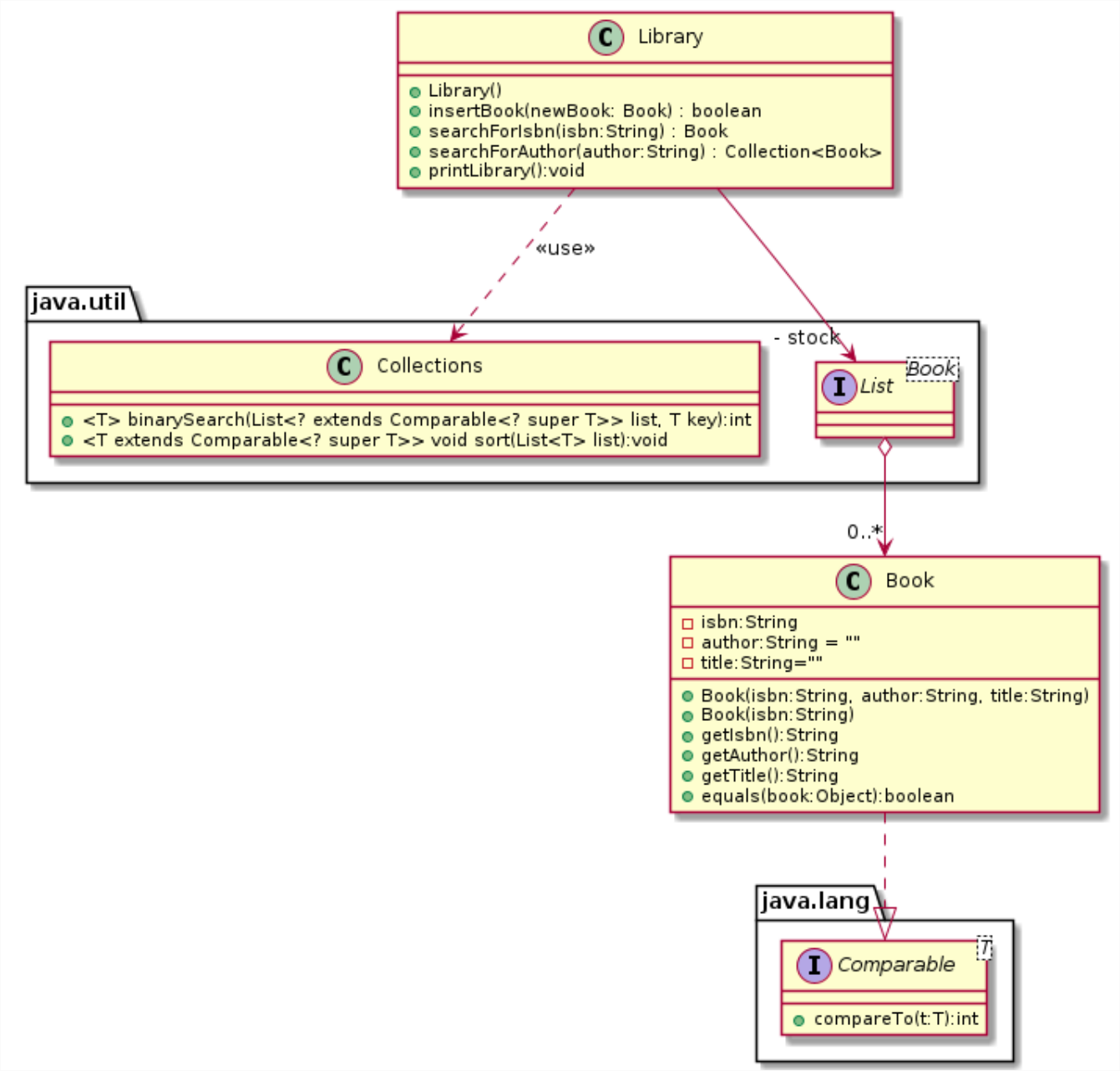
Some methods:

- `binarySearch(...)`
- `max(...)`
- `min(...)`
- `reverse(...)`
- `sort(...)`

[4] `https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/Collections.html`


Library
Library()
insertBook(newBook: Book) : boolean
searchByIsbn(isbn:string) : Book
printLibrary():void
«use»
java.util
Collections
<T> binarySearch(List<? extends Comparable<? super T>> list, T key):int
<T extends Comparable<? super T>> void sort(List<T> list):void
- stock
List
Book
0..*
Book
isbn:String
author:String = ""
title:String=""
Book(isbn:String, author:String, title:String)
Book(isbn:String)
getIsbn():String
getAuthor():String
getTitle():String
equals(book:Object):boolean
java.lang
Comparable
T
compareTo(t:T):int

Part 4


Library
Library()
insertBook(newBook: Book) : boolean
searchForIsbn(isbn:String) : Book
searchForAuthor(author:String) : Collection<Book>
printLibrary():void
«use»
java.util
Collections
<T> binarySearch(List<? extends Comparable<? super T>> list, T key):int
<T extends Comparable<? super T>> void sort(List<T> list):void
- stock
List
Book
0..*
Book
isbn:String
author:String = ""
title:String=""
Book(isbn:String, author:String, title:String)
Book(isbn:String)
getIsbn():String
getAuthor():String
getTitle():String
equals(book:Object):boolean
java.lang
Comparable
T
compareTo(t:T):int