Iterator Pattern (continued)

CSCI 2300

Key Elements of Iterator Pattern

- A class that represents a collection of objects
- Iterator interface
  - Java's Iterator interface
  - Your own Iterator interface
- When implementing an Iterator Pattern include:
  - A way to create an iterator of a collection
  - hasNext() method – true if there is another element in the collection
  - next() method – retrieves the next element of the collection
  - Optionally: remove() method – to remove the current element of the collection

Which of the following classes would benefit from implementing Iterator Pattern?

A. BankAccount

B. TicTacToePlayer

C. List – (list of students in a class, for example)

D. BookShelf

Java’s Iterator pattern interfaces

- Enumeration<E>
  - Access to each element of the collection
- Iterator<E>
  - Access to each element of the collection
  - Access to remove elements from the collection

java.util.Enumeration<E>

Modifier and Type	Method and Description
`boolean`	**`hasMoreElements()`** Tests if this enumeration (collection) contains more elements
`E`	**`nextElement()`** Returns the next element of this enumeration if this enumeration has at least one more element to provide. Throws NoSuchElementException – if no more elements exist

Suppose we have a class, `History`, containing a list of `Record` objects. This class implements `java.util.Enumeration<Record>` interface. **What does `nextElement()` method of this class return?**

A. History

B. Enumeration

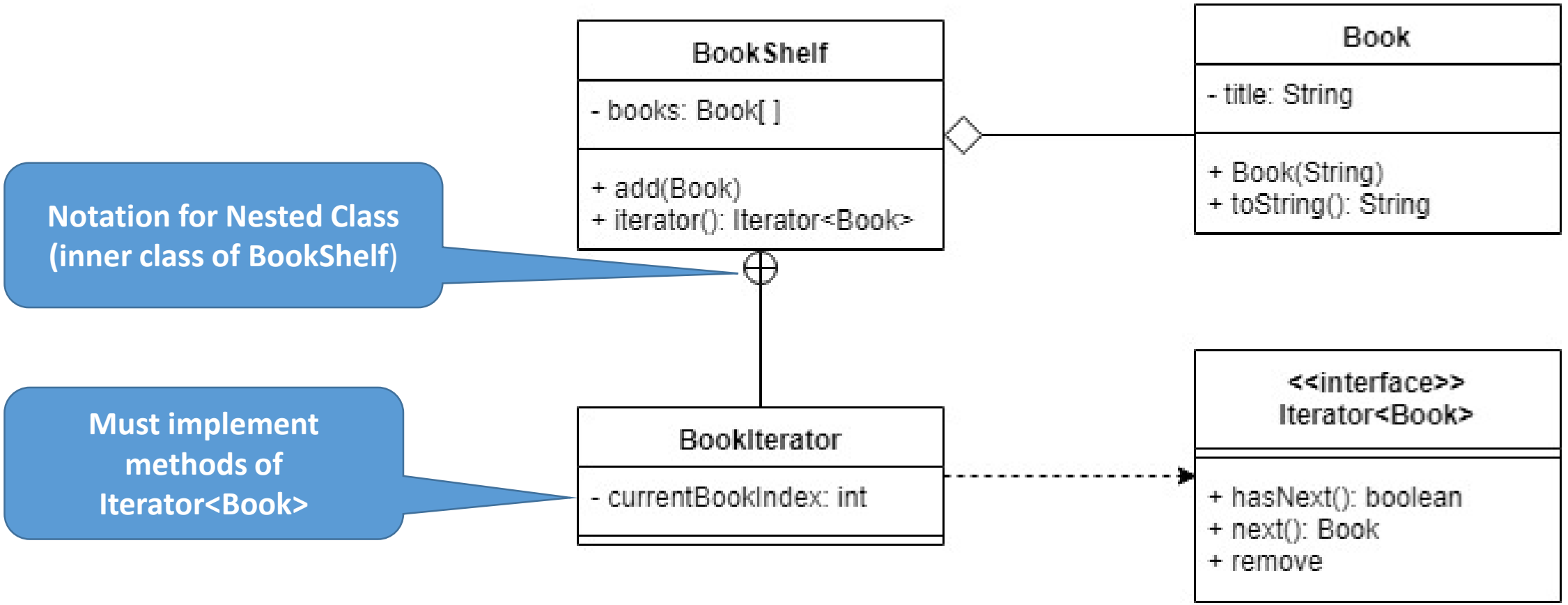
C. Record

D. boolean

java.util.Iterator<E>

Modifier and Type	Method and Description
`boolean`	**`hasNext()`** Returns true if the iteration has more elements
`E`	**`next()`** Returns the next element of this iteration. Throws NoSuchElementException – if no more elements exist
`void`	**`remove()`** Removes from the collection the last element returned by the iterator (optional)

Iterator Pattern Design



Iterator Pattern Implementation

```
public class BookShelf{
    // instance variables

    // constructor
    public BookShelf(){…}
    // Create an Iterator<Book> object
    public void add(Book b){…}
    public Iterator<Book> iterator(){
        return new BookIterator();
   }
   // BookIterator class here
}
```

BookIterator implementation

```
public class BookIterator implements Iterator<Book>{
    // instance variables
    // current book index for example

    // constructor
    public BookIterator(){…}

    public boolean hasNext(){…}
    public Book next(){…}
}
```

Iterating over a BookShelf

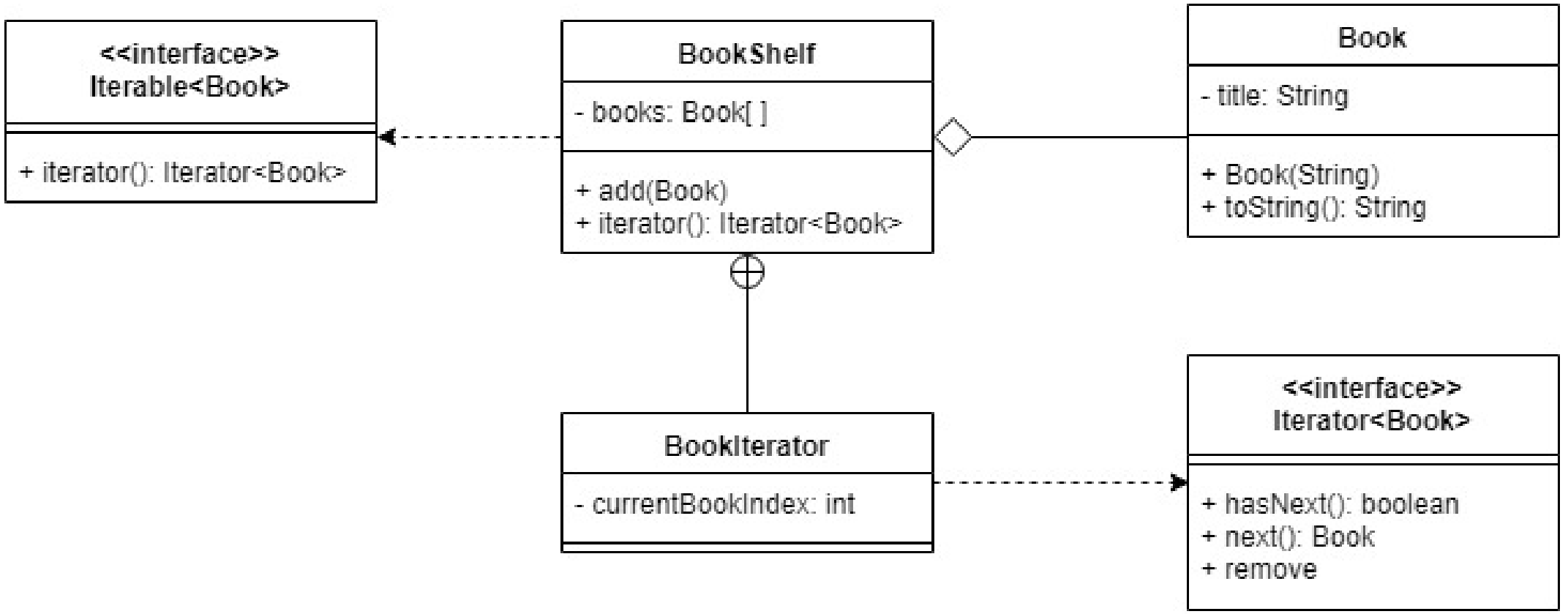
```
BookShelf shelf = new BookShelf();
shelf.add(new Book("Object Oriented Design"));
shelf.add(new Book("The Adventures of Tom Sawyer"));

Iterator<Book> bookIter = shelf.iterator();
while (bookIter.hasNext())
{
   System.out.println(bookIter.next());
}
```

`Iterable` interface

- In our example, we can iterate over a BookShelf
- The BookShelf is “iterable”
- Utilize Iterable interface, to improve design

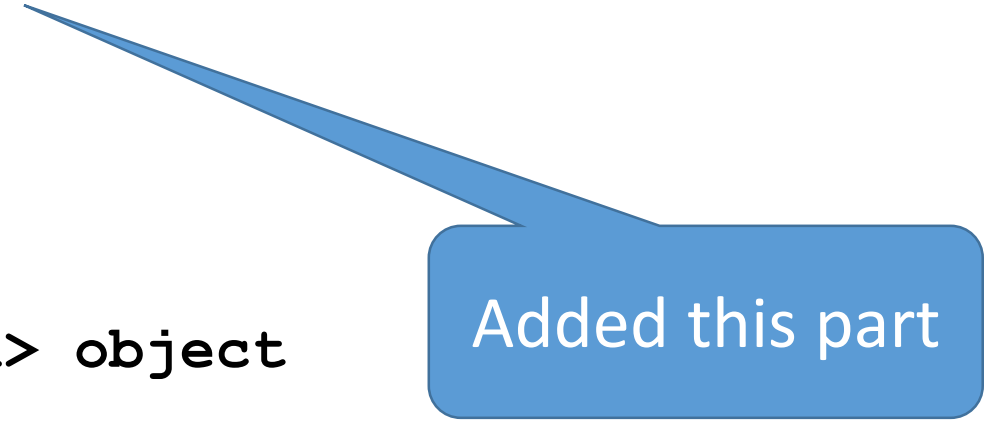
Iterable BookShelf Design



Iterable BookShelf Implementation

```
public class BookShelf implements Iterable<Book>{
    // instance variables

    // constructor
    public BookShelf(){…}
    // Create an Iterator<Book> object
    public void add(Book b){…}
    public Iterator<Book> iterator(){
        return new BookIterator();
   }
   // BookIterator class here
}
```



Iterating over an ‘Iterable’ BookShelf

```
BookShelf shelf = new BookShelf();
shelf.add(new Book(“Object Oriented Design”));
shelf.add(new Book(“The Adventures of Tom Sawyer”));

for(Book b: shelf)
{
   System.out.println(b);
}
```

Practice Exercise

- Write a Sentence class that
  - Takes a sentence in the constructor and stores it internally
  - Implements `Iterable<String>` interface
- Create a SentenceIterator<char> class that
  - Implements `Iterator<char>` interface
  - Returns the next character in the sentence on `next()`
  - Returns true if there are characters left in a sentence on `hasNext()`
  - You can use String's `charAt(int index)` method to get the character at the given index of the sentence String, and `length()` method to get the length of the sentence String.
- Create a Driver class that creates an instance of a Sentence and iterates over it, printing each character on a new line.
- Challenge: modify Sentence to iterate over words instead of characters