软件体系结构

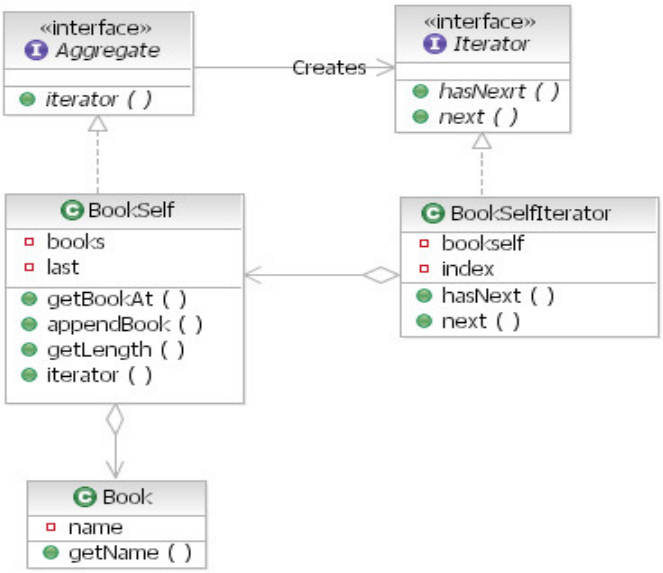
《软件体系结构作业九》

学　号 22920212204396

姓　名 黄子安

2024 年 4 月 26 日

1、针对 Iterator 的例子，将存储 Book 用的数组换成其他 Collection 并运行。



例子的类图如上所示，书架在逻辑上是线性表结构，更换成 Java 的 ArrayList 会比较适合，更符合逻辑特征，因此对 BookSelf 类做以下修改

```
import java.util.ArrayList;

public class BookShelf implements Aggregate {
    private ArrayList<Book> books;
    public BookShelf(int maxsize) {
        this.books = new ArrayList<>(maxsize);
    }
    public Book getBookAt(int index) {
        return books.get(index);
    }
    public void appendBook(Book book) {
        this.books.add(book);
    }
    public int getLength() {
        return this.books.size();
    }
    public Iterator iterator() {
        return new BookShelfIterator(this);
    }
}
```

因为 ArrayList 本身就是线性表，因此对于 BookselfIterator 类不需要做修改，直接根据下标值返回是否存在下一个元素和需要的时候返回下一个元素即可。对

于其他复杂的数据结构容器，需要定义好下一个元素是谁，比如对于树可以定义成 dfs 序上的节点从而化成链式结构，当然对于本题来说选用树作为存储容器是不合理的

```java
//实现Iterator接口即可将BookShelfIterator视为Iterator进行处理
public class BookShelfIterator implements Iterator {
    private BookShelf bookShelf;
    private int index;
    public BookShelfIterator(BookShelf bookShelf) {
        this.bookShelf = bookShelf;
        this.index = 0;
    }

    public boolean hasNext() {
        return index < bookShelf.getLength();
    }

    //next方法返回目前该书（Book的对象实例），然后把index推到下一个为位置
    public Object next() {
        Book book = bookShelf.getBookAt(index);
        index++;
        return book;
    }
}
```

运行代码，可以发现输出依旧相同，从逻辑上来说对客户端屏蔽了 Bookshelf 中的容器细节，即使换成了树等较为复杂的数据结构客户端的代码也不需要做任何的改变，从而体现了迭代器模式的设计理念

```
D:\Java17\bin\java.exe "-javaagent:D:\IdeaU\IntelliJ IDEA 2023.1.3\lib\idea_rt.jar=560
Around the World in 80 Days
Bible
Forrest Gump
Triumph

Process finished with exit code 0
```

2、针对 Iterator 的例子，设计一个 Specified 的 Iterator 并运行。

这次修改迭代器，使得只返回以字母 A 开头的书本，修改后具体的迭代器代码如下所示，判断是否存在下一个的时候会进行扫描，找到存在下一个以 A 开头的书本则停止，如果超过数组上限说明不存在，返回 false；next 函数也同理，这里返回后要把 index 再往后推一个位置，实现不断向前遍历

```
//实现Iterator接口即可将BookShelfIterator视为Iterator进行处理
public class BookShelfIterator implements Iterator {
    private BookShelf bookShelf;
    private int index;
    public BookShelfIterator(BookShelf bookShelf) {
        this.bookShelf = bookShelf;
        this.index = 0;
    }
    public boolean hasNext() {
        int idx = index;
        while(idx < bookShelf.getLength()
              && ! bookShelf.getBookAt(idx).getName().startsWith("A")) idx++;
        return idx < bookShelf.getLength();
    }
    //next方法返回目前该书（Book的对象实例），然后把index推到下一个为位置
    public Object next() {
        while(index < bookShelf.getLength()
              && ! bookShelf.getBookAt(index).getName().startsWith("A")) index++;
        return bookShelf.getBookAt(index++);
    }
}
```

修改下测试的代码，再加入基本以 A 开头的书，使得效果更加明显

```
public class Main {
    public static void main(String[] args) {
        BookShelf bookShelf = new BookShelf(4);
        bookShelf.appendBook(new Book("Around the World in 80 Days"));
        bookShelf.appendBook(new Book("Bible"));
        bookShelf.appendBook(new Book("Forrest Gump"));
        bookShelf.appendBook(new Book("Triumph"));
        bookShelf.appendBook(new Book("Alice in Wonderland"));
        bookShelf.appendBook(new Book("A Brief History of Time"));
        bookShelf.appendBook(new Book("Bible"));
        //返回值不指定到BookShelfIterator类变量
        //而是指定到Iterator类变量
        //因为我们不是利用shelfIterator的方法来写程序
        //而是打算利用Iterator的方法来写程序
        Iterator it = bookShelf.iterator();
        while (it.hasNext()) {
            Book book = (Book)it.next();
            System.out.println(book.getName());
        }
    }
}
```

运行后的结果如下图所示，符合对迭代器的预期

```
D:\Java17\bin\java.exe "-javaagent:D:\IdeaU\IntelliJ IDEA 2023.1.3\lib\idea_rt.jar=57100:D:\
Around the World in 80 Days
Alice in Wonderland
A Brief History of Time

Process finished with exit code 0
```