Дисциплина: *СД и ООП* **Тема:** *Списък*
Специалност: *КН 1 курс, 3 и 4 групи* **Седмица**: 0*7.05 - 10.05.2009 г.*

Задача: Информация за наличните книги в книжарница се съхранява в списък, който за всяка книга съдържа: каталожен номер, наименование, автор, цена и налично количество екземпляри. Списъкът е нареден във възходящ ред на каталожните номера. Да се реализира на езика Java клас за представяне на списък от книги, съдържащ:

- Конструктор за създаване на празен списък
- Метод за доставка на книга, който включва книгата в списъка, ако такава няма в него. В противен случай към наличното количество се добавя новополученото
- Метод за продажба на книга. Ако книгата я няма в списъка се активира изключение **BookNotFoundException**. В противен случай: ако исканото количество е по-малко от наличното, наличното количество се намалява с исканото и методът връща исканото количество, ако исканото количество е по-голямо или равно на наличното, книгата се изключва от списъка и методът връща наличното количество
- Метод за изпълнение на заявка с параметри: едномерен масив от книги и вид - доставка или продажба, който използва горните методи. Методът връща общ брой доставени или продадени книги
- Метод с параметър автор, който връща списък от наличните книги, написани от него

Реализация:

1.Клас **Book** за представяне на книга

```
public class Book {
	//Data
	int bookID;
	String title,author;
	int quantity;

	//Constructor
	public Book(int id,String t,String a,int q) {
		bookID = id;
		title = t;
		author = a;
		quantity = q;
	}

	//Methods
	public int compareTo(Book b) { return this.bookID - b.bookID; }

	public String toString() {
		return "(" + bookID + "," + title + "," + author + "," + quantity + ")";
	}
}
```

2.Клас **Request** за представяне на заявка

```
public class Request {
	//Data
	int kind; //1 - supplies, 2 - sales
	Book[] books;

	//Constructor
	public Request(int k,Book[] b) { kind = k; books = b; }
}
```

3.Клас **BookNotFoundException**

```
public class BookNotFoundException extends RuntimeException {
        public BookNotFoundException (String msg) {super(msg); }
}
```

4.Клас за представяне на книжарница

4.1. Клас **BookList2** - представяне чрез *обект на клас-реализация на списък*

```
public class BookList2 {
        //Data
        DoublyLinkedList data;
        //java.util.LinkedList<Book> data;

        //Constructor
        public BookList2() {
                data = new DoublyLinkedList();
                //data = new java.util.LinkedList<Book>();
        }

        //Public methods
        public int addBook(Book b) {
                boolean flag = false;
                int i = 0;
                for(; i < data.size(); i++) {
                        Book w = (Book)data.get(i);
                        //Book w = data.get(i);
                        if(b.compareTo(w) < 0) {
                                data.add(i,b);
                                flag = true;
                                break;
                        }
                        else if(b.compareTo(w) == 0) {
                                flag = true;
                                w.quantity += b.quantity;
                                break;
                        }
                }
                if(!flag) data.add(i,b);
                return b.quantity;
        }

        public int saleBook(Book b) throws BookNotFoundException {
                for(int i = 0; i < data.size(); i++) {
                        Book w = (Book)data.get(i);
                        //Book w = data.get(i);
                        if(b.compareTo(w) == 0) {
                                int q1 = w.quantity;
                                int q2 = b.quantity;
                                if (q1 > q2) {
                                        w.quantity = q1 - q2;
                                        return q2;
                                }
                                else {
                                        data.remove(i);
                                        return q1;
                                }
                        }
                }
                throw new BookNotFoundException(b.bookID + "  " + b.title + " not found");
        }

        public BookList2 getBooks(String author) {
                BookList2 newList = new BookList2();
                Iterator it = data.iterator();
                //java.util.Iterator<Book> it = data.iterator();
```

```
		while(it.hasNext()) {
			Book w = (Book)it.next();
			//Book w = it.next();
			if(author.equals(w.author))
				newList.addBook(w);
		}
		return newList;
	}

	public int doRequest(Request r) {
		int sum = 0;
		if(r.kind == 1)
			for(int i = 0; i < r.books.length; i++)
				sum += addBook(r.books[i]);
		else
			for(int i = 0; i < r.books.length; i++)
				try {
					sum += saleBook(r.books[i]);
				}catch (BookNotFoundException e) {
					System.out.println(e.toString());}
		return sum;
	}

	public void show() {
		Iterator it = data.iterator();
		//java.util.Iterator<Book> it = data.iterator();
		while(it.hasNext())
			System.out.println(it.next());
	}
}
```

4.2. Клас **BookList** - представяне чрез *линеен едносвързан списък*

```
class BookNode {
	//Data
	Book data;
	BookNode next;

	//Constructors
	BookNode(Book b) { data = b; next = null; }
	BookNode(Book b,BookNode n) { data = b; next = n; }
}

public class BookList {
	//Data
	BookNode first;

	//Constructor
	public BookList() {first = null; }

	//Public methods
	public int addBook(Book b) {
		if(first == null || b.compareTo(first.data) < 0) first = new BookNode(b,first);
		else if(b.compareTo(first.data) == 0) first.data.quantity += b.quantity;
		else {
			BookNode p = first;
			while(p.next != null)
				if(b.compareTo(p.next.data) > 0) p = p.next;
				else if(b.compareTo(p.next.data) == 0) {
					p.next.data.quantity += b.quantity;
					break;
				}
				else {
					p.next = new BookNode(b,p.next.next);
```

```
                                        break;
                                }

                                if(p.next == null) p.next = new BookNode(b);
                        }
                        return b.quantity;
                }

                public int saleBook(Book b) throws BookNotFoundException {
                        if(first == null)
                                throw new BookNotFoundException (b.bookID + "  " + b.title + " not found");
                        else if(b.compareTo(first.data) == 0) {
                                int q1 = first.data.quantity;
                                int q2 = b.quantity;
                                if(q1 > q2) {
                                        first.data.quantity = q1-q2;
                                        return q2;
                                }
                                else {
                                        first = first.next;
                                        return q1;
                                }
                        }
                        else {
                                BookNode p = first;
                                while(p.next != null)
                                        if(b.compareTo(p.next.data) == 0) {
                                                int q1 = p.next.data.quantity;
                                                int q2 = b.quantity;
                                                if (q1 > q2) {
                                                        p.next.data.quantity = q1 - q2;
                                                        return q2;
                                                }
                                                else {
                                                        p.next = p.next.next;
                                                        return q1;
                                                }
                                        }
                                        else p = p.next;
                                throw new BookNotFoundException(b.bookID + "  " + b.title + " not found");
                        }
                }

                public BookList getBooks(String author) {
                        BookList newList = new BookList();
                        BookNode p = first;
                        while(p != null) {
                                if(author.equals(p.data.author))
                                        newList.addBook(p.data);
                                p = p.next;
                        }
                        return newList;
                }

                public int doRequest(Request r) {…}

                public void show() {
                        BookNode p = first;
                        while(p != null) {
                                System.out.println(p.data);
                                p = p.next;
                        }
                }
}
```