Дисциплина: *СД и ООП* **Тема:** *Списък - продължение*
Специалност: *КН 1 курс, 3 и 4 групи* **Дата**: *28.04.2010 г.*

Задача 1: Да се състави програма на езика Java за създаване и обработка на списък от наличните книги в книжарница. За всяка книга списъкът съдържа: каталожен номер, наименование, автор, цена и налично количество екземпляри. Списъкът е нареден във възходящ ред на каталожните номера. Реализацията да включва:

- ✘ Клас **Book** за представяне на книга
- ✘ Клас **Request** за представяне на заявка
- ✘ Клас **BookNotFoundException**
- ✘ Клас **BookList** за представяне на списък от книги, съдържащ:

√Декларации на променливи за съхранение на данните
√Конструктор за създаване на празен списък
√Методи

- •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 собствено 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, 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
- •Метод за отпечатване на данните на наличните книги
- ✘ Клас за тестване

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 списък от книги

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);
                                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) { throw new UnsupportedOperationException();  }
```

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

Задача 2: Да се реализират следните класове за представяне на съобщения в пощенска кутия:

✘ Клас **Message** за представяне на email съобщение. Съобщението има получател, подател, дата на изпращане (обект от клас Date) и текст. Класът да съдържа :

√Декларации на променливи за съхранение на данните

√Конструктор с параметри получател и подател, който прави датата на съобщението равна на текущата дата

√Методи

- append, който прибавя ред към текста от съобщението
- toString, който превръща съобщението в низ като следния: ‘’From: <подател>\nTo: <получател>\n<дата>\n<текст>’’
- compareTo за сравнение на две съобщения по дата
- print, който отпечатва съобщението

✘ Клас **Mailbox** за представяне на пощенска кутия, който съхранява съобщения (обекти от клас Message), в намаляващ ред на датите им. Класът да съдържа:

√Декларации на променливи за съхранение на данните

√Конструктори

√Методи

- public void addMessage(Message msg), който съхранява msg
- public Message getMessage(int i), който връща i-то съобщение
- public void removeMessage(int i), който премахва i-то съобщение
- public void printAllMessages(), който отпечатва всички съобщения

Задача 3: Да се реализират:

✘ Клас **Country** за представяне на държава с полета за нейното име, население и площ. Класът да съдържа:

√Декларации на променливи за съхранение на данните

√Конструктор(и)

√Методи

- Методи за достъп до данните
- Метод, който определя гъстотата на населението (брой жители на единица площ)
- Метод за сравнение на две държави по име

✘ Клас **Continent** за представяне на континент, който има име и списък от държави на континента, нареден в азбучен ред на имената им. Класът да съдържа:

√Декларации на променливи за съхранение на данните

√Конструктор(и)

√Методи

- Методи за достъп до данните
- Метод, който определя страната с най-голяма площ
- Метод, който определя страната с най-многобройно население
- Метод, който определя страната с най-голяма гъстота на населението
- Метод, който определя площта, заета от държавите
- Метод, който извежда данните

✘ Клас **World** за представяне на списък от континенти. Класът да съдържа:

√Декларации на променливи за съхранение на данните

√Конструктор(и)

√Методи

- Методи за достъп до данните
- Метод, който определя страната с най-голяма площ в света
- Метод, който определя страната с най-многобройно население в света
- Метод, който определя страната с най-голяма гъстота на населението в света
- Метод, който определя площта, заета от всички държави
- Метод, който извежда данните за всички континенти