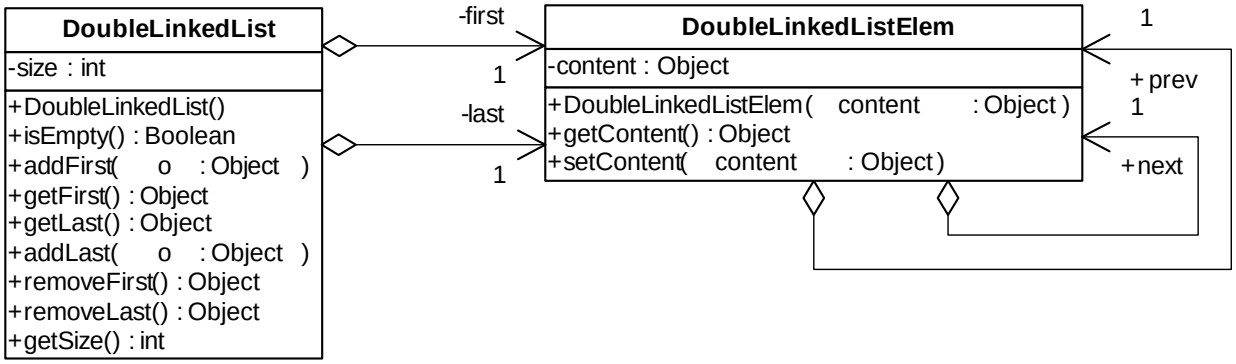
Technische Universität München
Fakultät für Informatik
D. Gmach, M. Rickert,
Dr. M. Schneider

WS 2005/2006
Lösungsvorschläge zu Blatt 12
27. Januar 2006

Übungen zu Einführung in die Informatik I

Aufgabe 48 Doppelt verkettete Liste in Java (Lösungsvorschlag)

a) UML-Klassendiagramm:



b)

```
public class DoubleLinkedList {

    private int size;
    private DoubleLinkedListElem firstElem;
    private DoubleLinkedListElem lastElem;


    public DoubleLinkedList() {
        firstElem = null;
        lastElem = null;
        size = 0;
    }

    public boolean isEmpty() {
        return size==0;
    }

    public void addFirst(Object o) {
        DoubleLinkedListElem newElem = new DoubleLinkedListElem(o)
            ;
        if (!isEmpty()) {
            firstElem.prev = newElem;
            newElem.next = firstElem;
        } else {
```

```
            lastElem = newElem;
        }
        firstElem = newElem;
        size++;
    }

    public Object getFirst() {
        return firstElem.getContent();
    }

    public Object removeFirst() {
        if (isEmpty())
            return null;
        DoubleLinkedListElem elem = firstElem;
        firstElem = firstElem.next;
        if (firstElem != null) {
            firstElem.prev = null;
        } else {
            lastElem = null;
        }
        size--;
        return elem.getContent();
    }

    public void addLast(Object o) {
        DoubleLinkedListElem newElem = new DoubleLinkedListElem(o)
            ;
        if (!isEmpty()) {
            lastElem.next = newElem;
            newElem.prev = lastElem;
        } else {
            firstElem = newElem;
        }
        lastElem = newElem;
        size++;
    }

    public Object getLast() {
        return lastElem.getContent();
    }

    public Object removeLast() {
        if (isEmpty())
            return null;
        DoubleLinkedListElem elem = lastElem;
        lastElem = lastElem.prev;
        if (lastElem != null) {
            lastElem.next = null;
        } else {
            firstElem = null;
        }
        size--;
        return elem.getContent();
    }
```

```
    public int getSize() {
        return size;
    }

    //only used for testing
    public static void main(String[] argv) {
        DoubleLinkedList myList = new DoubleLinkedList();
        System.out.println("Size: " + myList.getSize());
        System.out.println("List is empty: " + myList.isEmpty());
        System.out.println("Adding Element 1");
        myList.addFirst(new String("Element 1"));
        System.out.println("Size: " + myList.getSize());
        System.out.println("List is empty: " + myList.isEmpty());
        System.out.println("First Element: " + (String) myList.
           getFirst());
        System.out.println("Adding Element 2");
        myList.addLast(new String("Element 2"));
        System.out.print("Content of the list: ");
        DoubleLinkedListElem it = myList.firstElem;
        while (it != null) {
            System.out.print((String) it.getContent() + ", ");
            it = it.next;
        }
        System.out.println();
        System.out.println("Removing first Element");
        myList.removeFirst();
        System.out.println("Size: " + myList.getSize());
        System.out.println("First Element: " + (String) myList.
           getFirst());
        System.out.println("Removing first Element");
        myList.removeFirst();
        System.out.println("Size: " + myList.getSize());
        System.out.println("List is empty: " + myList.isEmpty());
    }
}

public class DoubleLinkedListElem {

    private Object content;
    public DoubleLinkedListElem next;
    public DoubleLinkedListElem prev;

    public DoubleLinkedListElem(Object content) {
        this.content = content;
        this.next = null;
        this.prev = null;
    }

    public void setContent(Object content) {
        this.content = content;
    }

    public Object getContent() {
```

```
            return this.content;
        }
    }
```

Aufgabe 49 Implementierung eines sortierbaren Bücherregals mit java.util.Vector<E> (Lösungsvorschlag)

a)
```
public class Book implements Comparable<Book> {
    private String author;
    private Publisher publisher;
    private String title;

    public Book() {
      this.author = "";
      this.publisher = new Publisher();
      this.title = "";
    }

    public Book(String author, String title, Publisher publisher) {
      this.author = author;
      this.publisher = publisher;
      this.title = title;
    }

    public Book(String author, String title, String publisher, int
       edition, int year) {
      this.author = author;
      this.publisher = new Publisher(publisher, edition, year);
      this.title = title;
    }

    public int compareTo(Book book) {
      return this.title.compareTo(book.getTitle());
    }

    public String getAuthor() {
      return this.author;
    }

    public Publisher getPublisher() {
      return this.publisher;
    }

    public String getTitle() {
      return this.title;
    }
  }
```

b)
```
import java.util.Vector;

public class BookShelf {
  private Vector<Book> shelf;
```

```
    public BookShelf() {
      this.shelf = new Vector<Book>();
    }

    public Book find(String title) {
      for (Book book : this.shelf) {
        if (book.getTitle().equals(title)) {
          return book;
        }
      }

      return null;
    }

    public int getBookCount() {
      return this.shelf.size();
    }

    public void insert(Book book) {
      this.shelf.add(book);
    }

    public void print() {
      for (Book book : this.shelf) {
        System.out.println(book.getTitle());
      }
    }

    public Book remove(String title) {
      Book book = this.find(title);

      if (null != book) {
        this.shelf.remove(book);
      }

      return book;
    }
  }
```

c)
```
  public class BookShelf {
    ...
    public void sort() {
      Collections.sort(this.shelf);
    }
  }
```

d)
```
  public class Test {
    public static void main(String[] args) {
      System.out.println("----------");

      BookShelf shelf = new BookShelf();
```

```
shelf.print();

System.out.println("————————————");

shelf.insert(new Book("Autor1", "Titel1", new Publisher()));
shelf.print();

System.out.println("————————————");

shelf.insert(new Book("Autor6", "Titel6", new Publisher()));
shelf.print();

System.out.println("————————————");

shelf.insert(new Book("Autor5", "Titel5", new Publisher()));
shelf.print();

System.out.println("————————————");

shelf.insert(new Book("Autor4", "Titel4", new Publisher()));
shelf.print();

System.out.println("————————————");

shelf.insert(new Book("Autor7", "Titel7", new Publisher()));
shelf.print();

System.out.println("————————————");

shelf.insert(new Book("Autor2", "Titel2", new Publisher()));
shelf.print();

System.out.println("————————————");

shelf.insert(new Book("Autor3", "Titel3", new Publisher()));
shelf.print();

System.out.println("————————————");

shelf.sort();
shelf.print();

System.out.println("————————————");

Book book = shelf.remove("Titel3");
shelf.print();

System.out.println("————————————");

book = shelf.remove("Titel1");
shelf.print();

System.out.println("————————————");
```

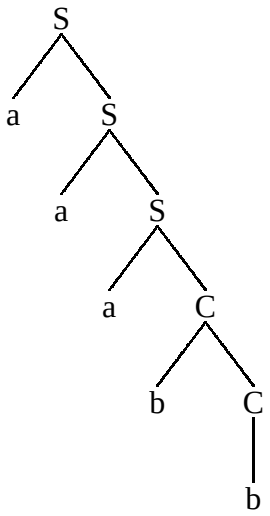
```
        book = shelf.remove("Titel5");
        shelf.print();

        System.out.println("----------");

        book = shelf.remove("Titel7");
        shelf.print();

        System.out.println("----------");

        book = shelf.remove("Titel2");
        shelf.print();

        System.out.println("----------");

        book = shelf.remove("Titel6");
        shelf.print();

        System.out.println("----------");

        book = shelf.remove("Titel4");
        shelf.print();

        System.out.println("----------");
    }
}
```

Aufgabe 50 Sprachen und Grammatiken (Lösungsvorschlag)

a)

$$
\begin{aligned}
N &= \{S, C\} \\
T &= \{a, b\} \\
P_1 &= S \rightarrow aS \\
P_2 &= S \rightarrow aC \\
P_3 &= C \rightarrow bC \\
P_4 &= C \rightarrow b
\end{aligned}
$$

b)



c)
```
let rec reverse list = match list with
    [] -> []
  | x::xs -> (reverse xs) @ [x];;

let rec check_terminal word = match word with
    'a'::[] -> true
  | 'b'::[] -> true
  | 'a'::xs -> check_terminal xs
  | 'b'::xs -> check_terminal xs
  | _ -> false;;

let check word =
  if (check_terminal word) then
    let rev_word = reverse word in
      let rec check_rec word = match word with
          'S'::'a'::xs -> check_rec ('S'::xs)
        | 'C'::'a'::xs -> check_rec ('S'::xs)
        | 'C'::'b'::xs -> check_rec ('C'::xs)
        | 'b'::xs -> check_rec ('C'::xs)
        | ['S'] -> true
        | _ -> false
      in check_rec rev_word
  else
    false;;
```

d)
```
check [];;
- : bool = false
check ['a';'a';'b'];;
- : bool = true
check ['b';'a';'a';'b'];;
- : bool = false
check ['a';'a'];;
- : bool = false
check ['a';'a';'a';'b';'b';'b';'b'];;
- : bool = true
check ['a'];;
- : bool = false
```

```
check ['b'];;
- : bool = false
check ['b';'C'];;
- : bool = false
check ['S'];;
- : bool = false
```