INFORMATIK I

Tutorium 12 — 17. Januar 2025

Name Last
Universität Münster

LaTeX-Vorlage von
Florian Sihler

Ringe und Verbundenheit – Verlobung?

Lösung 1: a) Die.java

```
public class Die {
    private int value;

    public Die() {toss();}

    public int getTossedValue() {return value;}
    public void toss() {
        value = (int)(Math.random() * 6) + 1;
    }
}
```

Lösung 1: b) Player.java

```
public class Player {
    private static int counter = 0;

    private int id;
    private String playerName;
    private Player nextPlayer; // default: null

    public Player(String playerName) {
        this.id = ++counter;
        this.playerName = playerName;
    }

    // getter and setter
}
```

Lösung 1: c) RingGame.java

Spielerliste erzeugen

- erster Spieler → `firstPlayer` und `playerToTheNext`
- ab zweitem Spieler: `playerToTheNext.setNextPlayer(player)`
- letzter Spieler: Verkettung mit `firstPlayer` (Kreisbildung)

Spieler entfernen

- falls `ringBearer == firstPlayer`: aktualisiere `firstPlayer = firstPlayer.getNextPlayer()`
- `while`-Schleife um Vorgänger von `ringBearer` zu finden (dessen Nachfolger muss also `ringBearer` sein!)
- dort neuen Nachfolger eintragen (`ringBearer` „*überspringen*")

Lösung 3

```
public class DoublyLinkedList {
    private DLLElement head, tail;

    public DoublyLinkedList() {
        head = new DLLElement();
        tail = new DLLElement();

        head.setNext(tail);
        tail.setPrevious(head);
    }
}
```

Lösung 3

```
public void insertFirst(Object object) {
```

Lösung 3

```
public void insertFirst(Object object) {
    DLLElement element = new DLLElement();
```

Lösung 3

```
public void insertFirst(Object object) {
    DLLElement element = new DLLElement();
    element.setObject(object);
```

Lösung 3

```
public void insertFirst(Object object) {
    DLLElement element = new DLLElement();
    element.setObject(object);

    element.setNext(head.getNext());
```

Lösung 3

```
public void insertFirst(Object object) {
    DLLElement element = new DLLElement();
    element.setObject(object);

    element.setNext(head.getNext());
    head.getNext().setPrevious(element);
```

Lösung 3

```
public void insertFirst(Object object) {
    DLLElement element = new DLLElement();
    element.setObject(object);

    element.setNext(head.getNext());
    head.getNext().setPrevious(element);
    head.setNext(element);
```

Lösung 3

```
public void insertFirst(Object object) {
    DLLElement element = new DLLElement();
    element.setObject(object);

    element.setNext(head.getNext());
    head.getNext().setPrevious(element);
    head.setNext(element);
    element.setPrevious(head);
}
```

Lösung 3

```
public void insertLast(Object object) {
    DLLElement element = new DLLElement();
    element.setObject(object);

    element.setPrevious(tail.getPrevious());
    tail.getPrevious().setNext(element);
    tail.setPrevious(element);
    element.setNext(tail);
}
```

Lösung 3

```
public void remove(Object object) {
```

Lösung 3

```
public void remove(Object object) {
    for (DLLElement current = head.getNext(); current != tail; ↩
    current = current.getNext()) { // iteriert durch Elemente
```

Lösung 3

```
public void remove(Object object) {
    for (DLLElement current = head.getNext(); current != tail; ↩
    current = current.getNext()) { // iteriert durch Elemente

            if (current.getObject().equals(object)) {
```

Lösung 3

```
public void remove(Object object) {
    for (DLLElement current = head.getNext(); current != tail; ↩
    current = current.getNext()) { // iteriert durch Elemente

            if (current.getObject().equals(object)) {
                current.getPrevious().setNext(current.getNext());
```

Lösung 3

```
public void remove(Object object) {
    for (DLLElement current = head.getNext(); current != tail; ↩
    current = current.getNext()) { // iteriert durch Elemente

            if (current.getObject().equals(object)) {
                current.getPrevious().setNext(current.getNext());
                current.getNext().setPrevious(current.getPrevious());
```

Lösung 3

```
public void remove(Object object) {
    for (DLLElement current = head.getNext(); current != tail; ↩
    current = current.getNext()) { // iteriert durch Elemente

            if (current.getObject().equals(object)) {
                current.getPrevious().setNext(current.getNext());
                current.getNext().setPrevious(current.getPrevious());
                return; // Abbruch der Suche
            }
    }
}
```

Lösung 3

```
@Override
public String toString() {
    String result = "[";
    for (DLLElement current = head.getNext(); current != tail; ↩
        current = current.getNext()) { // iteriert durch Elemente

            result += current.getObject(); // implizites toString()

            if (current != tail) result += ",␣";
    }
    result += "]";
    return result;
}
```

Das Finale rückt näher...

DIE NÄCHSTEN TUTORIEN

- Bereitet Fragen vor!

- **NUR VORAB GESTELLTE THEMEN UND FRAGEN WERDEN BEHANDELT!**
- Lieber einmal zu viel fragen – dafür bin ich da!

- Wir machen Übungsaufgaben aus Klausuren der letzten Jahre.

DIE NÄCHSTEN TUTORIEN

- Bereitet Fragen vor!

- **NUR VORAB GESTELLTE THEMEN UND FRAGEN WERDEN BEHANDELT!**
- Lieber einmal zu viel fragen – dafür bin ich da!
- Tutorium am `31.01.`!
- Wir machen Übungsaufgaben aus Klausuren der letzten Jahre.

Doubly linked lists are basically the programming equivalent of having trust issues:
always keeping tabs on both ends, just in case.

Name Last
Münster, 30. Januar 2025
name.last@uni-muenster.de