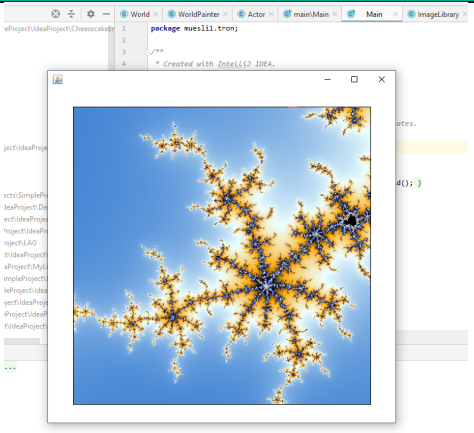
FOP Recap #10

Listen und verzeigerte Strukturen

TECHNISCHE UNIVERSITÄT DARMSTADT



Das steht heute auf dem Plan

TECHNISCHE UNIVERSITÄT DARMSTADT

Interface `List`
- Listen in Java
- Nutzung
- Listen vs. Arrays

Interface `Iterator`

Verzeigerte Datenstrukturen

Interface `List`

Listen in Java

TECHNISCHE UNIVERSITÄT DARMSTADT

- Listen in Java sind unterschiedlich zu Listen in Racket
- Der abstrakte Datentyp der Liste wird in Java durch das Interface `List` repräsentiert
- `List` ist generisch mit einem Typparameter E
- Wichtige Operationen, die in dem Interface festgelegt werden:
  - `size()`: Länge der Liste
  - `add(E element)`: Element am Ende hinzufügen
  - `remove(E element)`: Element aus Liste entfernen
  - `contains(E element)`: Prüfen ob Element in Liste ist
  - Viele weitere, siehe Dokumentation

Interface `List`

Nutzung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Im Package `java.util` finden sich einige Klassen, die `List` implementieren:
  - `ArrayList`
  - `LinkedList`
  - Eigene Implementation auch möglich
- Oft genutztes Muster:
  - Überall, wo Implementation egal ist (fast immer): Verwende statischen Typ `List<E>`
  - Wenn eine Liste erstellt werden muss, verwende Implementation aus Standardbibliothek

```
1  ArrayList<String> myList1 = new ArrayList<String>();
2  // Diamond Operator: String wird impliziert:
3  ArrayList<String> myList2 = new ArrayList<>();
4  // List ist Supertyp von ArrayList:
5  List<String> myList3 = new Arraylist<>();
```

Interface `List`

Listen vs. Arrays

TECHNISCHE UNIVERSITÄT DARMSTADT

Eigenschaft	Arrays	Listen
Gesamtgröße	fest	dynamisch
Implementation	immer gleich	dynamischer Typ kann variieren
Objektattribute	length	keins
Generics	zu vermeiden	Sehr wichtig
Hilfsmethoden	nur in `java.util.Arrays`	durch Interface festgelegt, `java.util.Collections`

Das steht heute auf dem Plan

TECHNISCHE UNIVERSITÄT DARMSTADT

Interface `List`

Interface `Iterator`

- Grundlagen
- Beispiele

Verzeigerte Datenstrukturen

Interface `Iterator`

Grundlagen

TECHNISCHE UNIVERSITÄT DARMSTADT

- Teil des Collections-Frameworks
- Weg, um über eine `Collection`, insbesondere auch `List` zu iterieren
- Generisch mit einem Typparameter E
- Repräsentiert ein Objekt, das über die zugrundeliegende `Collection` iteriert
- Das Interface definiert v.a. 2 Methoden:
  1. `boolean hasNext()`: liefert `true` genau dann, wenn es weitere Objekte im Iterator gibt
  2. `E next()`:
     liefert das nächste Element, wirft eine `NoSuchElementException`, wenn es keins mehr gibt

Interface Iterator

Beispiele — Gut

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
public <T> void foo(List<T> list){
    Iterator<T> iterator = list.iterator();
    while(iterator.hasNext()){
        T element = iterator.next();
        doSomething(element);
    }
}
```

Interface `Iterator`

Beispiele — Vorsicht: Häufiger Fehler!

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
public <T> void foo(List<T> list){
    // this is wrong!
    while(list.iterator().hasNext()){
        T element = list.iterator().next();
        doSomething(element);
    }
}
```

Das steht heute auf dem Plan

TECHNISCHE UNIVERSITÄT DARMSTADT

Interface `List`

Interface `Iterator`

Verzeigerte Datenstrukturen
- Grundideen
- Klasse `ListItem`
- Typische Form
- Mögliche interne Struktur
- Visuelle Repräsentation
- Zugriff auf Elemente
- Iterieren von Datenstrukturen
- Iterieren mit `ListItem`
- Beispiel-Implementierungen
- Typische Fehler

Verzeigerte Datenstrukturen

Grundideen

TECHNISCHE UNIVERSITÄT DARMSTADT

- Nutzen Objekte, die durch Referenzen (Zeiger/Pointer) aufeinander verweisen
- Starker Kontrast zu Arrays:
  - Haben veränderbare Länge
  - Effizientes Hinzufügen und Entfernen
  - Auch Nachteile: Können Endlos-Schleifen besitzen

Verzeigerte Datenstrukturen

Klasse `ListItem`

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
public class ListItem<T> {
    public T key;
    public ListItem<T> next;
}
```

Verzeigerte Datenstrukturen

Typische Form — `LinkedList`

TECHNISCHE UNIVERSITÄT DARMSTADT

- `add`: Hinzufügen von Objekten
- `remove`: Entfernen von Objekten
- `contains`: Prüfen, ob Objekte in Liste sind
- ...

Verzeigerte Datenstrukturen

Mögliche interne Struktur — `LinkedList`

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
public class ListItem<T> {
    public T key;
    public ListItem<T> next;
}
```

```java
public class LinkedList<T> {
    private ListItem<T> head;
    // private ListItem<T>tail;
}
```

Verzeigerte Datenstrukturen

Mögliche interne Struktur — Double Linked

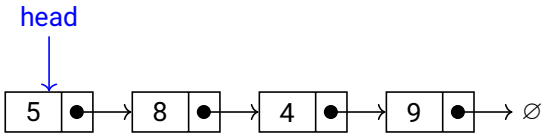
TECHNISCHE UNIVERSITÄT DARMSTADT

```java
public class ListItem<T> {
    public T key;
    public ListItem<T> next;
    public ListItem<T> prev;
}
```

Verzeigerte Datenstrukturen

Visuelle Repräsentation — `LinkedList<Integer>`

TECHNISCHE UNIVERSITÄT DARMSTADT



Verzeigerte Datenstrukturen
Zugriff auf Elemente

TECHNISCHE UNIVERSITÄT DARMSTADT

- Keinen (direkten) Zugriff durch einen Index
- Zugriff durch "Folgen der Zeiger"

Verzeigerte Datenstrukturen

Iterieren von Datenstrukturen — Arrays

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
public static <T> void print(T[] array) {
    System.out.println("Array:");
    for(T obj : array) {
        System.out.println("-> " + obj);
    }
}
```

```java
public static <T> void print(T[] array) {
    System.out.println("Array:");
    for(int index = 0; index < array.length; index++) {
        System.out.println("-> " + array[index]);
    }
}
```

Verzeigerte Datenstrukturen

Iterieren von Datenstrukturen — Listen

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
public static <T> void print(List<T> list) {
    System.out.println("List:");
    // possible for every class that implements Iterable
    for(T obj : list) {
        System.out.println("-> " + obj);
    }
}
```

```java
public static <T> void print(List<T> list) {
    System.out.println("List:");
    for(int index = 0; index < list.size(); index++) {
        System.out.println("-> " + list.get(index));
    }
}
```

Verzeigerte Datenstrukturen

Iterieren mit `ListItem`

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
public class ListItem<T> {
    public T key;
    public ListItem<T> next;
}
```

```java
public static <T> void print(ListItem<T> head) {
    System.out.println("List:");
    ListItem<T> pointer = head;
    while(pointer != null) {
        System.out.println("-> " + pointer.key);
        pointer = pointer.next;
    }
}
```

Verzeigerte Datenstrukturen

Iterieren mit `ListItem`

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
public class ListItem<T> {
    public T key;
    public ListItem<T> next;
}
```

```java
public static <T> void print(ListItem<T> head) {
    System.out.println("List:");
    for(ListItem<T> pointer = head; pointer != null;
                                pointer = pointer.next) {
        System.out.println("-> " + pointer.key);
    }
}
```

Verzeigerte Datenstrukturen

Beispiel-Implementierungen

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
public class ListItem<T> {
    public T key;
    public ListItem<T> next;
}
```

```java
public class LinkedList<T> {
    private ListItem<T> head;
    //....
}
```

Verzeigerte Datenstrukturen

Beispiel-Implementierungen — `addFirst`

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
public void addFirst(T item) {
    ListItem<T> newItem = new ListItem<T>();
    newItem.key = item;
    newItem.next = head;
    head = newItem;
}
```

Verzeigerte Datenstrukturen

Beispiel-Implementierungen — add

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
public void add(T item) {
    ListItem<T> newItem = new ListItem<T>();
    newItem.key = item;

    if(head == null) {
        head = newItem;
    }
    else {
        // ....
    }
}
```

Verzeigerte Datenstrukturen

Beispiel-Implementierungen — add

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
public void add(T item) {
    ....
    else {
        // Get last item
        ListItem<T> tail = head;
        while(tail.next != null) {
            tail = tail.next;
        }

        // Append item
        tail.next = newItem;
    }
}
```

Verzeigerte Datenstrukturen

Beispiel-Implementierungen — `remove`

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1  public boolean remove(T item) {
2      if(head != null && head.key.equals(item)) {
3          head = head.next;
4          return true;
5      }
6      else {
7          ....
8      }
9  }
```

Verzeigerte Datenstrukturen

Beispiel-Implementierungen — `remove`

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1  else {
2      ListItem<T> pointer = head;
3      // Search for previous node
4      while(pointer.next != null &&
5      pointer.next.key.equals(item) == false) {
6          pointer = pointer.next;
7      }
8      if(pointer.next == null) {
9          return false;
10     }
11     // Set pointers
12     pointer.next = pointer.next.next;
13     return true;
14 }
```

Verzeigerte Datenstrukturen

Beispiel-Implementierungen — `contains`

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
public boolean contains(T item) {
    ListItem<T> pointer = head;
    while(pointer != null) {
        if(pointer.key.equals(item)) {
            return true;
        }
        pointer = pointer.next;
    }
    return false;
}
```

Verzeigerte Datenstrukturen

Beispiel-Implementierungen — `get`

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
public T get(int index) {
    int position = 0;
    ListItem<T> pointer = head;
    while(pointer != null) {
        if(position == index) {
            return pointer.key;
        }
        pointer = pointer.next;
        position += 1;
    }
    throw new IndexOutOfBoundsException(index);
}
```

Verzeigerte Datenstrukturen

Beispiel-Implementierungen — `insert`

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
public boolean insert(int index, T item) {
    if(index < 0) { return false; }
    if(index == 0) {
        addFirst(item);
        return true;
    }
    else {
        // Create new item
        ListItem<T> newItem = new ListItem<T>();
        newItem.key = item;
        ....
    }
}
```

Verzeigerte Datenstrukturen

Beispiel-Implementierungen — `insert`

TECHNISCHE UNIVERSITÄT DARMSTADT

```
....
// Search for previous item
int position = 0;
ListItem<T> pointer = head;

while(pointer != null && position+1 < index) {
    pointer = pointer.next;
    position += 1;
}
....
```

Verzeigerte Datenstrukturen

Beispiel-Implementierungen — `insert`

TECHNISCHE UNIVERSITÄT DARMSTADT

```
....
if(pointer == null) {
    // Index was not reachable
    return false;
}
// Insert newItem between pointer and pointer.next
newItem.next = pointer.next;
pointer.next = newItem;
return true;
```

Verzeigerte Datenstrukturen

Typische Fehler

TECHNISCHE UNIVERSITÄT DARMSTADT

- Edge-Cases nicht beachtet, zum Beispiel:
  - `head` ist `null`
  - `remove` bezieht sich auf `head`
  - ....
- Zeiger werden vergessen, zum Beispiel:
  - `newItem.next` in `remove`
  - ....
- Erzeugen einer Schleife durch fehlerhaftes Setzen von Zeigern
  - Führt zu Endlos-Schleifen

Live-Coding!