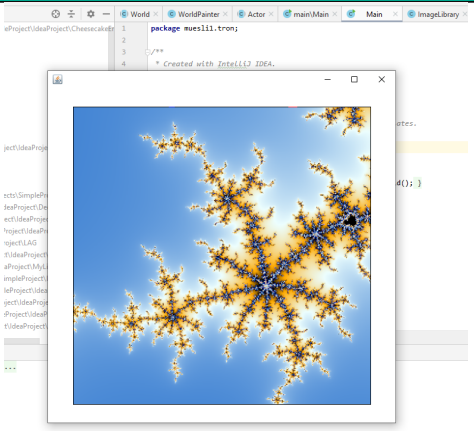
FOP Recap #6

Abstrakte Klassen



Themen für heute

TECHNISCHE UNIVERSITÄT DARMSTADT

Hinweis

Abstrakte Klassen

Statische Methoden und Attribute

Scopes, Bedingungsoperator

`String`, Casting

Das steht heute auf dem Plan

TECHNISCHE UNIVERSITÄT DARMSTADT

Hinweis
- Raumänderung
- Vererbung

Abstrakte Klassen

Statische Methoden und Attribute

Scopes, Bedingungsoperator

`String`, Casting

Hinweis
Raumänderung

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Nächste Woche in S402/101!**

Hinweis
Vererbung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Sehr wichtiger Klausurinhalt
- Fokus auf Interfaces und abstrakte Klassen

Das steht heute auf dem Plan

TECHNISCHE UNIVERSITÄT DARMSTADT

Hinweis

Abstrakte Klassen
- Syntax
- Wichtige Eigenschaften
- Zusammenhang

Statische Methoden und Attribute

Scopes, Bedingungsoperator

`String`, Casting

Abstrakte Klassen
Syntax

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1  public abstract class MyClassName {
2      ....
3  }
```

Syntax abstrakte Klasse:

Zugriffsmodifikatoren `abstract class` *Klassen-Name*
`extends` *Basis-Klassen-Name*
`implements` *Interface-Namen*

Abstrakte Klassen

Wichtige Eigenschaften

TECHNISCHE UNIVERSITÄT DARMSTADT

- Abstrakte Klassen können abstrakte Methoden deklarieren
- Es kann kein Objekt von einer abstrakten Klasse erstellt werden
- Abstrakte Klassen müssen nicht alle abstrakten Methoden ihrer Basis-Klassen implementieren
  - Weder die abstrakten Methoden ihrer abstrakten Basisklassen
  - Noch die Methoden der Interfaces, von denen sie direkt oder indirekt erben

Intermezzo: Live-Coding

Das steht heute auf dem Plan

TECHNISCHE UNIVERSITÄT DARMSTADT

Hinweis

Abstrakte Klassen

Statische Methoden und Attribute
- Allgemein
- Klassenmethoden
- Klassenattribute

Scopes, Bedingungsoperator

`String`, Casting

Statische Methoden und Attribute

Allgemein

TECHNISCHE UNIVERSITÄT DARMSTADT

- Methoden und Attribute gehören immer zu einer Klasse
- `static` Methoden und Attribute sind jedoch unabhängig von Objekten dieser Klasse
- Man kann auch ohne Objekte auf `static` Methoden und Attribute zugreifen
- Objektmethoden lassen sich jedoch weiterhin nur mit einem Objekt aufrufen

Statische Methoden und Attribute

Klassenmethoden

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
public class MyClassName {
    public static void test() {
        ....
    }
    public void omnom() {
        test();
    }
}
```

```java
MyClassName.test();
```

Statische Methoden und Attribute

Klassenattribute

TECHNISCHE UNIVERSITÄT DARMSTADT

- Haben programm-weit denselben Wert
- Unabhängig von jeglichen Objekten

Statische Methoden und Attribute

Klassenattribute

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
public class MyClassName {
    public static int myValueName = 5;
    public void omnom() {
        myValueName = -1;
    }
}
```

```java
MyClassName.myValueName = -24;
```

Das steht heute auf dem Plan

TECHNISCHE UNIVERSITÄT DARMSTADT

Hinweis

Abstrakte Klassen

Statische Methoden und Attribute

Scopes, Bedingungsoperator
- Klassen/Interfaces/Enums
- Attribute/Methoden
- Parameter/Lokale Variablen
- Shadowing
- Überblick
- Unterschied

Scopes

TECHNISCHE UNIVERSITÄT DARMSTADT

- Definieren in welchem Bereich Identifier sichtbar sind
- Möglicher Zugriff lässt sich dann (teils) durch Access Modifiers weiter einschränken

- Identifier sind zum Beispiel:
  - Klassen/Interfaces/Enums
  - Attribute
  - Variablen
  - Methoden
  - ...

Scopes

Klassen/Interfaces/Enums

TECHNISCHE UNIVERSITÄT DARMSTADT

- Sind bei uns normalerweise alle `public`
- Können dann überall importiert und genutzt werden

Scopes
Klassen/Interfaces/Enums

TECHNISCHE UNIVERSITÄT DARMSTADT

- Sind bei uns normalerweise alle `public`
- Können dann überall importiert und genutzt werden

- Ergänzung:
  - In jeder Datei gibt es genau eine Top-Level Klasse, die den Dateinamen trägt
  - Kann jedoch beliebig viele nicht `public` Klassen geben

Scopes
Attribute/Methoden

TECHNISCHE UNIVERSITÄT DARMSTADT

- Können je nach Access Modifier genutzt werden

Scopes

Parameter/Lokale Variablen

TECHNISCHE UNIVERSITÄT DARMSTADT

- Können nur in ihrer Methode genutzt werden
- Sind nur innerhalb ihres „geschweiften Klammerpaares“ zulässig

```
1  public String global = "global";
2  public void foo() {
3      String local = "local";
4      System.out.println(global); // -> "global"
5      System.out.println(local); // -> "local"
6  }
7  public void bar() {
8      System.out.println(global); // -> "global"
9      System.out.println(local); // -> Compiler-Error
10 }
```

Scopes
Shadowing

TECHNISCHE UNIVERSITÄT DARMSTADT

- Falls zwei Variable im Scope mit demselben Namen vorliegen
- Lässt sich dann mit `this` und `super` lösen

```
1  public class Auto {
2      public double maxSpeed;
3
4      public Auto(double maxSpeed) {
5          this.maxSpeed = maxSpeed;
6      }
7  }
```

Bedingungsoperator

Überblick

TECHNISCHE UNIVERSITÄT DARMSTADT

- Kurzform für if-else
- Nicht ganz indentisch, wegen Auswertungsreihenfolge

```
1  int number = -25;
```

```
1  int absValue;
2  if(number < 0) {
3      absValue = -number;
4  }
5  else {
6      absValue = number;
7  }
```

```
1  int absValue = number < 0? -number : number;
```

Bedingungsoperator

Unterschied

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
int[] array = new int[] {1, 2, 3};
int index = 1;
```

```java
array[index++] = index == 1? 2 : -5;
// -> array = {1, 2, -5}
```

```java
if(index == 1) {
    array[index++] = 2;
}
else {
    array[index++] = -5;
}
// -> array = {1, 2, 2}
```

Das steht heute auf dem Plan

TECHNISCHE UNIVERSITÄT DARMSTADT

Hinweis

Abstrakte Klassen

Statische Methoden und Attribute

Scopes, Bedingungsoperator

`String`, Casting
- Interner Aufbau
- `char`
- Beispiele
- `char`-Arithemtik
- Schon besser

`String`
Interner Aufbau

TECHNISCHE UNIVERSITÄT DARMSTADT

- Im Prinzip nur ein `char[]`
- Jeder `String` is unveränderbar

String
char

TECHNISCHE UNIVERSITÄT DARMSTADT

- Ist wie `int` ein primitiver Datentyp
- Belegt 2 Bytes Speicher statt wie ein `int` 4 Bytes
- Normale Verwendung:
  - Repräsentiert (im Normalfall) genau einen Buchstaben/Zeichen
  - Jedes Zeichen hat nach Unicode einen festen Zahlenwert zugeordnet

String
Beispiele

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1  String s = "ABC";
2  char c0 = s.charAt(0); // == 'A'
3  char c1 = s.charAt(1); // == 'B'
4  char c2 = s.charAt(2); // == 'C'
```

String
Beispiele

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
String s = "ABC";
char[] arr = s.toCharArray();
char c0 = arr[0]; // == 'A'
char c1 = arr[1]; // == 'B'
char c2 = arr[2]; // == 'C'
```

String
Beispiele

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
String s = "ABC";
String result = "";
result += s.charAt(2);
result += s.charAt(1);
result += s.charAt(0);
```

String
Beispiele

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1  String s = "ABC";
2  String result = "";
3  for(int i = s.length() - 1; i >= 0; i--) {
4      result += s.charAt(i);
5  }
6  System.out.println(result);
```

String
Beispiele

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
String s = "ABC";
String result = "";
for(int i = s.length() - 1; i >= 0; i--) {
    result += s.charAt(i);
}
System.out.println(result);
```

```
$ CBA
```

TECHNISCHE UNIVERSITÄT DARMSTADT

- In meinen Augen: Absolut schrecklich

```
1  char c = 'E';
2  char k = (char)(c + 1); // == 'F'
```

String
Schon besser

TECHNISCHE UNIVERSITÄT DARMSTADT

- Für Groß und Kleinschreibung!

```
1  char c = 'a';
2  char c2 = Character.toUpperCase(c); // == 'A'
3  boolean check = Character.isUpperCase(c2); // == true
4  char c3 = Character.toLowerCase(c2);
```

String
Richtig

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
String message = "hihi";
String upperOne = message.toUpperCase();
System.out.println(upperOne);

final Locale turkish = new Locale("tr");
String upperTwo = message.toUpperCase(turkish);
System.out.println(upperTwo);
```

String
Richtig

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
String message = "hihi";
String upperOne = message.toUpperCase();
System.out.println(upperOne);

final Locale turkish = new Locale("tr");
String upperTwo = message.toUpperCase(turkish);
System.out.println(upperTwo);
```

```
$ HIHI
$ HİHİ
```

Casting
Unterscheidung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Bei primitiven Datentypen
  - um einen Zahlenwert in einem anderen Typen zu speichern
  - passiert implizit oder explizit
- Bei Objekttypen
  - um den statischen Typen zu ändern
  - kann jedoch je nach dynamischen Typen fehlschlagen

Casting

Bei primitiven Datentypen

TECHNISCHE UNIVERSITÄT DARMSTADT

- Jeder Zahltyp hat einen Bereich an Zahlen, den er repräsentieren kann
- `long`, `int`, `char`, `short`, `byte` für Ganzzahlen
- `double`, `float` für Komma-Zahlen
- Hierbei erkennt man folgende Rangordnung der Bereiche:
- `double` > `float` > `long` > `int` > `char` > `short` > `byte`

Casting

Bei primitiven Datentypen — Widening Casting

TECHNISCHE UNIVERSITÄT DARMSTADT

- Passiert implizit, automatisch

```
1  int a = 5;
2  int b = 27;
3  long c = a - b;
```

Casting

Bei primitiven Datentypen — Widening Casting

TECHNISCHE UNIVERSITÄT DARMSTADT

- Passiert implizit, automatisch

```
1  char a = 5;
2  int b = 27;
3  long c = a - b;
```

Casting

Bei primitiven Datentypen — Narrowing Casting

TECHNISCHE UNIVERSITÄT DARMSTADT

- Muss explizit angegeben werden, da Verlust von Präzision stattfinden kann!

```
1  int a = 288;
2  char c = a; // ERROR!
3  char c2 = (char) a; // OK!
4
5  char c3 = a + 5; // ERROR!
6  char c4 = ((char) a) + 5; // ERROR!
7  char c5 = (char) (a + 5); // OK!
```

Casting
Bei Objekttypen

TECHNISCHE UNIVERSITÄT DARMSTADT

- Funktioniert nur ohne Fehler bei passenden dynamischen Typen!

```
1  class A { .... }
2  class B extends A { .... }
3  class C extends B { .... }
4
5  A a = ....;
6  C castedC = (C) a; // Nur möglich, wenn a dynamischen Typen von C
   ↪  oder Subtypen hat
```

Live-Coding!