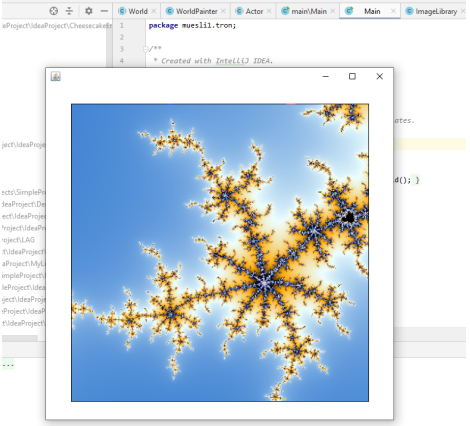
FOP Recap #2

Objekte, Attribute, Methoden



Hey, los gehts!

Heute aufn Menü

TECHNISCHE UNIVERSITÄT DARMSTADT

Organisation, Hinweise

`break`, `continue`, `if`, `else`

`if` und `else`

Import-Anweisungen

Objekte und Typen

Attribute vs lokale Variable

Enumeration

Das steht heute auf dem Plan

TECHNISCHE UNIVERSITÄT DARMSTADT

Organisation, Hinweise
- Was ist das Recap?
- Wann und wo findet das Recap statt?

`break`, `continue`, `if`, `else`

`if` und `else`

Import-Anweisungen

Objekte und Typen

Attribute vs lokale Variable

Enumeration

Organisation
Was ist das Recap?

TECHNISCHE UNIVERSITÄT DARMSTADT

- Das Recap ist ein zusätzliches und freiwilliges Angebot
- Keine Anwesenheitspflicht
- Keine Beantwortung von Fragen zu Hausübungen
- Stattdessen:
  - Wiederholung des Stoffes
  - Erklärung anhand von Beispielen
  - Rückfragen und Diskussion **erwünscht**, auch über Discord!

Organisation

Wann und wo findet das Recap statt?

TECHNISCHE UNIVERSITÄT DARMSTADT

- Jeden Mittwoch um 15:30 Uhr bis ca. 17:00 Uhr
- Verfügbar:
  - In Präsenz (regulär in S1|03 226)
  - per Live-Stream auf YouTube
  - Aufzeichnung nachträglich als YouTube-Video verfügbar

Das steht heute auf dem Plan

TECHNISCHE UNIVERSITÄT DARMSTADT

- Organisation, Hinweise
- `break`, `continue`, `if`, `else`
  - `break` und `continue`
- `if` und `else`
- Import-Anweisungen
- Objekte und Typen
- Attribute vs lokale Variable
- Enumeration

Ergänzung Schleifen

`break` und `continue`

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
for(int i = 0; i < 3; i++) {
    System.out.println("i = " + i + "!");
}
```

Ergänzung Schleifen

`break` und `continue`

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
for(int i = 0; i < 3; i++) {
    System.out.println("i = " + i + "!");
}
```

```
$ i = 0!
$ i = 1!
$ i = 2!
```

Ergänzung Schleifen

`break` und `continue`

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
for(int i = 0; i < 3; i++) {
    if(i == 1) {
        continue;
    }
    System.out.println("i = " + i + "!");
}
```

Ergänzung Schleifen

`break` und `continue`

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
for(int i = 0; i < 3; i++) {
    if(i == 1) {
        continue;
    }
    System.out.println("i = " + i + "!");
}
```

```
$ i = 0!
$ i = 2!
```

Ergänzung Schleifen

`break` und `continue`

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
for(int i = 0; i < 3; i++) {
    if(i == 1) {
        break;
    }
    System.out.println("i = " + i + "!");
}
```

Ergänzung Schleifen

`break` und `continue`

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
for(int i = 0; i < 3; i++) {
    if(i == 1) {
        break;
    }
    System.out.println("i = " + i + "!");
}
```

```
$ i = 0!
```

Das steht heute auf dem Plan

TECHNISCHE UNIVERSITÄT DARMSTADT

Organisation, Hinweise

`break`, `continue`, `if`, `else`

`if` und `else`

- Mit && sowie | |

Import-Anweisungen

Objekte und Typen

Attribute vs lokale Variable

Enumeration

if und else

TECHNISCHE UNIVERSITÄT DARMSTADT

```
if(condition1) {
    //...
}
else if(condition2) {
    //...
}
else {
    //...
}
```

```
if(condition1) {
    //...
}
else {
    if(condition2) {
        //...
    }
    else {
        //...
    }
}
```

if und else

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
boolean isMoving = true;
boolean lowOnGas = false;

if(isMoving == false) {
    car.startMotor();
}
else if(lowOnGas == false) {
    car.drive();
}
else {
    car.stopMotor();
}
```

```java
boolean isMoving = true;
boolean lowOnGas = false;

if(isMoving == false) {
    car.startMotor();
}
else {
    if(lowOnGas == false) {
        car.drive();
    }
    else {
        car.stopMotor();
    }
}
```

if und else

Mit && sowie ||

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1  int x = 25;
2  int y = 7;
3  int c = 12;
4  Robot r = new Robot(4, 0, DOWN, 12);
5  if((r.getX() == x && r.getY() == y) ||
6                  r.getNumberOfCoins() == c) {
7      r.move();
8  }
9  else {
10     r.turnOff();
11 }
```

`if` und `else`

Mit && sowie ||

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1  int x = 25;
2  int y = 7;
3  int c = 12;
4  Robot r = new Robot(4, 0, DOWN, 12);
5  if(r.getX() >= x || r.hasAnyCoins() || -c == x) {
6      r.move();
7  }
8  else {
9      r.turnOff();
10 }
```

Das steht heute auf dem Plan

TECHNISCHE UNIVERSITÄT DARMSTADT

Organisation, Hinweise

`break`, `continue`, `if`, `else`

`if` und `else`

Import-Anweisungen

Objekte und Typen

Attribute vs lokale Variable

Enumeration

Import-Anweisungen

TECHNISCHE UNIVERSITÄT DARMSTADT

- Beispiel package-Struktur:

- fopbot
  - Robot
  - Direction
  - World
- other
  - MyClass

MyClass.java

```java
package other;

import fopbot.Robot;

public class MyClass {
    public Robot myRobot;

    public void testCall() {
        myRobot.move();
    }
}
```

Das steht heute auf dem Plan

TECHNISCHE UNIVERSITÄT DARMSTADT

- Organisation, Hinweise
- `break`, `continue`, `if`, `else`
- `if` und `else`
- Import-Anweisungen
- Objekte und Typen
  - Typen
  - Objekte
  - Konstruktoren
  - Methodenaufruf
  - Spezialwert `null`
  - Pass-by-value, Pass-by-reference
- Attribute vs lokale Variable
- Enumeration

Objekte und Typen
Typen

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
public class CoolClassName {
    public int importantNumber;

    public void coolMethodName() {
        importantNumber += 1;
        boolean b = false;
        LemonTree tree = new LemonTree();
    }
}
```

Objekte und Typen
Typen

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1  public int importantNumber;
2  ....
3  boolean b = ....;
4  LemonTree tree = ....;
```

- Unterteilung:
  - Primitive Datentypen: `int`, `bool`, usw.
  - Objekt-Datentypen: `LemonTree`, `Robot`, usw.

TECHNISCHE UNIVERSITÄT DARMSTADT

Objekt:

Ein **Objekt** ist eine Instanz einer **Klasse**. Es hat seinen eigenen Speicher und die gespeicherten Werte in seinen Attributen können unabhängig von anderen Instanzen geändert werden.

Auf Objekten können **Methoden** aufgerufen werden. Dies geht auf primitiven Datentypen nicht. Welche **Methoden** verfügbar sind, hängt von der **Klasse** ab.

Um ein neues Objekt zu erstellen, muss der **new**-Operator verwendet werden. Hierbei wird der jeweilige **Konstruktor** auf dem neu erstellten Objekt implizit aufgerufen.

Nicht mehr genutzte Objekte werden von Java (irgendwann) automatisch gelöscht [Stichwort Garbage-Collection]

Objekte und Typen
Konstruktoren

TECHNISCHE UNIVERSITÄT DARMSTADT

LemonTree.java

```java
package mypackage;

public class LemonTree {
    public int numberOfLemons;
    public boolean fullyGrown;
    public String name = "NoName";

    public LemonTree(int lemons) {
        numberOfLemons = lemons;
        fullyGrown = false;
    }
}
```

Objekte und Typen
Konstruktoren

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
public LemonTree(int lemons) {
    numberOfLemons = lemons;
    fullyGrown = false;
}
```

Syntax Konstruktor:

Zugriffsmodifikatoren Klassen-Name (*Parameter1, Parameter2, ...*)

- Konstruktor legt fest, mit welchen Parametern ein neues Objekt von der Klasse erstellt werden kann und wie diese das neue Objekt beeinflussen

Objekte und Typen

Konstruktoren

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
public LemonTree(int lemons) {
    numberOfLemons = lemons;
    fullyGrown = false;
}
```

```java
LemonTree myTree = new LemonTree(25);
System.out.println(myTree.numberOfLemons);
System.out.println(myTree.fullyGrown);
```

Objekte und Typen

Konstruktoren

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
public LemonTree(int lemons) {
    numberOfLemons = lemons;
    fullyGrown = false;
}
```

```java
LemonTree myTree = new LemonTree(25);
System.out.println(myTree.numberOfLemons);
System.out.println(myTree.fullyGrown);
```

```
$ 25
$ false
```

TECHNISCHE UNIVERSITÄT DARMSTADT

LemonTree.java

```java
public class LemonTree {
    public int numberOfLemons;
    public boolean fullyGrown;
    public String name = "NoName";
}
```

Objekte und Typen

Konstruktoren — Standard-Konstruktor

TECHNISCHE UNIVERSITÄT DARMSTADT

LemonTree.java

```java
public class LemonTree {
    public int numberOfLemons;
    public boolean fullyGrown;
    public String name = "NoName";
}
```

```java
LemondTree myTree = new LemonTree(?????);
```

Objekte und Typen

Konstruktoren — Standard-Konstruktor

TECHNISCHE UNIVERSITÄT DARMSTADT

LemonTree.java

```java
public class LemonTree {
    .....

    public LemonTree() {
        // Automatisch generiert.
    }
}
```

Objekte und Typen

Konstruktoren — Standard-Konstruktor

TECHNISCHE UNIVERSITÄT DARMSTADT

LemonTree.java

```java
public class LemonTree {
    .....

    public LemonTree() {
        // Automatisch generiert.
    }
}
```

```java
LemondTree myTree = new LemonTree();
```

Objekte und Typen

Konstruktoren — Attribute

TECHNISCHE UNIVERSITÄT DARMSTADT

LemonTree.java

```java
package mypackage;

public class LemonTree {
    public int numberOfLemons;
    public boolean fullyGrown;
    public String name = "NoName";

    public LemonTree(int lemons) {
        numberOfLemons = lemons;
        fullyGrown = false;
    }
}
```

Objekte und Typen

Konstruktoren — Attribute

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1  LemonTree a = new LemonTree(5);
2  a.numberOfLemons = -2;
3  LemonTree b = new LemonTree(6);
4  b.fullyGrown = true;
5
6  System.out.println(a.fullyGrown);
7  System.out.println(b.numberOfLemons);
```

Objekte und Typen

Konstruktoren — Attribute

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
LemonTree a = new LemonTree(5);
a.numberOfLemons = -2;
LemonTree b = new LemonTree(6);
b.fullyGrown = true;

System.out.println(a.fullyGrown);
System.out.println(b.numberOfLemons);
```

```
$ false
$ 6
```

Objekte und Typen
Methodenaufruf

TECHNISCHE UNIVERSITÄT DARMSTADT

LemonTree.java

```java
public void grow() {
    fullyGrown = true;
}
public void water() {
    grow(); // Ohne Objekt??
    numberOfLemons += 1;
}
```

```java
LemonTree tree = new LemonTree(0);
tree.grow();
tree.water();
```

Objekte und Typen

Spezialwert `null`

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
LemonTree tree = new LemonTree(0);
LemonTree other = new LemonTree(5);
LemonTree third = null;

third.water(); // NullPointerException
third = new LemonTree(25);
third.grow();
```

Objekte und Typen

Pass-by-value, Pass-by-reference

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
public void doMorningRoutine(LemonTree tree) {
    tree.water();
}
```

```java
LemonTree a = new LemonTree(0);
doMorningRoutine(a);
System.out.println(a.numberOfLemons);
```

Objekte und Typen

Pass-by-value, Pass-by-reference

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
public void doMorningRoutine(LemonTree tree) {
    tree.water();
}
```

```java
LemonTree a = new LemonTree(0);
doMorningRoutine(a);
System.out.println(a.numberOfLemons);
```

```
$ 1
```

Objekte und Typen

Pass-by-value, Pass-by-reference

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
public void doMorningRoutine(int num) {
    num += 1;
}
```

```java
int a = 0;
doMorningRoutine(a);
System.out.println(a);
```

Objekte und Typen

Pass-by-value, Pass-by-reference

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
public void doMorningRoutine(int num) {
    num += 1;
}
```

```java
int a = 0;
doMorningRoutine(a);
System.out.println(a);
```

```
$ 0
```

Objekte und Typen

Pass-by-value, Pass-by-reference

TECHNISCHE UNIVERSITÄT DARMSTADT

- Ein Objekt belegt einen bestimmten Speicherplatz
- Mehrere Variablen können auf dasselbe Objekt verweisen
- Objekte werden per Referenz übergeben! Es wird keine Kopie erstellt.

Objekte und Typen

Pass-by-value, Pass-by-reference

TECHNISCHE UNIVERSITÄT DARMSTADT

```
1  LemonTree a = new LemonTree(0);
2  LemonTree b = a;
3  LemonTree c = b;
4
5  a.grow();
6  System.out.println(a.fullyGrown);
7  System.out.println(b.fullyGrown);
8  System.out.println(c.fullyGrown);
```

Objekte und Typen

Pass-by-value, Pass-by-reference

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
LemonTree a = new LemonTree(0);
LemonTree b = a;
LemonTree c = b;

a.grow();
System.out.println(a.fullyGrown);
System.out.println(b.fullyGrown);
System.out.println(c.fullyGrown);
```

```
$ true
$ true
$ true
```

Das steht heute auf dem Plan

TECHNISCHE UNIVERSITÄT DARMSTADT

Organisation, Hinweise

`break`, `continue`, `if`, `else`

`if` und `else`

Import-Anweisungen

Objekte und Typen

Attribute vs lokale Variable

- Initialisierung

Enumeration

Attribute vs lokale Variable

Initialisierung

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
public class DataContainer {
    public int data; // Attribut

    public void printData() {
        System.out.println(data);
    }
}
```

```java
public void abc() {
    int data; // Lokale Variable
    System.out.println(data);
}
```

Attribute vs lokale Variable

Initialisierung — Standardwerte

TECHNISCHE UNIVERSITÄT DARMSTADT

Name	Typ	Attribut Standardwert
`boolean`	Wahr/Falsch	`false`
`int`	Ganze Zahl	`0`
`double`	Gleitkommazahl	`0.0`
`String`	Zeichenkette	`null`
`LemonTree`	Eigene Klasse	`null`

Attribute vs lokale Variable

Initialisierung — Uninitialized local variable

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
int data; // Lokale Variable

if(....) {
    data = 5;
}

System.out.println(data); // ERROR!
```

Attribute vs lokale Variable

Initialisierung — Uninitialized local variable

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
int data; // Lokale Variable

while(....) {
    data = 5;
}

System.out.println(data); // ERROR!
```

Attribute vs lokale Variable

Initialisierung — Uninitialized local variable

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
int data = 0; // Lokale Variable

while(....) {
    data = 5;
}

System.out.println(data); // OK!
```

Das steht heute auf dem Plan

TECHNISCHE UNIVERSITÄT DARMSTADT

Organisation, Hinweise

`break`, `continue`, `if`, `else`

`if` und `else`

Import-Anweisungen

Objekte und Typen

Attribute vs lokale Variable

Enumeration

- Beispiel aus der Welt der FOPBot-Roboter
- Notation
- Methoden für Enumerationen
- Rückgabe

Enumeration

Beispiel aus der Welt der FOPBot-Roboter

TECHNISCHE UNIVERSITÄT DARMSTADT

Enumeration `Direction`

- `Direction.UP`
- `Direction.RIGHT`
- `Direction.DOWN`
- `Direction.LEFT`

```
1  public enum Direction {
2      UP,
3      RIGHT,
4      DOWN,
5      LEFT
6  }
```

Enumeration

Beispiel aus der Welt der FOPBot-Roboter

TECHNISCHE UNIVERSITÄT DARMSTADT

- Aufzählung von Konstanten
- können wie andere Konstanten verwendet werden
  z.B. `true` oder 1 – ohne jeweilige Operationen

- Bedeutung der Konstanten wird von Entwickler festgelegt

Operationen

Enumeration
Operationen

TECHNISCHE UNIVERSITÄT DARMSTADT

==

```
1  Robot robot = new Robot(0, 0, UP, 42);
2  Direction direction = robot.getDirection();
3  ...
4  if (direction == UP) {
5      ...
6  }
7  ...
8  if (direction.equals(RIGHT)) {
9      ...
10 }
```

Enumeration
Abkürzung

TECHNISCHE UNIVERSITÄT DARMSTADT

Anstatt z.B. `Direction.UP` kann auch nur UP geschrieben werden!

- einzelne Konstante
  ```
  import static fopbot.Direction.UP;
  ```
- alle Konstanten einer Enumeration
  ```
  import static fopbot.Direction.*;
  ```

Enumeration
Notation

TECHNISCHE UNIVERSITÄT DARMSTADT

Syntax Enumeration:

Zugriffsmodifikatoren Enumeration-Name { *Konstante1, Konstante2, ...* }

Best Case

- Namen für Enumerationen → Pascal Case – wie bei Namen für Klassen
  z.B. `MyDirection`
- Namen für Konstanten → Upper Case
  z.B. `MY_UP`, `MY_RIGHT`, `MY_DOWN` und `MY_LEFT`

TECHNISCHE UNIVERSITÄT DARMSTADT

`ordinal()` für Konstante

- liefert Position der Konstante in Enumeration

Beispiel für `Direction`

- `Direction.UP.ordinal()` $\rightarrow 0$
- `Direction.RIGHT.ordinal()` $\rightarrow 1$
- `Direction.DOWN.ordinal()` $\rightarrow 2$
- `Direction.LEFT.ordinal()` $\rightarrow 3$

TECHNISCHE UNIVERSITÄT DARMSTADT

`values()`

- liefert Array mit Konstanten der Enumeration

Beispiel für `Direction`

- `Direction.values()` → `[UP, RIGHT, DOWN, LEFT]`
- `Direction.values()[0]` → `UP`

TECHNISCHE UNIVERSITÄT DARMSTADT

`valueOf(String)`

- liefert Konstante mit gegebenem Namen

Beispiel für `Direction`

- `Direction.valueOf("UP")` → UP (Konstante)

Methoden

Rückgabe

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
public boolean checkNumber(int a) {
    return a > 5;
}
```

```java
boolean resultA = checkNumber(4);
boolean resultB = checkNumber(6);
System.out.println(resultA + " & " + resultB);
```

Methoden
Rückgabe

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
public boolean checkNumber(int a) {
    return a > 5;
}
```

```java
boolean resultA = checkNumber(4);
boolean resultB = checkNumber(6);
System.out.println(resultA + " & " + resultB);
```

```
$ false & true
```

Methoden
Rückgabe

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
public LemonTree createATree(int a) {
    return new LemonTree(a + 42);
}
```

```java
LemonTree resultA = createATree(4);
LemonTree resultB = createATree(6);
```

Methoden
Rückgabe

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
public LemonTree createATree(int a) {
    LemonTree localVar = new LemonTree(a + 42);
    localVar.grow();
    localVar.fullyGrown = false;

    return localVar;
}
```

Methoden
Rückgabe

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
public int runMe(int a) {
    if(a > 5) {
        return 3;
    }
    else {
        a = 2;
    }
    // ERROR! Return benötigt!
}
```

Methoden
Rückgabe

TECHNISCHE UNIVERSITÄT DARMSTADT

```java
public void growIt(LemonTree tree) {
    if(tree == null) {
        return; // Beendet Methode direkt
    }
    tree.grow();
    // Kein return benötigt! Nur optional
}
```

Live-Coding!