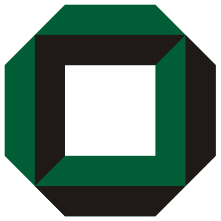
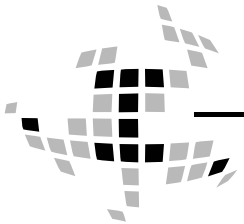


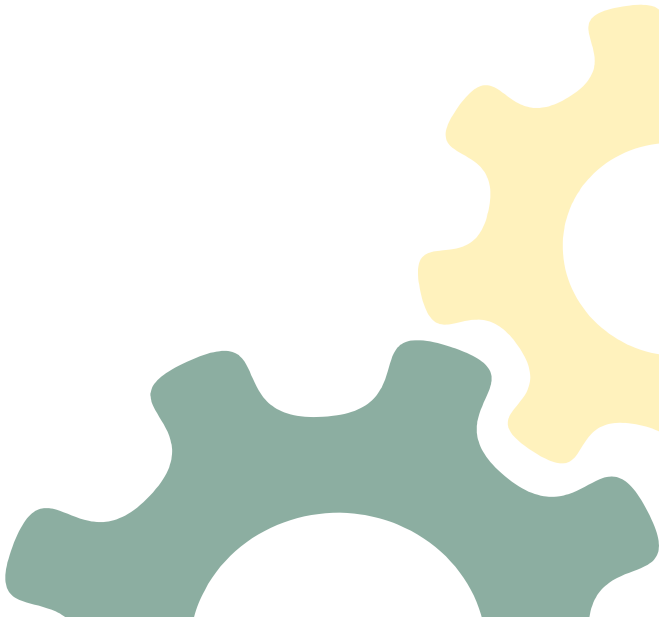
Universität Karlsruhe (TH)
Forschungsuniversität · gegründet 1825

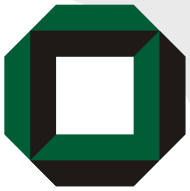
Softwaretechnik 1
Übung 6

16.07.2009



Fakultät für **Informatik**
Lehrstuhl für Programmiersysteme



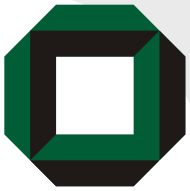


Aufgabe 1a)
Gegebene Methode

```
public static int getNumberOfDays(int month, int year) {
  int days = -1;
  if (year >= 1) {
    if (month == 1 || month == 3 || month == 5 || month == 7 ||
        month == 8 || month == 10 || month == 12) {
      days = 31;
    } else if (month == 4 || month == 6 || month == 9 ||
        month == 11) {
      days = 30;
    } else if (month == 2) {
      if (isLeapYear(year)) {
        days = 29;
      } else {
        days = 28;
      }
    }
  }
  return days;
}
```

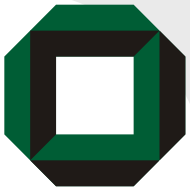
Aufgabe 1a)
Umwandlung in Zw. sprache

```
public static int getNumberOfDays(int month, int year) {
  int days = -1;
  if (year >= 1) {
    if (month == 1 || month == 3 || month == 5 || month == 7 ||
        month == 8 || month == 10 || month == 12) {
      days = 31;
    } else if (month == 4 || month == 6 || month == 9 ||
        month == 11) {
      days = 30;
    } else if (month == 2) {
      if (isLeapYear(year)) {
        days = 29;
      } else {
        days = 28;
      }
    }
  }
  return days;
}
```



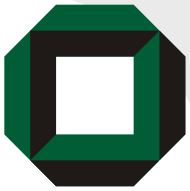
Aufgabe 1a) Umwandlung in Zw. sprache

```
010 int days = -1;
020 if (year >= 1) {
030   if (month == 1 || […] ) {
040 days = 31;
050   } else if (month == 4 || […]) {
060 days = 30;
070   } else if (month == 2) {
080     if (isLeapYear(year)) {
090 days = 29;
100     } else {
110 days = 28;
120     }
130   }
140 }
150 return days;
```



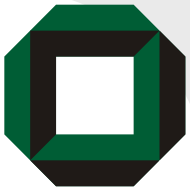
Aufgabe 1a) Umwandlung in Zw. sprache

```
010 int days = -1;
020 if (year >= 1) {
030   if (month == 1 || […] ) {
040 days = 31;
050   } else if (month == 4 || […]) {
060 days = 30;
070   } else if (month == 2) {
080     if (isLeapYear(year)) {
090 days = 29;
100     } else {
110 days = 28;
120     }
130   }
140 }
150 return days;
```



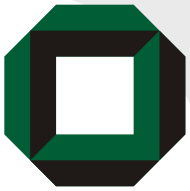
Aufgabe 1a) Umwandlung in Zw. sprache

```
010 int days = -1;
020 if not (year >= 1) goto 150;
030   if (month == 1 || […] ) {
040 days = 31;
050   } else if (month == 4 || […]) {
060 days = 30;
070   } else if (month == 2) {
080     if (isLeapYear(year)) {
090 days = 29;
100     } else {
110 days = 28;
120     }
130   }
150 return days;
```



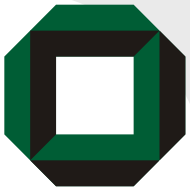
Aufgabe 1a)
Umwandlung in Zw. sprache

```
010 int days = -1;
020 if not (year >= 1) goto 150;
030   if (month == 1 || […] ) {
040 days = 31;
050   } else if (month == 4 || […]) {
060 days = 30;
070   } else if (month == 2) {
080     if (isLeapYear(year)) {
090 days = 29;
100     } else {
110 days = 28;
120     }
130   }
150 return days;
```



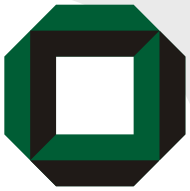
Aufgabe 1a) Umwandlung in Zw. sprache

```
010 int days = -1;
020 if not (year >= 1) goto 150;
030 if not (month == 1 || […] ) goto 50;
040 days = 31;
045 goto 150;
050 if not (month == 4 || […]) goto 70;
060 days = 30;
065 goto 140;
070 if not (month == 2) goto 150;
080     if (isLeapYear(year)) {
090 days = 29;
100     } else {
110 days = 28;
120     }
150 return days;
```



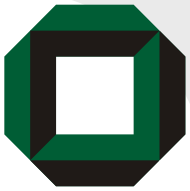
Aufgabe 1a)
Umwandlung in Zw. sprache

```
010 int days = -1;
020 if not (year >= 1) goto 150;
030 if not (month == 1 || […] ) goto 50;
040 days = 31;
045 goto 150;
050 if not (month == 4 || […]) goto 70;
060 days = 30;
065 goto 150;
070 if not (month == 2) goto 150;
080     if (isLeapYear(year)) {
090 days = 29;
100     } else {
110 days = 28;
120     }
150 return days;
```



Aufgabe 1a) Umwandlung in Zw. sprache

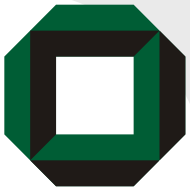
```
010 int days = -1;
020 if not (year >= 1) goto 150;
030 if not (month == 1 || […] ) goto 50;
040 days = 31;
045 goto 150;
050 if not (month == 4 || […]) goto 70;
060 days = 30;
065 goto 150;
070 if not (month == 2) goto 150;
080 if not (isLeapYear(year)) goto 110;
090 days = 29;
100 goto 150;
110 days = 28;
150 return days;
```



Aufgabe 1a) Zwischensprache

```
010 int days = -1;
020 if not (year >= 1) goto 150;
030 if not (month == 1 || […] ) goto 50;
040 days = 31;
045 goto 150;
050 if not (month == 4 || […]) goto 70;
060 days = 30;
065 goto 150;
070 if not (month == 2) goto 150;
080 if not (isLeapYear(year)) goto 110;
090 days = 29;
100 goto 150;
110 days = 28;
150 return days;
```

1. Transformiere in Zwischensprache
2. Fasse alle Folgen, die mit einem Sprung enden, zu je einem Grundblock zusammen

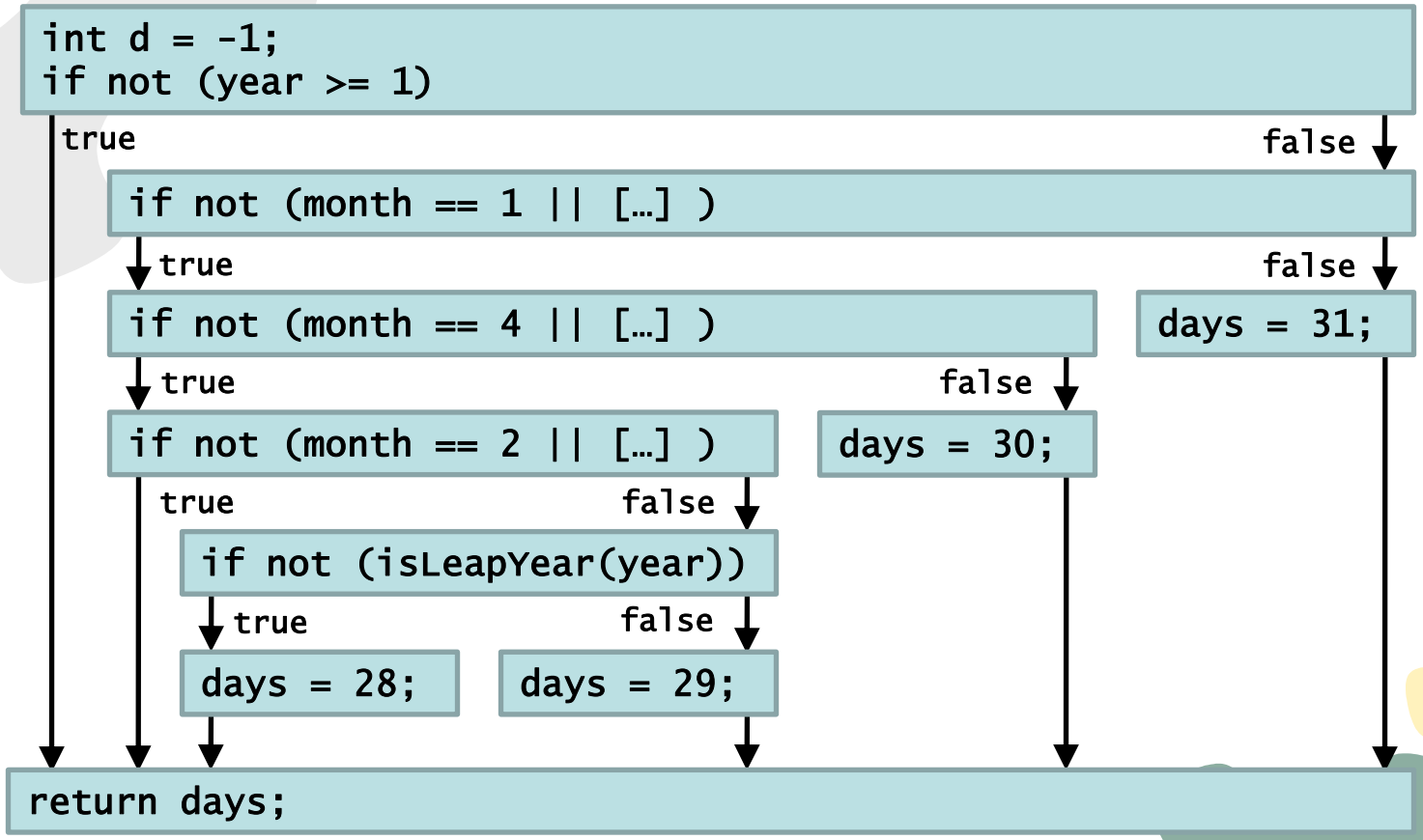


Aufgabe 1a) Grundblöcke

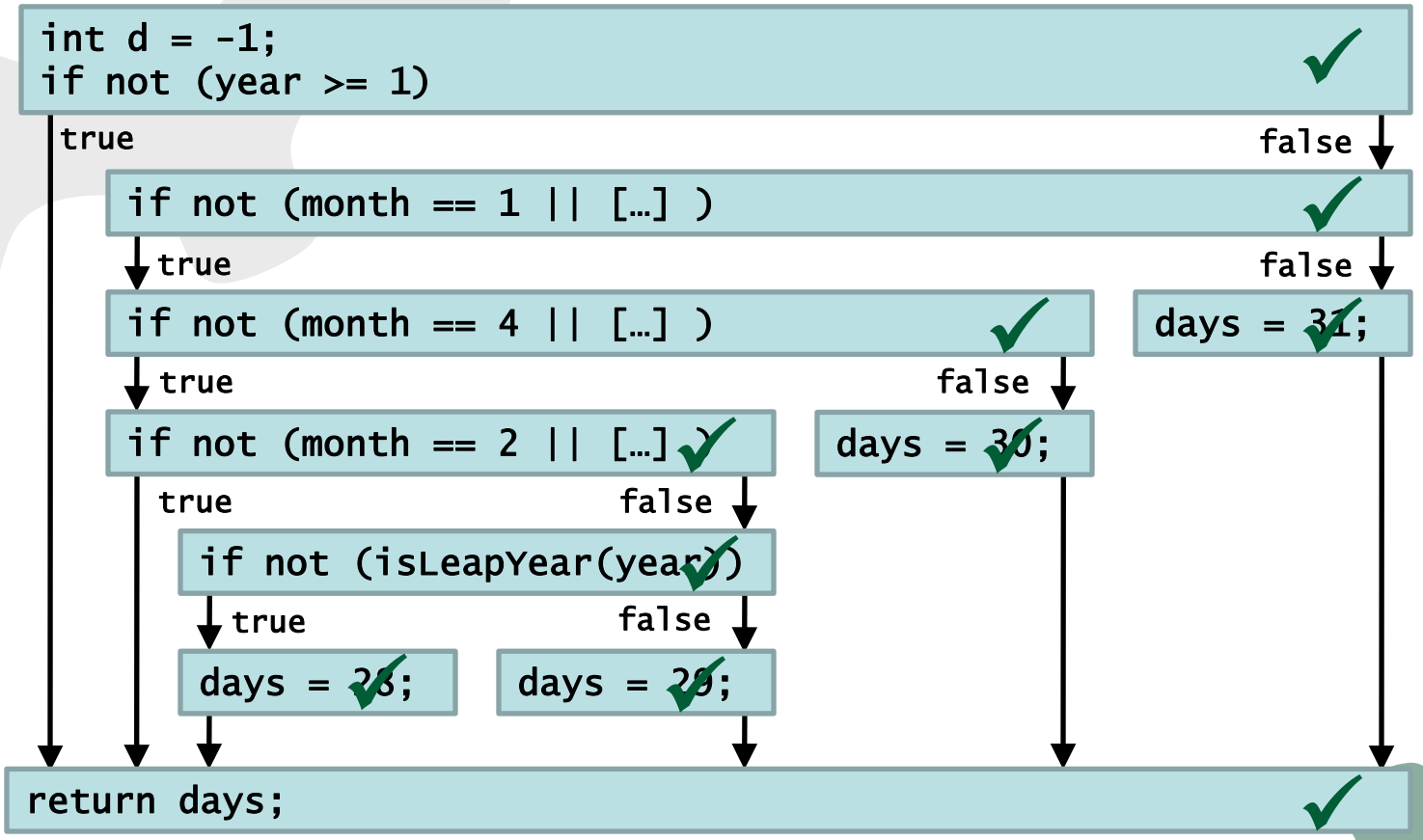
```
010 int days = -1;
020 if not (year >= 1) goto 150;
030 if not (month == 1 || […] ) goto 50;
040 days = 31;
045 goto 150;
050 if not (month == 4 || […]) goto 70;
060 days = 30;
065 goto 150;
070 if not (month == 2) goto 150;
080 if not (isLeapYear(year)) goto 110;
090 days = 29;
100 goto 150;
110 days = 28;
150 return days;
```

3. Prüfe, ob Eintritt nur am Anfang
4. Teile ggf. auf

Aufgabe 1a) Kontrollflussgraph

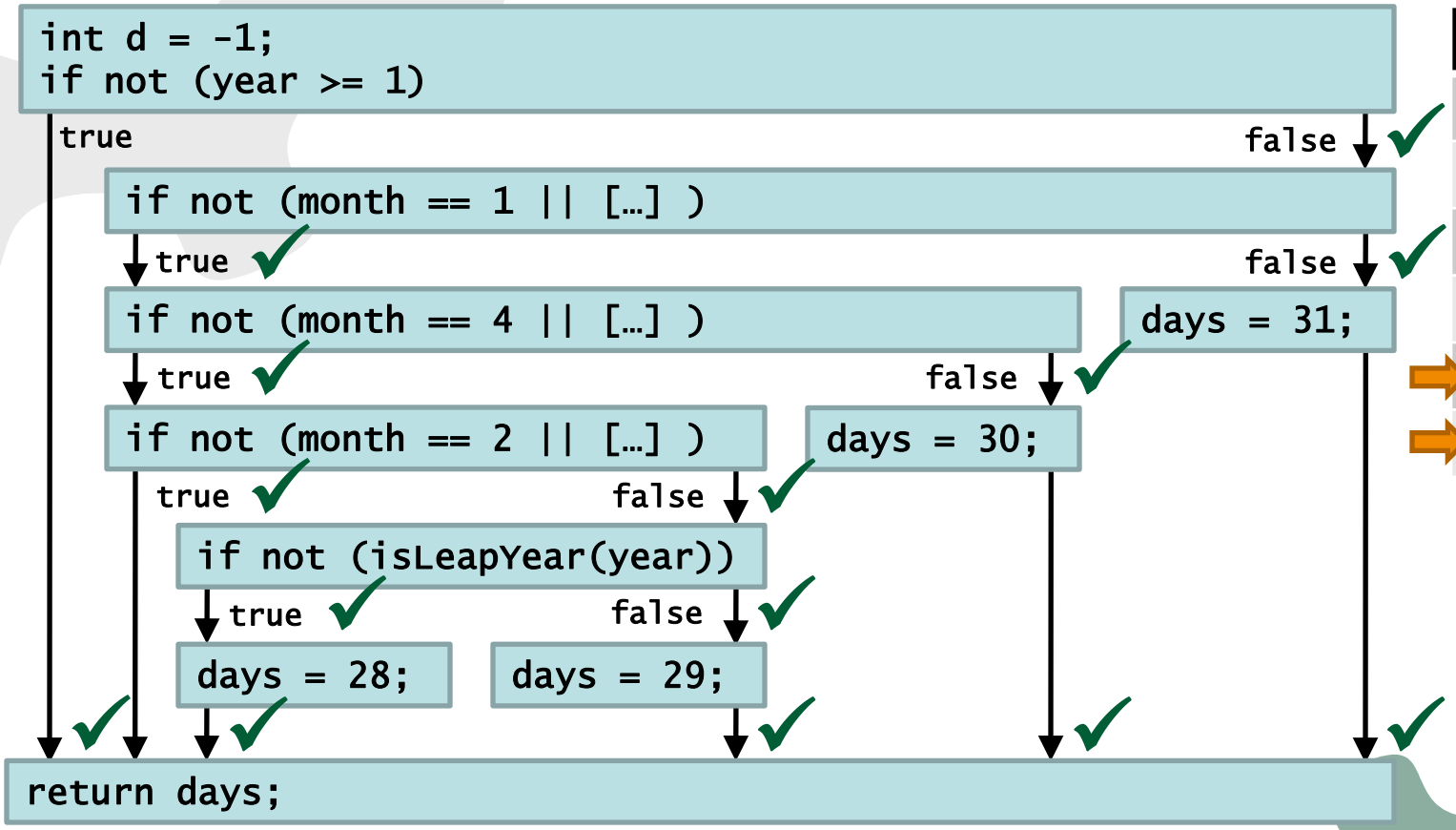


Aufgabe 1b)
Anweisungsüberdeckung

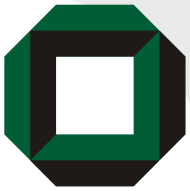


month	year
1	1
4	1
2	4
2	1

Aufgabe 1b) Zweigüberdeckung

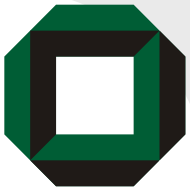


month	year
1	1
4	1
2	1
2	4
X	0
13	1



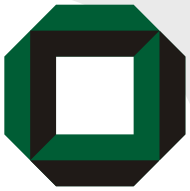
Aufgabe 1c) Bedingungsüberdeckung

year	(year % 4 == 0 &&	year % 100 != 0)	|| year % 400 == 0
1901	Falsch	Nicht ausgewertet	Falsch
1996	Wahr	Wahr	Nicht ausgewertet
2000	Wahr	Falsch	Wahr



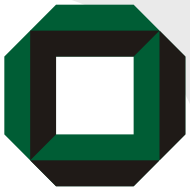
Aufgabe 2) Fehler

```
public final class Complex {
  private double real, imag;

  public Complex sub(Complex c) {
    double r = real - c.real;
    // Korrekte Zeile: double i = imag - c.imag;
    double i = imag - imag;
    real = r;
    imag = i;
    return this;
  }

  public String toString() {
    // Korrekte Zeile: return real + " + " + imag + "i";
    return real + "+" + imag + "i";
  }

  […]
```



Aufgabe 2) Fehler

```
  public Complex div(Complex c) {
    double tmp = c.modsq();
    double r = (real * c.real + imag * c.imag) / tmp;
    double i = (imag * c.real - real * c.imag) / tmp;
    real = r;
    imag = i;
    // Korrekte Zeile: return this;
    return new Complex(r, i);
  }
}
```



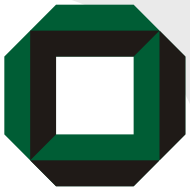
Aufgabe 2) Testklasse

```
import static org.junit.Assert.*;
import org.junit.*;

public class ComplexTest {
  private final static double DELTA = 0.00001;
  private Complex c1, c2;

  @Before public void setUp() {
    c1 = new Complex(1.2, 4.3);
    c2 = new Complex(2.1, 3.4);
  }

  @Test public void copy() {
    Complex c = new Complex(c1);
    assertNotSame(c1, c);
    assertEquals(1.2, c.real());
    assertEquals(4.3, c.imag());
  }
```

„Alle Berechnungen werden mit einem Fehler von höchstens 10^{-5} durchgeführt“

Aufbau der Testressourcen

DELTA wird hier nicht verwendet, da es sich um eine exakte Kopie handeln soll.

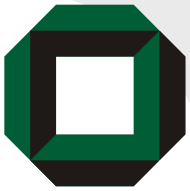


Aufgabe 2) Testklasse

```
@Test public void sub() {
  Complex c = c1.sub(c2);
  assertSame(c1, c);
  assertEquals(-0.9, c.real(), DELTA);
  assertEquals(0.9, c.imag(), DELTA);
}

@Test public void div() {
  Complex c = c1.div(c2);
  assertSame(c1, c);
  assertEquals(1.07326, c.real(), DELTA);
  assertEquals(0.30996, c.imag(), DELTA);
}

@Test public void stringRepresentation() {
  assertEquals("1.2 + 4.3i", c1.toString());
}  […]
}
```

„Achtung! Die komplexe Zahl **`this`** wird bei dieser Operation verändert und als Ergebnis zurückgeliefert.“

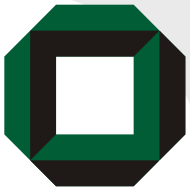
„Alle Berechnungen werden mit einem Fehler von höchstens 10^{-5} durchgeführt“



Aufgabe 3a)

```
public class StateMachine implements IStateMachine {
  private enum State { A, B, C, D, END }

  private State s = State.A;

  public void x() {
    switch (s) {
      case A:
        s = State.C;
        entryC();
        break;
      case B:
        exitB();
        s = State.C;
        entryC();
        break;
      case C:
        s = State.D;
        break;
      case D:
        s = State.END;
        break;
      default: throw new IllegalStateException();
    }
  }
```

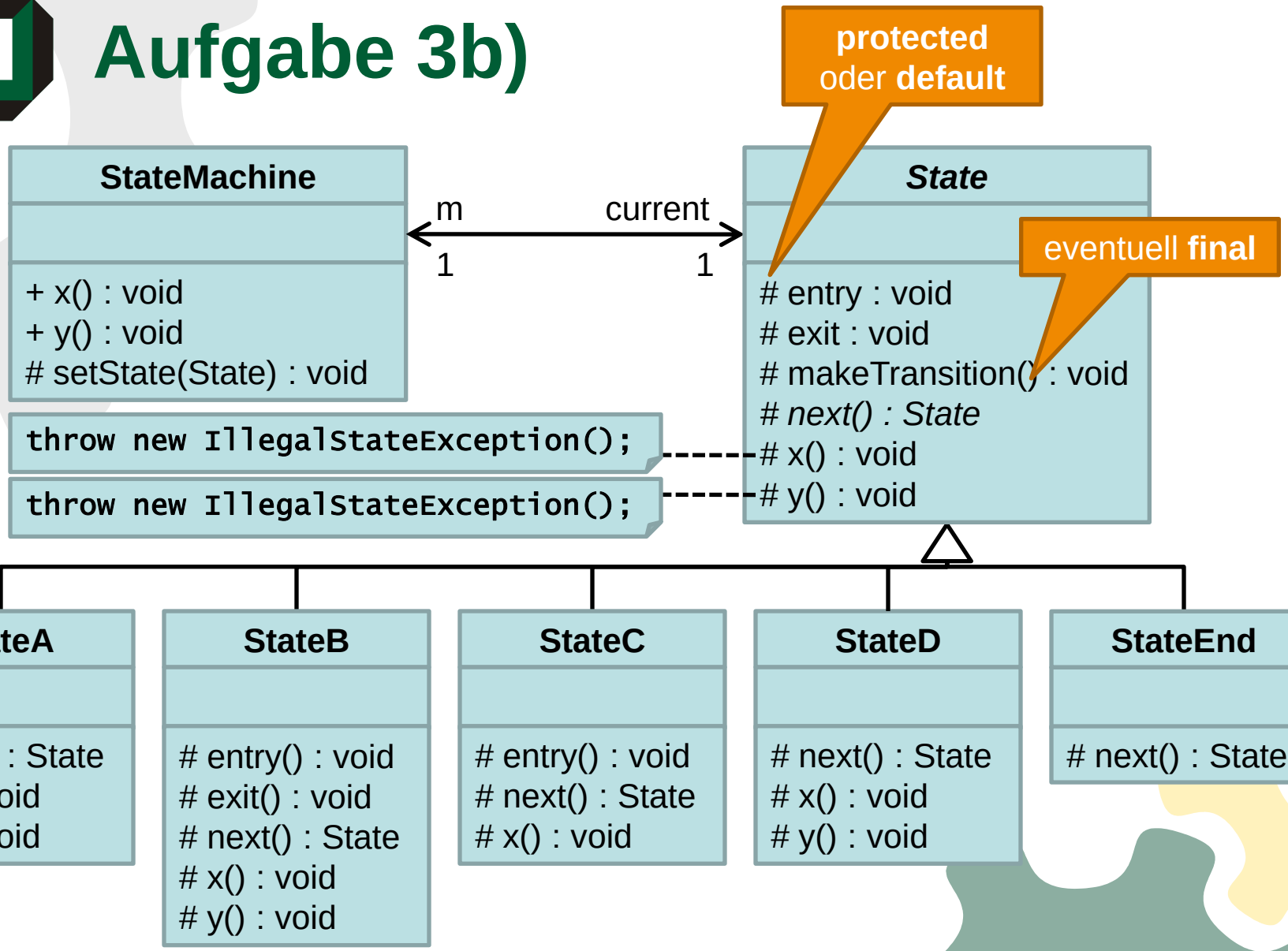
Aufruf von **exitB()** vor **entryC()**!

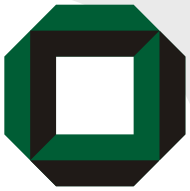


Aufgabe 3a)

```
public void y() {
  switch (s) {
    case A:
      s = State.B;
      entryB();
      break;
    case B:
      exitB();
      s = State.D;
      break;
    case D:
      s = State.C;
      entryC();
      break;
    default:
      throw new IllegalStateException();
  }
}

private void entryB() {}
private void exitB() {}
private void entryC() {}
}
```

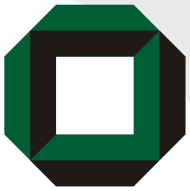
Aufgabe 3b)


protected
oder default
StateMachine
+ x() : void
+ y() : void
# setState(State) : void
m
1
current
1
State
eventuell final
# entry : void
# exit : void
# makeTransition() : void
# next() : State
# x() : void
# y() : void
throw new IllegalStateException();
throw new IllegalStateException();
StateA
next() : State
x() : void
y() : void
StateB
entry() : void
exit() : void
next() : State
x() : void
y() : void
StateC
entry() : void
next() : State
x() : void
StateD
next() : State
x() : void
y() : void
StateEnd
next() : State



Aufgabe 3c)

- Eingebettet explizit:
  - Vorteile:
    - Kompakt
    - Schnell
  - Nachteile:
    - Bei komplexen Automaten unübersichtlich
- Ausgelagert explizit:
  - Vorteile:
    - Flexibel
    - Bei komplexen Automaten Übersichtlicher
    - Die Kontextsensitivität (= Zustandsabhängigkeit) der Methoden braucht nicht mehr explizit verwaltet zu werden, statt dessen wird dynamische Polymorphie verwendet
    - Die Implementierungsarbeit wird auf verschiedene Klassen aufgeteilt (separation of concerns)
  - Nachteile:
    - Sehr viele Klassen bei komplexen Automaten



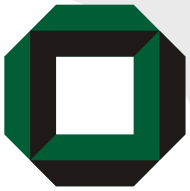
Aufgabe 3c)

- Zustandstabelle:
  - Vorteile:
    - Sehr kompakt
    - Sehr schnell
  - Nachteile:
    - Wenige Übergänge führen zu leeren Tabellen
    - Entry/Exit-Aktionen relativ aufwändig zu implementieren
    - Nur für einfache Automaten geeignet

Aufgabe 3c) Entry-/Exit-Aktionen

```
private void makeTransition(int i) {
  if (s == B) {
    exitB();
  }
  s = stateTable[i][s];
  if (s == ERROR)
    throw new IllegalStateException();
  }
  if (s == B) {
    entryB();
  } else if (s == C) {
    entryC();
  }
}
```

Wenn ein Fehler auftritt, wurde die Exit-Aktion schon ausgeführt.



Aufgabe 3c) Entry-/Exit-Aktionen

```
private void makeTransition(int i) {
  int oldS = s;
  s = stateTable[i][s];
  if (s == ERROR) {
    throw new IllegalStateException();
  }
  if (oldS == B) {
    exitB();
  }
  if (s == B) {
    entryB();
  } else if (s == C) {
    entryC();
  }
}
```