

Z.11 Fehlerbehandlung

Z.11 Fehlerbehandlung

- Programm beenden (`System.exit()`)
 - ◆ meist eine schlechte Idee
- Ausgabe einer Fehlermeldung
 - ◆ hilft nicht den Fehler zu überwinden
- spezieller Rückgabewert kennzeichnet Fehler
 - ◆ Konstruktoren haben keinen Rückgabewert
 - ◆ Was ist wenn die Methode den Wertebereich des Rückgabewerts bereits voll ausnutzt?
- Aufruf einer benutzerdefinierten Fehlerroutine
 - ◆ unschön
 - ◆ Was muss diese Methode tun?
- Lösung: Ausnahmebehandlung (Exceptions)!

Übungen zu Middleware

© Universität Erlangen-Nürnberg • Informatik 4, 2007

Z-Java-Exceptions.fm 2007-10-17 19.45

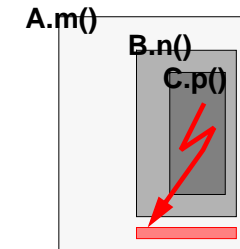
Z.1

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1 Was ist Ausnahmebehandlung?

Z.11 Fehlerbehandlung

- Weiterreichen des Programmflusses vom Fehlerursprung zur Fehlerbehandlung



- Verantwortlichkeit:
 - ◆ Autor eines Codestücks kann den Fehler erkennen, weiß jedoch nicht wie er behandelt werden soll.
 - ◆ Benutzer des Codes weiß was zu tun ist, hat jedoch nicht die Möglichkeit den Fehler zu erkennen.

Übungen zu Middleware

© Universität Erlangen-Nürnberg • Informatik 4, 2007

Z-Java-Exceptions.fm 2007-10-17 19.45

Z.2

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Was passiert bei einem Methodenaufruf?

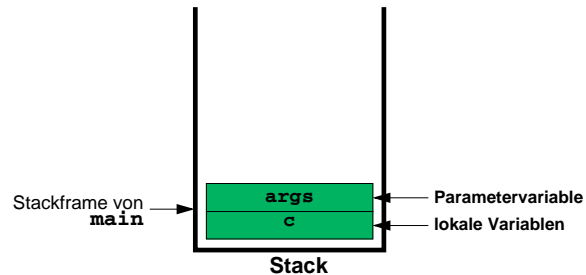
Z.11 Fehlerbehandlung

```

class Customer {
    void createAccount
        (Bank bank) {
        Account account =
            new Account();
        bank.newAccount(account, 5);
    }
}

class Bank {
    void newAccount
        (Account a, int i) {
        int counter = 0;
        ...
    }
}

class Main {
    public static void main(String
        args[]) {
        Customer c = new Customer();
        c.createAccount(new Bank());
    }
}
    
```



Übungen zu Middleware

© Universität Erlangen-Nürnberg • Informatik 4, 2007

Z-Java-Exceptions.fm 2007-10-17 19.45

Z.3

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Was passiert bei einem Methodenaufruf? (2)

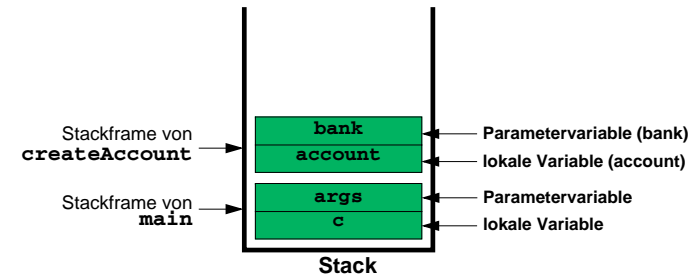
Z.11 Fehlerbehandlung

```

class Customer {
    void createAccount
        (Bank bank) {
        Account account =
            new Account();
        bank.newAccount(account, 5);
    }
}

class Bank {
    void newAccount
        (Account a, int i) {
        int counter = 0;
        ...
    }
}

class Main {
    public static void main(String
        args[]) {
        Customer c = new Customer();
        c.createAccount(new Bank());
    }
}
    
```



Übungen zu Middleware

© Universität Erlangen-Nürnberg • Informatik 4, 2007

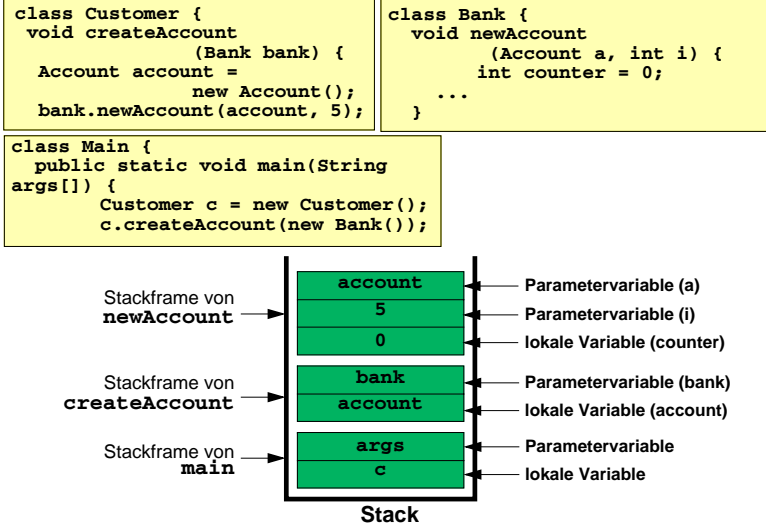
Z-Java-Exceptions.fm 2007-10-17 19.45

Z.4

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Was passiert bei einem Methodenaufruf? (3)

Z.11 Fehlerbehandlung



MW - Übung

Übungen zu Middleware
© Universität Erlangen-Nürnberg • Informatik 4, 2007

Z-Java-Exceptions.fm 2007-10-17 19.45

Z.5

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

4 Finally-Anweisung

Z.11 Fehlerbehandlung

- der **finally**-Block wird immer beim Verlassen eines **try**-Blockes ausgeführt
 - ◆ kann zum Aufräumen benutzt werden bei (un-)behandelten Ausnahmen

```
try {
    ...
} catch(...) {
    ... // Fehlerbehandlung
} finally {
    ... // Ressourcen freigeben
}
```

- ein **finally**-Block kann auch ohne **catch**-Block benutzt werden:

```
try {
    ...
    if (...) return;
    ...
} finally { ... }
```

MW - Übung

Übungen zu Middleware
© Universität Erlangen-Nürnberg • Informatik 4, 2007

Z-Java-Exceptions.fm 2007-10-17 19.45

Z.7

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3 Try, Throw und Catch-Anweisung

Z.11 Fehlerbehandlung

```
try {
    ...
    if (...) throw new MyException();
    ...
} catch(MyException e) {
    // exception handler
    ...
}
```

- **throw** wird benutzt um eine *Ausnahme* (Exception) zu werfen
- ein **catch**-Block muss direkt nach dem **try**-Block folgen
- es kann mehr als nur einen **catch**-Block geben
 - ◆ der passende **catch**-Block wird nach der Programmreihenfolge gesucht
- ein Methode muss nicht alle Exception fangen
 - ◆ nicht gefangene Exception werden automatisch an die aufrufende Methode weitergereicht

MW - Übung

Übungen zu Middleware
© Universität Erlangen-Nürnberg • Informatik 4, 2007

Z-Java-Exceptions.fm 2007-10-17 19.45

Z.6

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

5 throws-Anweisung

Z.11 Fehlerbehandlung

- unbehandelte Exceptions müssen bei der Methode angegeben werden:

```
class Test {
    void m() throws MyException {
        ...
        if (...) throw new MyException();
        ...
    }
}
```

MW - Übung

Übungen zu Middleware
© Universität Erlangen-Nürnberg • Informatik 4, 2007

Z-Java-Exceptions.fm 2007-10-17 19.45

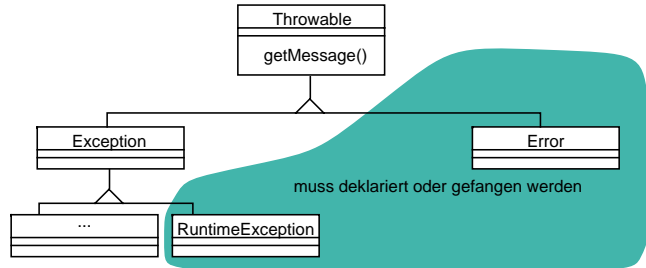
Z.8

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

6 Fehlerklassen

Z.11 Fehlerbehandlung

- alle Exceptions sind von **Throwable** abgeleitet
- Ausnahmen die beinahe überall auftreten können sind:
 - ◆ **Error**: Linkerfehler, Fehler im Format von Klassendateien, out of memory, ...
 - ◆ **RuntimeException**: array index, null pointer, illegal cast, ...
- Ausnahmen von Anwendungen sind von **java.lang.Exception** abgeleitet



Übungen zu Middleware

©Universität Erlangen-Nürnberg • Informatik 4, 2007

Z-Java-Exceptions.fm 2007-10-17 19.45

Z.9

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

7 Ausnahmen und Vererbung: Fehlerbehandlung

Z.11 Fehlerbehandlung

- Ausnahmebehandlung von Unterklassen durch mehrere catch-Blöcke
- Bemerkung: Die Oberklasse behandelt alle Unterklassen, die Oberklasse sollte daher immer am Ende "gefangen" werden:

```
class MathException {}
class ZeroDivideException extends MathException {}
class InvalidArgException extends MathException {}
try {
    ...
} catch (ZeroDivideException e) {
    ...
} catch (InvalidArgException e) {
    ...
} catch (MathException e) {
    ...
}
```

Übungen zu Middleware

©Universität Erlangen-Nürnberg • Informatik 4, 2007

Z-Java-Exceptions.fm 2007-10-17 19.45

Z.10

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

8 Beispiel

Z.11 Fehlerbehandlung

```
class TestException extends Exception {
    public TestException(String s) {super(s);}
}

public class Test {

    public void hello() throws TestException {
        if (...) throw new TestException("...an error msg...");
    }

    public void testIt() {
        try {
            hello();
            ...
        } catch (TestException t) {
            System.out.println("Exception raised:" + t.getMessage());
        } finally {
            // clean up
        }
    }
}
```

Übungen zu Middleware

©Universität Erlangen-Nürnberg • Informatik 4, 2007

Z-Java-Exceptions.fm 2007-10-17 19.45

Z.11

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

9 Ausnahmen und Vererbung: Throwing

Z.11 Fehlerbehandlung

- Können überschriebene Methoden andere Exceptions werfen, als die ursprüngliche Methode?
- Grundsatz:
 - ◆ Unterklassen können überall dort verwendet werden wo die Oberklasse erwartet wird.
 - ◆ Unterklassen sind "besser" als Oberklassen.
- das bedeutet:
 - ◆ Unterklassen dürfen keine zusätzlichen Exception werfen.
 - ◆ Unterklassen können Unterklassen von den deklarierten Ausnahmen der Oberklasse werfen.
 - ◆ Unterklassen dürfen keine Oberklassen von den ursprünglich deklarierten Ausnahmen werfen.

Übungen zu Middleware

©Universität Erlangen-Nürnberg • Informatik 4, 2007

Z-Java-Exceptions.fm 2007-10-17 19.45

Z.12

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

9 Beispiel

Z.11 Fehlerbehandlung

```
class E1 extends Exception {}
class E2 extends Exception {}
class E3 extends E2 {}
class A {
    void m() throws E2 {}
}
class B extends A {
    void m() throws ??? {}
}
```

■ ??? =

MW - Übung

Übungen zu Middleware
© Universität Erlangen-Nürnberg • Informatik 4, 2007

Z-Java-Exceptions.fm 2007-10-17 19.45

Z.13

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

9 Beispiel (2)

Z.11 Fehlerbehandlung

```
class E1 extends Exception {}
class E2 extends Exception {}
class E3 extends E2 {}
class A {
    void m() throws E2 {}
}
class B extends A {
    void m() throws ??? {}
}
```

■ ??? =

Falsch sind:

E1
Exception
...

Richtig sind:

E2
E3
keine

MW - Übung

Übungen zu Middleware
© Universität Erlangen-Nürnberg • Informatik 4, 2007

Z-Java-Exceptions.fm 2007-10-17 19.45

Z.14

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

10 Zusammenfassung: Exceptions

Z.11 Fehlerbehandlung

- werfen von Ausnahmen: `throw new MyException("...");`
- Programm-Block für den die Ausnahmebehandlung gilt: `try { ... }`
- Ausnahmebehandlung:

```
try {
    ... throw new MyException("..."); ...
} catch(MyException e) { ... }
```
- Zusätzlich: `finally`-Block
- Ausnahmen müssen von `Throwable` abgeleitet sein.
- Ausnahmen von Anwendungen sollten von `Exception` abgeleitet werden.
- Ausnahmen müssen bei der Methodendeklaration angegeben werden:
`void mymethod() throws ...`

MW - Übung

Übungen zu Middleware
© Universität Erlangen-Nürnberg • Informatik 4, 2007

Z-Java-Exceptions.fm 2007-10-17 19.45

Z.15

Reproduktion jeder Art oder Verwendung dieser Unterlage, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.