



Whitebox-Tests: Allgemeines

- Testen
- ▶ Strategien
 - ▶ Blackbox
 - ▶ Whitebox
 - ▶ Graybox
 - ▶ Zeitlicher Verlauf
 - ▶ Zusammenfassung
- Andere Bezeichnungen
Logic driven, Strukturelles Testen
 - Der Tester entwickelt Testfälle aus einer Betrachtung der Ablauflogik des Programms unter Berücksichtigung der Spezifikation
 - Intuitiv scheint es ausreichend zu sein, jede Programmanweisung mindestens einmal zu durchlaufen um das Programm zu testen (das entspricht *Statement Coverage*, s. nächste Folien), das reicht jedoch nicht aus
Man benötigt eigentlich ein erschöpfendes Testen aller Pfade

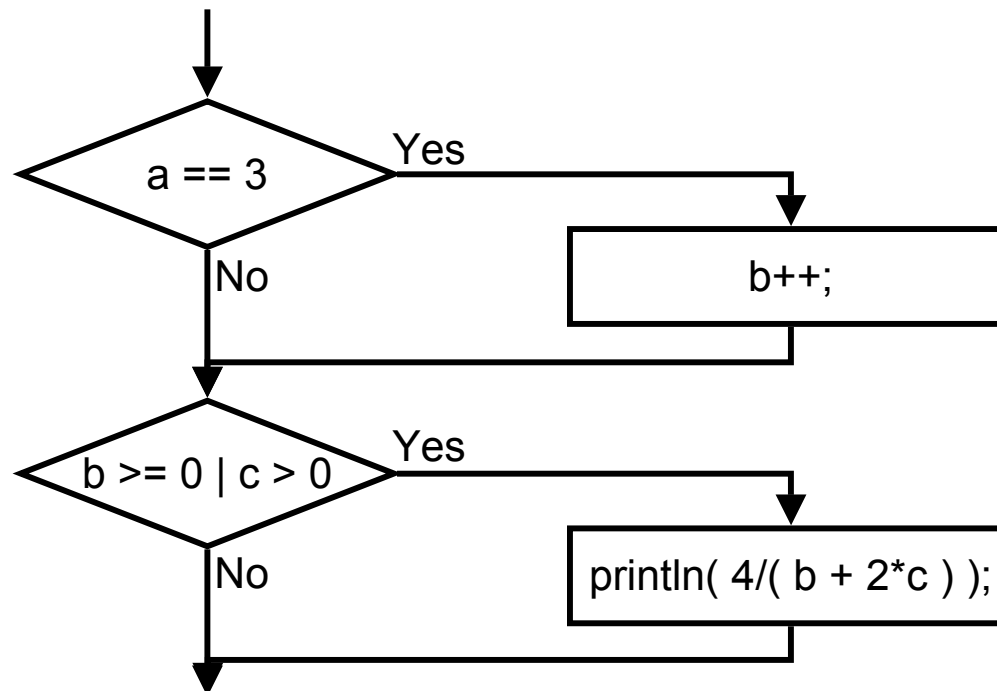


Whitebox-Tests: Pfade durch ein Programm

Testen

- ▶ Strategien
- ▶ Blackbox
- ▶ Whitebox
- ▶ Graybox
- ▶ Zeitlicher Verlauf
- ▶ Zusammenfassung

Für den Whitebox-Test betrachtet man
Pfade (A – E) durch ein Programm

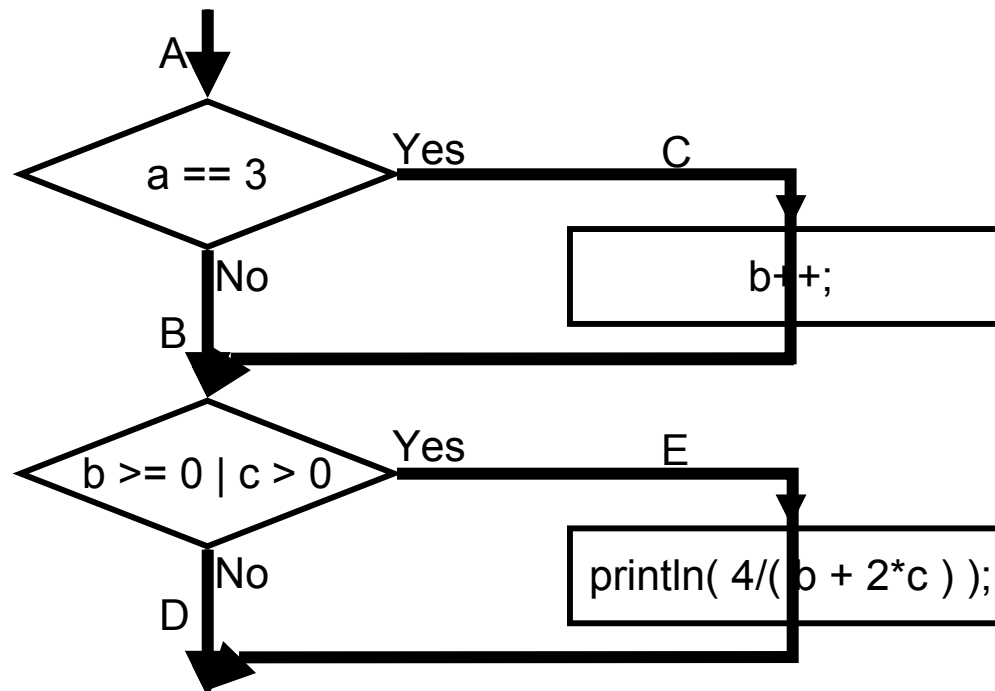




Whitebox-Tests: Pfade durch ein Programm

- Testen
- ▶ Strategien
 - ▶ Blackbox
 - ▶ Whitebox
 - ▶ Graybox
 - ▶ Zeitlicher Verlauf
 - ▶ Zusammenfassung

Für den Whitebox-Test betrachtet man
Pfade (A – E) durch ein Programm



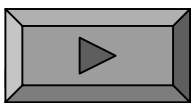
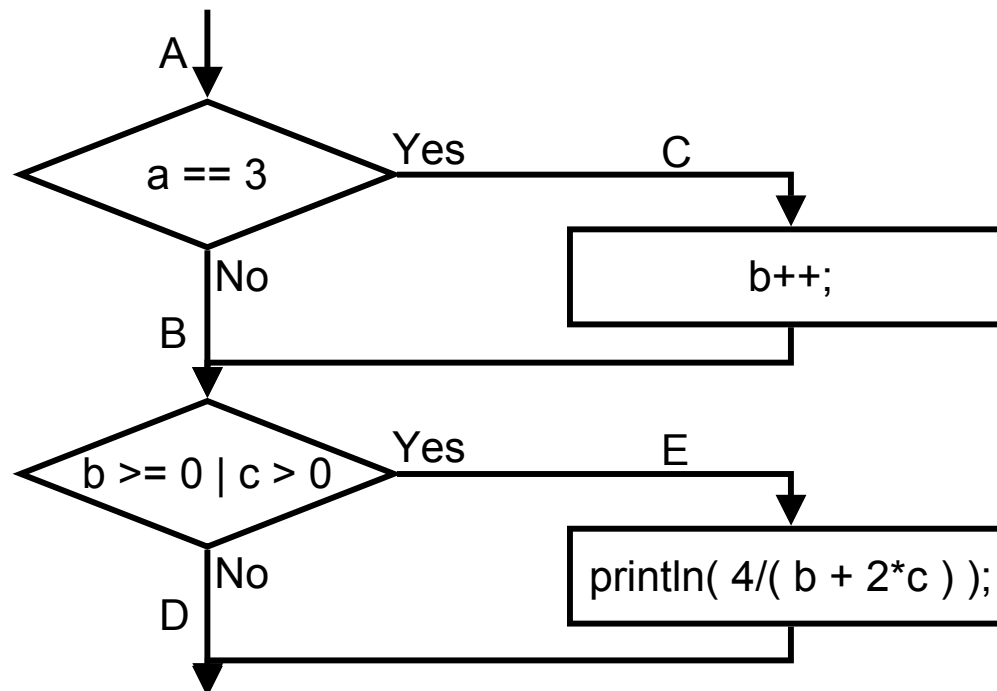


Whitebox-Tests: Partner-Diskussion

Testen

- ▶ Strategien
- ▶ Blackbox
- ▶ Whitebox
- ▶ Graybox
- ▶ Zeitlicher Verlauf
- ▶ Zusammenfassung

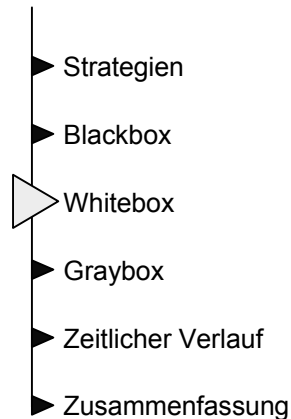
- Diskutieren Sie mit einem Partner:
 - Welche und wie viele Fälle muss man bei einem Whitebox-Test für das nachfolgende Programm testen, um sicher zu sein, dass es korrekt ist?
 - Welcher Fehler kann auftreten (falls überhaupt)?
Mit welchem Testfall würden Sie den Fehler finden?
 - Machen Sie sich Notizen
- Dauer: 3 Minuten





Anforderung an ein Überdeckungskriterium

Testen



Gesucht ist ein formales Überdeckungskriterium, das garantiert, dass das Programm mit geeigneten Testfällen durchlaufen wird, so dass potenzielle Fehler bemerkt werden (= Fehlverhalten auftreten)

Mögliche Überdeckungskriterien (engl.: Coverage Criteria)

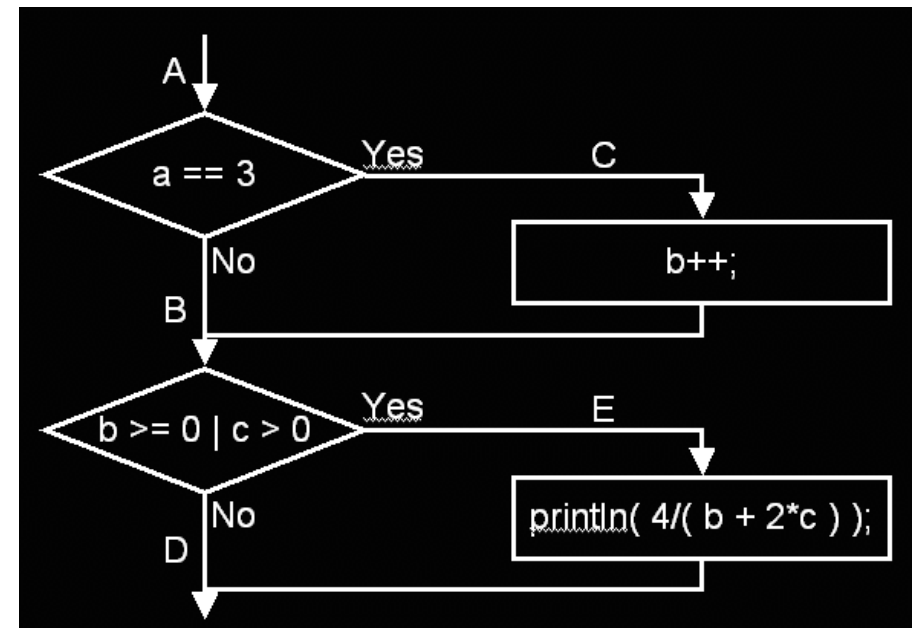
- Statement Coverage
- Decision Coverage, Branch Coverage
- Condition Coverage
- Decision/Condition Coverage
- Multiple Condition Coverage

Testen

- ▶ Strategien
- ▶ Blackbox
- ▶ Whitebox
- ▶ Graybox
- ▶ Zeitlicher Verlauf
- ▶ Zusammenfassung

- Programmcode in Java

```
int a, b, c;  
...  
if ( a == 3 )  
    b++;  
if ( b >= 0 | c > 0 )  
    println( 4/( b + 2*c ) );
```





Abdeckungskriterien für Whitebox-Tests: Statement Coverage

Testen

- ▶ Strategien
- ▶ Blackbox
- ▶ Whitebox
- ▶ Graybox
- ▶ Zeitlicher Verlauf
- ▶ Zusammenfassung

- Definition:
Jede Anweisung muss mindestens einmal durchlaufen werden
- Im Beispiel
Der folgende Testfall reicht für die vollständige Überdeckung nach dem Kriterium Statement Coverage aus
3, -2, 1

→ Dieser Test ist nicht ausreichend:
der Fehler würde nicht entdeckt werden!

- Neue Idee:
Verzweigungen und deren
Auswertungsergebnis anstelle von
Anweisungen berücksichtigen

`-1 >= 0 | 1 > 0`
→ true

```
int a, b, c;  
...  
if ( a == 3 )  
    b++;  
if ( b >= 0 | c > 0 )  
    println( 4/( b + 2*c ) );
```



Abdeckungstechniken für Whitebox-Tests: Decision Coverage, Branch Coverage

Testen

- ▶ Strategien
- ▶ Blackbox
- ▶ Whitebox
- ▶ Graybox
- ▶ Zeitlicher Verlauf
- ▶ Zusammenfassung

- Jede (binäre, Boole'sche) Entscheidung (engl.: *Decision*) muss einmal mit *true* und einmal mit *false* verlassen werden bzw.: Jeder Zweig (engl.: *Branch*), der auf eine Verzweigung folgt muss mindestens einmal durchlaufen werden

- Im Beispiel

Die folgenden Testfälle reichen für die vollständige Überdeckung nach dem Kriterium Decision Coverage aus

3, -2, 1
2, -1, 0

-1 >= 0 | 1 > 0
→ true

```
int a, b, c;  
...  
if ( a == 3 )  
    b++;  
if ( b >= 0 | c > 0 )  
    println( 4/( b + 2*c ) );
```



Abdeckungstechniken für Whitebox-Tests: Decision Coverage, Branch Coverage

Testen

- ▶ Strategien
- ▶ Blackbox
- ▶ Whitebox
- ▶ Graybox
- ▶ Zeitlicher Verlauf
- ▶ Zusammenfassung

- Jede (binäre, Boole'sche) Entscheidung (engl.: *Decision*) muss einmal mit *true* und einmal mit *false* verlassen werden bzw.: Jeder Zweig (engl.: *Branch*), der auf eine Verzweigung folgt muss mindestens einmal durchlaufen werden

- Im Beispiel

Die folgenden Testfälle reichen für die vollständige Überdeckung nach dem Kriterium Decision Coverage aus

3, -2, 1
2, -1, 0

→ Auch dieser Test ist nicht ausreichend:
der Fehler würde nicht entdeckt werden!

- Neue Idee:
Teilausdrücke statt des
Gesamtergebnisses berücksichtigen

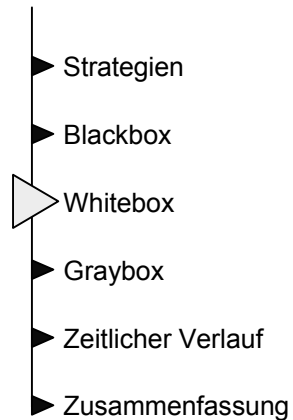
-1 >= 0 | 0 > 0
→ false

```
int a, b, c;  
...  
if ( a == 3 )  
    b++;  
if ( b >= 0 | c > 0 )  
    println( 4/( b + 2*c ) );
```



Abdeckungskriterien für Whitebox-Tests: Decision Coverage, Branch Coverage

Testen



Bemerkung

Normalerweise liefert Decision Coverage (DC) eine Obermenge von Statement Coverage (SC), es gibt jedoch Ausnahmen:

- Ein Programm hat keine Verzweigung, dann gibt es keine Testfälle (ist nicht wirklich relevant)
- Anweisungen in Exception-Handling-Programmteilen werden nicht zwangsläufig ausgeführt, weil dort häufig keine Verzweigungen existieren



Abdeckungstechniken für Whitebox-Tests: Condition Coverage

Testen

- ▶ Strategien
- ▶ Blackbox
- ▶ Whitebox
- ▶ Graybox
- ▶ Zeitlicher Verlauf
- ▶ Zusammenfassung

- Jeder Teilausdruck in jeder Bedingung muss alle möglichen Werte mindestens einmal annehmen und das Programm und alle Routinen müssen (wie bei Decision Coverage) mindestens einmal bei jedem Einstiegspunkt gestartet werden
- Im Beispiel
Die folgenden Testwerte reichen für die vollständige Überdeckung nach dem Kriterium Condition Coverage aus

3, -1, 1
2, -1, 0

$0 \geq 0 \mid 1 > 0$
→ true

```
int a, b, c;  
...  
if ( a == 3 )  
    b++;  
if ( b >= 0 | c > 0 )  
    println( 4/( b + 2*c ) );
```

4 / 2



Abdeckungstechniken für Whitebox-Tests: Condition Coverage

Testen

- ▶ Strategien
- ▶ Blackbox
- ▶ Whitebox
- ▶ Graybox
- ▶ Zeitlicher Verlauf
- ▶ Zusammenfassung

- Jeder Teilausdruck in jeder Bedingung muss alle möglichen Werte mindestens einmal annehmen und das Programm und alle Routinen müssen (wie bei Decision Coverage) mindestens einmal bei jedem Einstiegspunkt gestartet werden
- Im Beispiel
Die folgenden Testwerte reichen für die vollständige Überdeckung nach dem Kriterium Condition Coverage aus

3, -1, 1
2, -1, 0

→ Auch dieser Test ist nicht ausreichend:
der Fehler würde nicht entdeckt werden!

```
int a, b, c;  
...  
if ( a == 3 )  
    b++;  
if ( b >= 0 | c > 0 )  
    println( 4/( b + 2*c ) );
```

-1 >= 0 | 0 > 0
→ false



Abdeckungskriterien für Whitebox-Tests: Condition Coverage

Testen

- ▶ Strategien
- ▶ Blackbox
- ▶ Whitebox
- ▶ Graybox
- ▶ Zeitlicher Verlauf
- ▶ Zusammenfassung

- Problem bei der Condition Coverage:
Das Gesamtergebnis eines Ausdrucks in Verzweigungen wird nicht berücksichtigt, daher werden Zweige eventuell übersprungen

- Problem-Beispiel

```
if ( a & b )  
    println( .... );
```

Nach Condition Coverage werden nur zwei Testfälle benötigt, es reichen zum Beispiel:

- a == true, b == false
- a == false, b == true

In beiden Fällen wird aber nicht die println-Anweisung durchlaufen!

- Neue Idee
Teilausdrücke und Gesamtergebnis berücksichtigen



Abdeckungstechniken für Whitebox-Tests: Decision/Condition Coverage

Testen

- ▶ Strategien
- ▶ Blackbox
- ▶ Whitebox
- ▶ Graybox
- ▶ Zeitlicher Verlauf
- ▶ Zusammenfassung

- Jeder Teilausdruck in jeder Bedingung muss alle möglichen Werte mindestens einmal annehmen, jede Bedingung (als Ganzes) muss alle Ergebnisse mindestens einmal annehmen und das Programm und alle Routinen müssen mindestens einmal bei jedem Einstiegspunkt gestartet werden (wg. Exception-Handling)
- Im Beispiel
In diesem Beispiel reichen die Testwerte der Condition Coverage auch für die vollständige Überdeckung nach dem Kriterium Decision/Condition Coverage aus

3, -1, 1
2, -1, 0

$0 \geq 0 \mid 1 > 0$
→ true

```
int a, b, c;  
...  
if ( a == 3 )  
    b++;  
if ( b >= 0 | c > 0 )  
    println( 4/( b + 2*c ) );
```



Abdeckungstechniken für Whitebox-Tests: Decision/Condition Coverage

Testen

- ▶ Strategien
- ▶ Blackbox
- ▶ Whitebox
- ▶ Graybox
- ▶ Zeitlicher Verlauf
- ▶ Zusammenfassung

- Jeder Teilausdruck in jeder Bedingung muss alle möglichen Werte mindestens einmal annehmen, jede Bedingung (als Ganzes) muss alle Ergebnisse mindestens einmal annehmen und das Programm und alle Routinen müssen mindestens einmal bei jedem Einstiegspunkt gestartet werden (wg. Exception-Handling)
- Im Beispiel
In diesem Beispiel reichen die Testwerte der Condition Coverage auch für die vollständige Überdeckung nach dem Kriterium Decision/Condition Coverage aus

3, -1, 1
2, -1, 0

→ Der Fehler würde wiederum nicht entdeckt werden!

$-1 \geq 0 \mid 0 > 0$
→ false

- Neue Idee
Kombinationen von Teilausdrücken berücksichtigen

```
int a, b, c;  
...  
if ( a == 3 )  
    b++;  
if ( b >= 0 | c > 0 )  
    println( 4/( b + 2*c ) );
```



Abdeckungskriterien für Whitebox-Tests: Multiple Condition Coverage

Testen

- ▶ Strategien
- ▶ Blackbox
- ▶ Whitebox
- ▶ Graybox
- ▶ Zeitlicher Verlauf
- ▶ Zusammenfassung

- Alle möglichen Kombinationen von Teilausdrücken in jeder Bedingung und alle Einstiegspunkte in das Programm und alle Routinen müssen mindestens einmal durchlaufen werden
- Im Beispiel
Die folgenden Testwerte reichen für die vollständige Überdeckung nach dem Kriterium Decision/Condition Coverage aus

3, -1, 1
2, 0, 0
2, -1, 1
2, -1, 0

$0 \geq 0 \mid 1 > 0$
→ true

```
int a, b, c;  
...  
if ( a == 3 )  
    b++;  
if ( b >= 0 | c > 0 )  
    println( 4/( b + 2*c ) );
```



Abdeckungskriterien für Whitebox-Tests: Multiple Condition Coverage

Testen

- ▶ Strategien
- ▶ Blackbox
- ▶ Whitebox
- ▶ Graybox
- ▶ Zeitlicher Verlauf
- ▶ Zusammenfassung

- Alle möglichen Kombinationen von Teilausdrücken in jeder Bedingung und alle Einstiegspunkte in das Programm und alle Routinen müssen mindestens einmal durchlaufen werden
- Im Beispiel
Die folgenden Testwerte reichen für die vollständige Überdeckung nach dem Kriterium Decision/Condition Coverage aus

3, -1, 1
2, 0, 0
2, -1, 1
2, -1, 0

→ Der Fehler wird entdeckt, das Programm wird mit einer *ArithmeticException* abgebrochen

$0 \geq 0 \mid 0 > 0$
→ true

```
int a, b, c;  
...  
if ( a == 3 )  
    b++;  
if ( b >= 0 | c > 0 )  
    println( 4/( b + 2*c ) );
```

4 / 0
→ Fehler!



Abdeckungskriterien für Whitebox-Tests: Multiple Condition Coverage

Testen

- ▶ Strategien
- ▶ Blackbox
- ▶ Whitebox
- ▶ Graybox
- ▶ Zeitlicher Verlauf
- ▶ Zusammenfassung

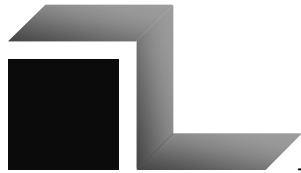
- Alle möglichen Kombinationen von Teilausdrücken in jeder Bedingung und alle Einstiegspunkte in das Programm und alle Routinen müssen mindestens einmal durchlaufen werden
- Im Beispiel
Die folgenden Testwerte reichen für die vollständige Überdeckung nach dem Kriterium Decision/Condition Coverage aus

3, -1, 1
2, 0, 0
2, -1, 1
2, -1, 0

→ Der Fehler wurde entdeckt, das Programm wird mit einer *ArithmeticException* abgebrochen

$-1 \geq 0 \mid 1 > 0$
→ true

```
int a, b, c;  
...  
if ( a == 3 )  
    b++;  
if ( b >= 0 | c > 0 )  
    println( 4/( b + 2*c ) );
```



Abdeckungskriterien für Whitebox-Tests: Multiple Condition Coverage

Testen

- ▶ Strategien
- ▶ Blackbox
- ▶ Whitebox
- ▶ Graybox
- ▶ Zeitlicher Verlauf
- ▶ Zusammenfassung

- Alle möglichen Kombinationen von Teilausdrücken in jeder Bedingung und alle Einstiegspunkte in das Programm und alle Routinen müssen mindestens einmal durchlaufen werden
- Im Beispiel
Die folgenden Testwerte reichen für die vollständige Überdeckung nach dem Kriterium Decision/Condition Coverage aus

3, -1, 1
2, 0, 0
2, -1, 1
2, -1, 0

→ Der Fehler wurde entdeckt, das Programm wird mit einer *ArithmeticException* abgebrochen

- Eine Menge von Testfälle, die MCC erfüllt, erfüllt auch SC, DC, CC und D/CC

$-1 \geq 0 \mid 0 > 0$
→ false

```
int a, b, c;  
...  
if ( a == 3 )  
    b++;  
if ( b >= 0 | c > 0 )  
    println( 4/( b + 2*c ) );
```



Abdeckungskriterien in der Übersicht

Testen

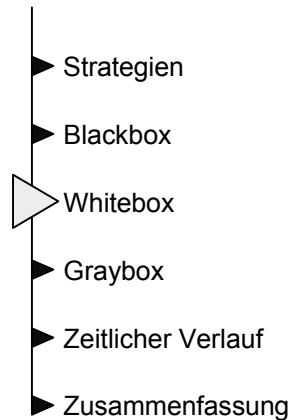
- ▶ Strategie
- ▶ Blackbox
- ▶ Whitebox
- ▶ Graybox
- ▶ Zeitliche
- ▶ Zusammenfassung

	1	2	A	B	C	D	E	F	G	H	I	J	K	L	M	N
	1			Testfallwerte					Überdeckung							
	2			a	b	c	b'		1 (if)	1 Bed.	2	3 (if)	3a	3b	3 Ges.	4
	3															
	4															
	5															
	6															
	7															
	8															
	9															
	10		Statement Coverage													
	11		Decision Coverage													
	12		Condition Coverage													
	13		Dec./Cond. Coverage													
	22		Multiple Condition Coverage													
	24		Statement Coverage	nicht erfüllt	jede Anweisung ausgeführt											
	25		Decision Coverage	nicht erfüllt	jede Bedingung einmal true, einmal false											
	26		Condition Coverage	nicht erfüllt	jede Teilbedingung einmal true, einmal false											
	27		Dec./Cond. Coverage	nicht erfüllt	jede Bedingung und jede Teilbedingung einmal true, einmal false											
	28		Multiple Condition Coverage	nicht erfüllt	Teilbedingungen in allen Kombinationen											
	29															
	30															

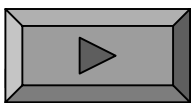


Whitebox-Tests: Bisherige Erfahrungen

Testen



- Multiple Condition Coverage ist ein minimales Testkriterium für Programme mit Mehrfach-Bedingungen
- Hoffnung:
Nach einem erschöpfenden Pfad-Test ist das Programm vollständig getestet
- Diskutieren Sie mit einem Partner:
 - Wird die Hoffnung erfüllt?
 - Falls ja, geben Sie eine Begründung an!
 - Falls nein, geben Sie ein Gegenbeispiel an!
- Dauer: 3 Minuten



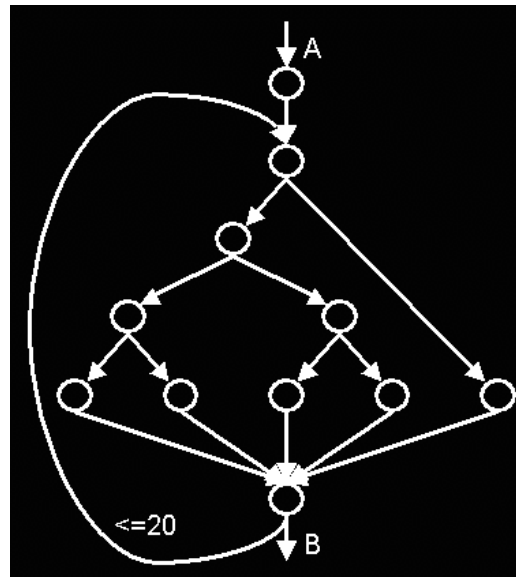


Whitebox-Tests

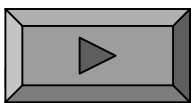
Testen

- ▶ Strategien
- ▶ Blackbox
- ▶ Whitebox
- ▶ Graybox
- ▶ Zeitlicher Verlauf
- ▶ Zusammenfassung

- Die Anzahl eindeutiger Pfade durch ein Programm ist meistens sehr hoch; Beispiel eines Programm-Ablaufplans



- Die eindeutigen Pfade durch dieses Programm entspricht der Anzahl der Möglichkeiten von A nach B zu kommen
- Frage:
Wie viele Möglichkeiten gibt es, von A nach B zu kommen?





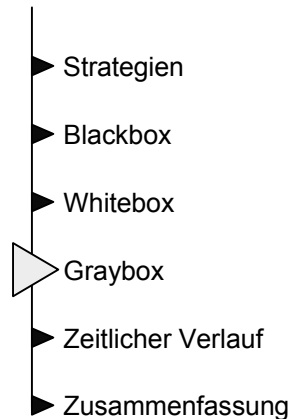
Testen

- ▶ Strategien
- ▶ Blackbox
- ▶ Whitebox
- ▶ Graybox
- ▶ Zeitlicher Verlauf
- ▶ Zusammenfassung

Fragen?



Testen



- Graybox-Tests versuchen, die Vorteile von Blackbox- und Whitebox-Testverfahren zu kombinieren

Vorgehen

- Soll-Überdeckungsgrad festlegen, z.B. *Decision Coverage*
- Zunächst Blackbox-Tests durchführen
 - Zur Überprüfung der Funktionalität
 - Überdeckung wird (im Hintergrund) mitprotokolliert
- Dann Whitebox-Test durchführen
 - Analyse der nicht überdeckten Programmteile
 - Korrektur des Programms (Entfernen unnötiger Teile) oder
 - Erstellen zusätzlicher Testfälle
 - Testen, bis der vordefinierte Überdeckungsgrad erreicht ist