Karlsruher Institut für Technologie

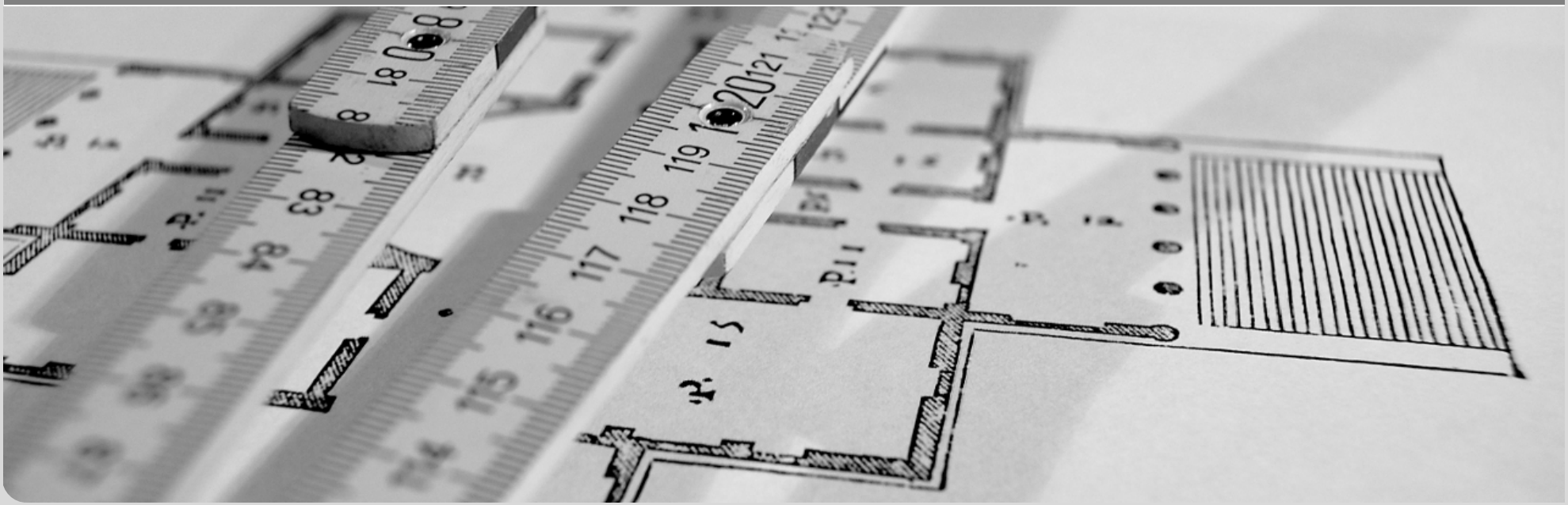
SDQ Software Design and Quality

Softwaretechnik - 7. Tutorium

Tutorium Nr. 17

Kay Schmitteckert | 16.07.2015

INSTITUT FÜR PROGRAMMSTRUKTUREN UND DATENORGANISATION (IPD)



Zu...

Übungsblatt 6

Zum Aufwärmen

Quiz

Quiz

Aussage	wahr	falsch
Eine Komponente kann immer nur mit einem einzigen Dekorierer versehen werden		
Entwurfsmuster erleichtern die Kommunikation im Team		
Eine Komposition ist eine Aggregation, bei der die Teile keine Daseinsberechtigung ohne das Ganze haben		
Das Lastenheft ist eine Verfeinerung des Pflichtenhefts		

Quiz

Aussage	wahr	falsch
Eine Komponente kann immer nur mit einem einzigen Dekorierer versehen werden		×
Entwurfsmuster erleichtern die Kommunikation im Team		
Eine Komposition ist eine Aggregation, bei der die Teile keine Daseinsberechtigung ohne das Ganze haben		
Das Lastenheft ist eine Verfeinerung des Pflichtenhefts		

Quiz

Aussage	wahr	falsch
Eine Komponente kann immer nur mit einem einzigen Dekorierer versehen werden		×
Entwurfsmuster erleichtern die Kommunikation im Team	×	
Eine Komposition ist eine Aggregation, bei der die Teile keine Daseinsberechtigung ohne das Ganze haben		
Das Lastenheft ist eine Verfeinerung des Pflichtenhefts		

Quiz

Aussage	wahr	falsch
Eine Komponente kann immer nur mit einem einzigen Dekorierer versehen werden		×
Entwurfsmuster erleichtern die Kommunikation im Team	×	
Eine Komposition ist eine Aggregation, bei der die Teile keine Daseinsberechtigung ohne das Ganze haben	×	
Das Lastenheft ist eine Verfeinerung des Pflichtenhefts		

Quiz

Aussage	wahr	falsch
Eine Komponente kann immer nur mit einem einzigen Dekorierer versehen werden		×
Entwurfsmuster erleichtern die Kommunikation im Team	×	
	×	
Eine Komposition ist eine Aggregation, bei der die Teile keine Daseinsberechtigung ohne das Ganze haben	×	
Das Lastenheft ist eine Verfeinerung des Pflichtenhefts		×

Quiz

Aussage	wahr	falsch
Testfälle sollten schon während der Implementierungsphase geschrieben werden		
Kontrollflussorientierte Tests gehören zu den statischen Testverfahren		
Durchsichten sind auf Code und Testfälle anwendbar		
Eine Komposition ist eine Aggregation, bei der die Teile keine Daseinsberechtigung ohne das Ganze haben		
Das Lastenheft ist eine Verfeinerung des Pflichtenhefts		

Quiz

Aussage	wahr	falsch
Testfälle sollten schon während der Implementierungsphase geschrieben werden	×	
Kontrollflussorientierte Tests gehören zu den statischen Testverfahren		
Durchsichten sind auf Code und Testfälle anwendbar		
Eine Komposition ist eine Aggregation, bei der die Teile keine Daseinsberechtigung ohne das Ganze haben		
Das Lastenheft ist eine Verfeinerung des Pflichtenhefts		

Quiz

Aussage	wahr	falsch
Testfälle sollten schon während der Implementierungsphase geschrieben werden	×	
Kontrollflussorientierte Tests gehören zu den statischen Testverfahren		×
Durchsichten sind auf Code und Testfälle anwendbar		
Eine Komposition ist eine Aggregation, bei der die Teile keine Daseinsberechtigung ohne das Ganze haben		
Das Lastenheft ist eine Verfeinerung des Pflichtenhefts		

Quiz

Aussage	wahr	falsch
Testfälle sollten schon während der Implementierungsphase geschrieben werden	×	
Kontrollflussorientierte Tests gehören zu den statischen Testverfahren		×
Durchsichten sind auf Code und Testfälle anwendbar	×	
Eine Komposition ist eine Aggregation, bei der die Teile keine Daseinsberechtigung ohne das Ganze haben		
Das Lastenheft ist eine Verfeinerung des Pflichtenhefts		

Quiz

Aussage	wahr	falsch
Testfälle sollten schon während der Implementierungsphase geschrieben werden	×	
Kontrollflussorientierte Tests gehören zu den statischen Testverfahren		×
Durchsichten sind auf Code und Testfälle anwendbar	×	
Eine Komposition ist eine Aggregation, bei der die Teile keine Daseinsberechtigung ohne das Ganze haben	×	
Das Lastenheft ist eine Verfeinerung des Pflichtenhefts		

Quiz

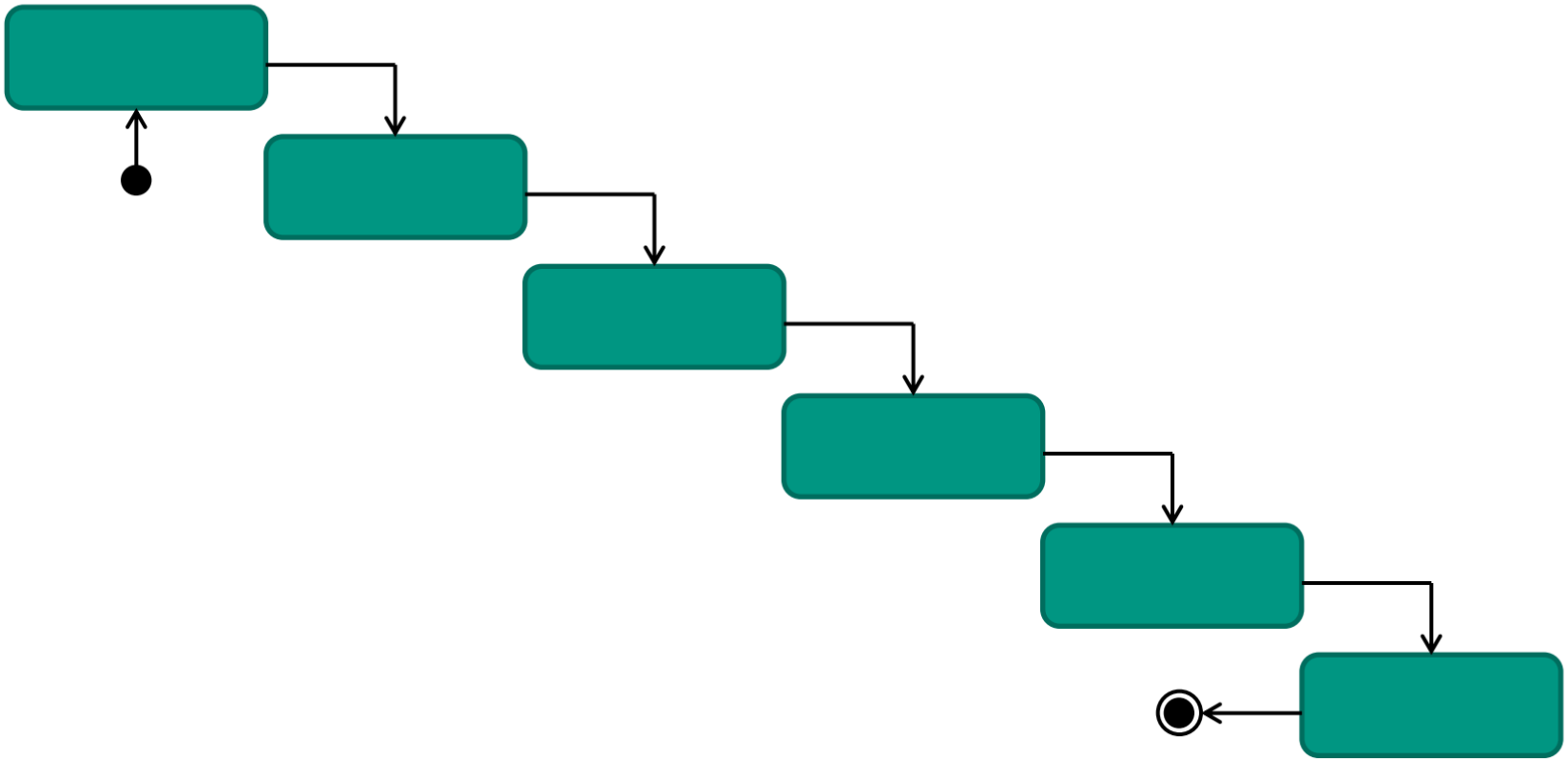
Aussage	wahr	falsch
Testfälle sollten schon während der Implementierungsphase geschrieben werden	×	
Kontrollflussorientierte Tests gehören zu den statischen Testverfahren		×
Durchsichten sind auf Code und Testfälle anwendbar	×	
Eine Komposition ist eine Aggregation, bei der die Teile keine Daseinsberechtigung ohne das Ganze haben	×	
Das Lastenheft ist eine Verfeinerung des Pflichtenhefts		×

Lastenheft / Pflichtenheft

1. Zielbestimmung
2. Produkteinsatz
3. Produktumgebung
4. Funktionale Anforderungen
5. Produktdaten
6. Nichtfunktionale Anforderungen
7. Globale Testfälle
8. Systemmodelle
   a. Szenarien
   b. Anwendungsfälle
   c. Objektmodell
   d. Dynamische Modelle
   e. Benutzerschnittstelle – Bildschirmskizzen, Navigationspfade
9. Glossar

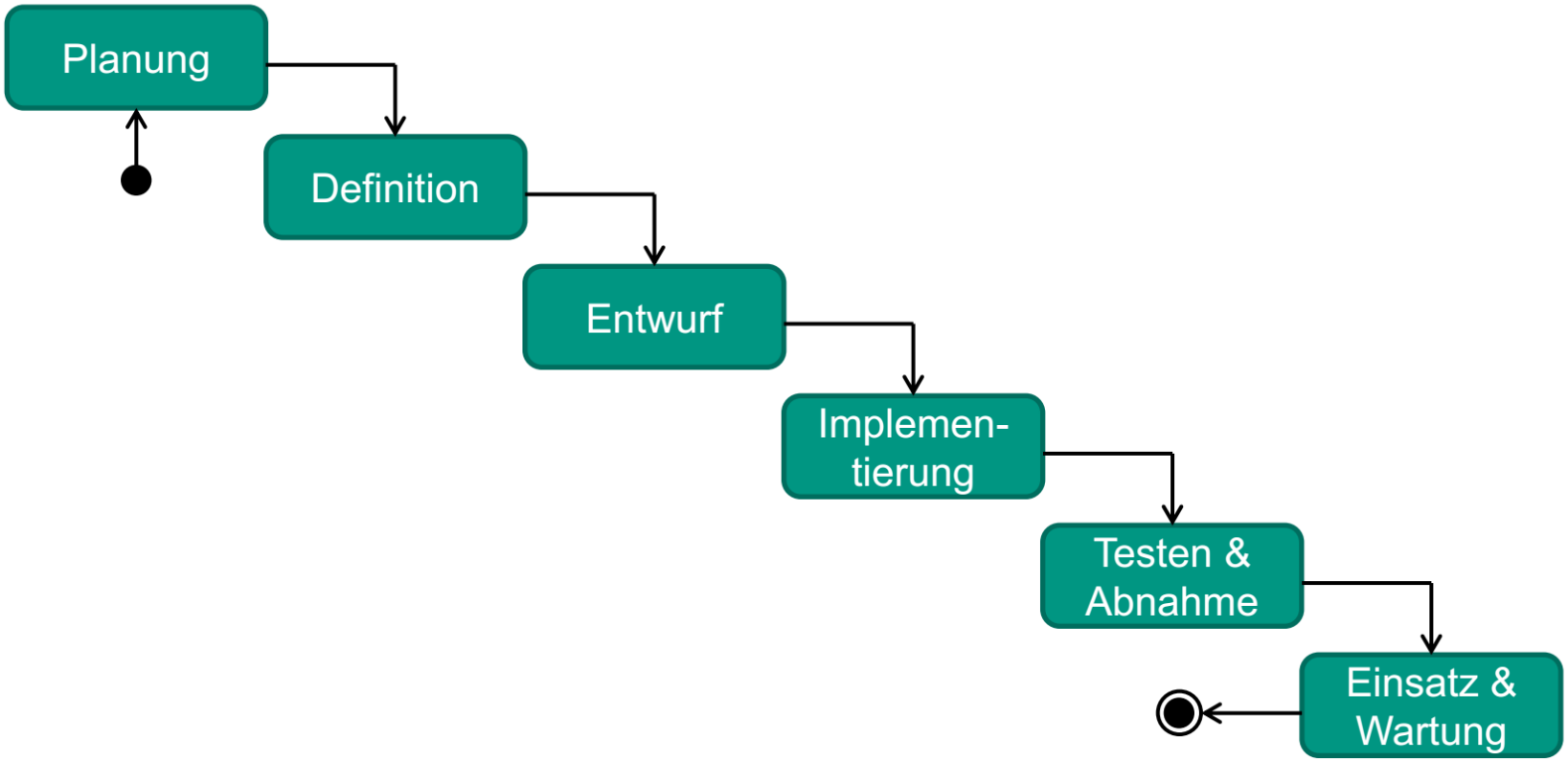
Aufwärmen

Vervollständigen Sie folgendes Schaubild des Wasserfallmodells, indem Sie in die vorgegebenen Kästchen die Namen der Phasen des Wasserfallmodells in der richtigen Reihenfolge eintragen



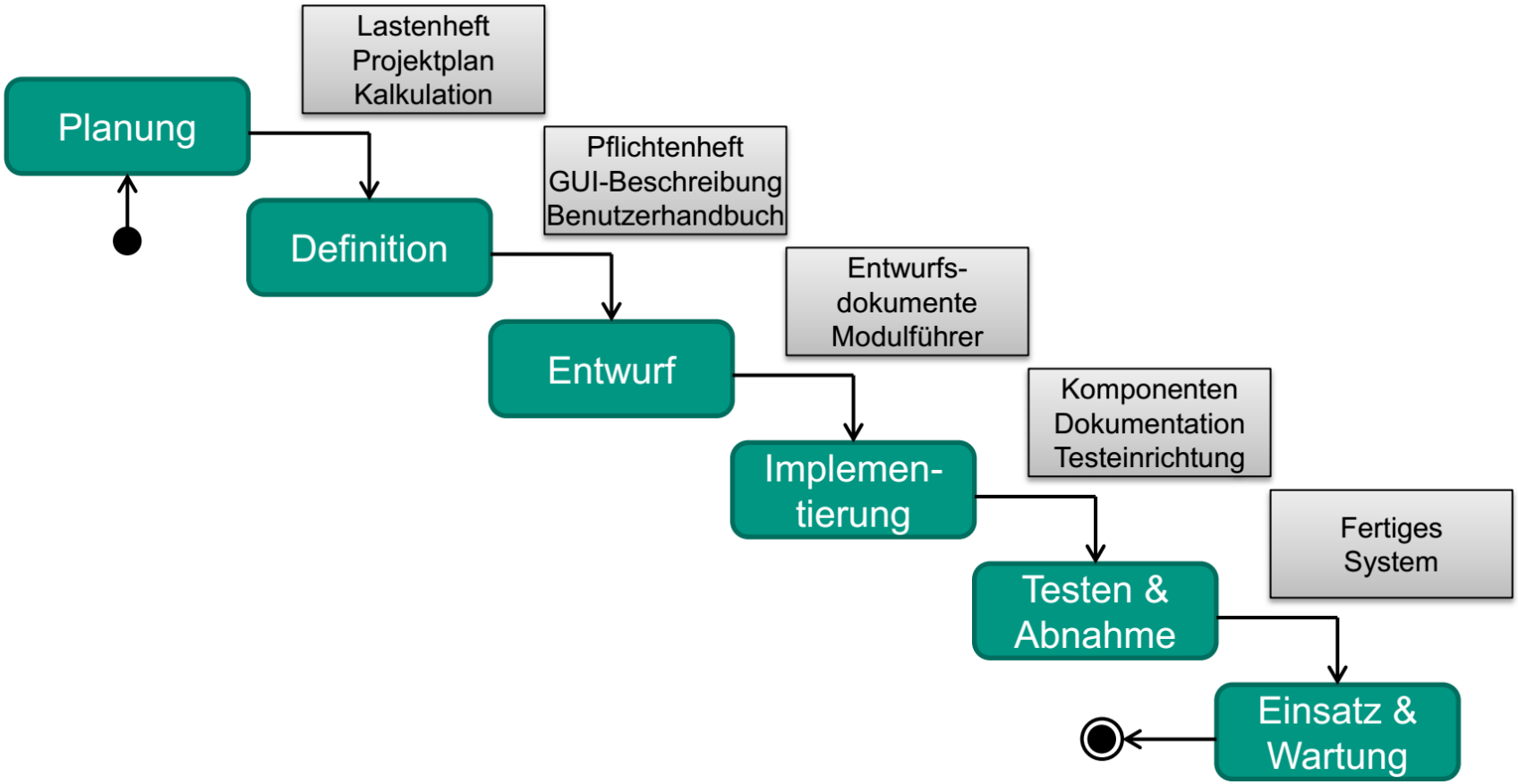
Aufwärmen

Vervollständigen Sie folgendes Schaubild des Wasserfallmodells, indem Sie in die vorgegebenen Kästchen die Namen der Phasen des Wasserfallmodells in der richtigen Reihenfolge eintragen



Aufwärmen

Vervollständigen Sie folgendes Schaubild des Wasserfallmodells, indem Sie in die vorgegebenen Kästchen die Namen der Phasen des Wasserfallmodells in der richtigen Reihenfolge eintragen



Prozessmodelle

Prozessmodelle

- Programmieren durch Probieren
- Wasserfallmodell
- V-Modell
- Prototypenmodell
- Iteratives Modell
- Synchronisiere und Stabilisiere
- Agile Methoden (spez. Extreme Programming)

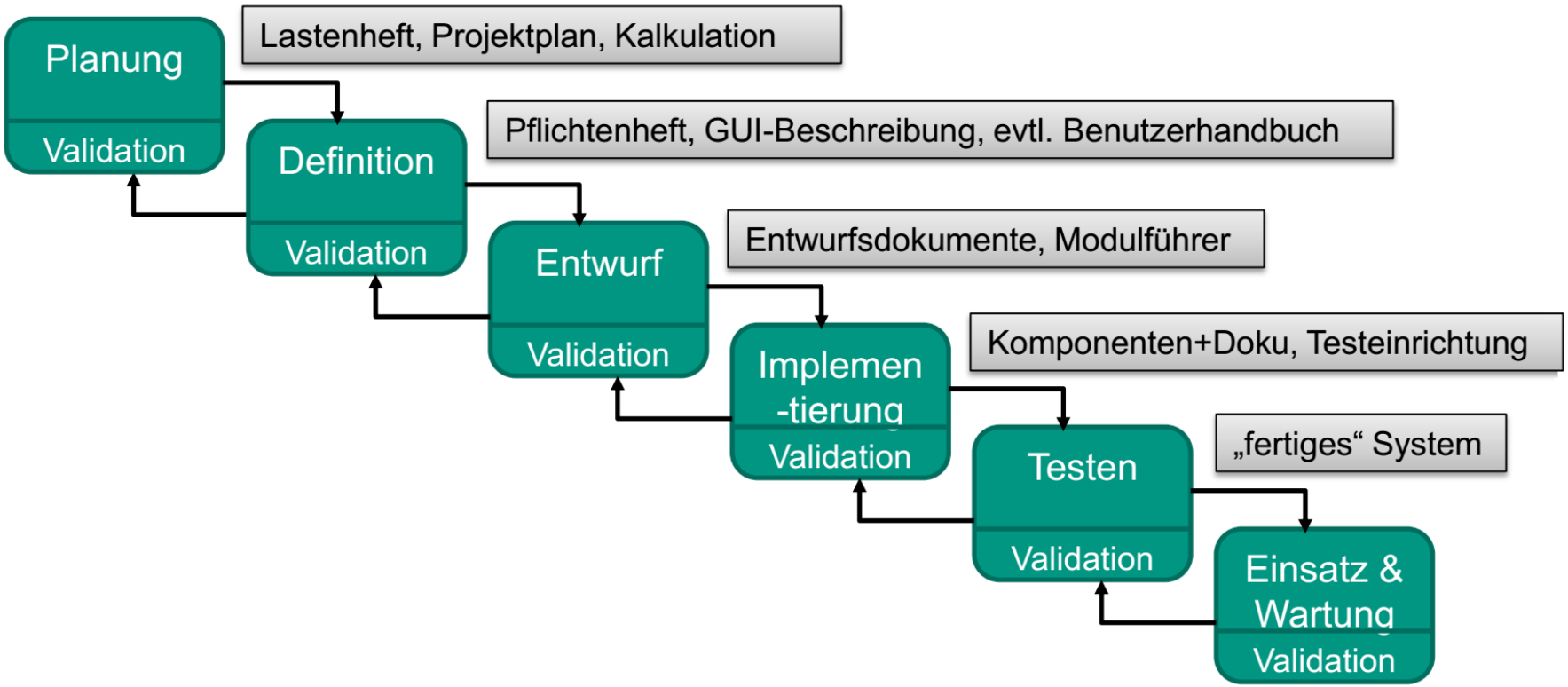
Programmieren durch Probieren

- Vorgehen
  - Vorläufiges Programm erstellen
  - Über Anforderungen, Entwurf, Testen und Wartung nachdenken
  - Programm entsprechend anpassen
- Vorteile
  - Sofort Code, kein „nutzloser“ Zusatzaufwand
- Nachteile
  - Code schlecht strukturiert, da Entwurfsphase fehlt
  - Mangelhafte Aufgabenerfüllung, da keine Anforderungsanalyse
  - Wartung/Pflege kostspielig, da Programm nicht darauf vorbereitet
  - Keine Dokumentation
  - Keine Teamarbeit möglich, da Aufgabenaufteilung nicht vorgesehen

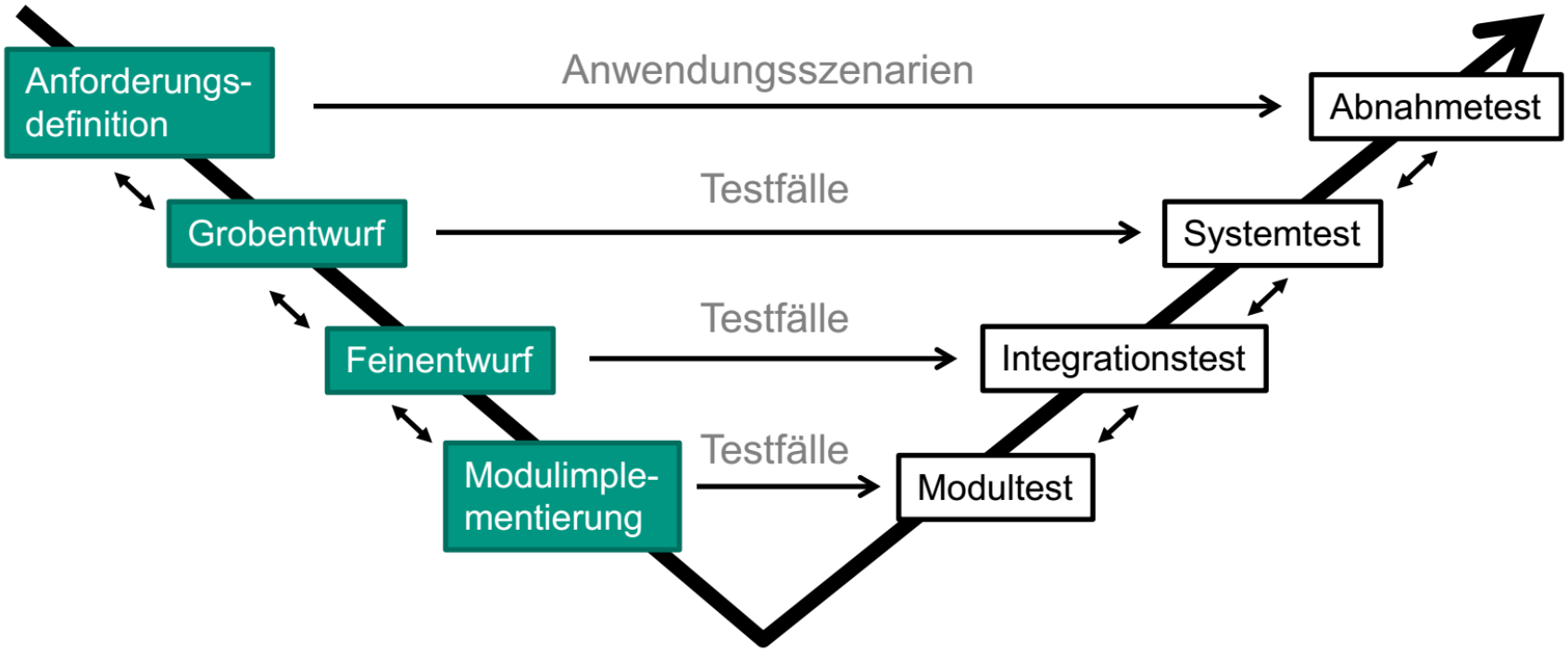
Wasserfallmodell

- Vorgehen
  - Fest vorgegebene Reihenfolge der Aktivitäten
  - Am Ende jeder Phase steht ein fertiges Dokument
    $\rightarrow$ „Dokumentgetriebenes“ Modell
- Vorteile
  - Einfach
  - Verständlich
- Nachteile
  - Benutzerbeteiligung nur in Definitionsphase
  - Keine phasenübergreifende Rückkopplung vorgesehen
    $\rightarrow$ Fehlersuche und Korrektur problematisch
  - Parallelisierungspotential möglicherweise nicht ausgeschöpft
    $\rightarrow$ Markteinführung verzögert sich unnötig
  - Zwang zur genauen Spezifikation schlecht verstandener Funktionen
    $\rightarrow$ Entwurf, Implementierung und Testen von später nutzlosem Code

Wasserfallmodell



V-Modell


Anforderungs-
definition
Anwendungsszenarien
Abnahmetest
Grobentwurf
Testfälle
Systemtest
Feinentwurf
Testfälle
Integrationstest
Modulimple-
mentierung
Testfälle
Modultest

Prototyp

- ist geeignet, wenn ohne Experimentation keine vollständige Spezifikation erstellt werden kann
- kann Arbeitsmoral und Vertrauen zwischen Anbieter und Kunden stärken
- Prototyp wird weggeworfen

Erweiterung Prototyp: Iteratives Modell

- Versuch, mehr weiter zu verwenden als beim Prototypmodell
- Idee: Teile der Funktionalität lassen sich klar definieren und realisieren
- Funktionalität Schritt für Schritt erstellen und dem Produkt „hinzufügen“
- Gleiche Vorteile und Einsatzgebiete wie Prototypmodell

Synchronisiere und Stabilisiere

- Auch „Microsoft-Modell“
- Ansatz
  - Organisation der 200 Programmierer eines Projektes (z.B. Windows 95) in „kleinen Hacker-Teams“
    $\rightarrow$ Freiheit für eigene Ideen/Entwürfe
  - Regelmäßige Synchronisation (nächtlich)
  - Regelmäßige Stabilisierung (Meilensteine, 3 Monate)
- Drei Phasen
  - Planungsphase (3-12 Monate)
  - Entwicklungsphase in 3 Meilensteinen (9 Monate)
  - Stabilisierungsphase (3-8 Monate)

Synchronisiere und Stabilisiere

- Pro
  - Effektiv durch kurze Produktzyklen
  - Priorisierung nach Funktionen
  - Natürliche Modularisierung nach Funktionen
  - Fortschritt auch ohne vollständige Spezifikation möglich
  - Viele Entwickler arbeiten in kleinen Teams und damit genau so effektiv wie wenige
  - Rückmeldungen können frühzeitig einfließen
- Kontra
  - Ungeeignet für manche Art von Software – z.B. bei enger Kopplung der Komponenten
  - Mündliche Arbeitsweise: Ad-hoc-Prozesse in jedem Team, kein Lernen über Teamgrenzen
  - Alle 18 Monate sind 50% des Codes überarbeitet worden (hohe Kosten)

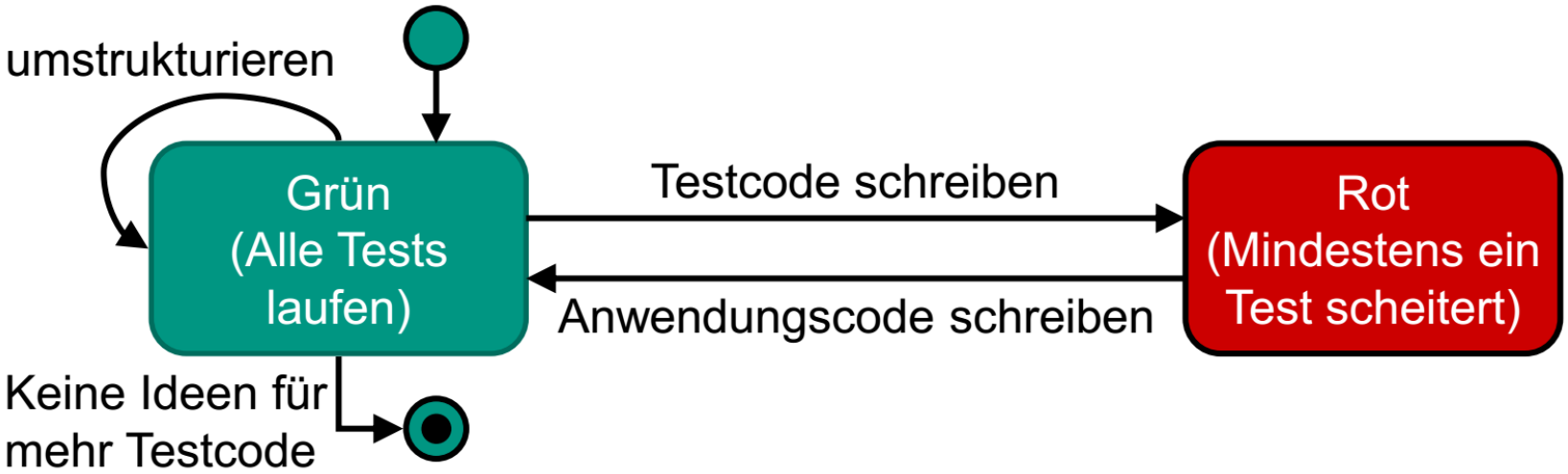
Agile Prozesse

- Minimum an Vorausplanung
- Inkrementelle Planung
- Vermeiden unterstützender Dokumente
- Schnelle Reaktion auf Änderungen
- Einbeziehung des Kunden in die Entwicklung
- Vertreter: z.B. Extreme Programming (XP), Scrum

Extreme Programming

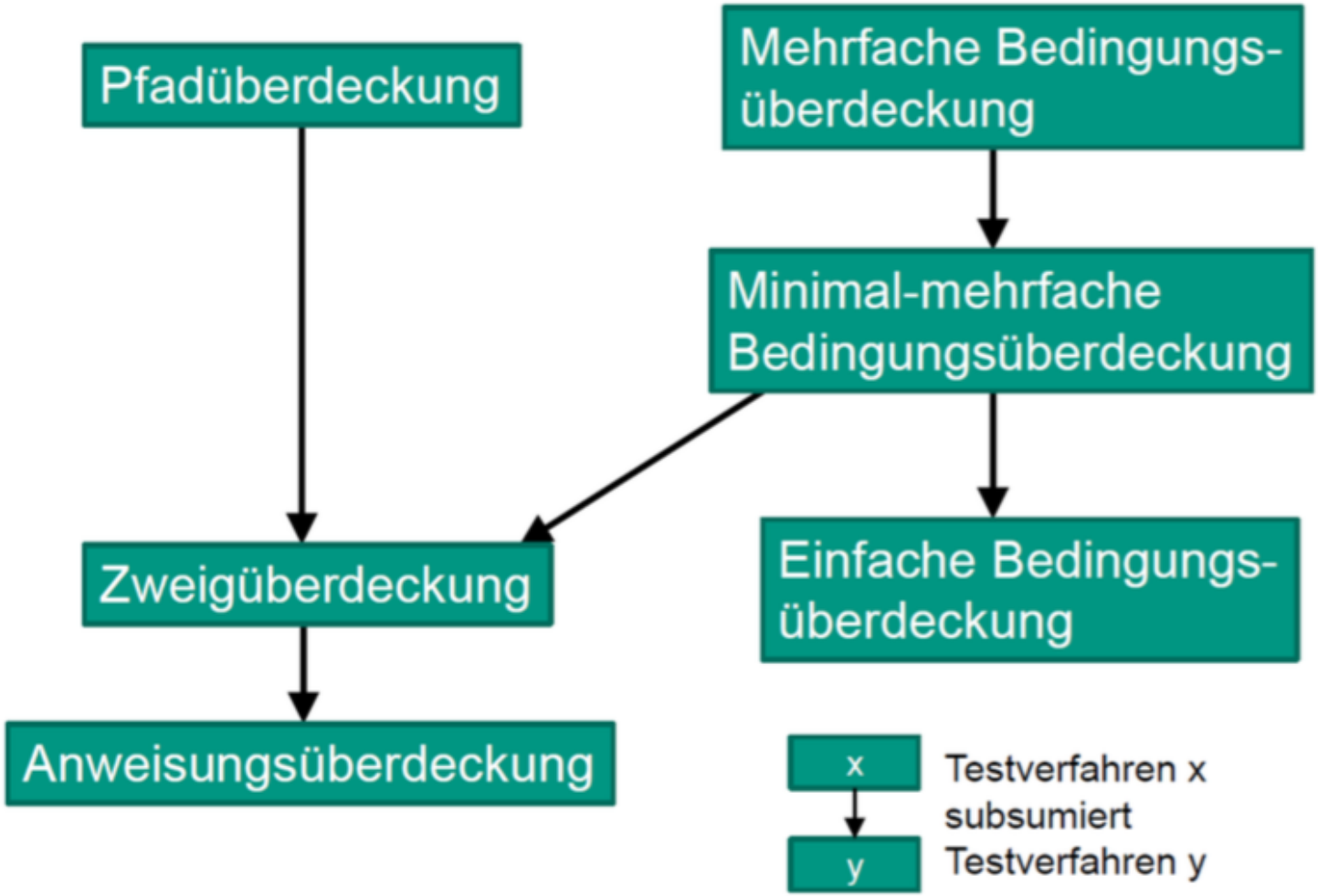
- Keine Code-Inspektionen
- ... dafür: Paarprogrammierung
  $\rightarrow$ Höhere Qualität des Codes beim Entstehen
- Testgetriebene Entwicklung
  - Programmierer schreiben automatische Komponententests
  - Kunde spezifiziert Akzeptanztests, die das Team implementiert
  - Andere Testarten auch möglich, z.B. Stresstest
  - Testausführung häufig und automatisch
- Inkrementeller Entwurf durch Umstrukturierungen
- Aktive Beteiligung des Kunden
- Fortlaufende Integration und Umstrukturierung des Codes
- Gemeinsamer Code-Besitz

Testgetriebene Entwicklung - Zustandsdiagramm



- Testcode vor Anwendungscode schreiben
- Kleine Schritte (nicht mehr als eine Methode)
- Inkrementeller Entwurf (nur so viel, wie gebraucht wird; kein vorausschauender Entwurf)

Subsumieren



Tutorienübersicht

Tutorium 1 / Kennenlerntutorium

- Versionskontrolle: GIT vs SVN
- Maven

Tutorium 2

- Wasserfallmodell
- Lastenheft & Durchführbarkeitsanalyse
- UML

Tutorium 3

- Aktivitätsdiagramme
- Sequenzdiagramme
- Zustandsdiagramme
- Linguistische Analyse

Tutorium 4

- Übersicht: Entwurfsmuster
- Entkopplungsmuster
  - Adapter
  - Beobachter
  - Vermittler

Tutorium 5

- Variantenmuster
  - Schablonenmethode
  - Kompositum
  - Dekorierer
- Sequenzdiagramme

Tutorium 6

- Einführung Parallelität
- Kontrollflussgraph

Tutorium 7

- Übersicht Prozessmodelle
- Subsumierung

Aufgaben

Kontrollflussorientierte Testverfahren - Klausuraufgabe SS 2009

a) Kreuzen Sie an, ob die Aussage wahr oder falsch ist

Aussage	wahr	falsch
Die Anweisungsüberdeckung subsumiert die einfache Bedingungsüberdeckung		
Die Zweigüberdeckung subsumiert die minimalmehrfache Bedingungsüberdeckung		
Die Pfadüberdeckung subsumiert die Anweisungsüberdeckung		
Bei der mehrfachen Bedingungsüberdeckung ist die Größe der minimalen Testfallmenge unabhängig davon, ob Kurzauswertung vorgenommen wird oder nicht		

Kontrollflussorientierte Testverfahren - Klausuraufgabe SS 2009

a) Kreuzen Sie an, ob die Aussage wahr oder falsch ist

Aussage	wahr	falsch
Die Anweisungsüberdeckung subsumiert die einfache Bedingungsüberdeckung		×
Die Zweigüberdeckung subsumiert die minimalmehrfache Bedingungsüberdeckung		
Die Pfadüberdeckung subsumiert die Anweisungsüberdeckung		
Bei der mehrfachen Bedingungsüberdeckung ist die Größe der minimalen Testfallmenge unabhängig davon, ob Kurzauswertung vorgenommen wird oder nicht		

Kontrollflussorientierte Testverfahren - Klausuraufgabe SS 2009

a) Kreuzen Sie an, ob die Aussage wahr oder falsch ist

Aussage	wahr	falsch
Die Anweisungsüberdeckung subsumiert die einfache Bedingungsüberdeckung		×
Die Zweigüberdeckung subsumiert die minimalmehrfache Bedingungsüberdeckung		×
Die Pfadüberdeckung subsumiert die Anweisungsüberdeckung		
Bei der mehrfachen Bedingungsüberdeckung ist die Größe der minimalen Testfallmenge unabhängig davon, ob Kurzauswertung vorgenommen wird oder nicht		

Kontrollflussorientierte Testverfahren - Klausuraufgabe SS 2009

a) Kreuzen Sie an, ob die Aussage wahr oder falsch ist

Aussage	wahr	falsch
Die Anweisungsüberdeckung subsumiert die einfache Bedingungsüberdeckung		×
Die Zweigüberdeckung subsumiert die minimalmehrfache Bedingungsüberdeckung		×
Die Pfadüberdeckung subsumiert die Anweisungsüberdeckung	×	
Bei der mehrfachen Bedingungsüberdeckung ist die Größe der minimalen Testfallmenge unabhängig davon, ob Kurzauswertung vorgenommen wird oder nicht		

Kontrollflussorientierte Testverfahren - Klausuraufgabe SS 2009

a) Kreuzen Sie an, ob die Aussage wahr oder falsch ist

Aussage	wahr	falsch
Die Anweisungsüberdeckung subsumiert die einfache Bedingungsüberdeckung		×
Die Zweigüberdeckung subsumiert die minimalmehrfache Bedingungsüberdeckung		×
Die Pfadüberdeckung subsumiert die Anweisungsüberdeckung	×	
Bei der mehrfachen Bedingungsüberdeckung ist die Größe der minimalen Testfallmenge unabhängig davon, ob Kurzauswertung vorgenommen wird oder nicht	×	

Kontrollflussorientierte Testverfahren - Klausuraufgabe SS 2009

b) Beschreiben Sie, wie man eine `do-while`-Schleife in die strukturerhaltende Zwischensprache aus der Vorlesung überführen kann. Geben Sie zur Veranschaulichung das entstehende Zwischensprachprogramm für unten stehende `do-while`-Schleife an

```
do {
   statements;
} while (condition);
```

Kontrollflussorientierte Testverfahren - Klausuraufgabe SS 2009

Karlsruher Institut für Technologie

Musterlösung

1. `do`-Zeile entfernen
2. Befehle `statements` der Schleife übernehmen
3. `while (condition)` in
   `if (condition) goto „1. Befehl in der Schleife“`
   umwandeln (Keine Negation der Bedingung)

Programm in Zwischensprache:

```
(10   -)
20   statements;
30   if (condition) goto 20
```

Kontrollflussorientierte Testverfahren - Klausuraufgabe SS 2009

c) Gegeben sei folgende Java-Methode:

```
01  public static double median(double[] d) {
02    double median = Double.NaN;
03    if (d != null && d.length > 0) {
04      if (d.length == 1) { median = d[0];
05      } else {
06        Arrays.sort(d);  // sortiert d aufsteigend
07        int mid = d.length / 2;
08        if (d.length % 2 != 0) { median = d[mid];
09        } else { median = (d[mid - 1] + d[mid]) / 2; }
10      }
11    }
12    return median;
13  }
```

Begründen Sie: Was wäre die Folge, wenn man das && in Zeile 3 durch ein & ersetzt?

Kontrollflussorientierte Testverfahren - Klausuraufgabe SS 2009

Musterlösung

Keine Kurzauswertung $\rightarrow$ Bei `null` als Eingabe gäbe es eine `NullPointerExeption` bei `d.length`

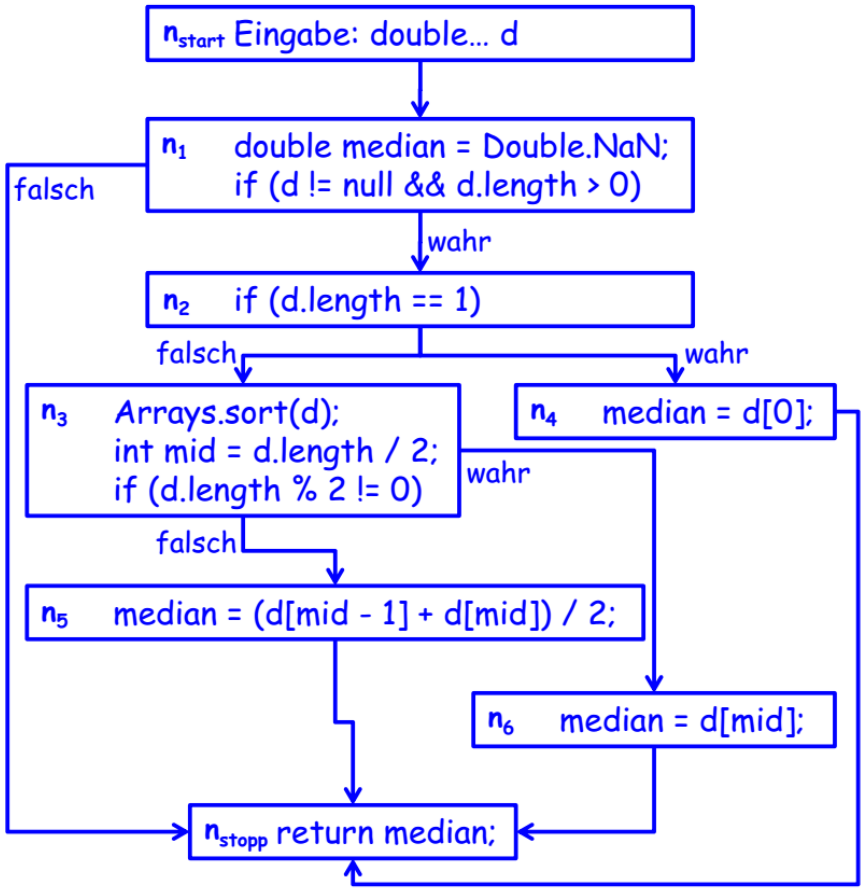
Kontrollflussorientierte Testverfahren - Klausuraufgabe SS 2009

d) Erstellen Sie auf der folgenden Seite den Kontrollflussgraphen der Methode `median(...)`. Bitte schreiben Sie den Quelltext in Kästchen, Verweise auf die Zeilennummern der Methode sind nicht ausreichend

```
01  public static double median(double[] d) {
02    double median = Double.NaN;
03    if (d != null && d.length > 0) {
04      if (d.length == 1) { median = d[0];
05      } else {
06        Arrays.sort(d);  // sortiert d aufsteigend
07        int mid = d.length / 2;
08        if (d.length % 2 != 0) { median = d[mid];
09        } else { median = (d[mid - 1] + d[mid]) / 2; }
10      }
11    }
12    return median;
13  }
```

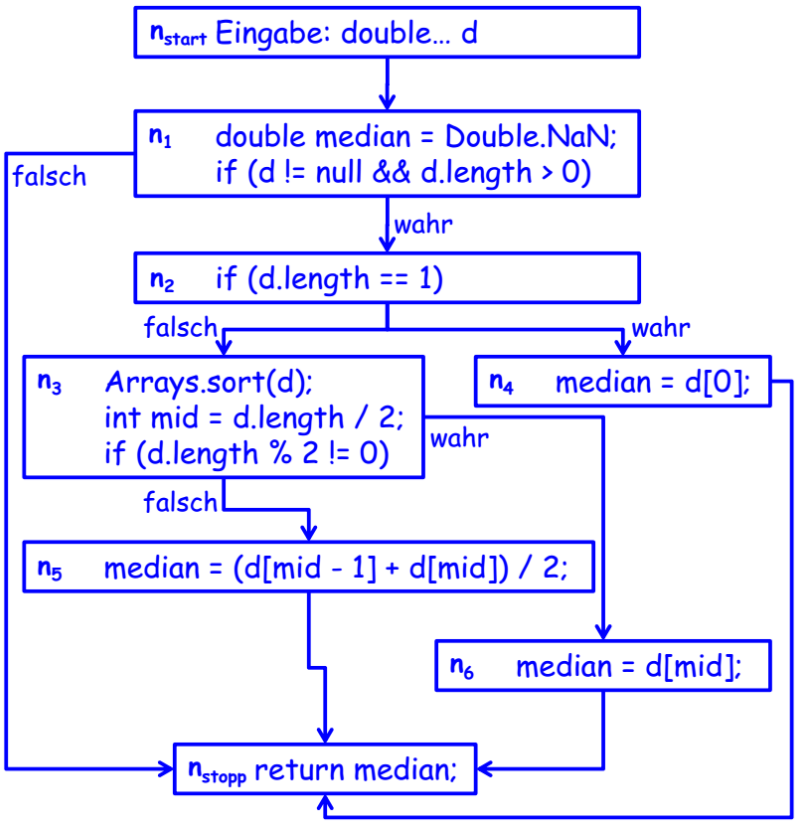
Kontrollflussorientierte Testverfahren - Klausuraufgabe SS 2009

Musterlösung



Kontrollflussorientierte Testverfahren - Klausuraufgabe SS 2009

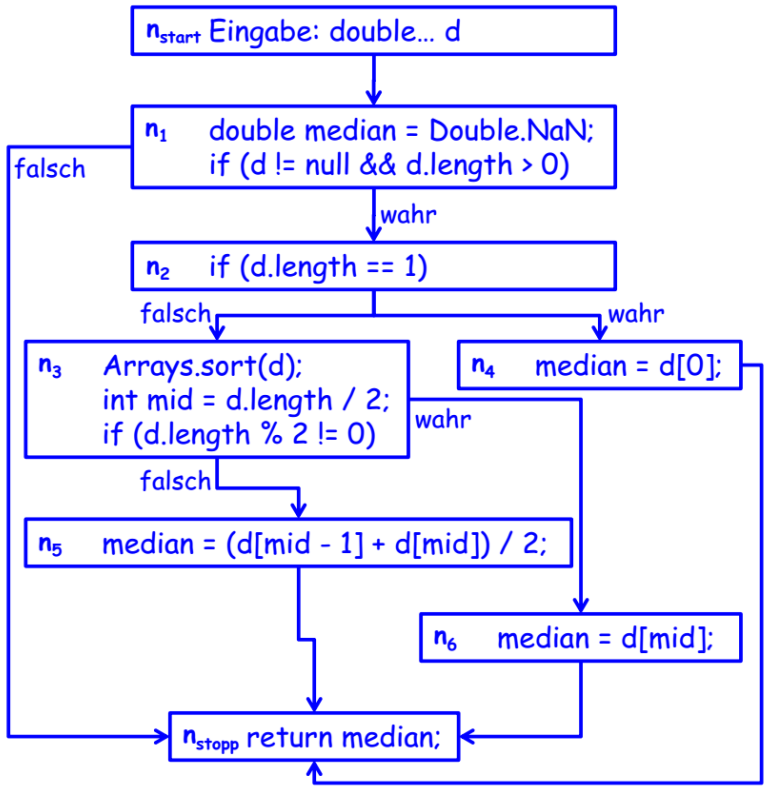
e) Geben Sie eine minimale Testfallmenge an, welche für die Methode `median(...)` die Anweisungsüberdeckung erfüllt. Geben Sie die durchlaufenen Pfade an



Kontrollflussorientierte Testverfahren - Klausuraufgabe SS 2009

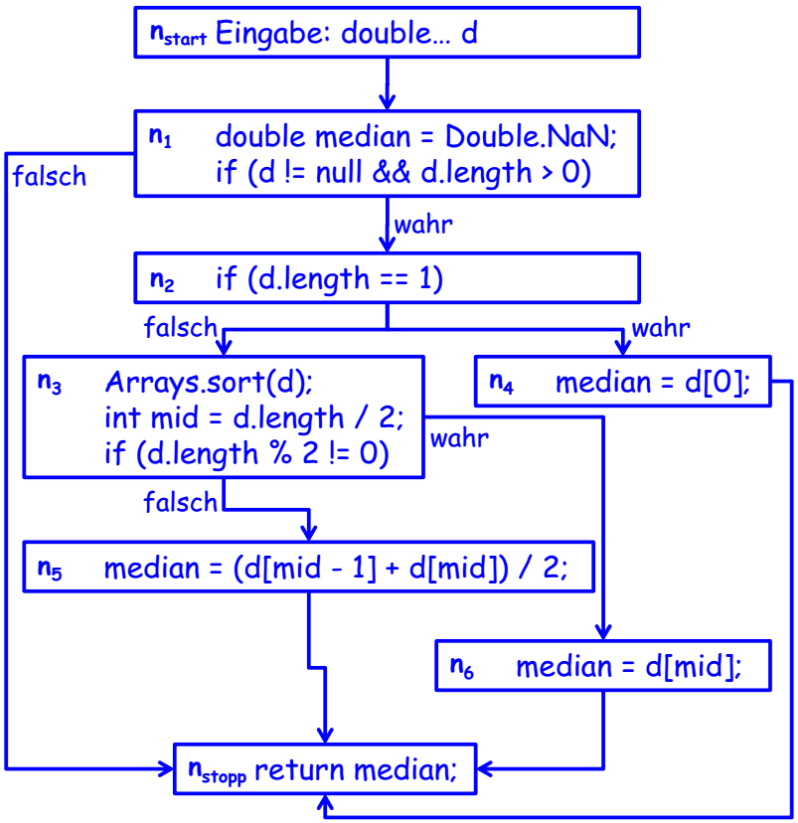
Musterlösung

$$\{1\} : n_{start}, n_1, n_2, n_4, n_{stopp}$$
$$\{1,2\} : n_{start}, n_1, n_2, n_3, n_5, n_{stopp}$$
$$\{1,2,3\} : n_{start}, n_1, n_2, n_3, n_6, n_{stopp}$$



Kontrollflussorientierte Testverfahren - Klausuraufgabe SS 2009

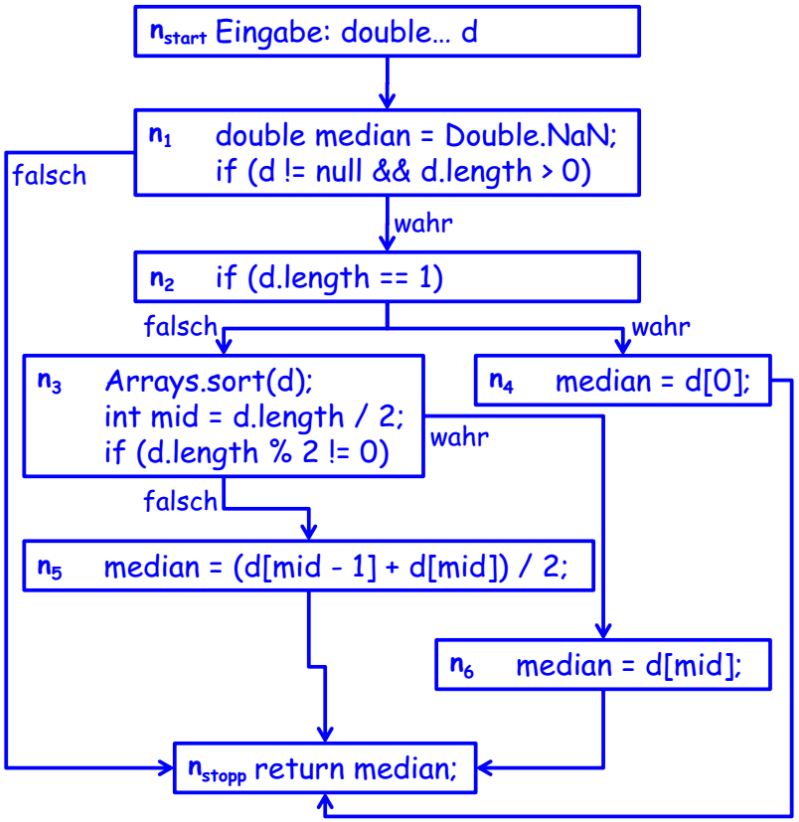
f) Ergänzen Sie die Testfallmenge aus e) so, dass Sie eine min. Menge erhalten, welche die Zweigüberdeckung für die Methode `median(...)` erfüllt. Geben Sie für die neuen Testfälle die durchlaufenen Pfade an



Kontrollflussorientierte Testverfahren - Klausuraufgabe SS 2009

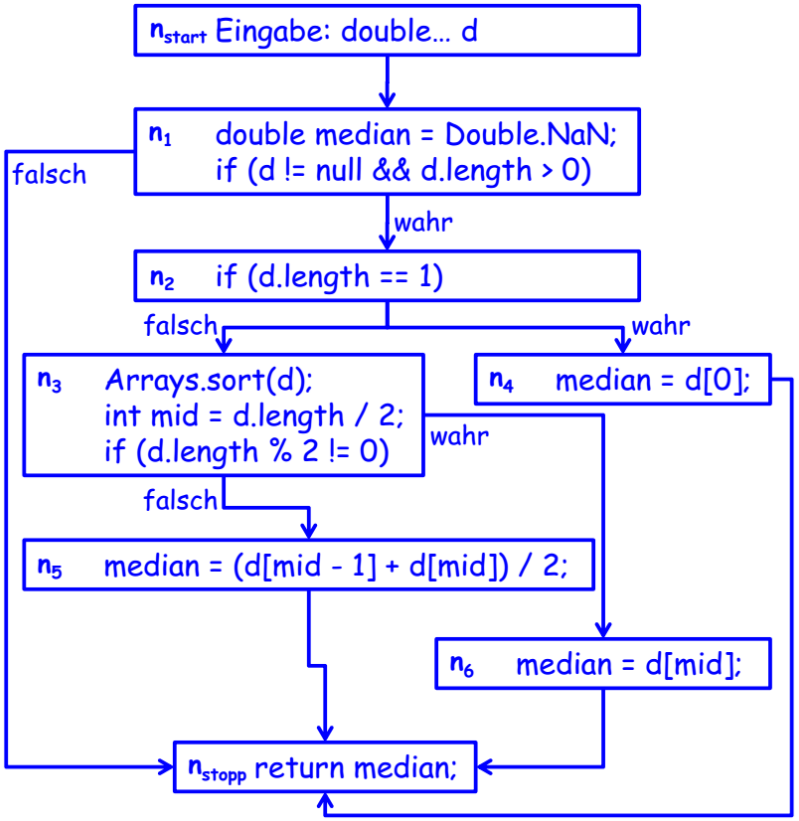
Musterlösung

null oder $\{\}$: n_{start}, n_1, n_{stopp}



Kontrollflussorientierte Testverfahren - Klausuraufgabe SS 2009

g) Erfüllt die minimale Testfallmenge aus f), welche die Zweigüberdeckung erfüllt, auch die Pfadüberdeckung für die Methode `median(...)`? Begründen Sie Ihre Antwort



Kontrollflussorientierte Testverfahren - Klausuraufgabe SS 2009

Musterlösung

Ja, es gibt in diesem Fall so viele Pfade wie Zweige (allgemein gilt das nicht)

Qualitätssicherung - Klausuraufgabe SS 2011

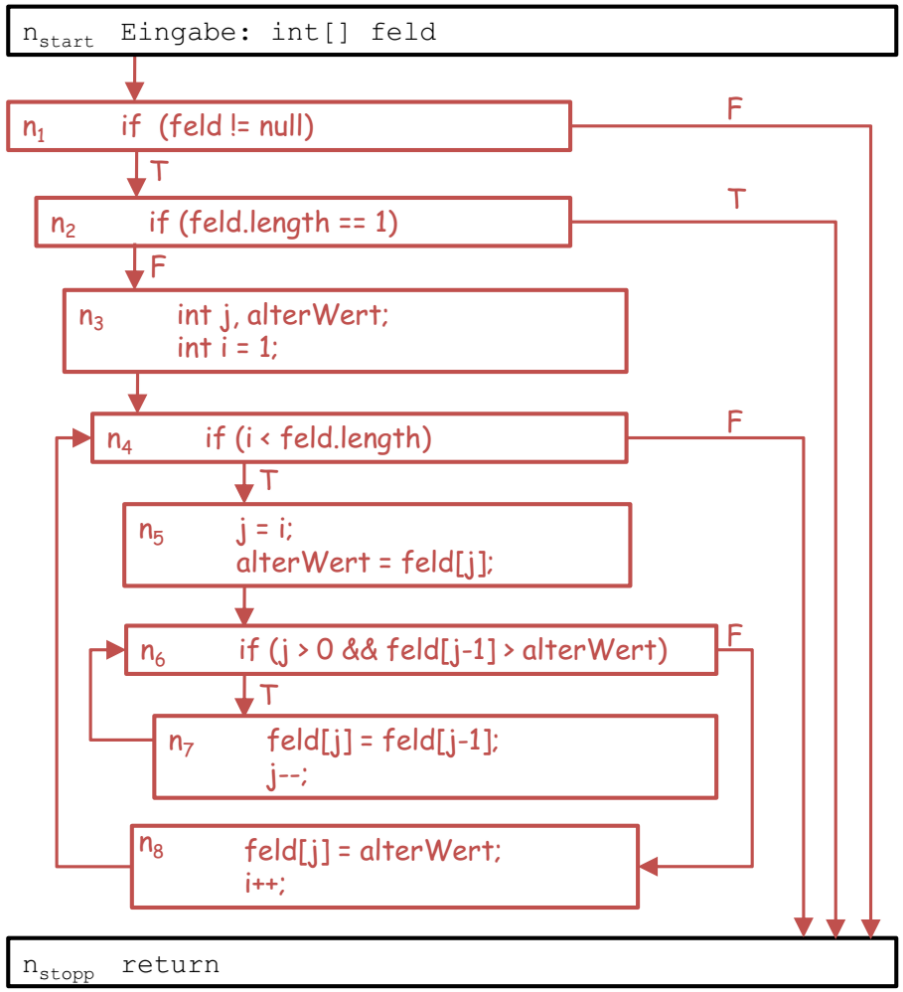
a) Gegeben sei die Methode `sortiere(...)`, die ein Feld von Ganzzahlen mittels Sortieren durch Einfügen aufsteigend sortiert:

```
01  public void sortiere(int[] feld) {
02    if (feld != null) {
03      if (feld.length == 1) { return; }
04      else {
05        int j, alterWert;
06        for (int i = 1; i < feld.length; i++) {
07          j = i;
08          alterWert = feld[i];
09          while (j > 0 && feld[j - 1] > alterWert) {
10            feld[j] = feld[j - 1]; j--;
11          }
12        feld[j] = alterWert;
13        } } } }
```

Erstellen Sie den Kontrollflussgraphen der Methode `sortiere(...)`

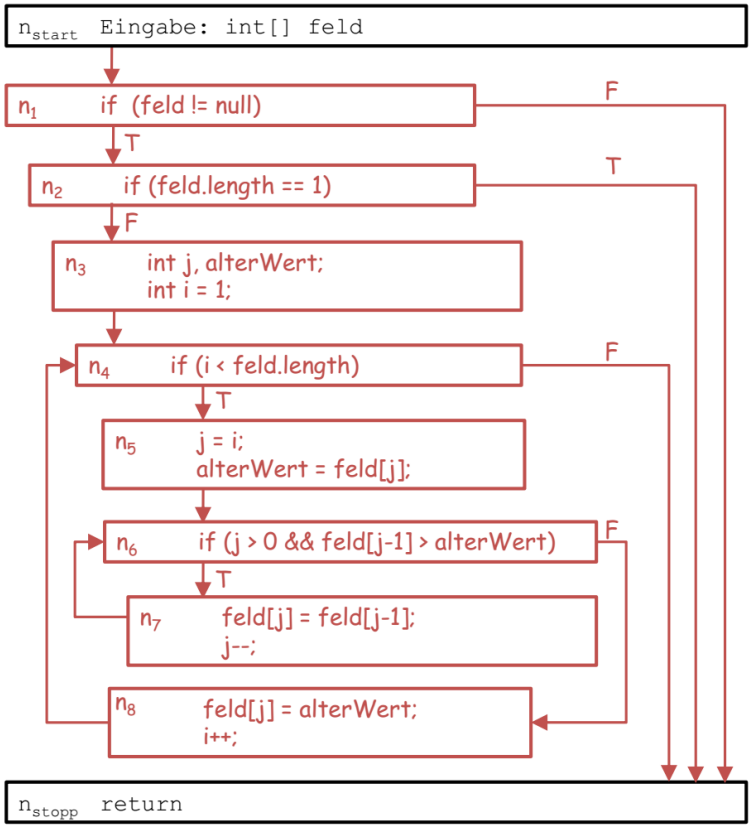
Qualitätssicherung - Klausuraufgabe SS 2011

Musterlösung



Qualitätssicherung - Klausuraufgabe SS 2011

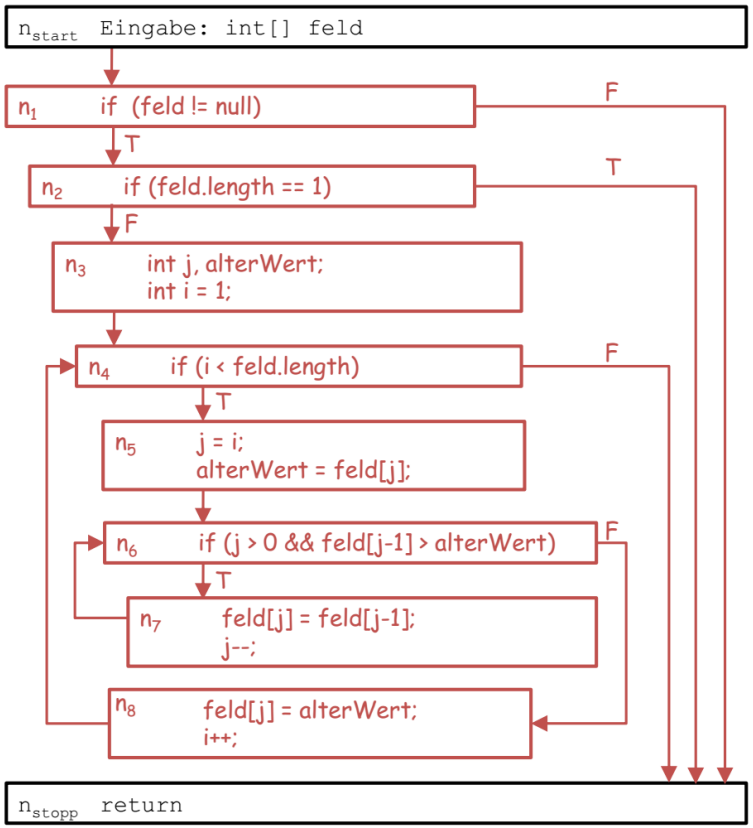
b) Geben Sie eine minimale Testfallmenge (Eingabevektor) an, die die Anweisungsüberdeckung für die Methode `sortiere(...)` erfüllt. Geben Sie die durchlaufenen Pfade an



Qualitätssicherung - Klausuraufgabe SS 2011

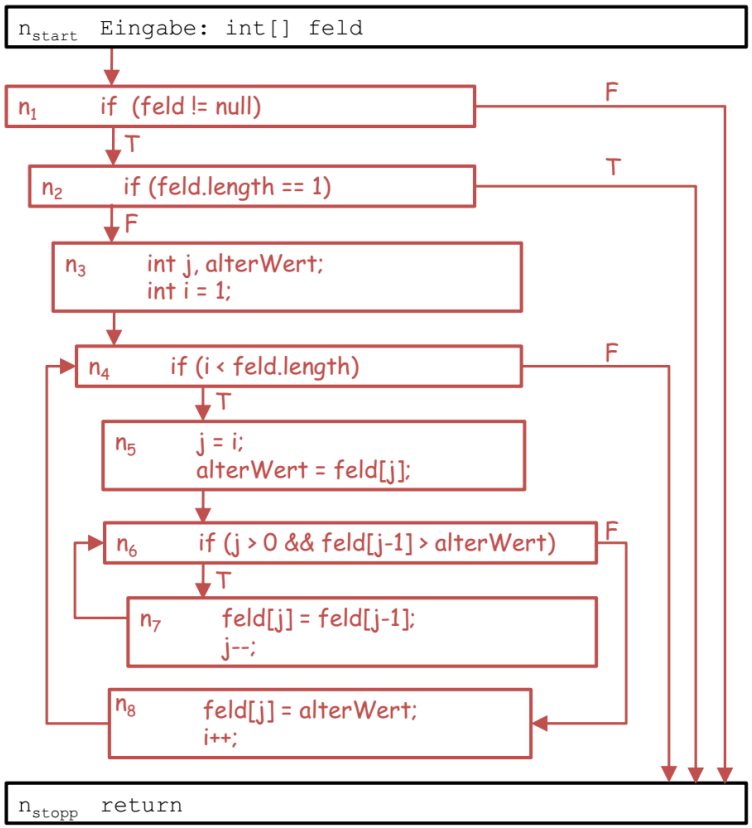
Musterlösung

$$\{2, 1\} : n_{start}, n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_6, n_8, n_4, n_{stopp}$$



Qualitätssicherung - Klausuraufgabe SS 2011

c) Ergänzen Sie die Testfallmenge aus Teil b) so, dass Sie eine minimale Menge erhalten, die die Zweigüberdeckung der Methode `sortiere(...)` erfüllt. Geben Sie für die neue Testfallmenge die durchlaufenen Pfade an



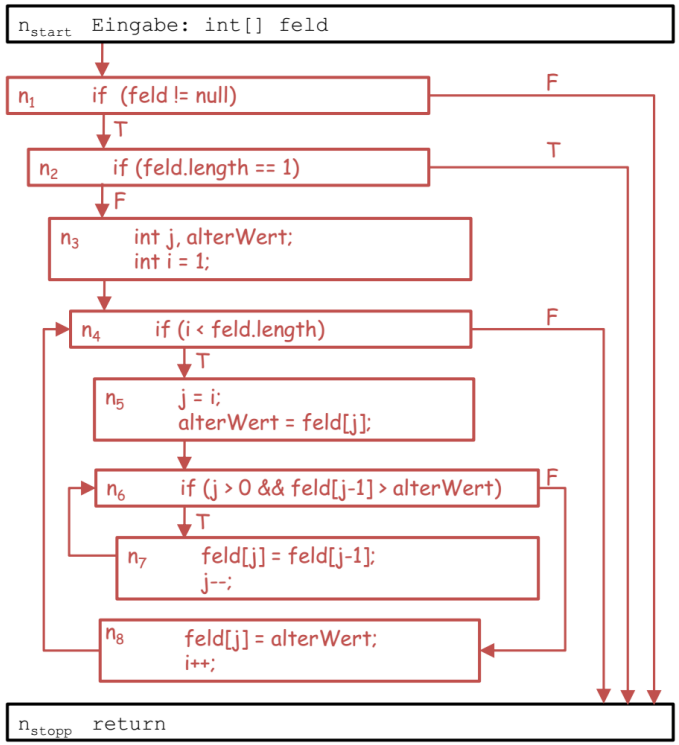
Qualitätssicherung - Klausuraufgabe SS 2011

Musterlösung

$$\{2,1\} : n_{start}, n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_6, n_8, n_4, n_{stopp}$$

$$\text{null} : n_{start}, n_1, n_{stopp}$$

$$\{1\} : n_{start}, n_1, n_2, n_{stopp}$$



Qualitätssicherung - Klausuraufgabe SS 2011

d) Was passiert, wenn die Bedingung der `while`-Schleife aus Zeile 9 wie folgt abgeändert wird? `j > 0 & feld[j - 1] > alterWert`

```
01  public void sortiere(int[] feld) {
02    if (feld != null) {
03      if (feld.length == 1) { return; }
04      else {
05        int j, alterWert;
06        for (int i = 1; i < feld.length; i++) {
07          j = i;
08          alterWert = feld[i];
09          while (j > 0 && feld[j - 1] > alterWert) {
10            feld[j] = feld[j - 1]; j--;
11          }
12        feld[j] = alterWert;
13        } } } }
```

Qualitätssicherung - Klausuraufgabe SS 2011

Musterlösung

Keine Kurzauswertung $\rightarrow$ Wenn `j` den Wert 0 besitzt, wird versucht, auf das Element -1 des Feldes `feld` zuzugreifen:
`ArrayIndexOutOfBoundException`

Organisatorisches

Klausur

- Nicht vergessen für Klausur und Übungsschein anzumelden!

- Hauptklausur: 10.08.2015, 14 Uhr
- Nachklausur: 06.10.2015, 11 Uhr
  - Ein Punkt pro Minute
- Restliche Übungsblätter kann man bald beim Übungsleiter abholen

Viel Glück bei der Klausur!