Ausdrucksstarke ausführbare Schnittstellenspezifikationen für Java

Peter Müller
Jörg Meyer
Arnd Poetzsch-Heffter
Fernuniversität Hagen

Gliederung

- Einführung in Schnittstellenspezifikationen
- Ausführbare Schnittstellenspezifikationen
- Spezifikation von Seiteneffekten
- Implementierung
- Zusammenfassung

Einführung

- Zielsetzung
  - Präzise Dokumentation
  - Test-Unterstützung
  - Verifikation
- Spezifikation des Verhaltens von Programmen
  - Vor- und Nachbedingungen für Methoden
  - Invarianten für Klassen
- Spezifikationstechniken
  - deklarativ: basierend auf Logik
  - ausführbar: basierend auf Ausdrücken der Programmiersprache

Ausführbare Spezifikationen

- Vorteile
  - einfache Erlernbarkeit
  - einfache Anwendung
  - gute Unterstützung für Test und Fehlersuche
- Existierende Spezifikationssprachen
  - Eiffel
  - JaWa (JIT ‘98)

Probleme existierender Sprachen

```
interface I {
   public void m();
}

class T implements I {
   int a;
   List b;

   public void m()
      pre    a > 0
      post   a-- >= 0   { ... }
}
```

Probleme existierender Sprachen

- unklare Semantik

```
interface I {
   public void m();
}

class T implements I {
   int a;
   List b;

   public void m()
      pre    a > 0
      post   a-- >= 0   { ... }
}
```

Probleme existierender Sprachen

- unklare Semantik
- fehlende Unterstützung von Abstraktion

```
interface I {
   public void m();
}

class T implements I {
   int a;
   List b;

   public void m()
      pre    a > 0
      post   a-- >= 0   { ... }
}
```

Probleme existierender Sprachen

- unklare Semantik
- fehlende Unterstützung von Abstraktion
- mangelhafte Beschreibung von Seiteneffekten auf Geflechten

```
interface I {
   public void m();
}

class T implements I {
   int a;
   List b;

   public void m()
      pre    a > 0
      post   a-- >= 0   { ... }
}
```

Programmbeispiel

```
class Component  {
   Container  parent;  int x;  int y;  Object placeHolder;
   ...  }

class Container extends Component  {
   int ncomponents;
   Component[] component;
   int getCount()                          { return ncomponents; }
   Component getComponent(int i)    { return component[i]; }
   Component add(Component c)       { ... }
   ... }
```

Spezifikationsprimitive

- seiteneffektfreie Java-Audrücke
  - kein schreibender Attributzugriff
  - keine Objekterzeugung
  - nur seiteneffektfreie Methoden: Observer
- spezielle Spezifikationsausdrücke
  - Zugriff auf Werte im Vorzustand: old(exp)
  - beschränkte Quantifizierung über int:
    forall (i: n..m): exp(i)
  - Primitive für Objekterzeugung, Erreichbarkeit usw.

Eine Methodik zur Abstraktion

- Beschreibung abstrakter Eigenschaften durch Observer

Eine Methodik zur Abstraktion

- Beschreibung abstrakter Eigenschaften durch Observer
  - Auszeichnung vorhandener Methoden

```
observer int getCount()                          { ... }
observer Component getComponent(int i)           { ... }
```

Eine Methodik zur Abstraktion

- Beschreibung abstrakter Eigenschaften durch Observer
  - Auszeichnung vorhandener Methoden

```
observer int getCount()                          { ... }
observer Component getComponent(int i)           { ... }
```

  - Hinzufügen neuer Methoden

```
observer boolean contains(Component c)           { ... }
observer boolean wf()                            { ... }
```

Eine Methodik zur Abstraktion

- Beschreibung abstrakter Eigenschaften durch Observer
  - Auszeichnung vorhandener Methoden

```
observer int getCount()                          { ... }
observer Component getComponent(int i)           { ... }
```

  - Hinzufügen neuer Methoden

```
observer boolean contains(Component c)           { ... }
observer boolean wf()                            { ... }
```

- Spezifikation von Observern
  - informell
  - durch andere Observer

Spezifikationsbeispiel

Eigenschaften von add:

Voraussetzungen:

Garantien:

Component add(Component c)

pre

post

Spezifikationsbeispiel

Eigenschaften von add:

Voraussetzungen:

– Container und Component wohlgeformt

Garantien:

Component add(Component c)

pre

wf() &&

c.wf() &&

post

Spezifikationsbeispiel

Eigenschaften von add:

Voraussetzungen:

- Container und Component wohlgeformt
- c in keinem Container

Garantien:

```
Component add(Component c)
  pre
     wf()                          &&
     c.wf()                        &&
     c.getParent()==null
  post
```

Spezifikationsbeispiel

Eigenschaften von add:

Voraussetzungen:

- Container und Component wohlgeformt
- c in keinem Container

Garantien:

- Container bleibt wohlgeformt

```
Component add(Component c)
  pre
     wf()                          &&
     c.wf()                        &&
     c.getParent()==null
  post
     wf()                          &&
```

Spezifikationsbeispiel

Eigenschaften von add:

Voraussetzungen:

- Container und Component wohlgeformt
- c in keinem Container

Garantien:

- Container bleibt wohlgeformt
- add gibt c zurück

```
Component add(Component c)
  pre
    wf()                          &&
    c.wf()                        &&
    c.getParent()==null
  post
    wf()                          &&
    result == old(c)              &&
```

Spezifikationsbeispiel

Eigenschaften von add:

Voraussetzungen:

- Container und Component wohlgeformt
- c in keinem Container

Garantien:

- Container bleibt wohlgeformt
- add gibt c zurück
- Anzahl der Elemente steigt um 1

```
Component add(Component c)
  pre
     wf()                                  &&
     c.wf()                                &&
     c.getParent()==null
  post
     wf()                                  &&
     result == old(c)                      &&
     getCount()==old(getCount())+1         &&
```

Spezifikationsbeispiel

Eigenschaften von add:

Voraussetzungen:

- Container und Component wohlgeformt
- c in keinem Container

Garantien:

- Container bleibt wohlgeformt
- add gibt c zurück
- Anzahl der Elemente steigt um 1
- c wird Element mit höchstem Index

```
Component add(Component c)
 pre
    wf()                                  &&
    c.wf()                                &&
    c.getParent()==null
 post
    wf()                                  &&
    result == old(c)                      &&
    getCount()==old(getCount())+1         &&
    getComponent(getCount())==old(c)
```

Spezifikationsbeispiel

Eigenschaften von add:

Voraussetzungen:

- Container und Component wohlgeformt
- c in keinem Container

Garantien:

- Container bleibt wohlgeformt
- add gibt c zurück
- Anzahl der Elemente steigt um 1
- c wird Element mit höchstem Index
- schon enthaltene Komponenten bleiben unverändert

```
Component add(Component c)
 pre
    wf()                                  &&
    c.wf()                                &&
    c.getParent()==null
 post
    wf()                                  &&
    result == old(c)                      &&
    getCount()==old(getCount())+1         &&
    getComponent(getCount())==old(c)
```

?

Spezifikationsbeispiel

Eigenschaften von add:

Voraussetzungen:

- Container und Component wohlgeformt
- c in keinem Container

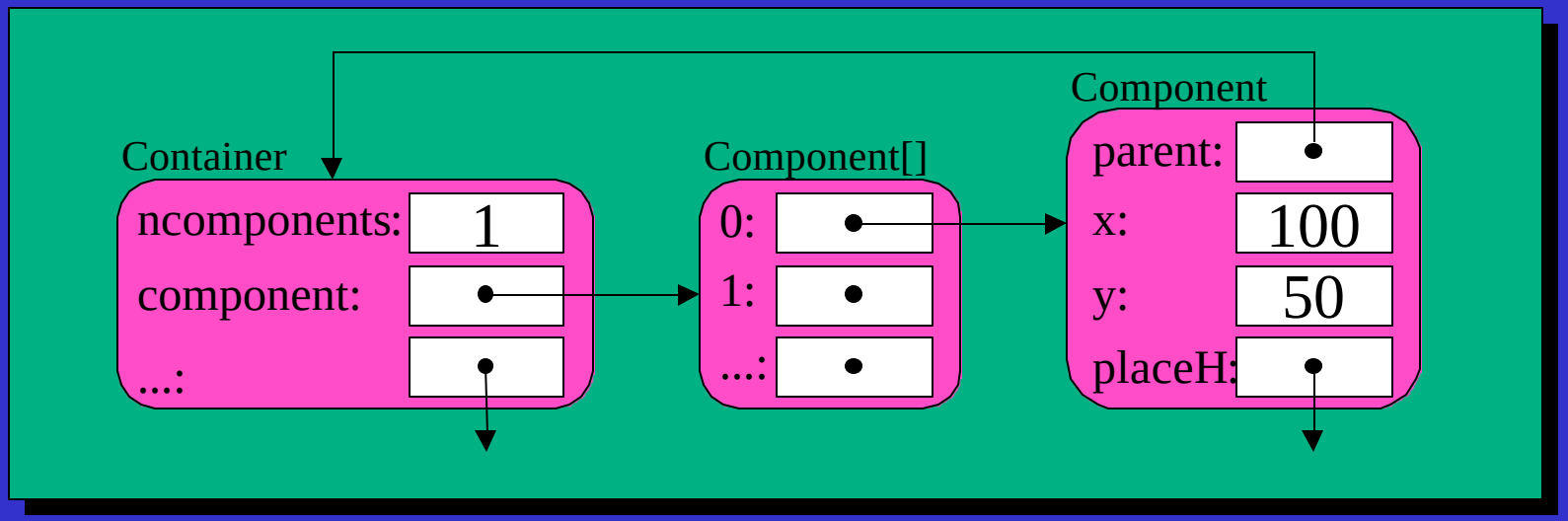
Garantien:

- Container bleibt wohlgeformt
- add gibt c zurück
- Anzahl der Elemente steigt um 1
- c wird Element mit höchstem Index
- schon enthaltene Komponenten bleiben unverändert
- bis auf das parent-Feld bleibt c unverändert

```
Component add(Component c)
 pre
    wf()                                  &&
    c.wf()                                &&
    c.getParent()==null
 post
    wf()                                  &&
    result == old(c)                      &&
    getCount()==old(getCount())+1         &&
    getComponent(getCount())==old(c)
```

?

Spezifikation von Frame Properties

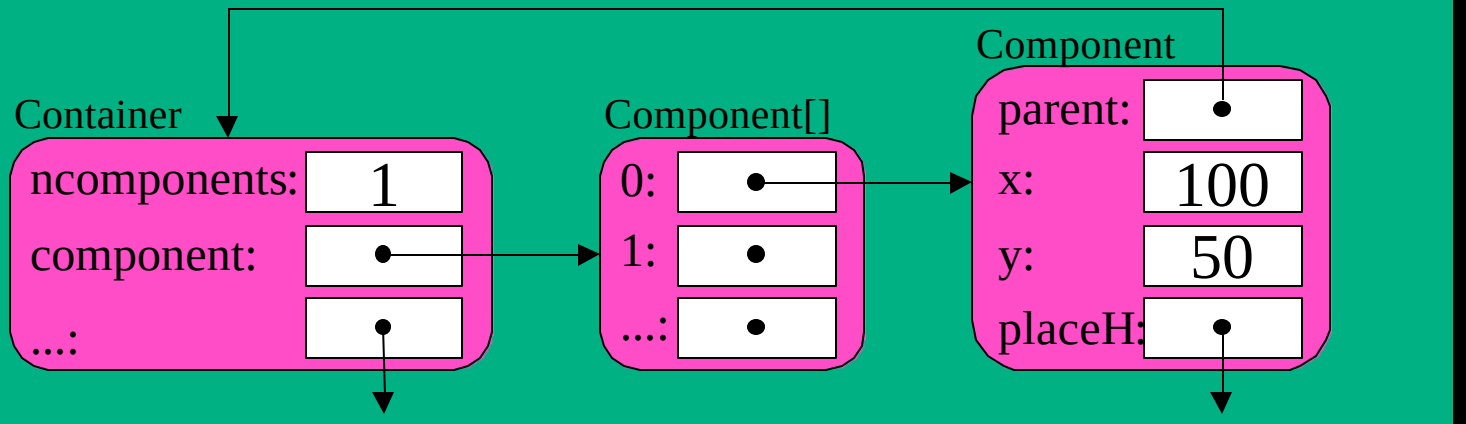


Anforderungen:

- flexible Beschreibung von (Teil-)Strukturen
- Unterstützung von Abstraktion
- frühzeitige Entdeckung von Verletzungen
- effizient zu implementieren

Beschreibung von Objektstrukturen

- Beschreibung als Menge von Locations
- Basistyp `structure` mit Ausdrücken < >, <e.f>, +
- Beschreibung durch Observer-Methoden
  - vordefinierter Observer getReachLocs




```
observer structure ohneParent()
   { return placeHolder.getReachLocs() + <this.x> + <this.y>; }
```

Der unchanged-Ausdruck

unchanged(S): Struktur S bleibt unverändert

```
forall (i: 1..old(getCount()):
        unchanged(getComonent(i).ohneParent())
```

Mögliche Umsetzungen von unchanged(S)

1. Kopieren und Speichern der Struktur S im Vorzustand
   Vergleichen der Struktur mit der Kopie im Nachzustand

2. Markieren aller Locations von S im Vorzustand
   Dynamische Überprüfung bei jedem schreibenden Attributzugriff

Implementierungstechnik

- zu jedem Attribut ein Markierungsattribut
- zu jeder Struktur jeweils eine Methode zum
  - Markieren der Struktur
  - Löschen der Markierung
- statt schreibendem Attributzugriff Aufruf einer statischen Methode
  - Überprüfen der Markierung (ggf. Exception)
  - Schreiben des Wertes

Eigenschaften der Implementierung

- Übersetzung in reinen Java-Code
  - Implementierung als reiner Präprozessor
  - keine Änderungen der VM notwendig
- Laufzeit- und Speicher-Overhead ca. 100%
- Probleme:
  - Reflection
  - Java Native Interface

Zusammenfassung

- seiteneffektfreie Spezifikationen
- Abstraktion durch Observer-Methoden
- Spezifikation von Seiteneffekten auf Geflechten
  - Beschreibung von Objektstrukturen
  - effiziente Überprüfung der Spezifikation
  - einfache Implementierung

Weitere Informationen

- Arbeiten zur Spezifikation und Verifikation
  www.informatik.fernuni-hagen.de/pi5/publications.html