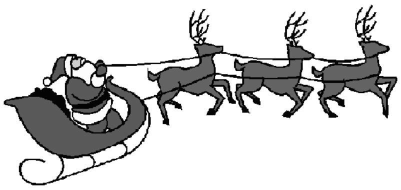
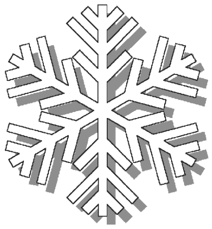
Andreas Podelski
Fachbereich 14 – Informatik
Universität des Saarlandes,
Max-Planck-Institut für Informatik





Programmiersprachen (WS 1996/97)

19. Dezember 1996

Übungsblatt 9

Abdelwaheb Ayari
Hubert Baumeister

Abgabe: 9. Januar 1997

Aufgabe 1: (3 Punkte)
Beschreibe ein Verfahren mit dem man Vorkommen von super.f($c_1, \ldots, c_n$) der Form

```
...
C f(C1 x1,..,Cn xn) {
   ...
   super.f(c1,..,cn);
   ...
}
..
```

aus den Instanzmethoden einer Klassenhierarchie entfernen kann. **Tip:** Definiere geeignete neue Instanzmethoden.
Führe das Verfahren an folgendem Beispiel durch:

```
class A {
  int a;
  void init() {
    a = 10;
  }
}
```

```
class B extends A {
  void print(){
    System.out.println(a);
 }
}
```

```
class C extends B {
  void init(){
    super.init();
    a++ ;
  }
}
```

```
class Test {
  public static void main(String[] args) {
    B b = new B(), c = new C(),
    b.init(); b.print();
    c.init(): c.print();
  }
}
```

Aufgabe 2: (2 Punkte)
Gegeben sei folgende Schnittstelle:

```
interface RATIONAL {
   int zähler();
   int nenner();
   RATIONAL plus(RATIONAL n);
   RATIONAL minus(RATIONAL n);
   RATIONAL times(RATIONAL n);
   RATIONAL divide(RATIONAL n);
}
```

Implementiere eine Klasse, deren Objekte rationale Zahlen sind und die `RATIONAL` implementieren. Rationale Zahlen werden erzeugt mit `Rational(x)` für eine ganze Zahl x und `Rational(z,n)` für z/n. Implementiere die Methode `String toString()` für rationale Zahlen, die eine rationale Zahl mit Zähler z und Nenner n in gekürzter Darstellung z'/n' liefert.
Teste die Klasse an folgenden Beispiel. Sei $r_1 =$ `new Rational(2)` und $r_2 =$ `new Rational(1,4)`, dann lasse r_1.`plus`(r_2), r_1.`minus`(r_2), r_1.`times`(r_2) und r_1.`divide`(r_2) berechnen und ausgeben.

Aufgabe 3: (5 Punkte)
Implementiere Klassen für Stacks und Queues von beliebigen Objekten mit Hilfe von Feldern und ohne Klassen aus dem Packet `java.util` zu verwenden (s.a. Übungsblatt 8). Objekte der Klassen Stack und Queue sollen die Nachrichten `boolean isEmpty()`, `void add(Object o)`, `void addAll(Object[] a)` und `Object remove()` verstehen. Wenn der Stack oder die Queue leer ist, soll `remove` den Wert `null` liefern. Außerdem soll die Methode `String toString()` implementiert werden, die eine Darstellung eines Stacks, bzw. einer Queue liefert. Entwirf eine geeignete Vererbungs- und Typhierarchie. Überlege ob es sinnvoll ist zusätzlich zu `Stack` und `Queue` weitere Klassen und Schnittstellen zu definieren. Gibt es einen Platz für ein Analogon zur Signatur `STORE` vom Übungsblatt 8? Als Test erzeuge einen Stack und eine Queue der Zahlen 1 bis 10 und gib den Stack bzw. die Queue einmal als ganzes aus (mit der Methode `toString`) und einmal elementweise.

Zusatzaufgabe: (3 Punkte) Überlege, wie man priority Queues in obige Hierarchie einbauen kann und implementiere Deinen Vorschlag.