Problem beim Vertauschen von Elementen in verketteten Listen

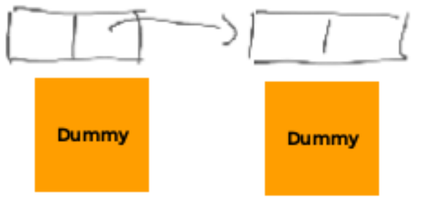
Element an Stelle 0 zu vertauschen, ist ein Sonderfall, weil der Vorgänger nicht existiert.

Lösung 1:

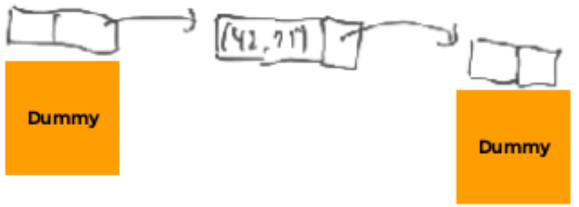
"Künstlichen" Vorgänger erzeugen.

Zwei Dummies statt einem in jeder Liste.

Leere Liste:



Nicht-leere Liste:



Lösung 1a:

Nur ein Dummy, aber Elemente bilden einen Kreis.

Problem beim Vertauschen von Elementen in verketteten Listen

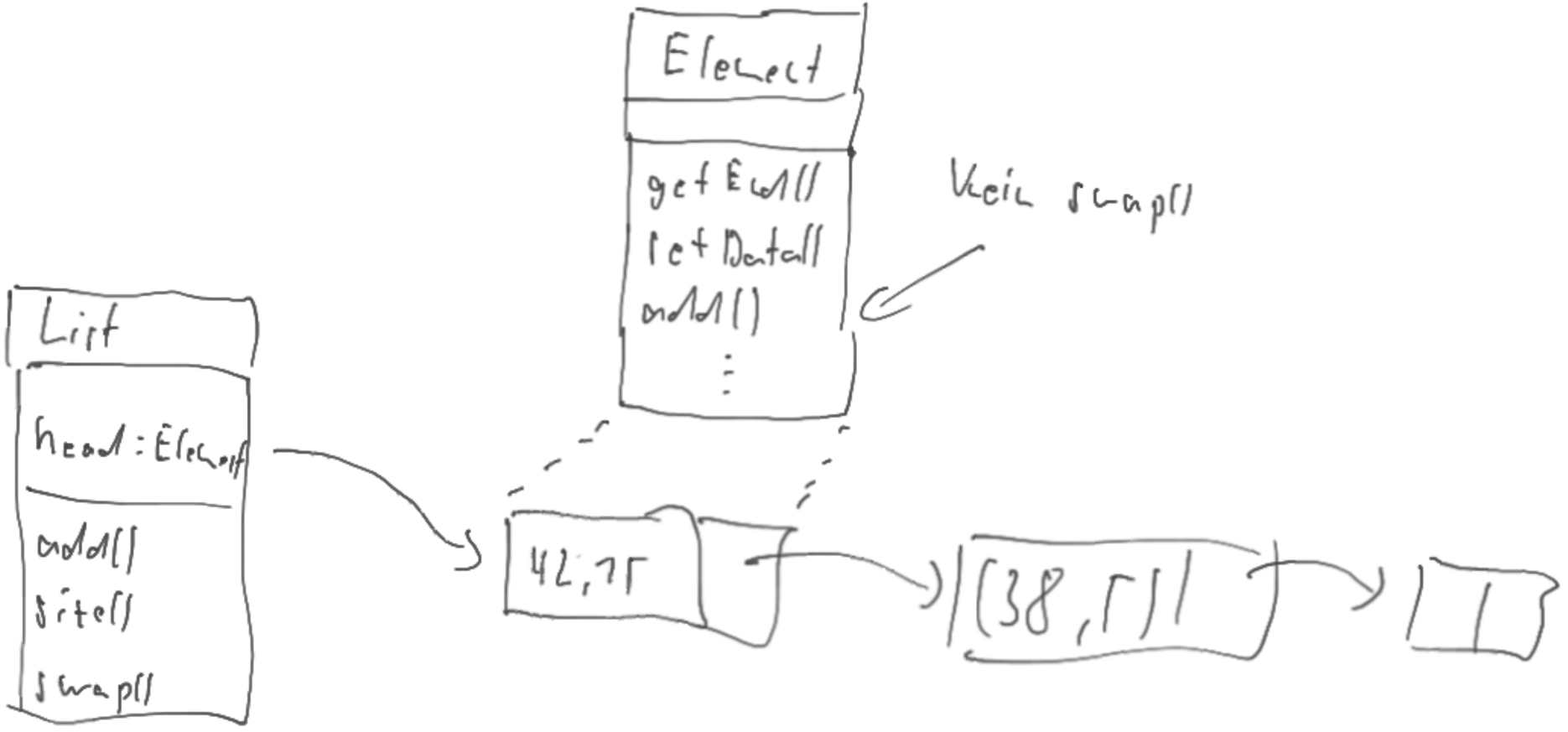
Element an Stelle 0 zu vertauschen, ist ein Sonderfall, weil der Vorgänger nicht existiert.

Lösung 2:

Zusätzliche Klasse "List"

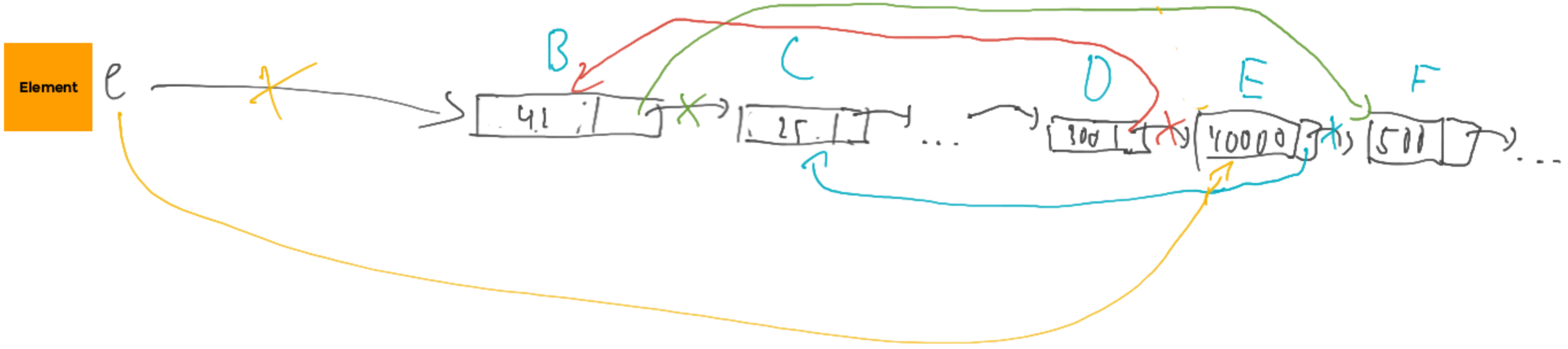
implementiert Methoden, die in Element nicht sinnvoll umgesetzt werden können.

bietet auch API einer Liste an, z.B. add() und size().



In main():

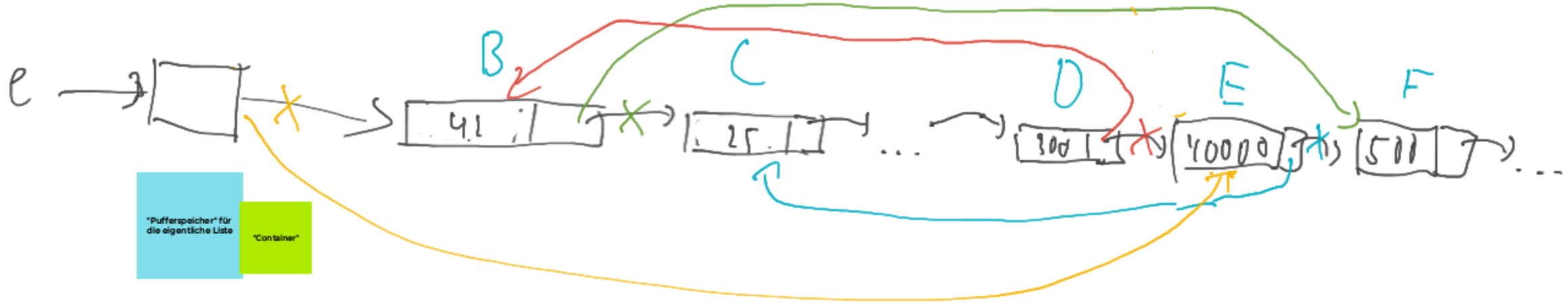
Bisherige Situation:



Änderung von e durch Aufruf von e.swap() ist kompliziert.

In main():

Neuer Ansatz:

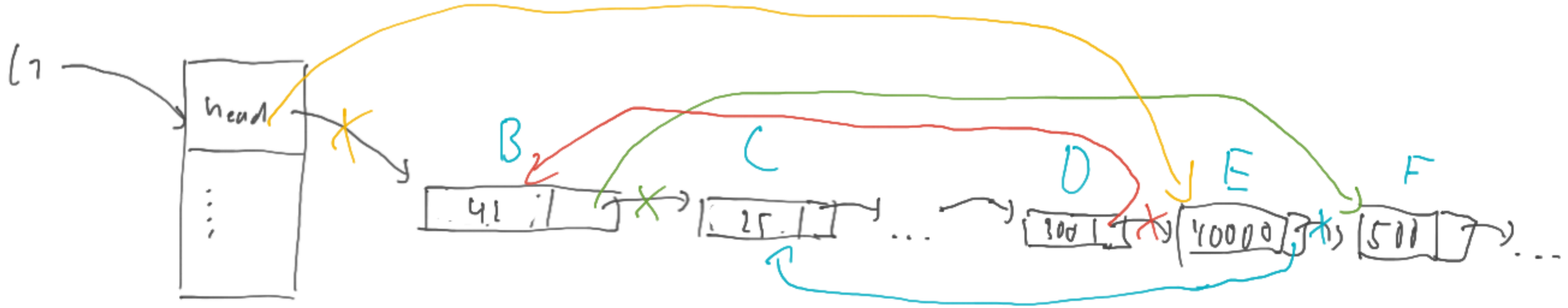


"Pufferspeicher" für die eigentliche Liste

"Container"

Neue Klasse: List

In main():

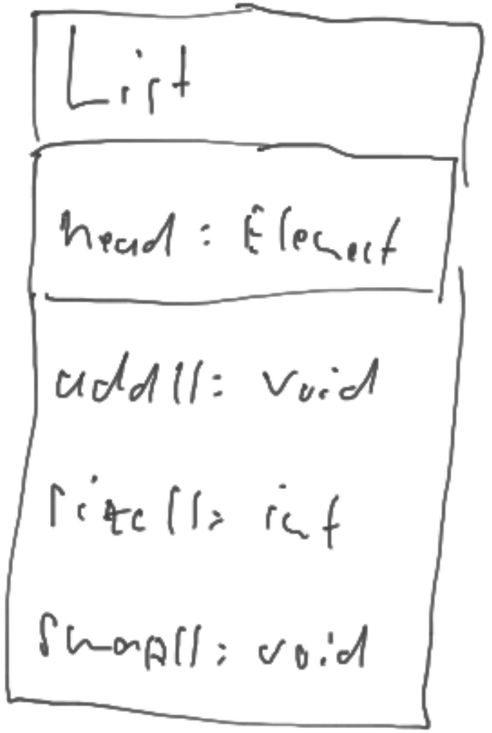


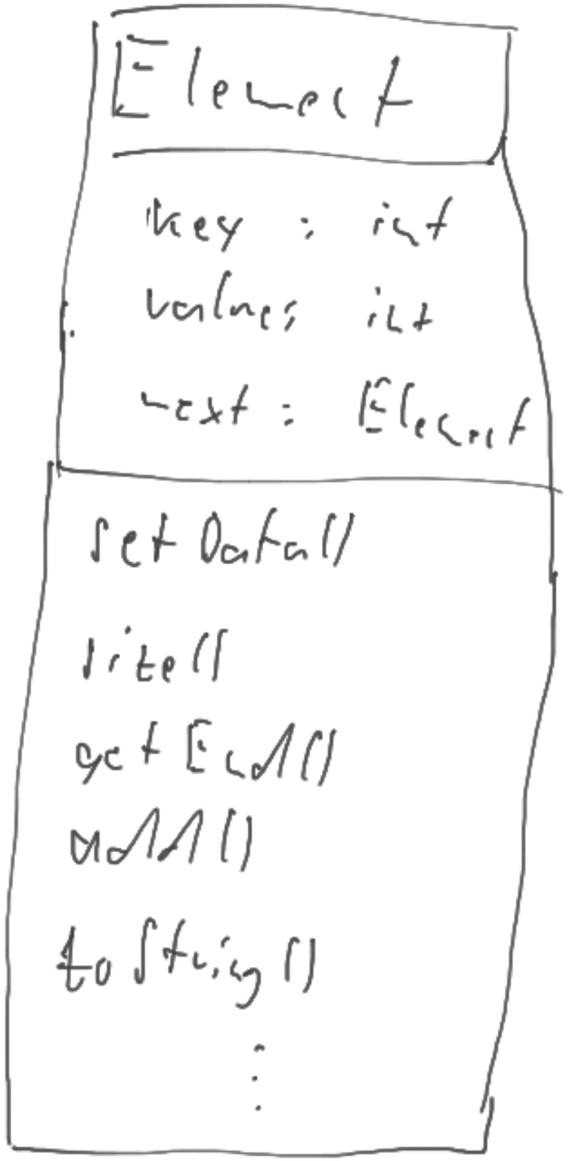
B.next = F

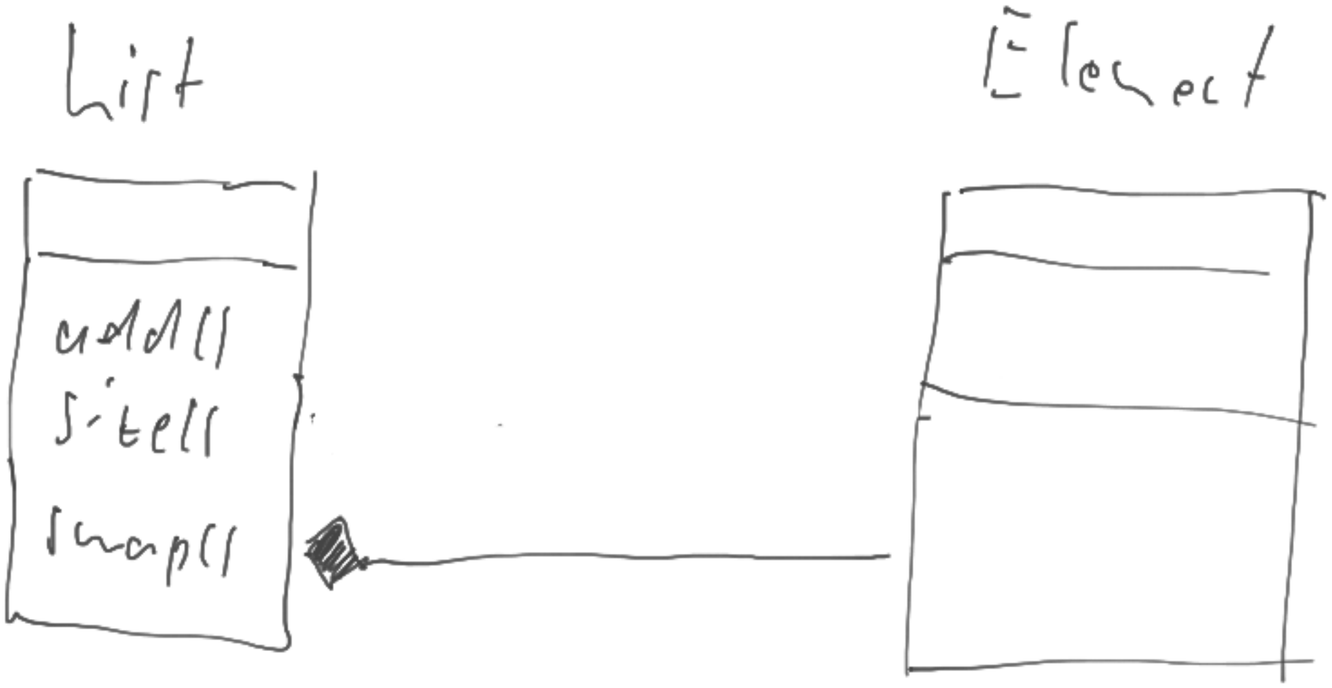
E.next = C

D.next = B

head = E


List
head : Element
add() : void
size() : int
show() : void


Element
key : int
value : int
next : Element
setData()
size()
getEl()
add()
toString()


List
addAll
sizeAll
snapAll
Elements