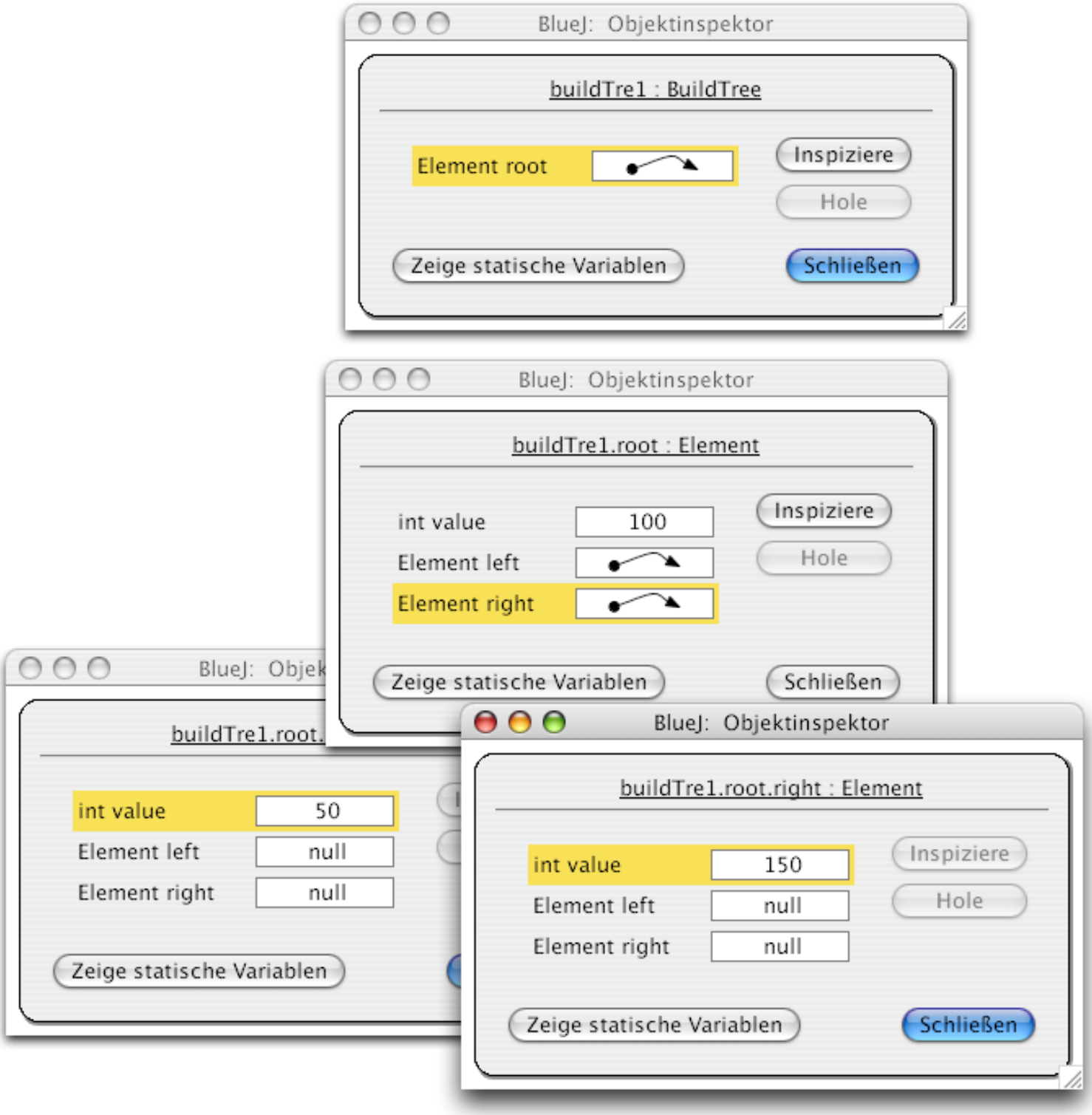
Ein kleiner Baum wird gebaut...

```
public class Element {
    int value;
    Element left, right;

    public Element(int n) {
        value = n;
        left = null;  right = null;
    }

    public void show() {
        System.out.println("  "+value);
    }
}
```

```
public class BuildTree {
    Element root;

    public BuildTree()    {
        root = new Element(100);
        root.left = new Element(50);
        root.right = new Element(150);
    }
}
```

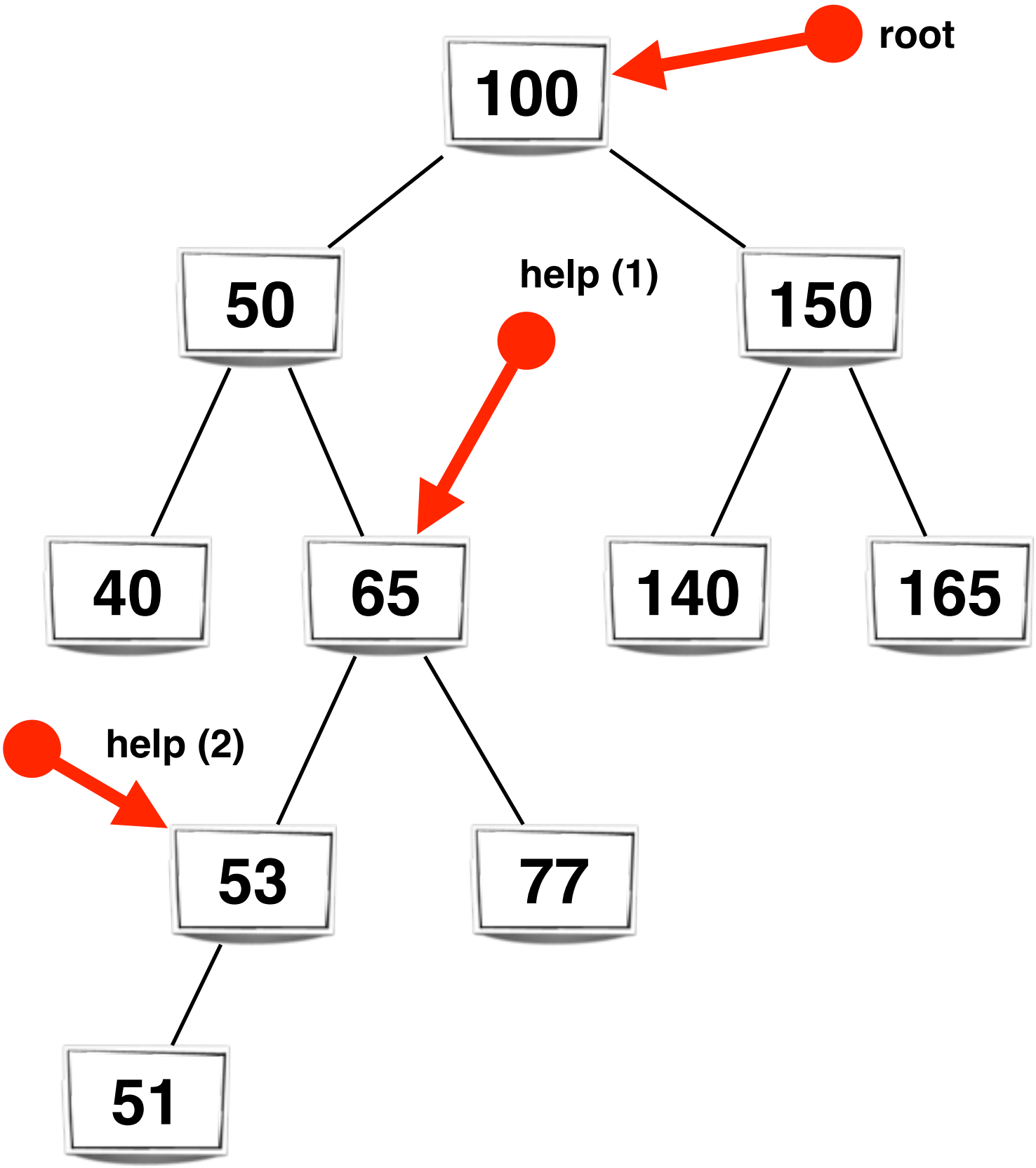
...und untersucht



Ein großer Baum wird gebaut...

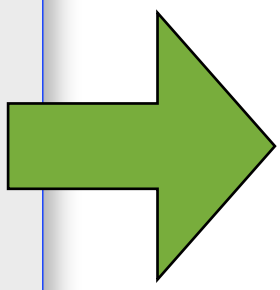
```
public class BuildTree {
  Element root;

  public BuildTree()  {
        root     = new Element(100);
        root.left    = new Element(50);
        root.right  = new Element(150);

        root.left.left    = new Element(40);
        root.left.right  = new Element(65);
        root.right.left  = new Element(140);
        root.right.right   = new Element(165);

        Element help  = root.left.right;
        help.left    = new Element(53);
        help.right  = new Element(77);
        help    = help.left;
        help.left    = new Element(51);
  }
}
```

und so sieht er aus:

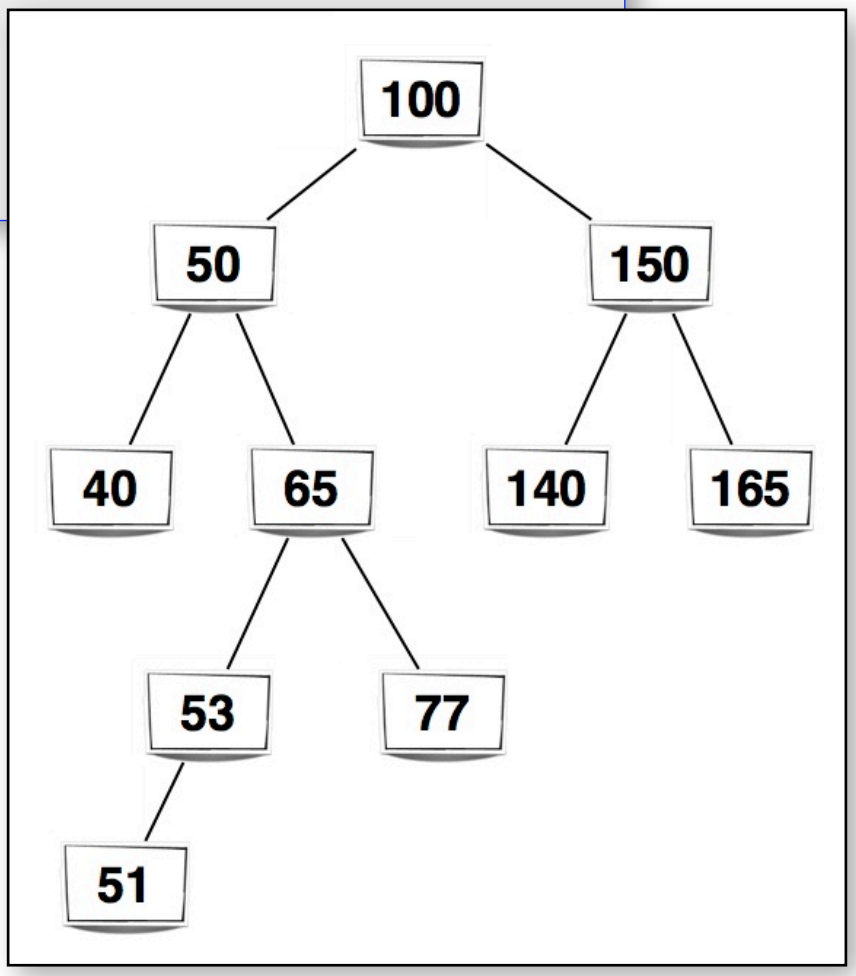


Ein Baum wird ausgegeben...

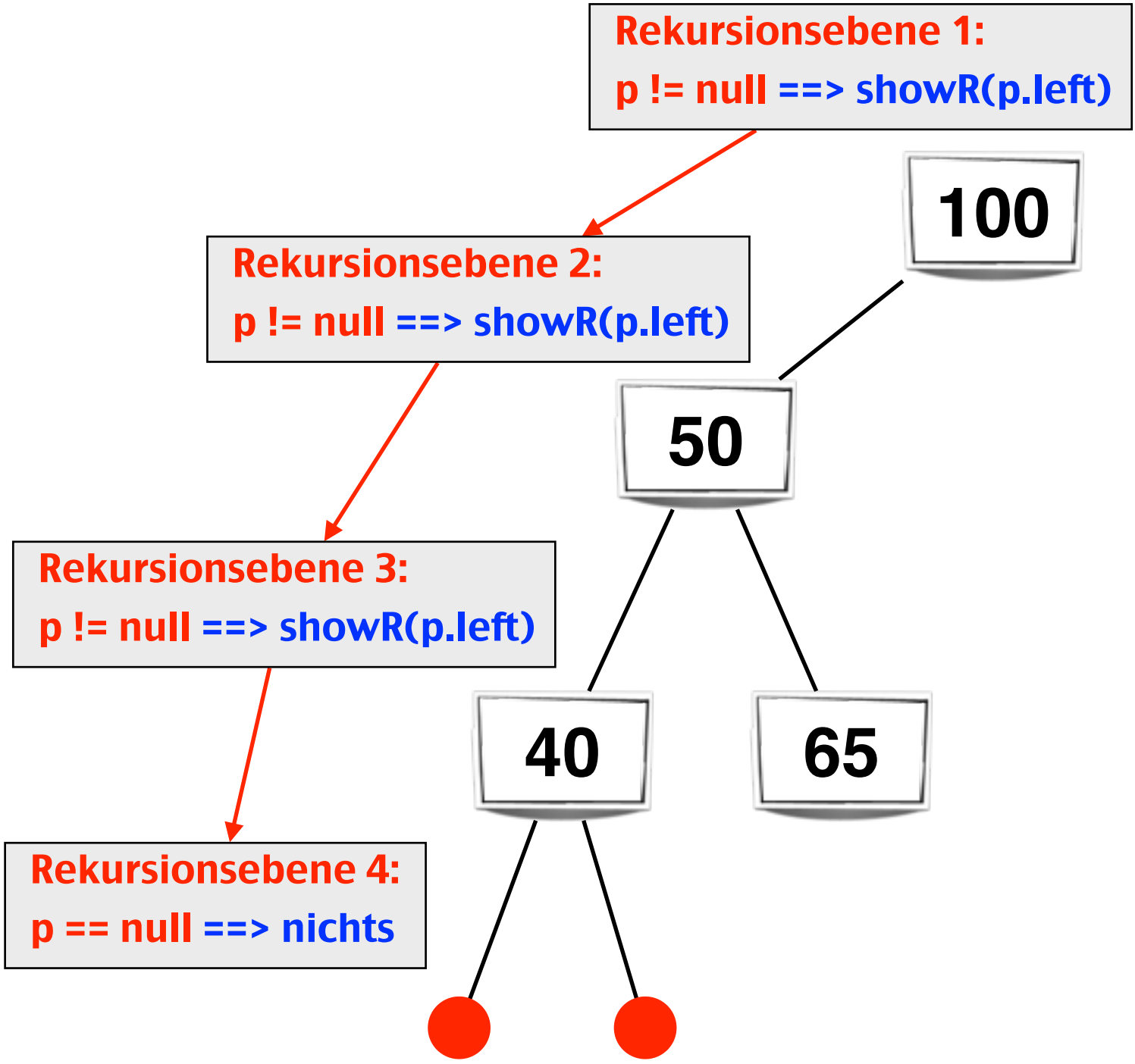
```
public void show() {
    showR(root);
  }

  private void showR(Element p) {
    if (p != null)  {
      showR(p.left);
      p.show();
      showR(p.right);
    }
  }
```



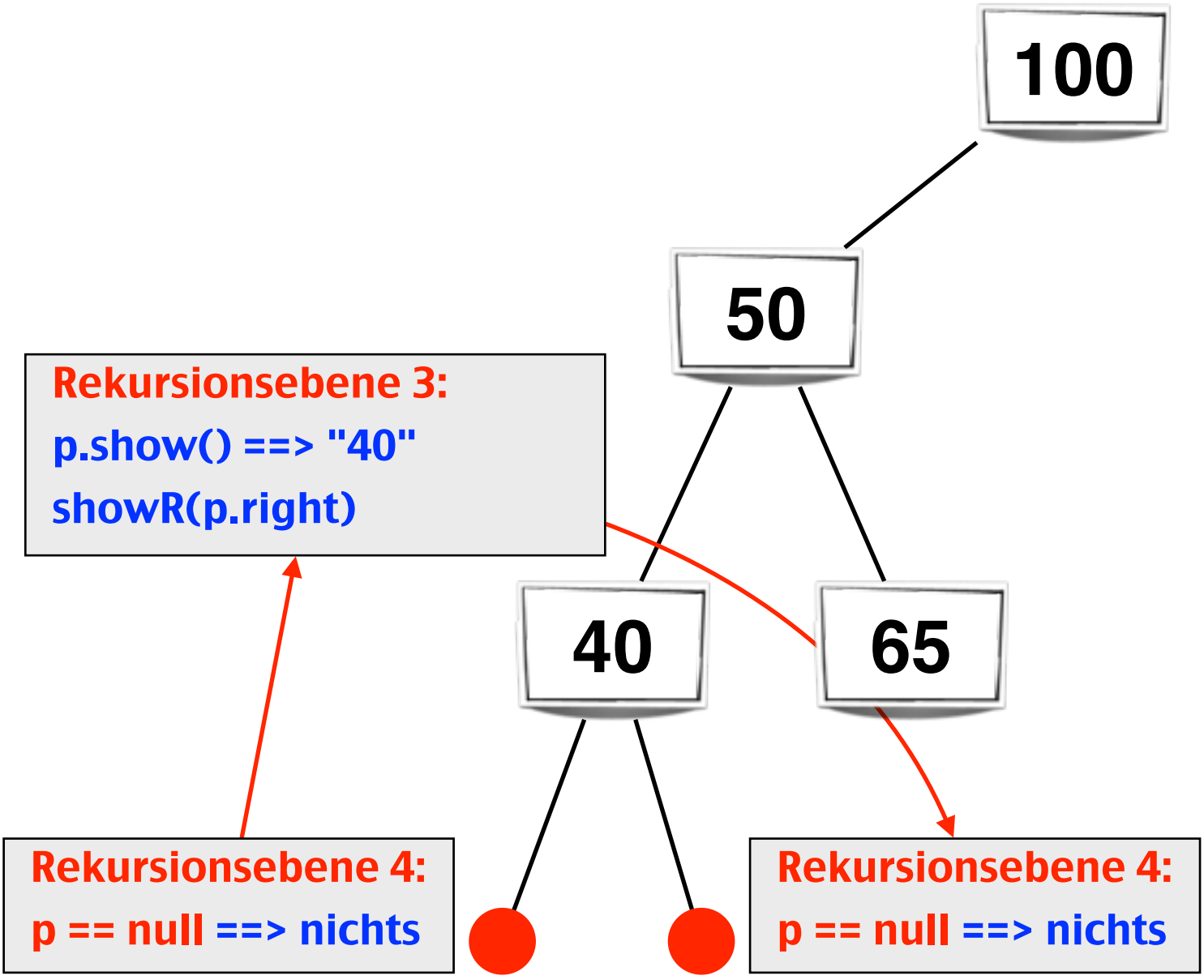
```
40
50
51
53
65
77
100
140
150
165
```



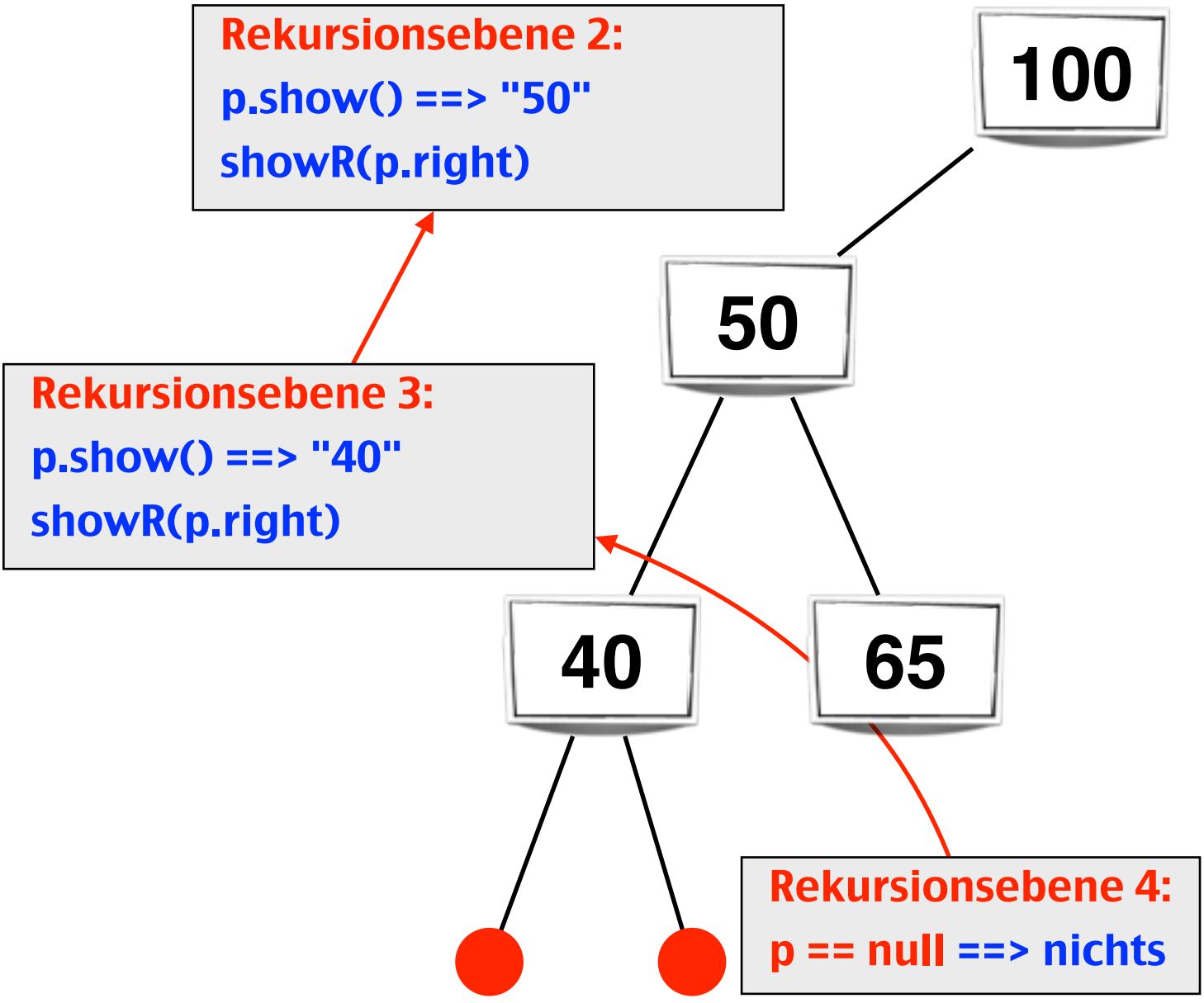
Die showR()-Methode



Die showR()-Methode



Die showR()-Methode



Die insert()-Methode

```
Falls Wurzel leer
    Dann
        Lege neues Element Wurzel an
    Sonst
        Falls v < Wurzel.v
            Fuege v in linken Teilbaum ein
        Sonst
            Fuege v in rechten Teilbaum ein
```

```
private Element insertR(int v, Element tree) {
    if (tree==null)
        tree = new Element(v);
    else if (v < tree.value)
        tree.left = insertR(v,tree.left);
    else
        tree.right = insertR(v,tree.right);
    return tree;
}
```

Die insert()-Methode

```
private Element insertR(int v, Element tree) {
    if (tree==null)
        tree = new Element(v);
    else if (v < tree.value)
        tree.left = insertR(v,tree.left);
    else
        tree.right = insertR(v,tree.right);
    return tree;
}
```

Aufgaben

1. Überlegen Sie sich, wie Sie ihren Mitschülern die Arbeitsweise der rekursiven **insert()**-Methode verdeutlichen können und entwickeln Sie eine entsprechende Präsentation (Folie, Powerpoint etc.).
2. Entwerfen Sie eine **nicht-rekursive** Methode zum Einfügen in einen binären Suchbaum.
3. Entwerfen Sie eine **rekursive isMember(int v)**-Methode, die den Wert TRUE zurück liefert, wenn eine Zahl v in dem Baum vorhanden ist.

Eine iterative insert()-Methode

```
public void insertIt(int v)
{   if (root==null) root = new Element(v);
    else
    {    Element h = root;
         while(h != null)
         {    if (v < h.value)
              {
                   if (h.left == null)
                   {    h.left = new Element(v);  break; }
                   else h = h.left;
              }
              else
              {
                   if (h.right == null)
                   {    h.right = new Element(v);  break; }
                   else
                        h = h.right;
              }
         }
    }
}
```

Lösung von **Marcel L.** und **Christian P.**, März 2010