**Aufgabe 3:**
Gegeben sei folgendes Interface…

```
import visitor.Baum.Knoten;

public interface Visitor<T>
{
    public void visit(Knoten<T> current);
}
```

…sowie folgende Baumimplementierung:

```
public class Baum<E extends Comparable<E>>
{
  public static class Knoten<E>
  {
    Knoten<E> left;
    Knoten<E> right;
    E content;
  }

  Knoten<E> root;

  public void einfuegen(E content) { … }

  protected void traversiere(Visitor<E> visitor)
  {
    if (root == null) return;
    else traversiere(root, visitor);
  }

  protected void traversiere(Knoten<E> current, Visitor<E> visitor)
  {
    if (current.left != null) traversiere(current.left, visitor);
    visitor.visit(current);
    if (current.right != null) traversiere(current.right, visitor);
  }
}
```

Die `traversiere`-Methode wird benutzt, um den Baum komplett zu durchlaufen. Dabei ruft die `traversiere`-Methode bei jedem Knoten des Baums einmal die `visit`-Methode des übergebenen `Visitors` auf. Ziel dieser Aufgabe ist die Implementierung einer `size`-Methode, die mit Hilfe eines von Ihnen geschriebenen `Visitors` und der vorgegebenen `traversiere`-Methode, die Anzahl der Knoten im Baum bestimmt.

a) Schreiben Sie dazu eine Klasse, die das `Visitor`-Interface implementiert. Der von Ihnen implementierte `Visitor` soll in obiger `traversiere`-Methode benutzt werden, um die Anzahl der Knoten im Baum zu bestimmen.
b) Ergänzen Sie die Klasse `Baum`, um eine Methode `size`, die Ihre `Visitor`-Implementierung aus Teilaufgabe a) instanziiert, die `traversiere`-Methode aufruft und die vom `Visitor` gezählte Anzahl der Elemente zurückgibt.