1999 Foundations of Programming

NOTES ON THE MODEL ANSWER TO PAPER 11 QUESTION 2

The actual wording of the compiler message is:

   Invalid left hand side of assignment.

The problem is that the pseudo-argument this is null so there is no type information available and the compiler will not allow the assignment of a new Link to this. The compiler does not know, of course, that this is associated with start which does have the correct type Link.

There is no way to fix the program by altering this statement alone. In essence, this has refer to an instantiation of a Link so that a valid statement like this.next = new Link(k) can be used. In consequence the main() method needs to ensure that start refers to a Link before allowing start.put(...) to be used. The toString() method will need adjusting too as will be seen.

A suitable, correct, version of the program is:

```
public class ListTest
 { public static void main(String[] args)
    { Link start = new Link(4);
      start.put(7);  start.put(11);
      System.out.println("List elements:  " + start);
    }
 }

class Link
 { private int val;
   private Link next;

   public Link(int n)
    { this.val = n;
      this.next = null;
    }

   public void put(int k)
    { if (this.next == null)
         this.next = new Link(k);
      else
         this.next.put(k);
    }

   public String toString()
    { return this.val + (this.next == null ? "" : "  " + this.next.toString());
    }
 }
```

An appropriate sum() method is:

```
   public int sum()
    { return this.val + (this.next == null ? 0 : this.next.sum());
    }
```

An appropriate reverse() method is:

```
   public Link reverse()
    { if (this.next == null)
         return this;
      else
       { Link temp = this.next.reverse();
         temp.put(this.val);
         return temp;
       }
    }
```