Visualising Java Data Structures as Graphs

John Hamer

Department of Computer Science

University of Auckland

J.Hamer@cs.auckland.ac.nz

THE UNIVERSITY OF AUCKLAND
NEW ZEALAND

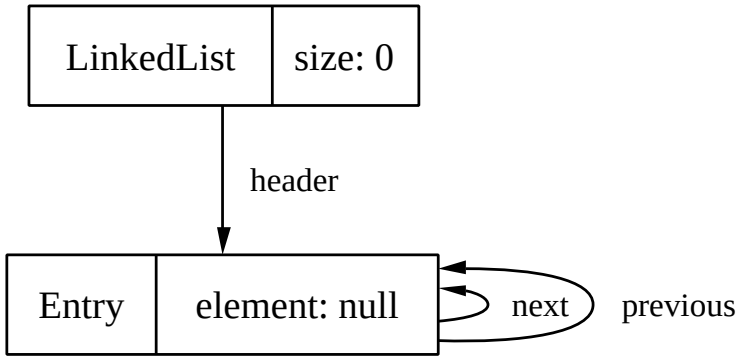
The Idea

- Student code calls the static method `Dot.drawGraph(`*`whatever`*`)`
- *`whatever`* can be any Java object.
- `Dot.drawGraph`
  - traverses the object's fields using Java reflection
  - outputs a GraphViz format graph description to a text file
  - runs the GraphViz processor to produce a PNG (or EPS, etc.) picture
- Student views the sequence of pictures using a standard viewer

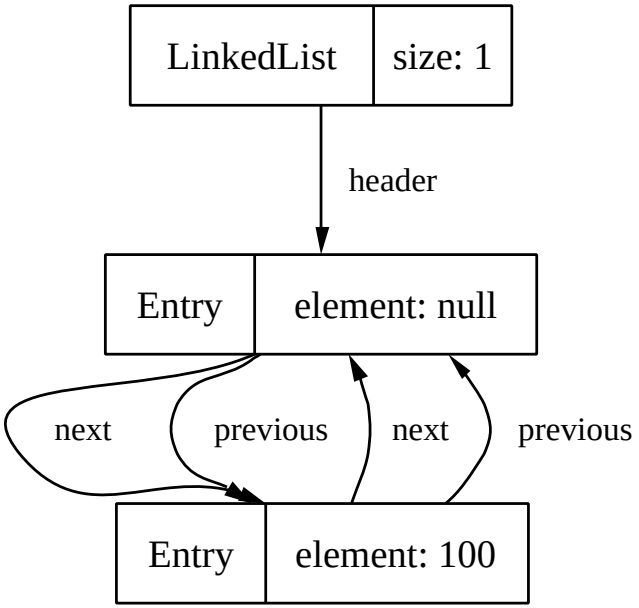
Example


```
public static void main( String[] args ) {
  List xs = new LinkedList( );
  for( int i = 0; i < 4; i++ ) {
    Dot.drawGraph( xs );
    xs.add( new Integer(i+100) );
  }
}
```

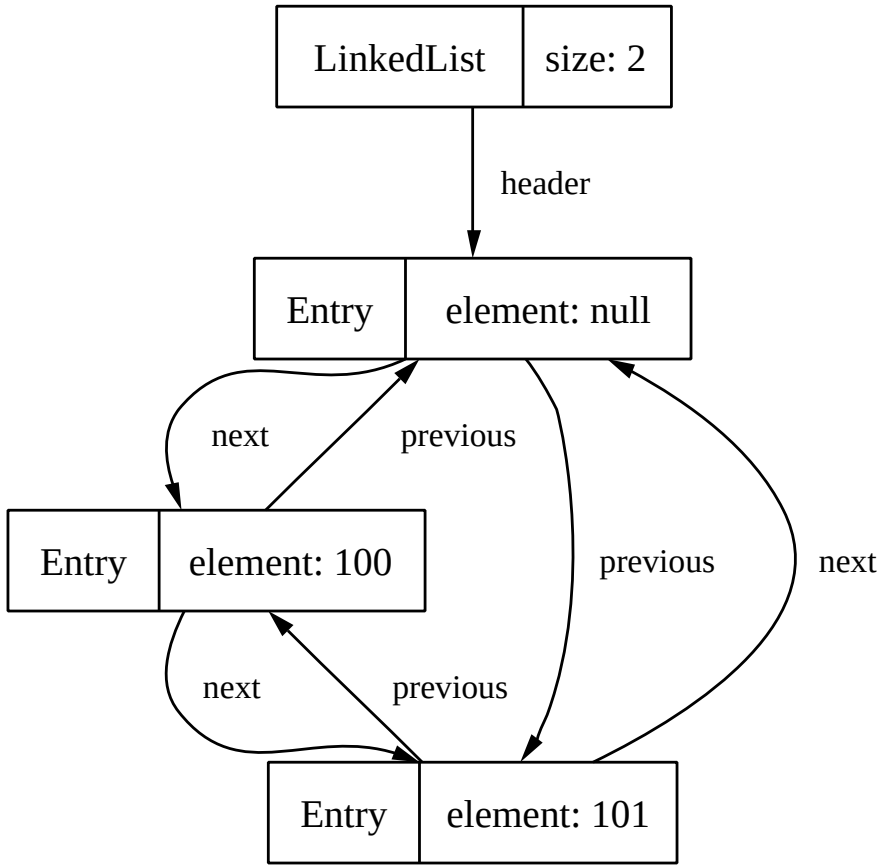
Frame #1



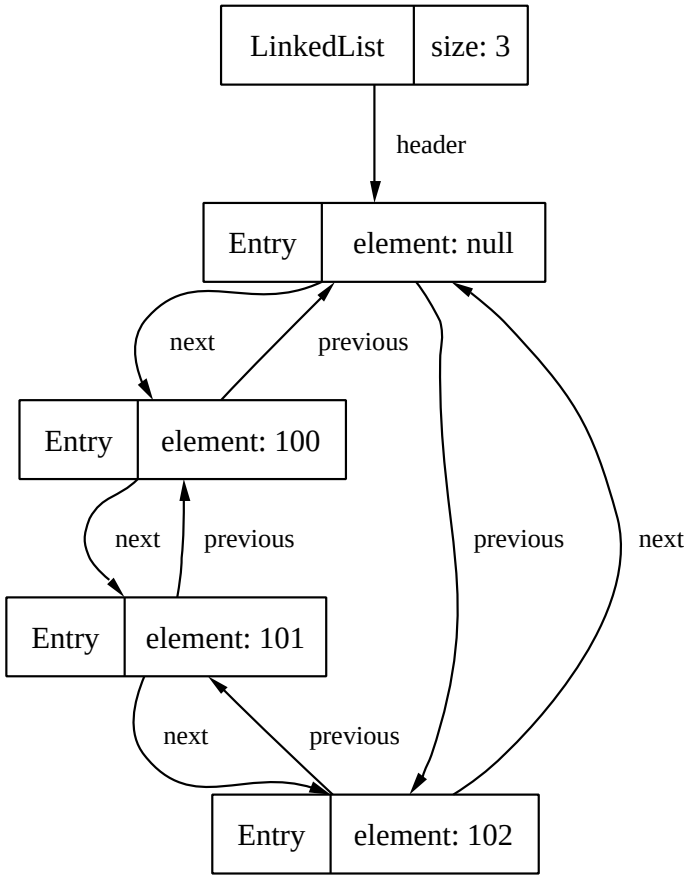
Frame #2



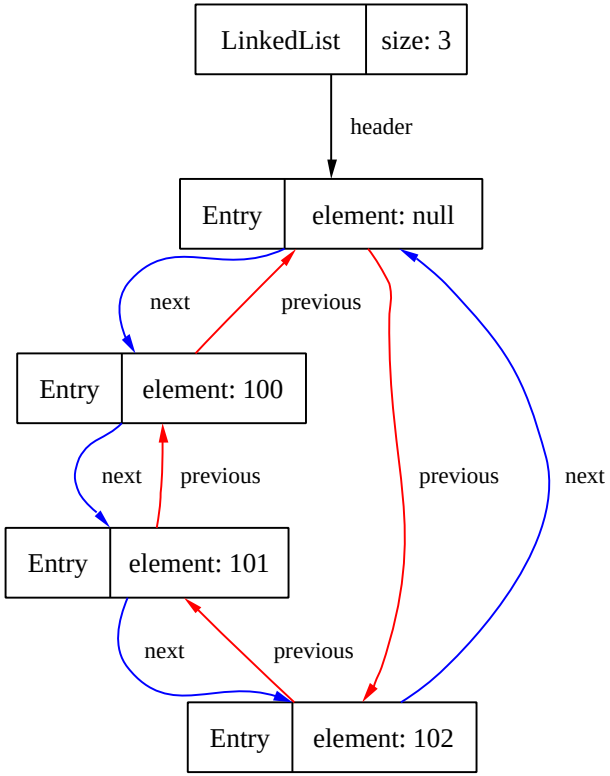
Frame #3



Frame #4



Frame #4 coloured




```
Dot.Context ctx = Dot.defaultContext( );
ctx.setFieldAttribute( "next", "color=blue" );
ctx.setFieldAttribute( "previous", "color=red" );
```

About GraphViz

- GraphViz is a widely used, freely available graph drawing program, developed at ATT; see `www.graphviz.org`
- Layout is completely automatic and (generally) æsthetically pleasing.
- Text input for nodes and edges, with optional attributes (colour, node shape, labels, fonts, etc.).
- Output to a variety of formats (PNG, EPS, SVG, . . .)

Related work

GraphViz

- Brocard —Perl interface to GraphViz for visualising data structures; also regular expressions, grammars, XML, call graph, profiling, ...
- North & Koutsofios —visual debugger, vdbx

Visualisation

- Thomas Naps' `Visualiser` class. Canned collection of visualisations: numeric arrays (bar, scattergram, data views), general arrays, stacks, queues, linked lists, binary trees, general trees, graphs, networks.

Principles

- Students must be engaged in active learning;
- tools need to be simple to use;
- avoid distracting students from substantive course material;
- for instructors, minimise the effort required to integrate tools into the curriculum;
- software must be reliable.

Features of our tool

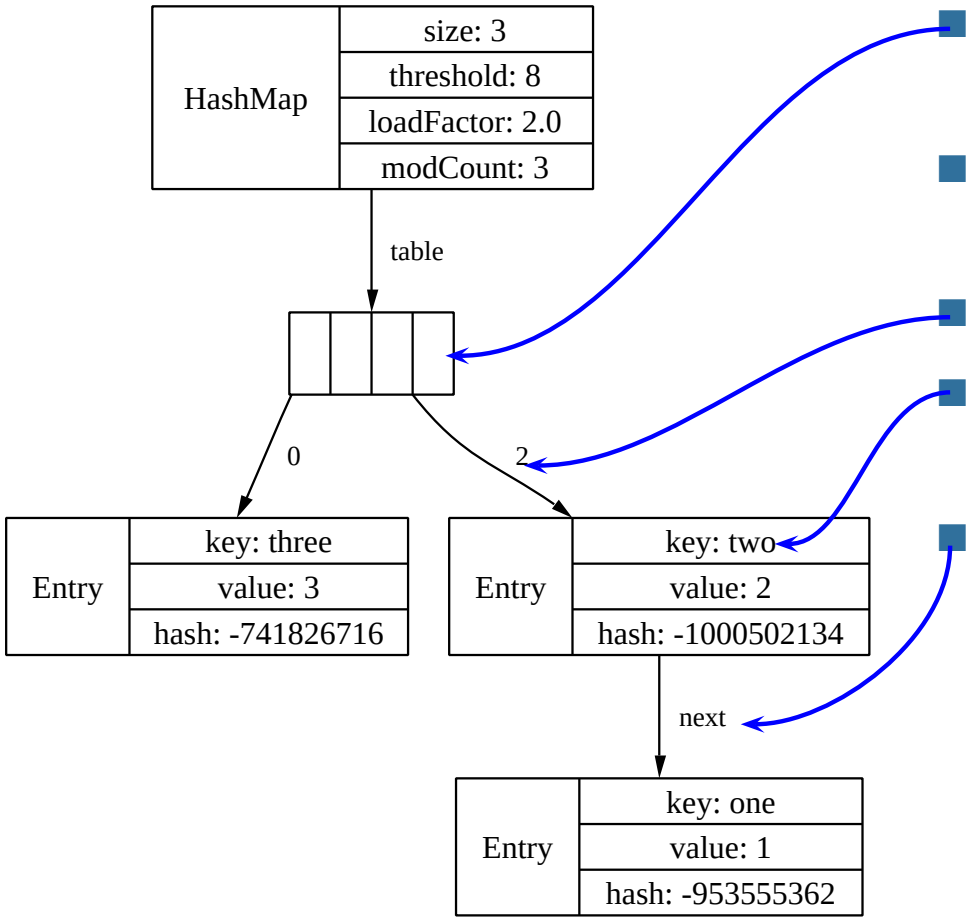
- trivial to setup and easy to use (source < 600 lines);
- active learning —students decide where to place the calls to `drawGraph`, what to elide;
- connects code with the Java data model;
- usable on any Java program; no specific programming conventions necessary;
- allows "wrong" data structures to be viewed (as well as correct ones);
- configuration allows broad and precise elision of detail;
- visualisations can be incorporated in reports, www pages, and presentations.

Overcoming student misconceptions

Java has a simple data model, right?

- Strings are objects, but string constants *look* like primitive values.
- Assignment of objects is by reference, primitive types by value.
- Object arrays hold references, not values.
- 2-dimensional arrays are constructed from 1-d arrays (is it row or column order?)
- Static fields are not part of any object.
- Inheritance means objects are often not the same as their declared types.

Visualising the Java data model



- Arrays are displayed with elements juxtaposed.
- Values in primitive arrays are shown inline.
- Object arrays just contain links.
- Primitive fields are shown inside the object's node.
- Object fields are shown as labelled arcs.

Degrees of faithfulness

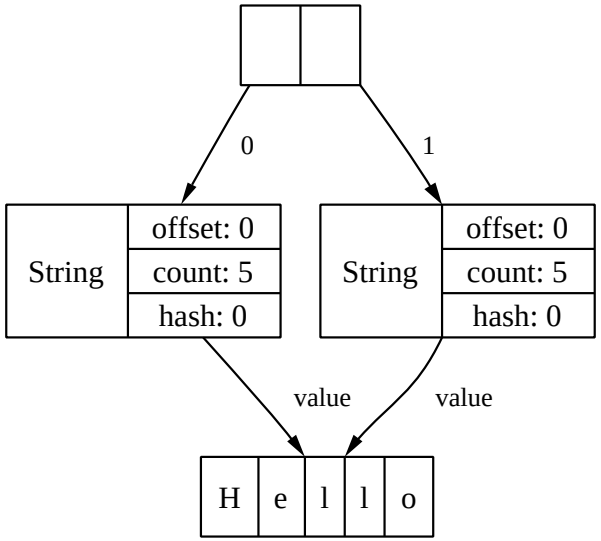
Three different views of `String`

- Show the full internal state of `String`.
- Acknowledge `String` is an object, but hide the internal state.
- Pretend `String` is a primitive value (not an object).

These views apply to any object, not just `String`.

The Full Monty


```
String x = "Hello";
String y = new String(x);
Dot.drawGraph( new String[]{ x, y } );
```

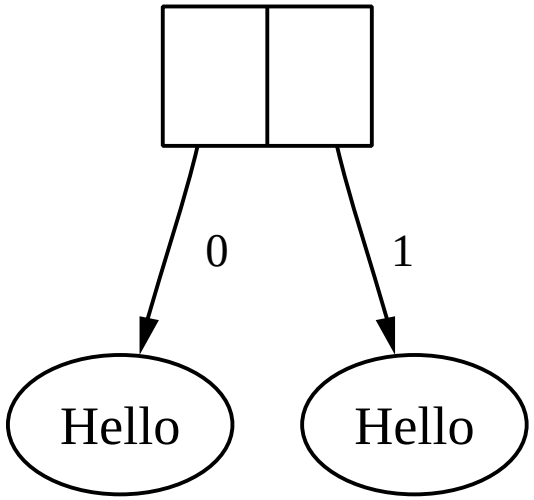


\+ Useful in explaining the memory consumption of substring operations, or as an example of a sharing data structure.

– Clutters the visualisation.

– Details are a distraction (e.g., explaining `hash`).

Hide the internal state



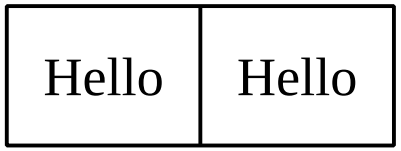
\+ Visualisation respects reference semantics.

\+ More compact.

– Internal sharing is not shown.

- Can be used with any object, by calling the `toString` method.

Pretend it's primitive



- \+ Most compact.
- – Visualisation contradicts reference semantics.
- Can be used with any object, by calling the `toString` method.

Limitations and future work

- GraphViz has limited support for node shapes, label placement, ...
- Graphs of, e.g., Java AWT components, can be immense. Drawing even a simple `Button` will bring in every interface component!
- Work in progress on integration with a debugger (Jacob Tseng). Extended a Java IDE debugger with a "draw" command. Graphs are updated at each breakpoint.
- Also, "draw" command extension to the BeanShell (an interactive Java interpreter), provided by a first-year student.
- More elision controls.
- Experimental features for dynamically selecting attributes (e.g., red nodes in a red-black tree are displayed in red).
- Interactive graphs —select a node and expand or elide.

Summary and conclusions

- Light-weight, general purpose visualisation tool for Java.
- Useful in elucidating the Java data model, especially reference semantics.
- Less suitable for classical array data structures (c.f., Naps), or OOP (but see, e.g., UMLGraph `http://www.spinellis.gr/sw/umlgraph/`)
- Freely available from `http://www.cs.auckland.ac.nz/~j-hamer`

A view of an “Arne” Tree