

2D User Interfaces - Exercises

Tutorial 5

Creating a Treeview

Marcus Tönnis, Patrick Maier

Fachgebiet Augmented Reality (FAR)
Technische Universität München

Overview

- Eclipse Tips&Tricks
- Introduction
- A word (or two) about data structures
- Advanced Canvas/Graphics features
- Assignment and Tips

Eclipse Tips and Tricks

- Why always write the code by hand?
 - For-loops: type `for` and press Ctrl-Space
 - Try-catch block: type `try` and press Ctrl-Space
 - `System.out.println();`: Type `syso` and press Ctrl-Space (but better use a logging API)
 - What other shortcuts are available?
Go to Window -> Preferences -> Java -> Editor -> Templates
- Ok – Code already written but it needs change
 - Adding a try-catch block: Highlight the code block and type `Alt-S` then `W` and the type `Enter`
 - Same code is called multiple times (BAD!!!! Code duplication is error prone), e.g.:

```
if ("test".equals(node.getName())) {
    System.out.println(node.getName());
}
```

Highlight the code to exclude (here `node.getName()`) and type `Alt-Shift-L` to exclude a local variable
- Code deleted – organize imports: `Ctrl-Shift-O`

Introduction

- Goal: browse the entire list at once
- Collapsible subtrees
- Example: see picture



A word (or two) about data structures

- Our dataset is a recursive structure => a tree

```
import java.util.Vector;

public class Node {
    public Node( String name ) { ... }
    private final Vector childrenNodes;
    ...
}
```

- Note: Remember the more fine granular variant applying the Composite Pattern

A word (or two) about data structures (ctd)

- Useful operations on child nodes
 - Add a child node

```
Node n = new Node("Root");  
n.addElement(new Node("Child"));
```

- Retrieve a child node
- Note: Vector only stores Objects, thus casting is required

```
Node c = n.elementAt(i); // doesn't work - return value is an Object  
Node c = (Node) n.elementAt(i); // explicit cast needed
```

A word (or two) about data structures (ctd)

- Recursive traversing

- Required is a public method in the Node class
- Remember from previous exercise: `dump()` method

```
public String dump() {
    String result = "";
    for( int i=0; i < size(); i++ ) {
        Node current = (Node)elementAt(i);
        String sel = current.selected ? "selected " : "";
        if (current.size() != 0) {
            result += "<group " + sel + "name=\"\" + current.name
                + "\" >\n" + current.dump() + "</group>\n";
        } else {
            result += "<item " + sel + "name=\"\" + current.name
                + "\" />\n";
        }
    }
    return result;
}
```

Note: Trees may be large – String concatenation is slow => better use `StringBuffer`

Advanced Canvas/Graphics features

- Adopting recursion for painting the view
 - Remember: Canvas needs method `void paint(Graphics g)`
 - Suggestion: add recursive `paint(Graphics g)` method to the class responsible for rendering a Node (when you apply the MVC paradigm)
 - In `Canvas.paint`, just call `rootnode.paint(g)`
- Problem: how to remember position?
 - The `Graphics` class has a helpful method: `translate(x,y)`
 - Shifts coordinate system to new position
 - Painting now can use its own 'local' coordinate system starting from 0, 0
 - Don't forget to shift it back ;-)

Exercise Assignment

- Write a midlet which allows the user to..
 - view a tree structure (loaded from XML)
 - expand and collapse nodes
- Questions
 - How to deal with indentation?
 - How to handle sliders (the vertical height and position indicator to the right)

Exercise Assignment (ctd)

- When you finished your work
 - Come by in our office (with appointment) and present your work before the next tutorial is held (for today's exercise before the tutorial session on 08.01.09 at 13.00h)
 - If you are fast, you can present your solution in the lab session (18.12.08)
 - Room: 03.13.035
- Q/A: maierp@in.tum.de / toennis@in.tum.de
- See you back here on 18.12.08 at 13.00h for the lab session