JAVA DAYS ´99




# Java™ Foundation Classes: Advanced Topics

Onno Kluyt
Sr. Product Manager
Sun Microsystems, Inc

Sun microsystems

# What are we working on?

- Performance, Performance!!!
- Some new features
  - Input validation
  - Text, HTML enhancements
  - New key binding scheme
  - Top level frame decoration
  - AWT implementing Accessibility API






© 1999 Sun Microsystems GmbH

# Agenda




- Advanced JFC Programming Hints
  - Using Models in JFC Effectively
  - Threads
- JFC Enhancements
  - Input Validation
  - Text & HTML components
- Roadmap



© 1999 Sun Microsystems GmbH

JAVA DAYS ´99




# Using JFC Models Effectively

Sun microsystems

# Why MVC?




- Historically praised for reusability and maintainability
  - Also helps performance!
  - Moves control of data into app space
- You determine the optimal way of handling:
  - Allocation
  - Representation
  - Binding between data model and view



© 1999 Sun Microsystems GmbH

Sun microsystems

# Not all components have models




- A component that has a value, has a model:

```
BoundedRangeModel, ButtonModel,
ComboBoxModel, Document, ListModel,
ListSelectionModel, MutableComboBoxModel,
SingleSelectionModel, TableColumnModel,
TableModel, TreeModel, TreeSelectionModel
```

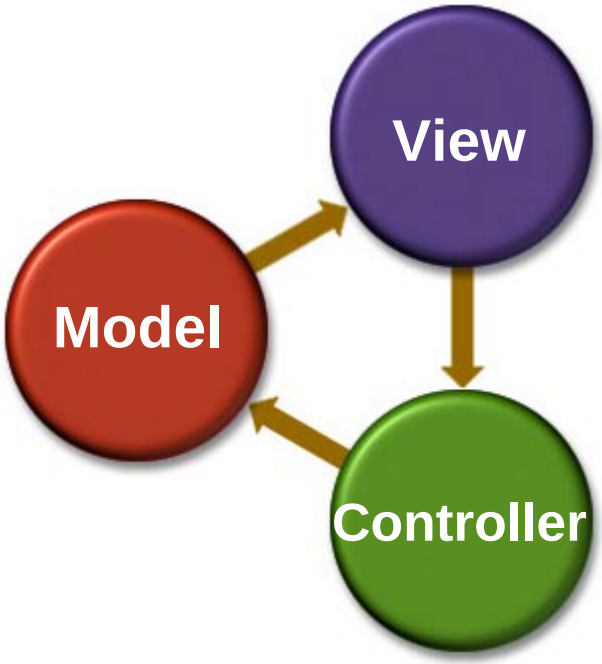


© 1999 Sun Microsystems GmbH

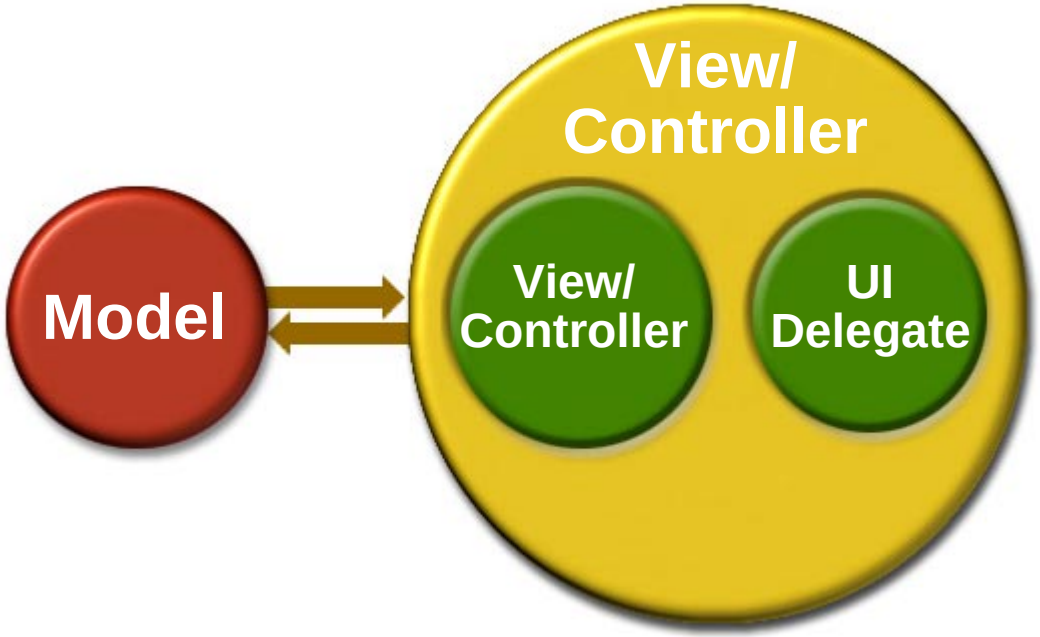
Sun microsystems

# JFC MVC Architecture

## Traditional MVC




## MVC Hybrid

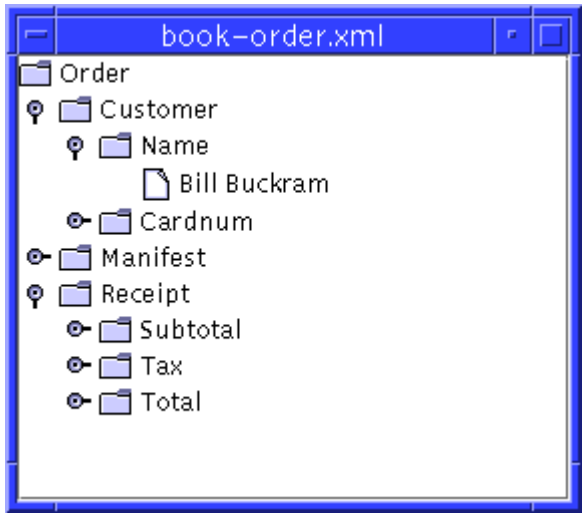


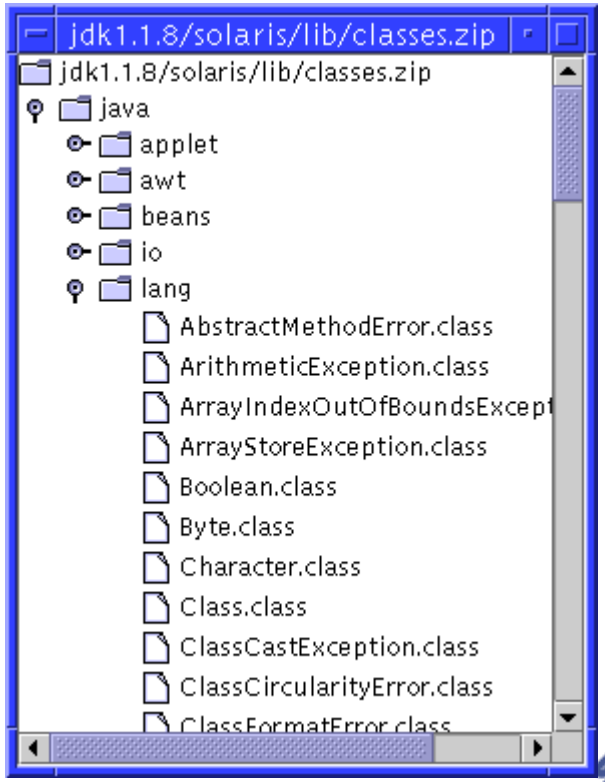



© 1999 Sun Microsystems GmbH

# What does this enable?

- Many combinations of views and models
  - XML TreeModel can be viewed with a JTree
  - Jar TreeModel can be viewed with a JTree

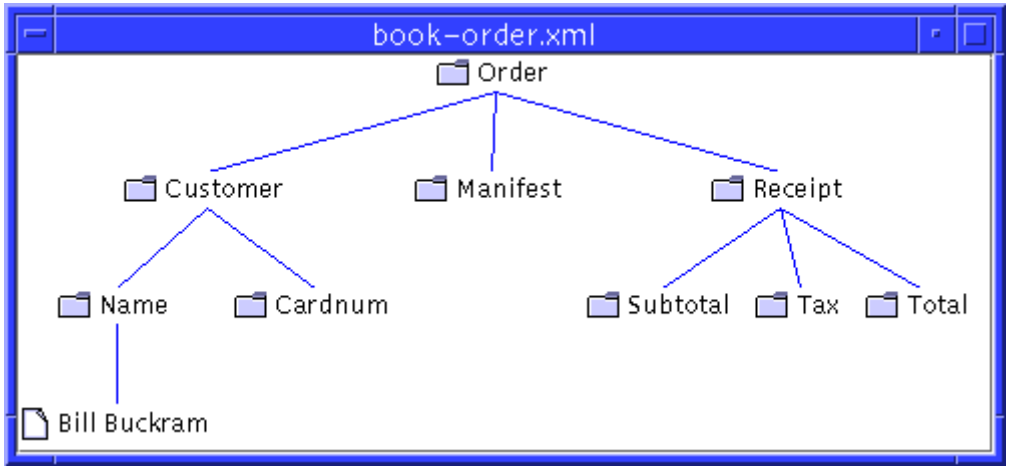


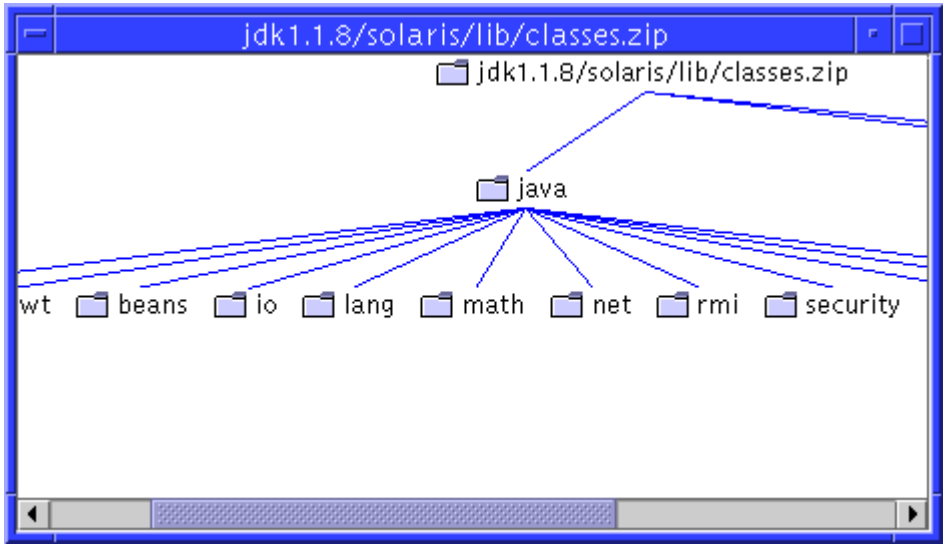






© 1999 Sun Microsystems GmbH

# One Model, Configurable View

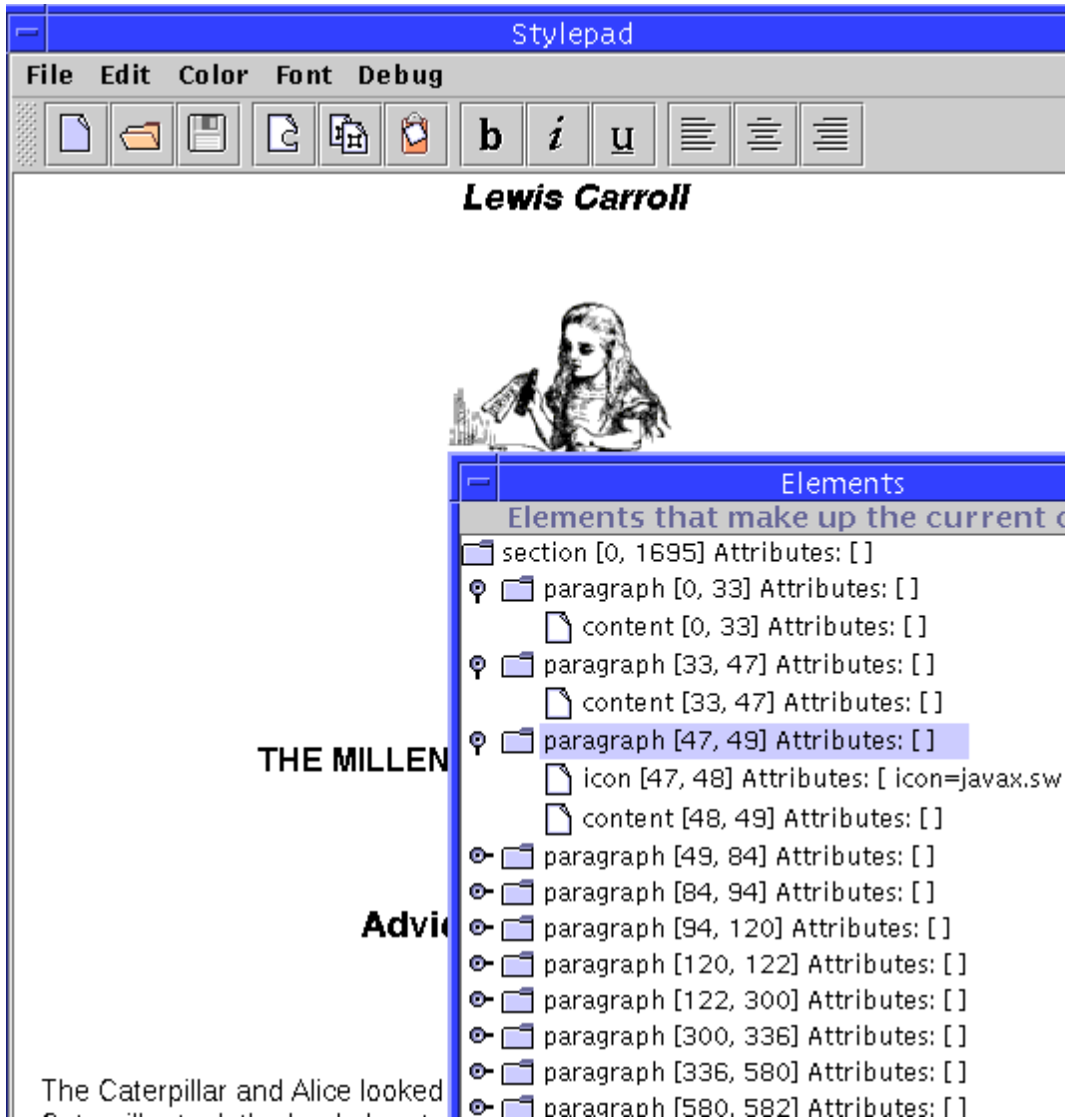





© 1999 Sun Microsystems GmbH

# One Model, Multiple Views






© 1999 Sun Microsystems GmbH

# Combining Models: The Decorator Pattern




- Models can be combined or linked to filter data from the ‘real’ model






© 1999 Sun Microsystems GmbH

Sun microsystems

# Ignoring MVC




```
JComboBox c = new JComboBox();
for (int i = 0; i < 200 ; i++) {
    // Note that you will often be
    // copying from existing data here
    c.addItem(new Integer(i));
}
```

- Takes 1000 microseconds
- Doesn’t scale well



© 1999 Sun Microsystems GmbH

Sun microsystems

# Using MVC




```
Vector v = new Vector();
for (int i = 0; i < 200 ; i++) {
      v.addElement(new Integer(i));
}
JComboBox c = new JComboBox(v);
```

- Takes only 50 microseconds
- Scales extremely well



© 1999 Sun Microsystems GmbH

Sun microsystems

# JTable - Big Model (Simple)




```
public class SpreadsheetModel extends AbstractTableModel {
   Object[][] array = new Object[1000][1000];
   public Object getValueAt(int row, int col) {
      return array[row][column];
   }

   public void setValueAt(Object val, int row, int col) {
      array[row][col] = val;
   }
}
```

- Uses 4MB before first cell is set



© 1999 Sun Microsystems GmbH

Sun microsystems

# JTable - Big Model (Sparse)




```
public class SpreadsheetModel extends AbstractTableModel {
   Hashtable matrix = new Hashtable();
   public Object getValueAt(int row, int col) {
      return matrix.get(new Point(row, col));
   }

   public void setValueAt(Object val, int row, int col) {
      matrix.put(new Point(row, col), val));
   }
}
```

- Uses about 1KB before first cell is set



© 1999 Sun Microsystems GmbH

Sun microsystems

JAVA DAYS ´99

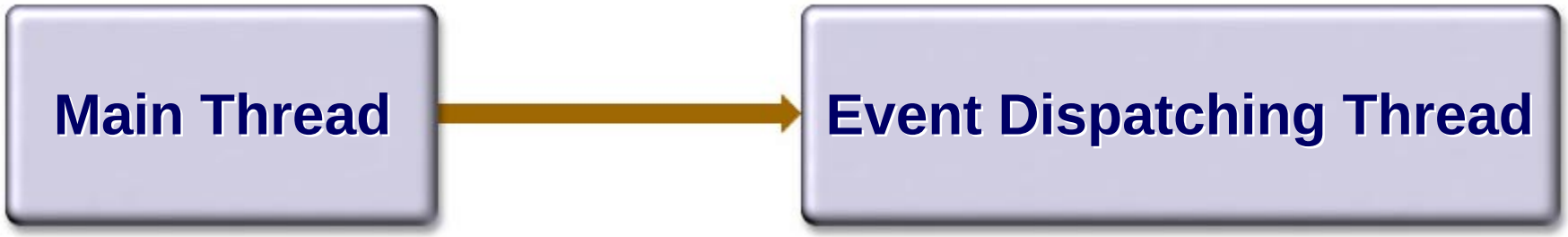



# Threads

Sun microsystems

# AWT Threads

```
MyApplication.main()
MyApplet.init()
```






© 1999 Sun Microsystems GmbH

# Threads - Be Careful






© 1999 Sun Microsystems GmbH

# Threads Are Good




- When to use:
  - If it might take a while - use a thread
  - If it might block - use a thread
  - If it needs to occur later - use a timer
  - Likewise if it needs to occur periodically
- Human interface goals:
  - User action produces immediate feedback
  - Let them know what is going on
  - Let them interrupt what is going on
  - Let them know why they are waiting



© 1999 Sun Microsystems GmbH

Sun microsystems

# Rules for Threads




- GUI access must occur on the event dispatching thread:

```
SwingUtilities.invokeLater (Runnable X)
SwingUtilities.invokeAndWait (Runnable x)
```

- It is OK to construct components on a worker thread



© 1999 Sun Microsystems GmbH

Sun microsystems

# SwingWorker Utility Class




- To perform a task on another thread

```
SwingWorker worker = new SwingWorker() {
    protected Object construct() {
        return getValueFromDatabase();
    }
    protected void finished() {
        showValueFromDatabase(get());
    }
};
worker.start();
```



© 1999 Sun Microsystems GmbH

Sun microsystems

# Example: Thread Downloads Web Search Results




```
while(null != (item = in.readLine())) {
    buffer.add(item);
    if ((buffer.size() >= 10) && gui.isReady()) {
        final Object[] items = itemBuffer.toArray();
        Runnable doUpdateList = new Runnable() {
            public void run() {
                gui.updateList(items);
            }
        };
        SwingUtilities.invokeLater(doUpdateList);
        buffer.clear();
    }
}
```



© 1999 Sun Microsystems GmbH

Sun microsystems

# Asynchronous Layout



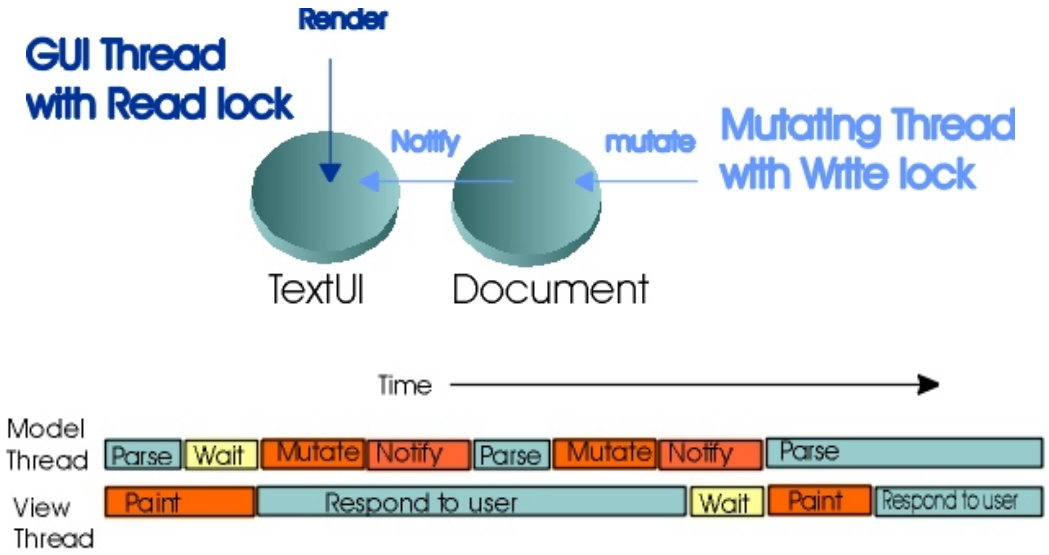

- More responsive UI
- Updates from the model are more efficient
- Layout can be computed incrementally
- Building block that can be used in conjunction with synchronous views



© 1999 Sun Microsystems GmbH

Sun microsystems

# Concurrency Support




- Support asynchronous model mutations
- Access to View is serialized



© 1999 Sun Microsystems GmbH

JAVA DAYS ´99




# Validating Input

Sun microsystems

# The Past

- Goal: validate input before user moves the focus elsewhere
- Listening for **FOCUS_LOST** events
- Glass pane mania



© 1999 Sun Microsystems GmbH

# InputVerifier

- Before focus moves within a Window: consult the current focus owners InputVerifier
- JComponent "**`inputVerifier`**" property

```
public abstract class InputVerifier {
    public abstract boolean verifyInput(JComponent input);
    public boolean shouldYieldFocus(JComponent input) {
        return verifyInput(input);
    }
}
```



© 1999 Sun Microsystems GmbH

# Using an InputVerifier

```
class VerifyDate extends InputVerifier {
      Date date;
      DateFormat df = DateFormat.getDateInstance()

public boolean verify(JComponent input) {
      try {
      date = df.parse(((JTextField)input).getText());
      return true;
      }
      catch (ParseException e) {
      return false;
      }
   }
   // … shouldYieldFocus()
}
```



© 1999 Sun Microsystems GmbH

# Using an InputVerifier




```
class VerifyDate extends InputVerifier {
      Date date;
      DateFormat df = DateFormat.getDateInstance();

      // … verify()

      public boolean shouldYieldFocus(JComponent input) {
          if (verify(input)) {
              ((JTextField)input).setText(df.format(date));
              return true;
          }
          else {
              input.getToolkit().beep();
              return false;
          }


      }
}
```



© 1999 Sun Microsystems GmbH

Sun microsystems

JAVA DAYS ´99

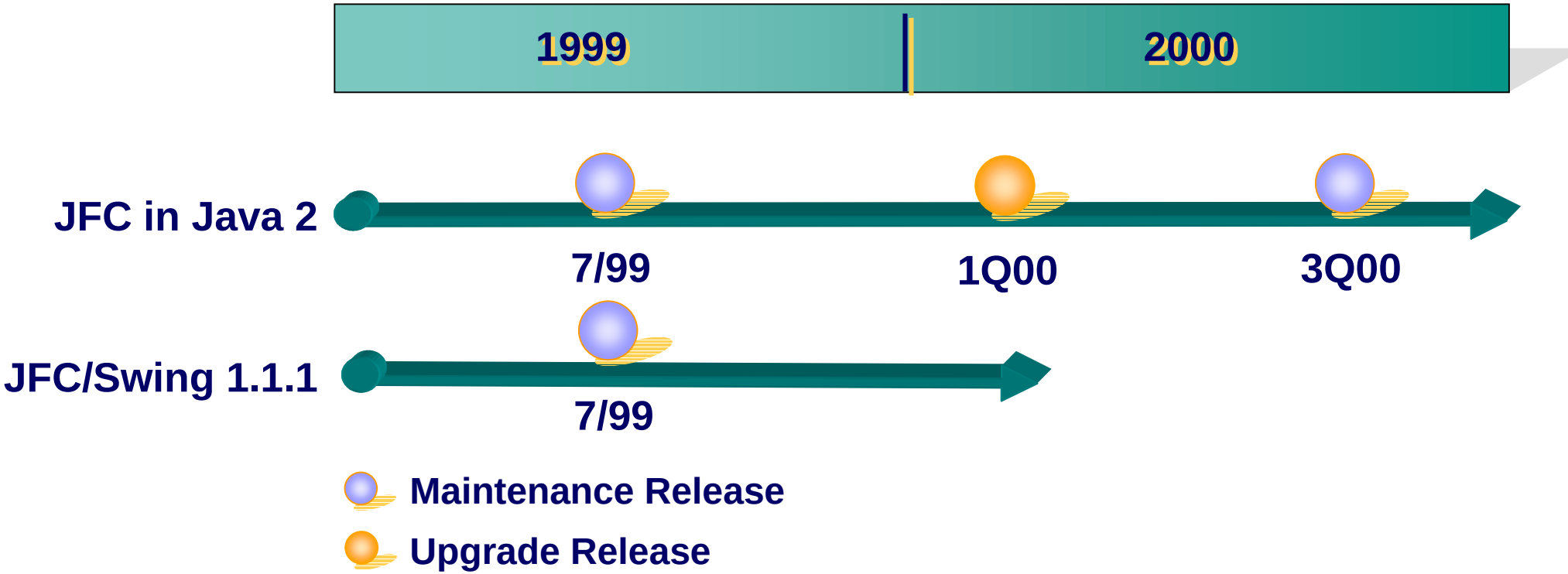



# Roadmap and Summary

Onno Kluyt
Sr. Product Manager
Sun Microsystems, Inc

Sun microsystems

# Roadmap




## Performance and new features benefits: The Java™ 2 Platform



© 1999 Sun Microsystems GmbH

# Information Pointers




- The JFC web site - http://java.sun.com/jfc
- The Swing Connection - http://java.sun.com/jfc/tsc
- JFC Directory - http://java.sun.com/jfc/directory
- Feedback - swing-feedback@java.sun.com



© 1999 Sun Microsystems GmbH

Sun microsystems