JDK 1.2 (currently b3)

- Attractive for applications, less so for applets
  - browser problems with 1.1 on some platforms, 1.2 ???
  - same event model, all 1.1 code should run under 1.2
  - augments JDK 1.1.X in several ways

- JFC/Swing (Java Foundation Classes)
  - light(er)weight widgets, also can be used with 1.1.X, so this is a possibility for applications/applets (unclear on applet use)
  - many new widgets, JButton, JFrame, JPanel, JTree, and many others, see javadoc
  - default double-buffering, debuggable graphics, ...
JFC

- Portable Look and Feel (plaf)
  - Java Look and Feel (aka Metal)
  - Windows look and feel (won’t work under Unix)
  - Motif look and feel
  - other “extras” and make-your-own

- Using JFC/Swing can result in VERY SLOW applications
  - might be faster with a JIT (just-in-time compiler)
  - slow on ultrasparcs, not tested (by ola) under NT/95

- New widgets can save time, but cumbersome under 1.1.x
  - see demo directory in the 1.1.x hierarchy
  - it’s the future, but it’s not serializable
Harpoon Ideas

- What are the tricky parts of Harpoon, what should be done first, what should be done second

- What’s a “figure”/“drawable”
  - how are figures moved?
  - how are figure resized?

- How is a presentation saved?
  - YAMAL is one choice
  - serializable is another choice
Decorator and Composite pattern

- **Composite pattern shown by AWT Container/Component**
  - character, row/column, composite
  - what is a component? what is a container?
  - useful for whole/part hierarchies, differentiate between leaf/composite, but both are components
  - what is the ABC? where are the children?

- **What about layout? Why is this tricky?**
  - How is text resized, what about fonts

- **Decorator: add responsibility, but no subclassing, what about a button with a border? A panel with a scrollbar?**
  - Visual component is base class/interface
  - decorator implements Visual, but stores a component and defers operations to component