A key claim by proponents of OO Software is that it increases reusability.
How?
Old days: Subroutine Libraries

- A library is a collection of useful subroutines.
- Little extensibility/customizability.
- *Callbacks* possible:

```c
#include "integration.h"

// integration.h declares:
// double integrate(double, double, double*(double))

// A local function
double myFunc(double x) { sin(exp(x)); }

// A call to a library routine
double area = integrate(0.0, b, &myFunc);
// Library routine calls back to myFunc.
```
• Simple libraries are collections of concrete classes.
  • Library objects are at a low level only.
• **Frameworks** are collections of concrete classes, interfaces and abstract classes such that
  • the library classes routinely call back to client level code
  • Abstract classes, concrete classes and interfaces intended to be specialized or realized by client code are called *Extension points*
  • Client objects may be used by library objects
Framework Class Diagram

Application Frameworks

- Incomplete applications
- Client code completes the application

**Example: Document/Editor Framework**

- Many applications are basically editors
- Functionality of the File menu is essentially the same.

(See `javax.swing.text.EditorKit` and `javax.swing.text.Document` for a similar example.)
abstract class Document {

    abstract public void writeTo(Writer w)
                 throws IOException;

    abstract public void readFrom(Reader r)
                 throws IOException;

    public save() {
        ... writeTo(w); modified = false; ...}

    public open() {
        ... readFrom(r); modified = false; ...}

    public void setModified() { modified = true; }

    public boolean getModified() { return modified; }

    public void setFile(File f) { file = f; }
}
abstract class EditorComponent extends Component {
    abstract public void setDocument(Document e);
    abstract public Document getDocument();
    abstract public void AddMenus(MenuBar b);
}

interface AbstractFactory {
    Document makeDocument();
    EditorComponent makeEditorComponent();
}
The Framework code

- Is parameterized by an AbstractFactory
- Handles New, Open, Close, Save, Save As, Print, and Exit actions.
- Also actions on Frames such as resize, minimize, move, close, etc (mostly inherited from Frame).
class DocEditorFrameWork {
    // Constructor
    DocEditorFrameWork(AbstractFactory f) { ... }
    ...
}

class Main {
    static public void main( ... ) {
    ...
        new DocEditorFrameWork( new Factory() );
    ...
    }
}