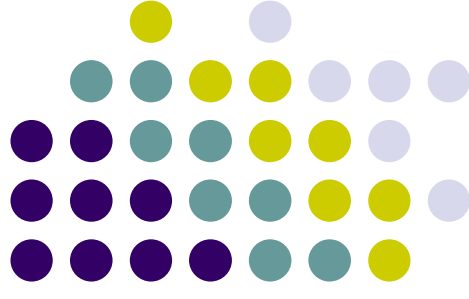
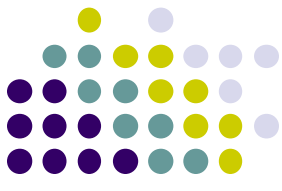


1st week discussion

ICS 52: Introduction to Software Engineering

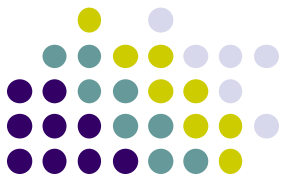
Instructor: Prof. Dan Frost
TA: Tiago Proenca
tproenca@ics.uci.edu
03.30.2011





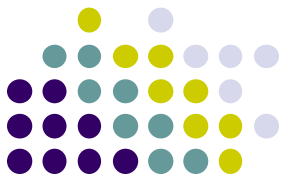
Notice

- **Contact info**
 - tproenca@ics.uci.edu
 - Please, include '[ICS52]' in subject
- **Office hour**
 - 1:00-2:00 pm on Mondays at ICS2 110
- **Grading**
 - TA and Professor will grade all homework and exams.



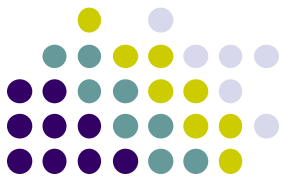
Outline

- History of Java
- Homework #A
- Introduction to SWT



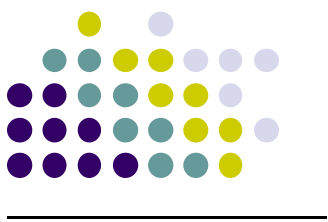
Early days

- Java “NOT” developed for World Wide Web
- Late 70s Bill Joy wanted to rewrite UNIX
 - C++ inadequate for short effective programs
- 1991 – Stealth Project: Smart consumer electronic devices
 - *“A large, distributed, heterogeneous network of consumer electronic devices all talking to each other”*
- Development of an independent language
 - Initially named Oak, then renamed to Java



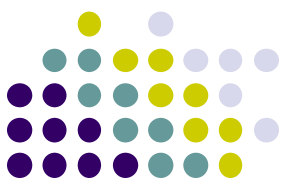
WWW

- Initial efforts of Java not successful
- 1994, WWW came alive
- Java was relaunched as the language of Internet apps
- Sun formally announced Java at SunWorld'95.
- Netscape Inc. announced that it would incorporate Java support in their browser
- Microsoft followed



Java Features

- **Platform independence**
 - Write a program once, compile it once, and run it anywhere.
 - Compromise: VM
- **Reliability:**
 - Crash and reboot not acceptable.
 - multiple-inheritance
 - implicit garbage collection thereby providing efficient memory utilization and higher reliability
 - eliminate all unsafe constructs used in C and C++ by only providing data structures within objects
- **Security**
 - pointers were excluded



History of Java

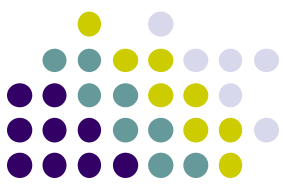
- Timeline of java:
 - <http://www.java.com/en/javahistory/timeline.jsp>
 - http://en.wikipedia.org/wiki/Java_version_history
- Java 1.0 (1996)
 - C-like language
 - Automatic garbage collection, applet of web page
- Java 1.1 (1997)
 - More scalable event model for GUI programming
 - Introduction of inner classes
- Java 1.2 (1998)
 - Introduction of Collections framework
 - Incremental improvement of Swing GUI components, other libraries and APIs
 - Different java platforms: J2EE, J2SE, J2ME
- Java 1.3 (2000)
- Java 1.4 (2002)
 - Introduction of assertions, regular expressions, and exception chaining into the core language
 - A logging API
- Java 5 (2004)
 - Introduction of generics, enumerations, and enhanced "for each" loop into core language.
- Java 6 (2006)
 - Dramatic performance improvements for the core platform and Swing.
 - Many GUI improvements
- Java 7 (scheduled for late 2010)
 - A new library for parallel computing on Multi-core processors
 - Swing Application Framework making Swing applications easier to create.





Run from Command-prompt

- **Windows:**
- <http://download.oracle.com/javase/tutorial/getStarted/cupojava/win32.html>
- **Linux:**
- <http://download.oracle.com/javase/tutorial/getStarted/cupojava/unix.html>



Run from Command-prompt

1. Install Java
2. Add the Java binary directory to the Path environment variable
3. Execute the compiler (**javac**) or interpreter (**java**) from any directory from the command-prompt



Assignment1: Enhance a mini library system

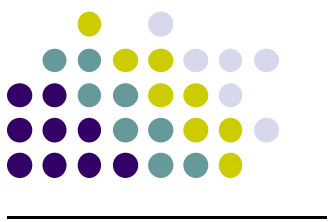
- What you have: LibFiles.zip
 - Book.java
 - BookCollection.java
 - Cardholder.java
 - ChList.java
 - EnterISBNDialog.java
 - Lib.java
 - LibFrame.java
 - Titles.dat



Assignment1: Enhance a mini library system

- What you need do:
 - Fix warnings: `javac -Xlint *.java`
 - Implement search by Title, Author and Keyword
(Use *Iterators* defined in `BookCollection.java`)
 - Implement four functionalities:
 - Put a main in `Book.java`
 - Add exit and save function
 - Add a Check In functionality for checked out books
 - Permit user to reserve a book (`book.java`)

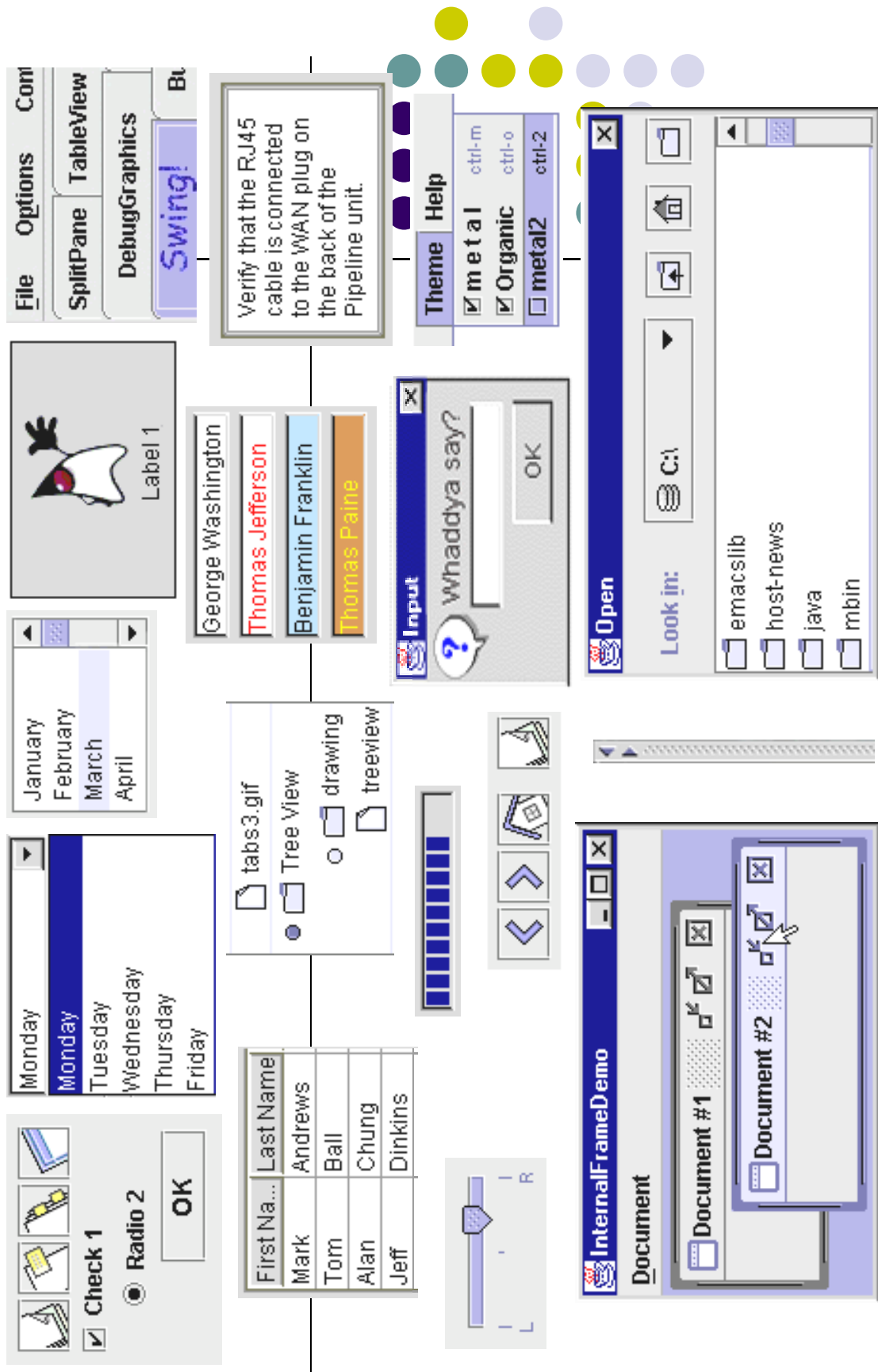
Details on: <http://www.ics.uci.edu/~frost/ics52/hwA.html>

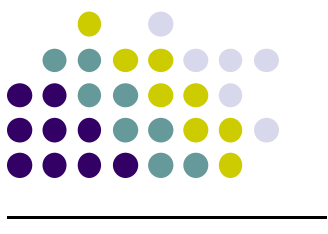


AWT vs. Swing

- Important to choose appropriate GUI components to develop a java program
- Two basic GUI components
 - AWT (Abstract Window Toolkit)
 - Connection between your app and the native GUI
 - Swing
 - Implements components based on AWT
 - Entirely JAVA, lightweight UI framework
- AWT
 - Good for simple applet development or development that targets a specific platform
- Swing
 - Good for most other JAVA GUI development

Using SWING to build user interface



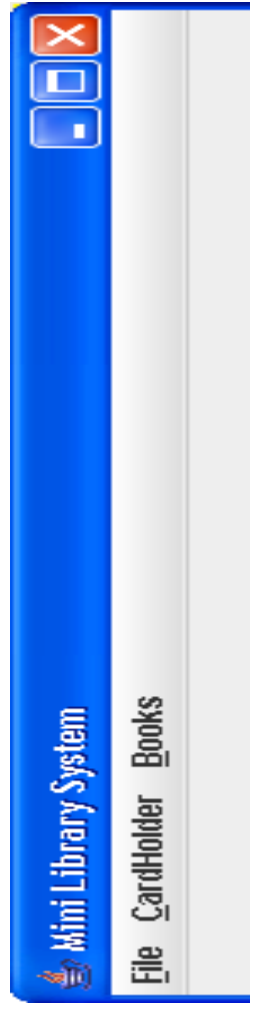
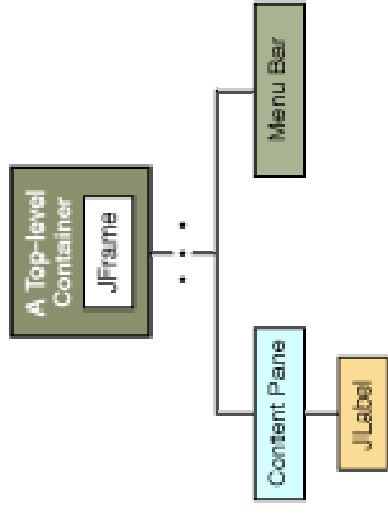
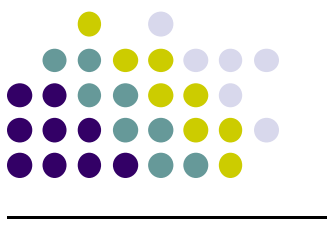


Swing

- Build the UI
- Attach the listeners to components (ones that will wait for some event, e.g. buttons)
- Java JFC/Swing Tutorial
 - <http://download.oracle.com/javase/tutorial/uiswing/start/index.htm>

JFrame

- It's a window with title, border, (optional) menu bar and user-specified components.
- It can be moved, resized, minimized.
- *The LibFrame class in the assignment extends JFrame*



JDialog [EnterISBNDialog]

- Like a frame, a dialog box is a separate window.
- Unlike a frame, however, a dialog box is not completely independent.
- Every dialog box is associated with a frame, which is called its parent.



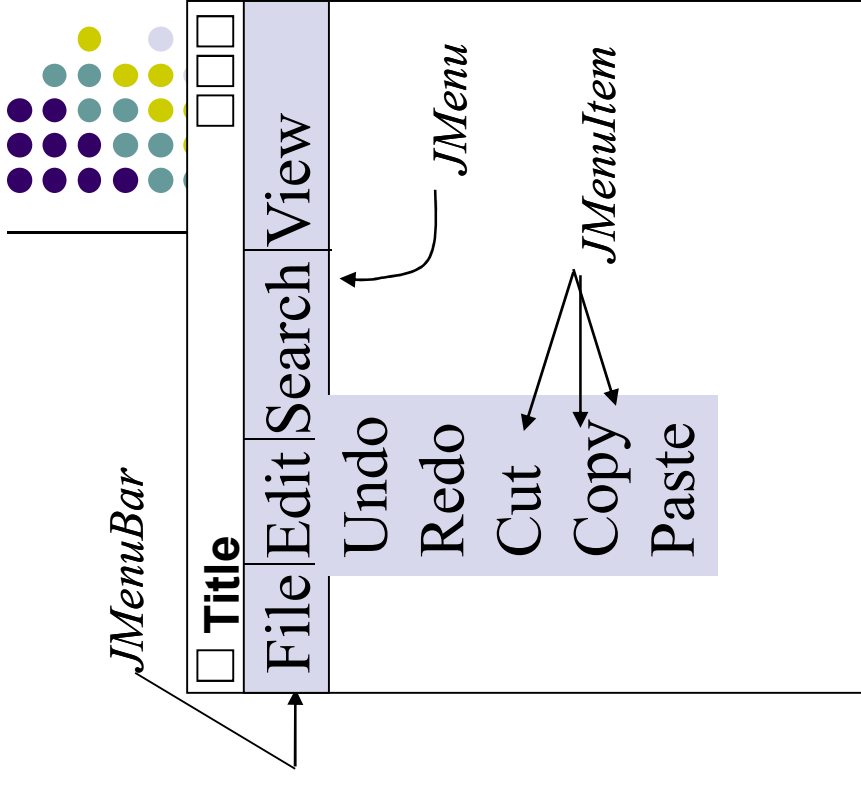
```
cancelButton.addActionListener(buttonListener); // 1
okButton.addActionListener(buttonListener);   // 2
.....
Object object = event.getSource();           // 3
if (object == okButton)                      // 4
{
.....
```

- **Create dialog boxes for the other box searches (title, author, keyword)**

JMenuBar

```
public class MenuExFrame extends JFrame // 1
    implements ActionListener           // 2
{
    public MenuExFrame() {
        // Set up frame itself – title,size,location

        JMenuBar menuBar = new JMenuBar( ); // 7
        setJMenuBar( menuBar );           // 8
        JMenu fileMenu = new JMenu( "File" ); // 9
        menuBar.add( fileMenu );          // 10
        JMenu editMenu = new JMenu( "Edit" ); // 11
        .....
        JMenuItem item; // 12
        item = new JMenuItem ( "Undo" ); // 13
        item.addActionListener( this ); // 14
        editMenu.add( item ); // 15
        item = new JMenuItem ( "Redo" ); // 16
        item.addActionListener( this ); // 17
    }
}
```



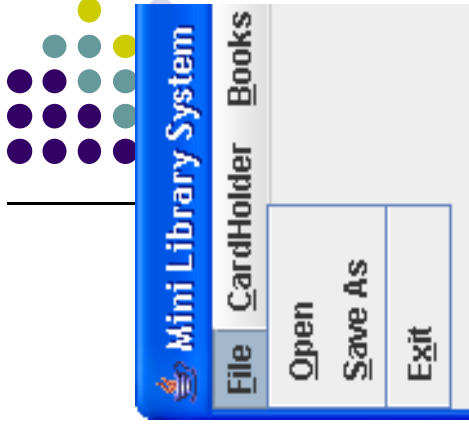
- Create a menu bar and add menus to it (lines 7, 10)
- Add the menu bar to the frame (8)
- Create menu objects and add items to them (lines 11, 15)
- Create menu items objects for your actions (lines 13, 16)

JMenuBar [LibMenuBar]

```
JMenu fileMenu, chMenu, bkMenu; // 1
JMenuItem menuItem; // 2
// Build the File menu.
fileMenu = new JMenu("File"); // 4
fileMenu.setMnemonic(KeyEvent.VK_F); // 5
add(fileMenu); // 6

// attach to it a group of JMenuItems
menuItem = new JMenuItem("Open", KeyEvent.VK_O); // 8
menuItem.addActionListener(new FileOpenListener()); // 9
fileMenu.add(menuItem); // 10
.....

class FileOpenListener implements ActionListener { // 11
    public void actionPerformed(ActionEvent e) { // 12
        parentFrame.loadData(); // 13
    } // 14
```



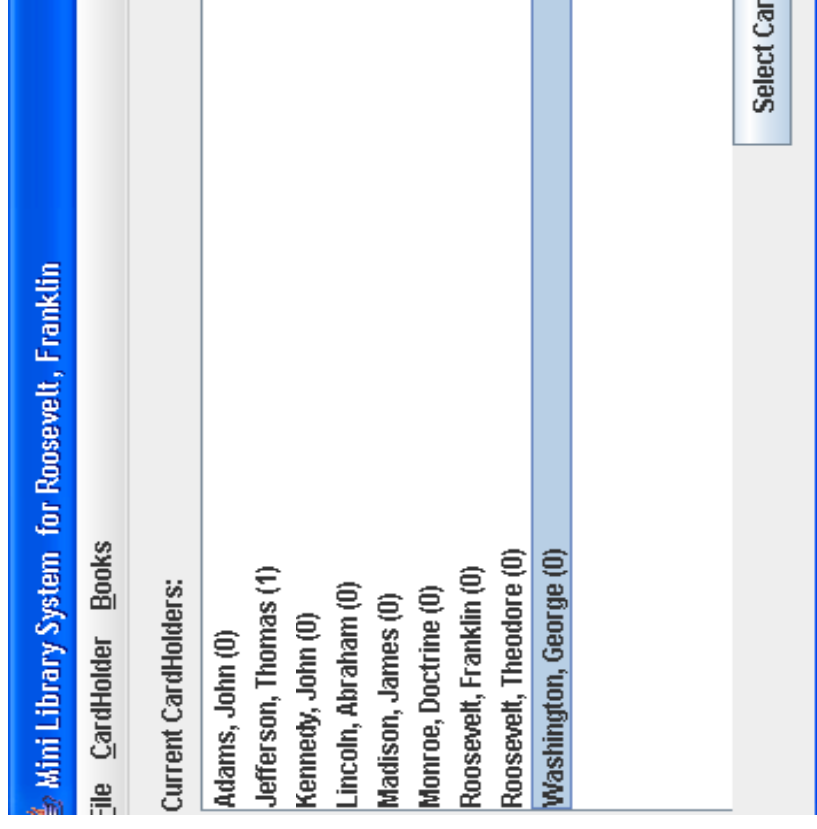
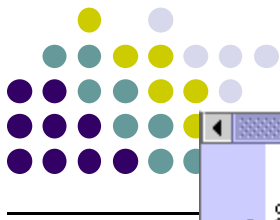
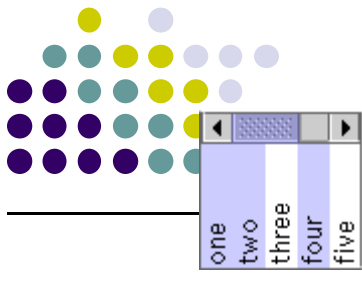
JList

- Allows users to select one or more items from a list
- Each `JList` has a

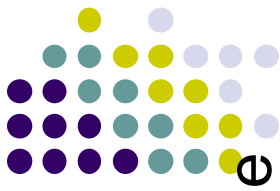
`ListSelectionModel`

- keeps track of selected items
- enforces selection mode (single, single interval, multiple interval)
- notifies listeners of selection changes

```
DefaultListModel chListModel = new DefaultListModel(); // 1
chList = new JList(chListModel); // 2
chList.setSelectionMode(ListSelectionModel.SINGLE_SELECTION); // 3
chList.setSelectedIndex(0); // 4
```

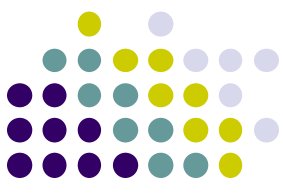


**** `SelectCardHolder()` in `LibFrame` might come in useful for other book searches.**



Main Steps

- To make any graphic program work we must be able to create windows and add content to them.
- To make this happen we must:
 1. Import the swing packages.
 2. Set up a top-level container.
 3. Fill the container with GUI components.
 4. Install listeners for GUI Components.
 5. Display the container.

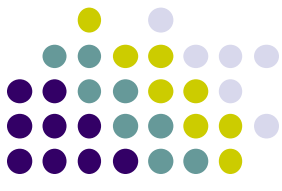


Quick Listener Example

```
 JButton button = new JButton();
 button.addActionListener(new MyActionListener());

 class MyActionListener extends ActionListener {
     public void actionPerformed(ActionEvent e) {
         JOptionPane.showMessageDialog(this, "hello");
     }
 }
```

Serialization

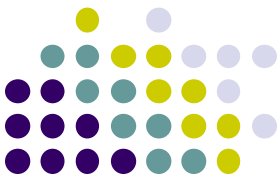


- *Object serialization* provides the ability to save an object, and its current state, so that it can be reused in another program
- The idea that an object can “live” beyond the program execution that created it is called ***persistence***
- Object serialization is accomplished using:
 - `java.io.Serializable` interface
 - `java.io.Externalizable` interface
 - Only classes that implement one of two interfaces can be serializable.
 - `ObjectOutputStream` and `ObjectInputStream` classes take charge of serialization
- The `writeObject` method is used to serialize an object
 - Manipulates stored object and add new object
- The `readObject` method is used to deserialize an object
 - Reads object stored by `writeObject`

Serialization

From LibFrame.java

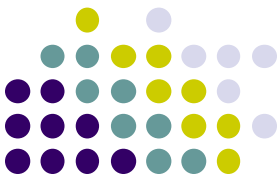
```
public void saveData()  
.....  
FileOutputStream fileOut = new FileOutputStream(file);  
ObjectOutputStream out = new ObjectOutputStream(fileOut);  
out.writeObject(lib.getChList()); // serialize  
out.writeObject(lib.getBookCollection()); // serialize
```

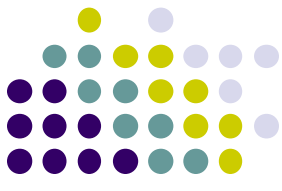


Serialization

From LibFrame.java

```
public void loadData()  
.....  
FileInputStream fileIn = new FileInputStream(file);  
ObjectInputStream in = new ObjectInputStream(fileIn);  
lib.loadChList((ChList)in.readObject());  
lib.loadBookCollection((BookCollection)in.readObject());
```





Resources

- Java API
 - <http://java.sun.com/j2se/1.5.0/docs/api/>
- Java Developer's Almanac
 - <http://www.exampledepot.com/>
- Java JFC/Swing Tutorial
 - <http://download.oracle.com/javase/tutorial/uiswing/start/index.html>