Java as a Second Language

Week 5: GUIs

CS3773
Simson Garfinkel

This week: Graphical User Interfaces

Java classes are like C++ classes, but much simpler.

- Single inheritance
- Easier-to-read syntax
- No operator overloading (but there is method overloading)
- Easy to make class (static) and instance variables.
- Easy to initialize variables.

Outline for Today:

- Redux on Homework 3
- Subversion (how to use source code control)
- GUIs
- JFrame & JButton

Homework 3 Redux

Overall, most people did a good job!

Common mistakes:

- ChargedBot should extend ChaseBot
  - chasing is a common function.
  - rangeToObject(o) & bearingToObject(o) belong in ChaseBot
  - ChaseBot needs to be re-implemented to use rangeToObject().
- Nobody bothered to submit for Schelling.
  - 10 points for submitting
  - 10 for code quality
  - 10 for getting something that worked.

Nobody got 100; don't worry about that. This course will be curved.

Homework 4 redux

Overall, very good. But please try to write code that follows the assignment, rather than just beats the validator.

Don't do this:

```
} catch (NumberFormatException e) {
    System.out.println("'foobar' is not a number.");
}
```

Do this:

```
} catch (BadLineException e) {
    System.out.printf"'%s' is not a number.",e.errorValue);
}
```

Trying to "beat the validator" is cheating.

Don't do this:

```
//String does not contain a parsable integer.
catch (NumberFormatException e){
    System.out.println("line 4: foobar is not a number.");
}
```

Do this:

```
//String does not contain a parsable integer.
catch (NumberFormatException e){
    System.out.printf("line %d: foobar is not a number.",lineNumber);
}
```

Why is beating the validator bad?

It gives the impression that code works, when it doesn't.

It presents the impression that you understand how to solve a problem, when you don't.

It won't work if the underlying problem changes – or if the validator changes.

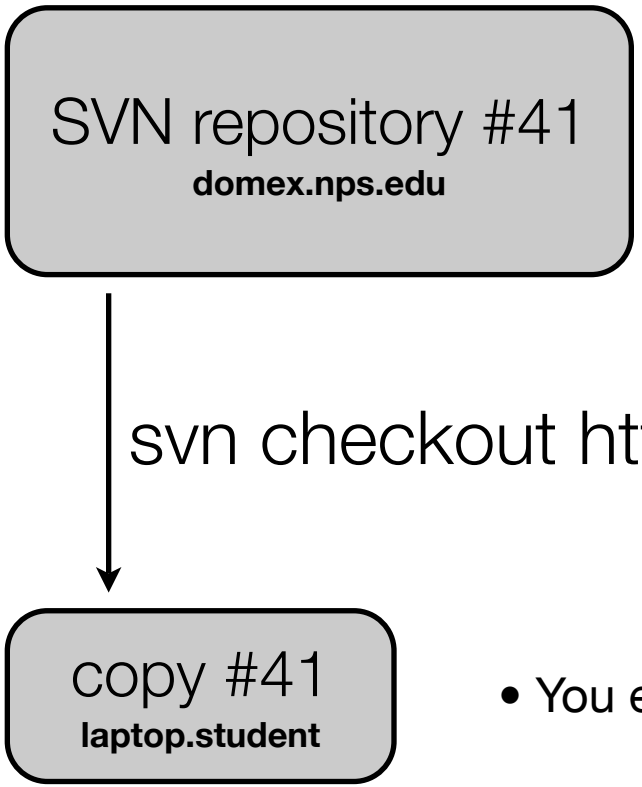
But most importantly:

PRACTICES LIKE THIS CAN KILL PEOPLE WHEN PROGRAMS ARE DEPLOYED IN THE REAL WORLD.

Subversion

http://subversion.tigris.org/

http://domex.nps.edu/cs3773/svn is the course subversion repository

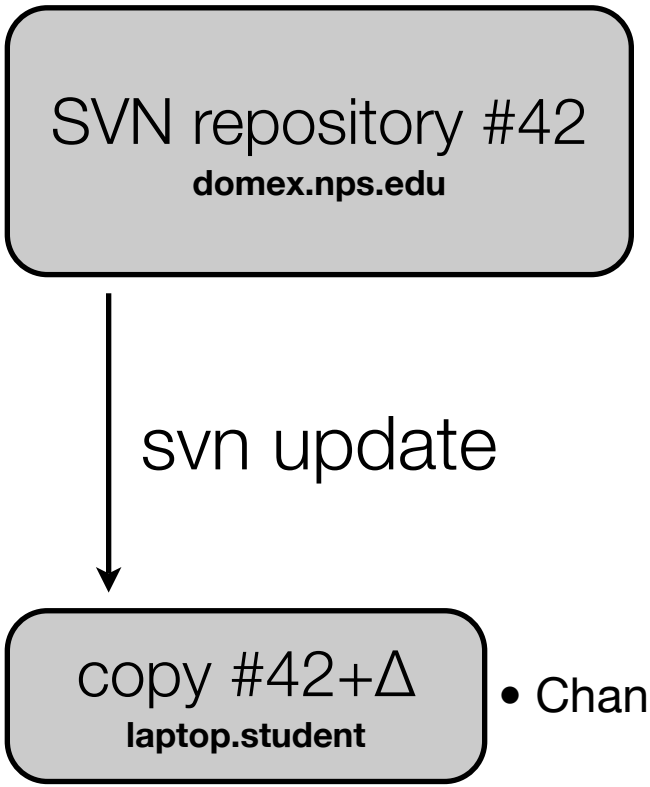


Master copy of source code.

- Current version
- All previous versions
- Comments for each commit

- You edit your local copy

When there is a change on the repository, use the "update" command to get a change.



To get from #41 to #42:

1. Insert "refresh()" after "if" statement in Bots.java
2. Delete line "return;" after the " @param */" in Charge.java
3. Delete the file "nothing.txt~"

- Changes *applied* to your source code

svn commands that you'll be using:

svn checkout — gets a copy of the repository

svn help — prints a list of the commands

svn update — applies changes in repository to your source

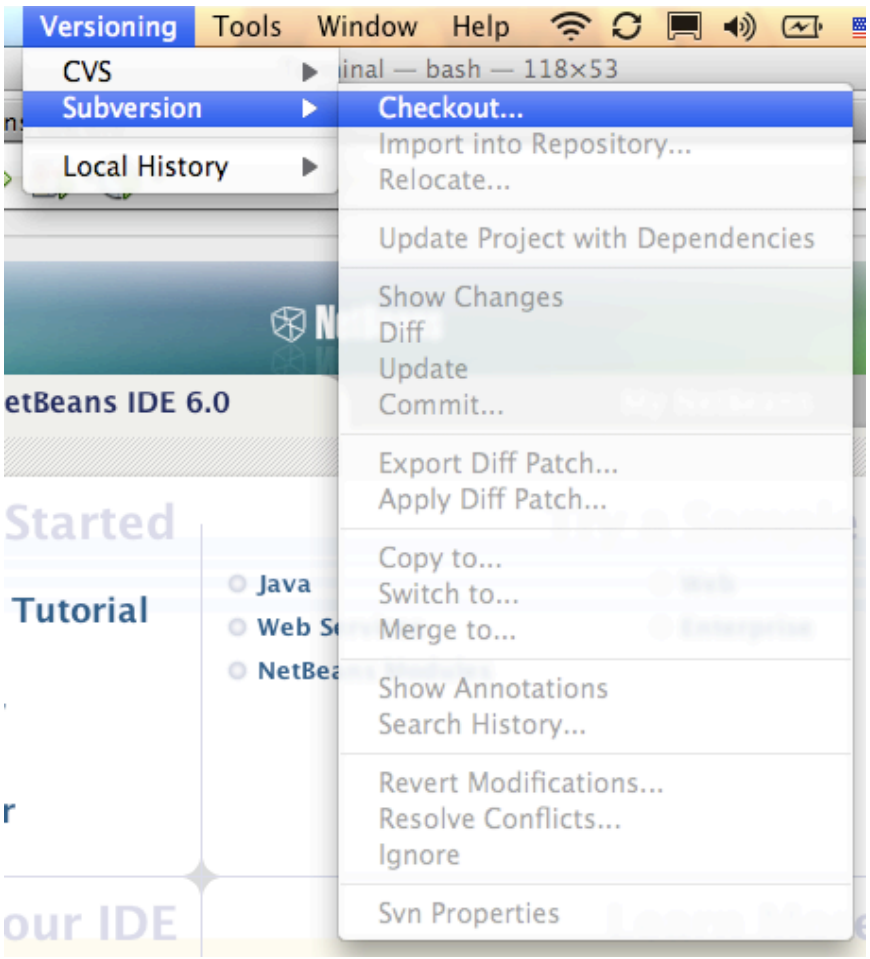
Other commands you may find useful:

svn info — prints information about the repository

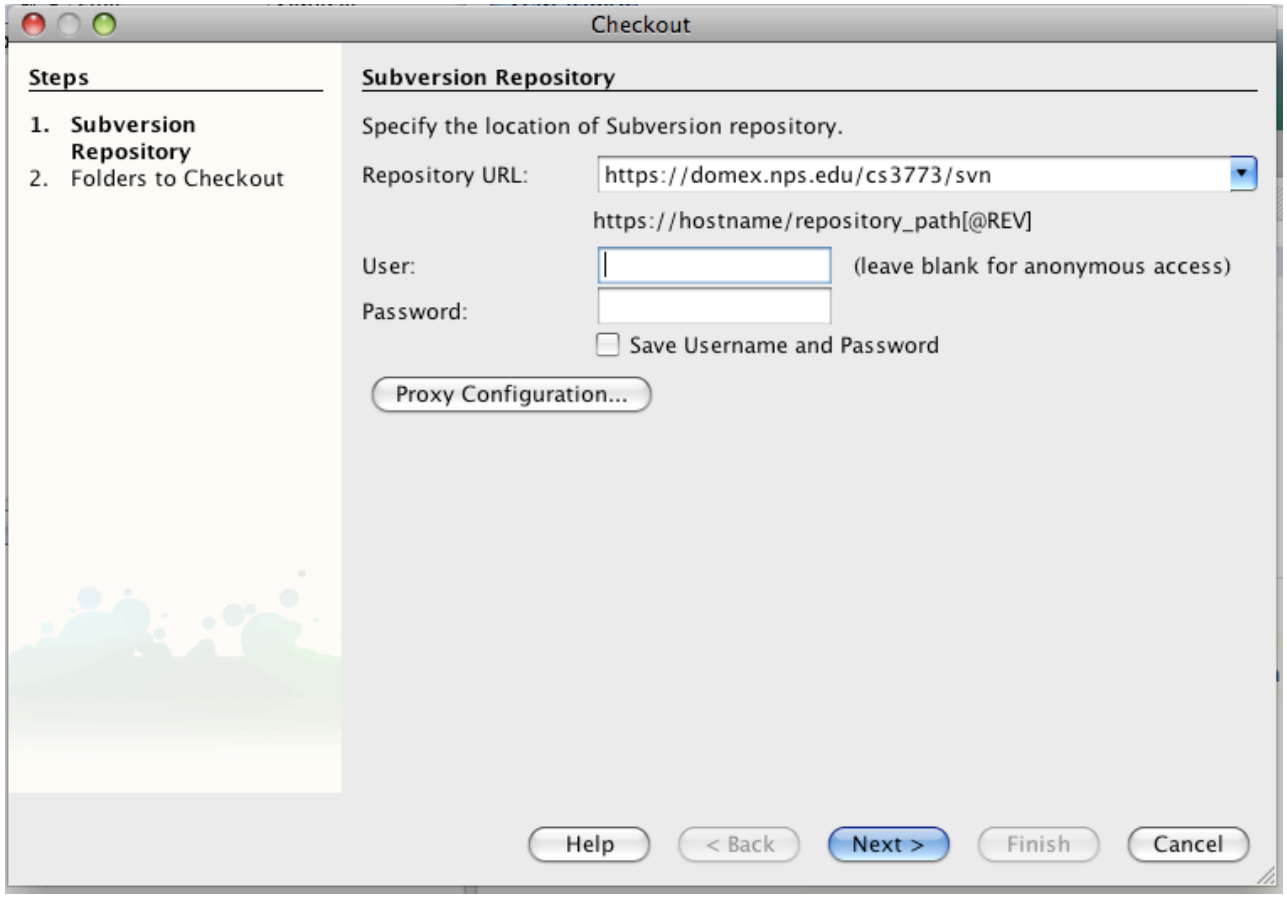
svn log — prints the repository log

svn revert — removes your local changes

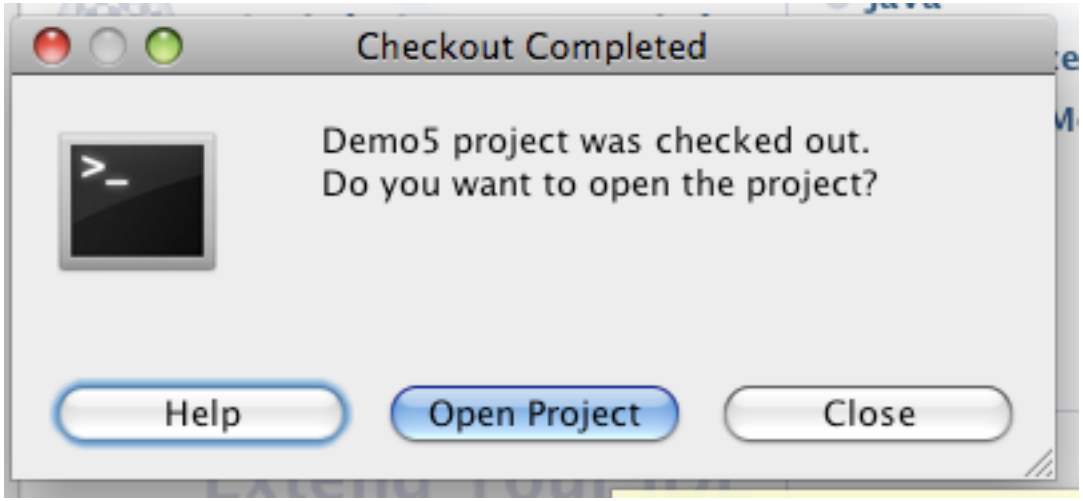
Select "Checkout"



Specify the Repository URL



After the checkout, open the project



What is a computer interface?

What is a computer interface?

A way for interfacing with a computer

What is a computer interface?

A way for interfacing with a computer

- CLI - Command Line Interface

What is a computer interface?

A way for interfacing with a computer

- CLI - Command Line Interface
- GUI - Graphical User Interface

What is a computer interface?

A way for interfacing with a computer

- CLI - Command Line Interface
- GUI - Graphical User Interface

What is a computer interface?

A way for interfacing with a computer

- CLI - Command Line Interface
- GUI - Graphical User Interface

A way for communicating within a computer program

What is a computer interface?

A way for interfacing with a computer

- CLI - Command Line Interface
- GUI - Graphical User Interface

A way for communicating within a computer program

- API — Application Programmer Interface

What is a computer interface?

A way for interfacing with a computer

- CLI - Command Line Interface
- GUI - Graphical User Interface

A way for communicating within a computer program

- API — Application Programmer Interface

What is a computer interface?

A way for interfacing with a computer

- CLI - Command Line Interface
- GUI - Graphical User Interface

A way for communicating within a computer program

- API — Application Programmer Interface

A way for communicating with the future:

What is a computer interface?

A way for interfacing with a computer

- CLI - Command Line Interface
- GUI - Graphical User Interface

A way for communicating within a computer program

- API — Application Programmer Interface

A way for communicating with the future:

- Data Structures

What is a computer interface?

A way for interfacing with a computer

- CLI - Command Line Interface
- GUI - Graphical User Interface

A way for communicating within a computer program

- API — Application Programmer Interface

A way for communicating with the future:

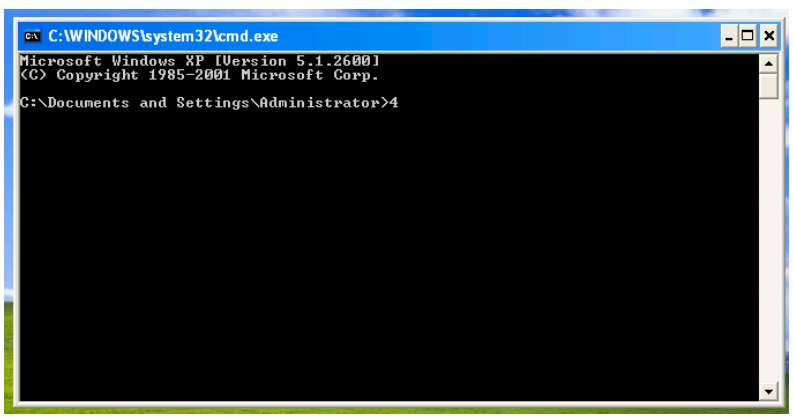
- Data Structures
- Serialization Interface

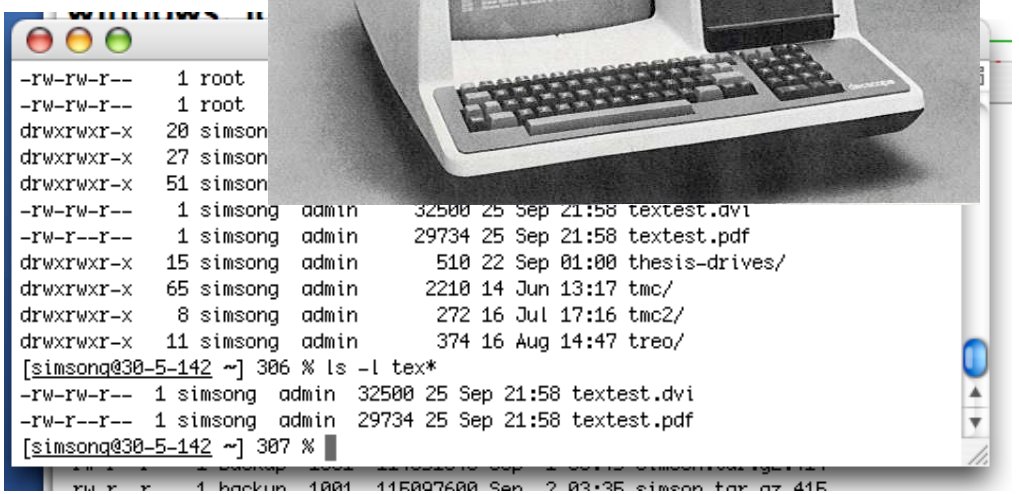
The Command Line Interface

Originally developed with teletypes & printing terminals

"Glass Teletypes"

- xterm
- Terminal.app
- command.com
- cmd.sys

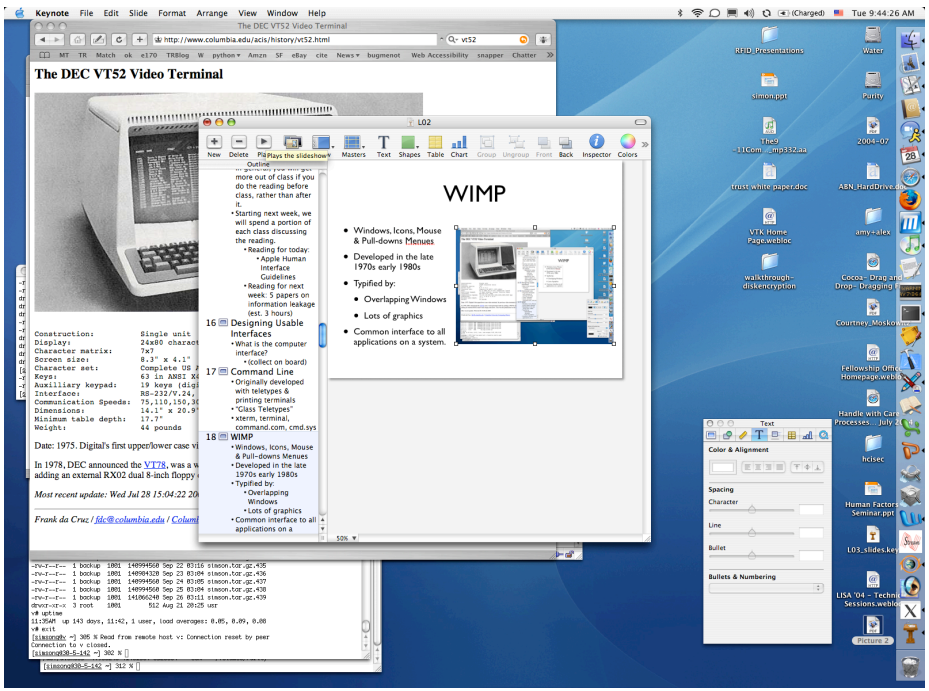




WIMP: Windows, Icons, Mouse & Pull-down menus

Developed in the late 1970s early 1980s

- Xerox Parc
- Apple
- Microsoft



Typified by:

- Overlapping Windows
- Lots of graphics

Big idea:

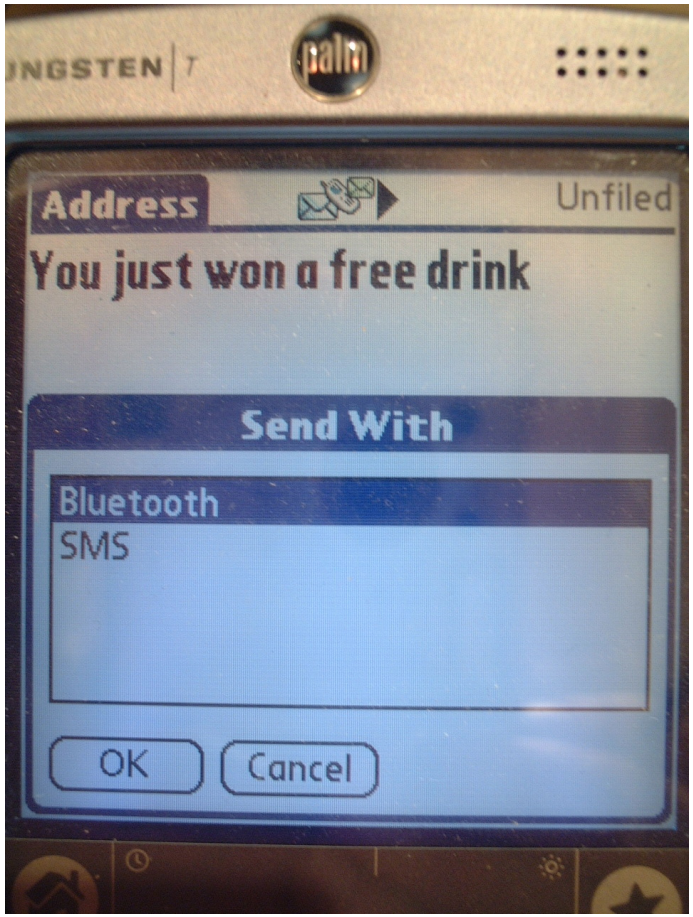
- A common interface to all applications on a system.

Tiny WIMP

PalmOS

Pocket PC

Symbian



Alternative Interfaces: Non-WIMP

Speech

Gesture

- hands
- eye-tracking

Tactile

Dance

How do we know if an interface is “usable?”

Usability: “I know it when I see it.”

- **satisfaction**: Interfaces we enjoy using
- **efficiency**: Interfaces we are fast at using
- **learnability**: Interfaces that we can use without asking for help
- **errors**: Interfaces that we can use accurately
- **memorability**: Interfaces we can use after time

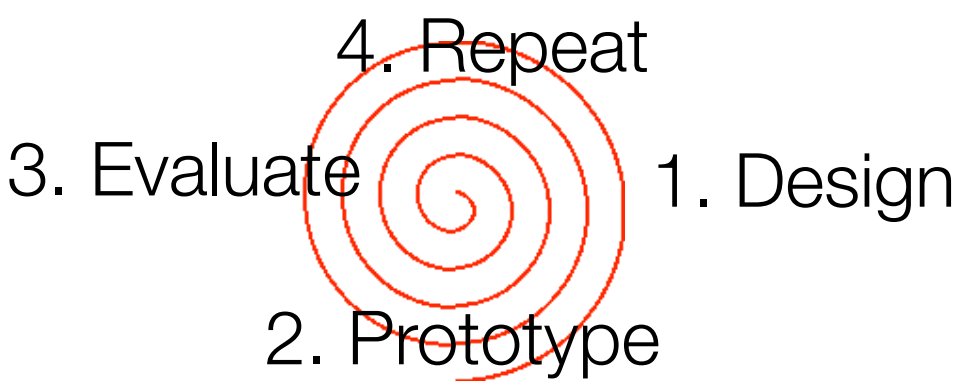
Notice that this is different than “accessible.”

Building a usable interface: The Design Cycle

Task Analysis

- What problem is the user really trying to solve?

Iterative Design (usability spiral)



Keep the customer in the picture!

Task Analysis

Observe existing work practices

Create scenarios

Create “customers” / “personas”

- Sally in accounting
- Bob the new user

Discuss ideas with end-users

Show prototypes

- Try out ideas before committing to software

Does Task Analysis Always Make sense?

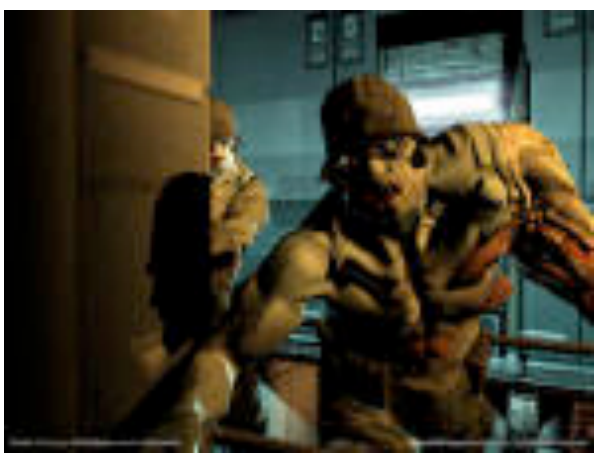
Q: What is the task that a user in this game is trying to solve?

Game-based answers:

- Maintain situational awareness
- Locate & kill enemies

Marketing-based answers:

- Rapidly master the game controls
- Advance through the game story
- Purchase the next version



“Doom as a user interface for process management”

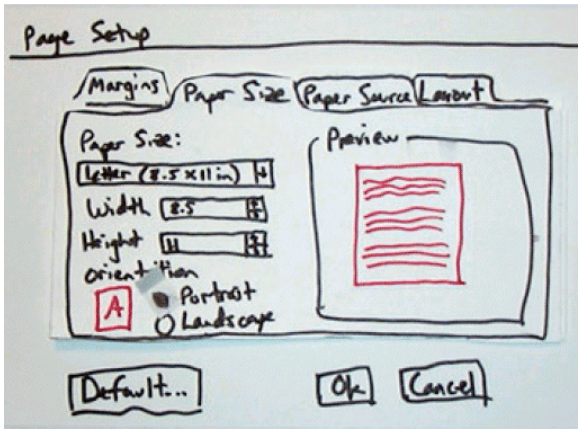
2001 Proceedings of the SIGCHI conference on Human factors in computing systems

http://citeseer.ist.psu.edu/chao01doom.html

"Paper" Prototyping

Build a mock-up

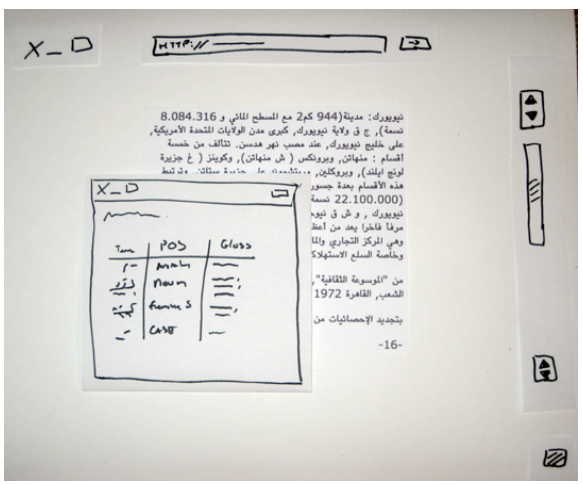
- Paper!
- Adobe Illustrator / Photoshop
- NetBeans Interface Designer



http://www.snyderconsulting.net/

Advantages:

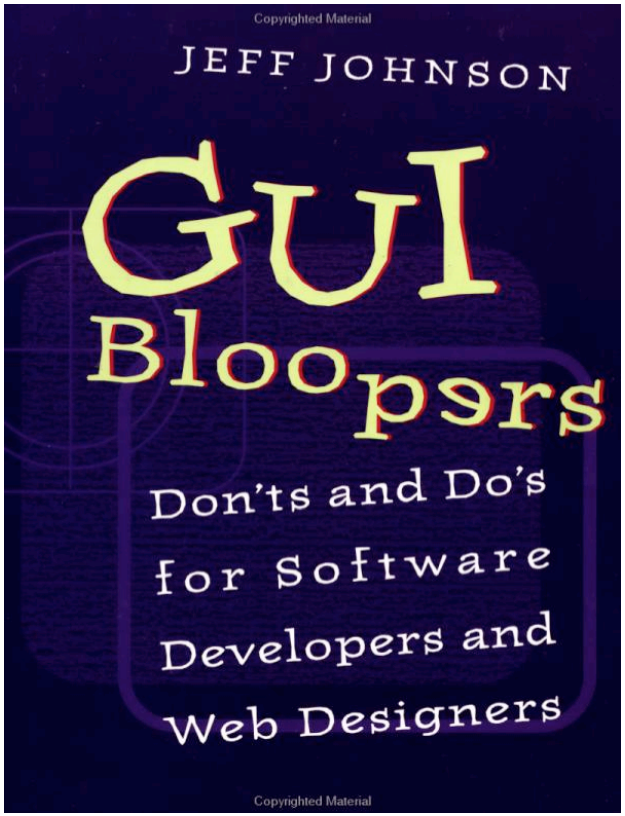
- Helps you to understand the problem.
- Gets feedback before you start to code.
  http://www.useit.com/alertbox/20030414.html



http://www.alistapart.com/

Designing usable interfaces

Jeff Johnson, GUI Bloopers:
Don’t and Do’s for Software Developers
and Web Designers, Morgan Kaufmann, 2000



Principle #1
Focus on the users and their tasks, not the technology

For whom is this product being designed?

What is the product for?

What problems do the users have now?

What are the skills and knowledge of the users?

How do users conceptualize and work with their data?

Principle #2
Consider function first, presentation later

Does not mean “worry about the user interface later!”

Develop a conceptual model

Keep it as simple as possible, but no simpler

Develop a lexicon (***)

Principle #3:
Conform to the users' view of the task

Strive for naturalness

Use the user's vocabulary, not your own

Keep program internals inside the program

(remember, the implementation can change!)

Principle #4
Don’t complicate the user’s task

Common tasks should be easy

Don’t give users extra problems to solve

- Converting a file format from TIFF to JPG for web publishing
- Installing program “A” in order to install program “B”
- Looking up information one screen to type it on another

Principle #5
Promote Learning Inside the Interface

Think “outside-in,” not “inside-out” — The user wants to solve a problem, not learn how to use your program!

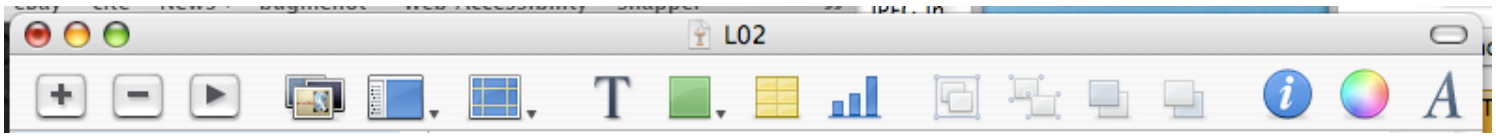
Be careful of ambiguity

- “He saw the woman with the telescope”
- Icons that don’t make sense

Be consistent so there is something to learn!

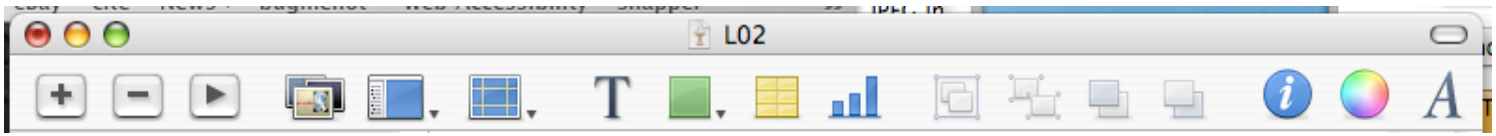
Icon Bars (Principle #5)

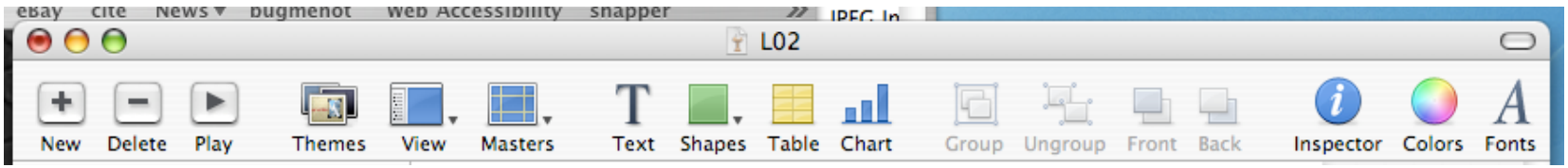
What do these icons mean?



Icon Bars (Principle #5)

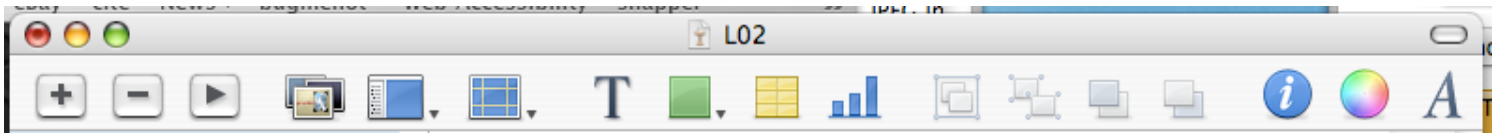
What do these icons mean?

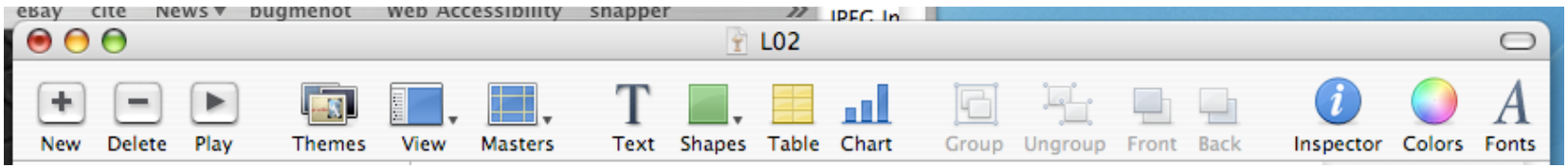




Icon Bars (Principle #5)

What do these icons mean?

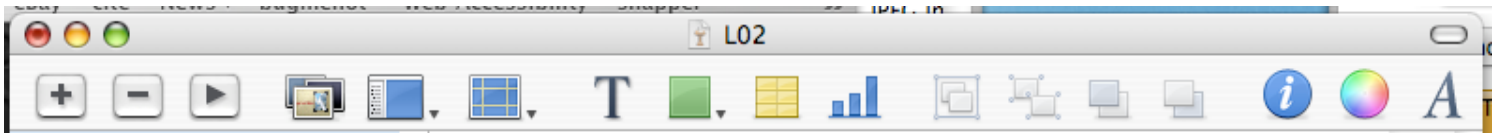


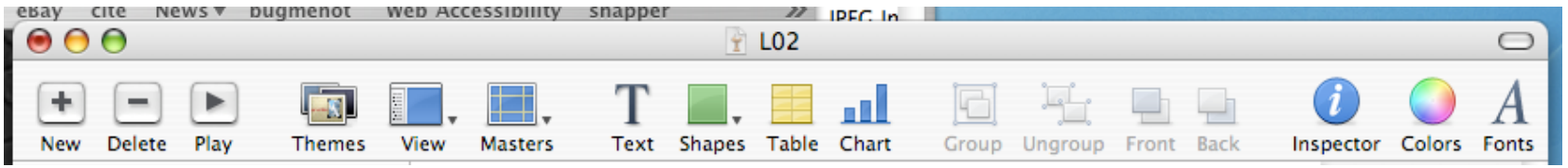


How about if we just used text?

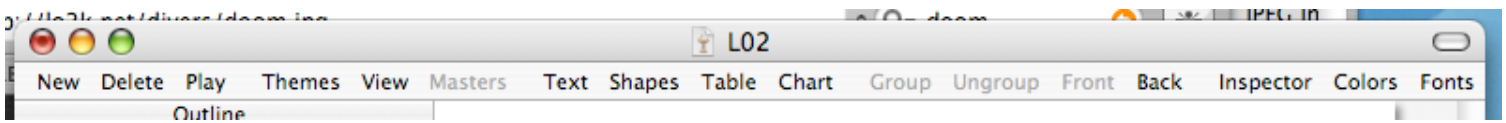
Icon Bars (Principle #5)

What do these icons mean?





How about if we just used text?



Principle #6
Deliver information, not just data

Design displays carefully

The screen belongs to the user

Preserve display inertia

The Two Most Important Principles!

Principle 7: Design for responsiveness

- Many users will forgive a bad interface, as long as it is fast.

Principle 8: Try it out on users, then fix it!

- Testing and iteration are the keys to good interface design.

- In most cases, programmers design for themselves... is that a good thing? Is that a bad thing?

User Interfaces are Hard to Design

You are not the user

- Most software engineering is about communicating with other programmers
- UI is about communicating with users

The user is always right

- Consistent problems are the system's fault

... but the user is not always right

- user's aren't designers

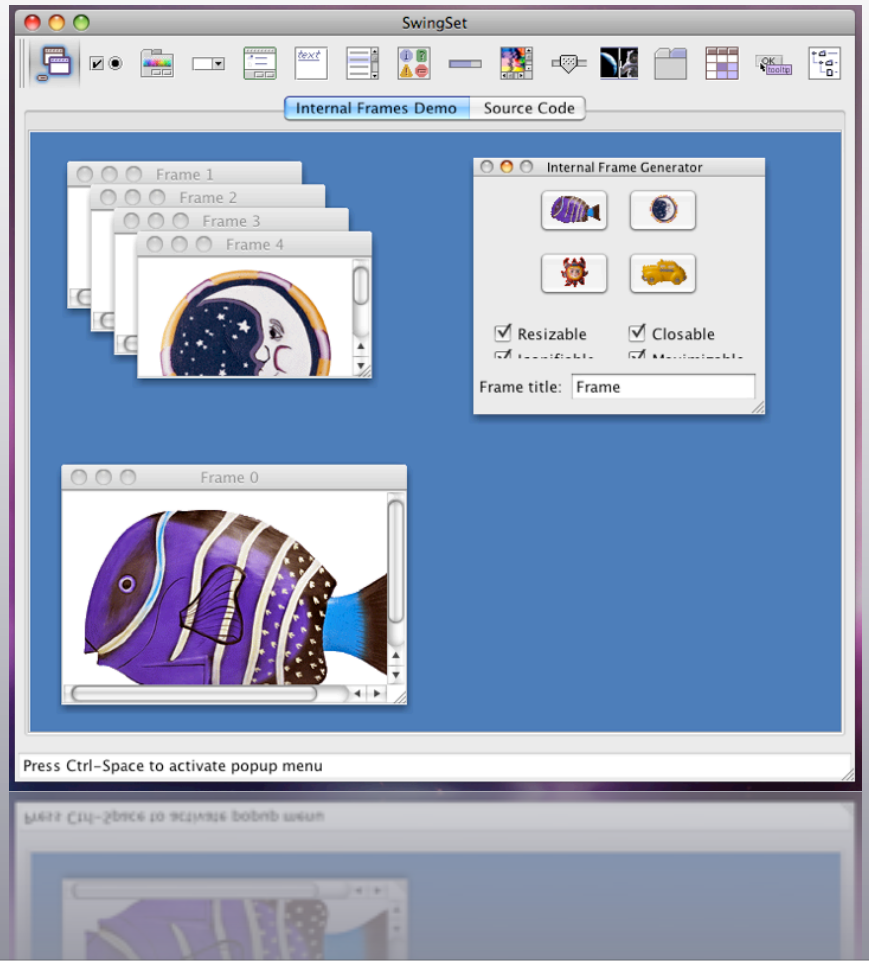
UI's are half the game:

Myers & Rosson, "Survey on user interface programming", CHI '92

User Interfaces account for 50% of:

- Design time
- Implementation time
- Maintenance time
- Code Size

(probably more now!)



Making GUIs with Java & Swing

GUIs in Java

AWT - Abstract Window Toolkit

- import java.awt.*;
- Introduced with Java 1.0
- Used *native implementations* for windows, buttons, text objects, menues, etc.
- A disaster: buttons were different on every platform!
- Included with Internet Explorer

JFC/Swing

- Introduced with Java 1.2
- Microsoft refused to adopt (because it hated Sun by then)

SWT - Standard Widget Toolkit (part of Eclipse)

& others

javax.swing
JFRAME

java.lang.Object
  java.awt.Component
    java.awt.Container
      java.awt.Window
        java.awt.Frame
          javax.swing.JFrame

Everything that appears on the screen is drawn inside a JFrame

Frames have:

- contentPane
- Other stuff

http://java.sun.com/j2se/1.5.0/docs/api/javax/swing/JFrame.html

http://java.sun.com/docs/books/tutorial/uiswing/components/rootpane.html

Let's look at a simple Swing Program:

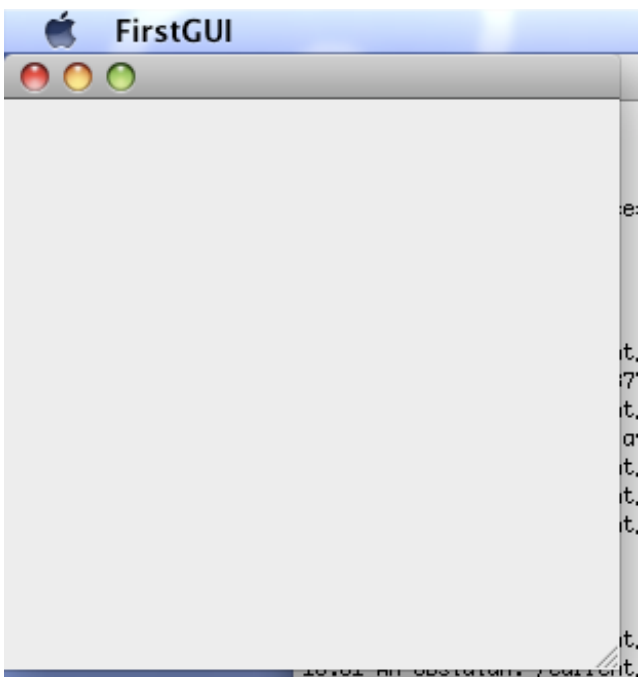
```
import javax.swing.*;

public class FirstGUI {
   public static void main(String[] args) {
       JFrame frame = new JFrame();
       frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   }
}
```

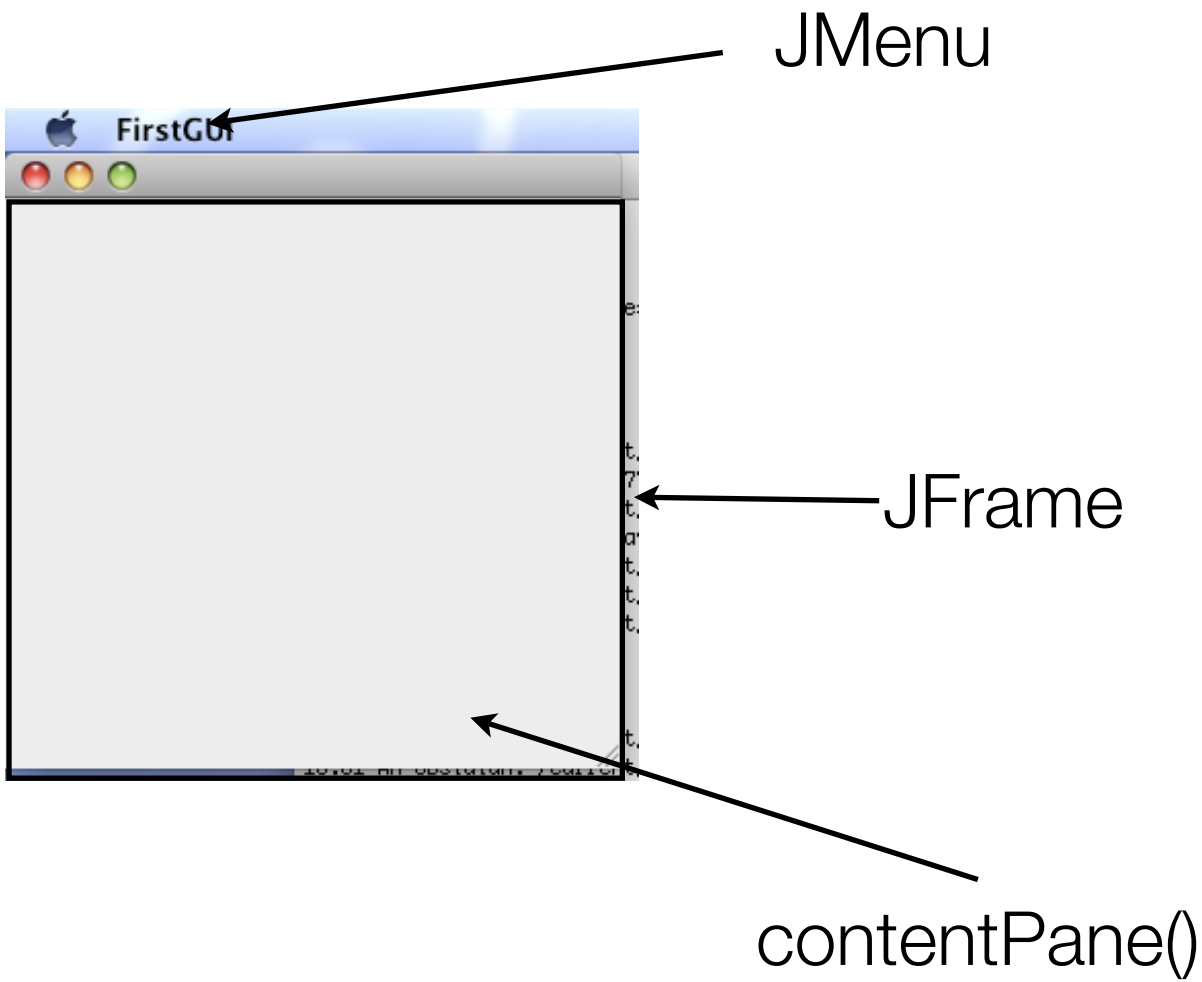
Let's look at a simple Swing Program:

```
import javax.swing.*;

public class FirstGUI {
   public static void main(String[] args) {
       JFrame frame = new JFrame();
       frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
       frame.setVisible(true);
   }
}
```

Let's look at a simple Swing Program:

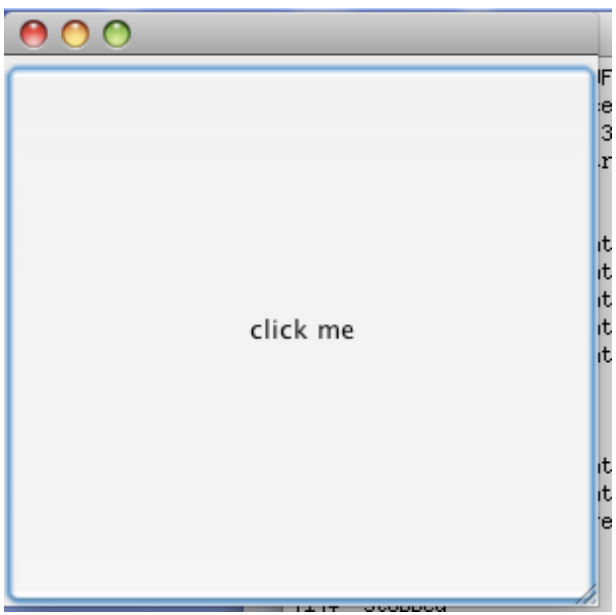
```
import javax.swing.*;

public class FirstGUI {
   public static void main(String[] args) {
       JFrame frame = new JFrame();
       frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
       frame.setSize(300,300);
       frame.setVisible(true);
   }
}
```




JMenu
FirstGUI
JFrame
contentPane()

Add a clickable button.

```
public class FirstGUI {
    public static void main(String[] args) {
        JFrame frame = new JFrame();
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        JButton button = new JButton("click me");
        frame.getContentPane().add(button);
        frame.setSize(300,300);
        frame.setVisible(true);
    }
}
```



Swing components allow HTML for titles...

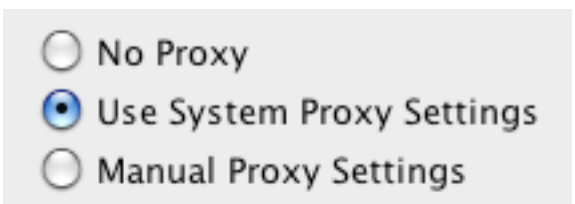
```
JButton button =
new JButton("<html><b>bold</b><br><i>itallic</i></html>");
```

javax.swing
JButton

java.lang.Object
  java.awt.Component
    java.awt.Container
      javax.swing.JComponent
        javax.swing.AbstractButton
          javax.swing.JButton

Terminology:

- *title* or *text* — what's displayed on the button
- *enabled/disabled* — Can the button be clicked?
- *pressed* — the button is being clicked on
- *selected* — is the the *selected* state
- *icon* — image displayed by the button
- *ButtonGroup* — a set of mutually exclusive buttons



Reference:

http://java.sun.com/docs/books/tutorial/uiswing/components/button.html

http://java.sun.com/j2se/1.5.0/docs/api/javax/swing/JButton.html