Western

UNIVERSITY · CANADA

Topic 7

Focusing on Users and their Tasks (Chapter 7)

Computer Science 2212b
Introduction to Software Engineering
Winter 2014

Jeff Shantz
jeff@csd.uwo

7.1 – User Centered Design

Software development should focus on the needs of users

- Understand your users
- Design software based on an understanding of the users' tasks
- Ensure users are involved in decision making processes
- Design the user interface following guidelines for good usability
- Have users work with and give their feedback about prototypes, on-line help and draft user manuals

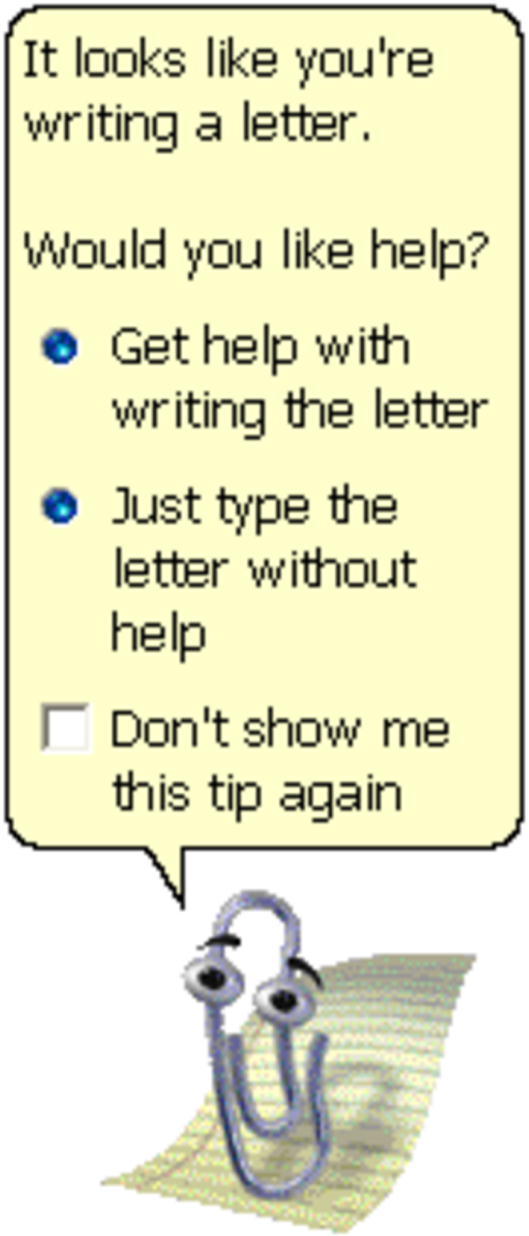
The Importance of Focusing on Users

- Reduced training and support costs
- Reduced time to learn the system
- Greater efficiency of use
- Reduced costs by only developing features that are needed
- Reduced costs associated with changing the system later
- Better prioritizing of work for iterative development
- Greater attractiveness of the system, so users will be more willing to buy and use it

7.2 – Characteristics of Users

Software engineers must develop an understanding of the users

- Goals for using the system
- Potential patterns of use
- Demographics: age, educational background, etc.
- Knowledge of the domain and of computers
- Physical ability
- Psychological traits and emotional feelings


It looks like you're writing a letter.
Would you like help?
Get help with writing the letter
Just type the letter without help
Don't show me this tip again

7.3 – Basics of User Interface Design

- User interface design should be done in conjunction with other software engineering activities.
- Do use case analysis to help define the tasks that the UI must help the user perform.
- Do iterative UI prototyping to address the use cases.
- Results of prototyping will enable you to finalize the requirements.

Usability vs. Utility

Does the system provide the *raw capabilities* to allow the user to achieve their goal?

- This is *utility*.

Does the system allow the user to *learn* and to *use* the raw capabilities easily?

- This is *usability*.

***Both* utility and usability are essential.**

- They must be measured in the context of particular types of users.

Aspects of Usability

Usability can be divided into four separate aspects:

1. Learnability
   - The speed with which a new user can become proficient with the system.
2. Efficiency of use
   - How fast an expert user can do their work.

Aspects of Usability

Usability can be divided into four separate aspects:

3. Error handling

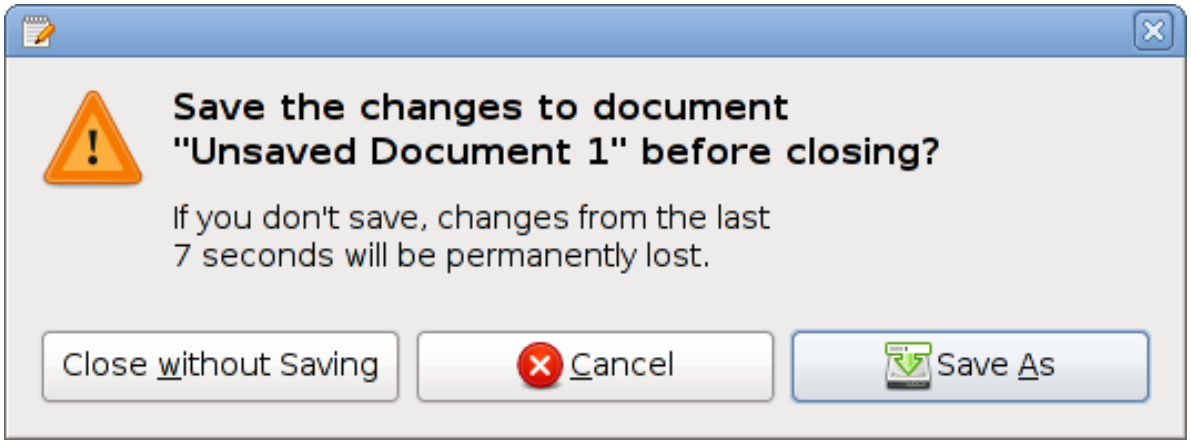
- The extent to which it prevents the user from making errors, detects errors, and helps to correct errors.
- Think of spell checking and quitting in Microsoft Word

4. Acceptability

- The extent to which users like the system.

Terminology of User Interface Design

- **Dialog:** A specific window with which a user can interact, but which is not the main UI window.



- **Mode:** A situation in which the UI restricts what the user can do.
- **Modal dialog:** A dialog in which the system is in a very restrictive mode.

Terminology of User Interface Design

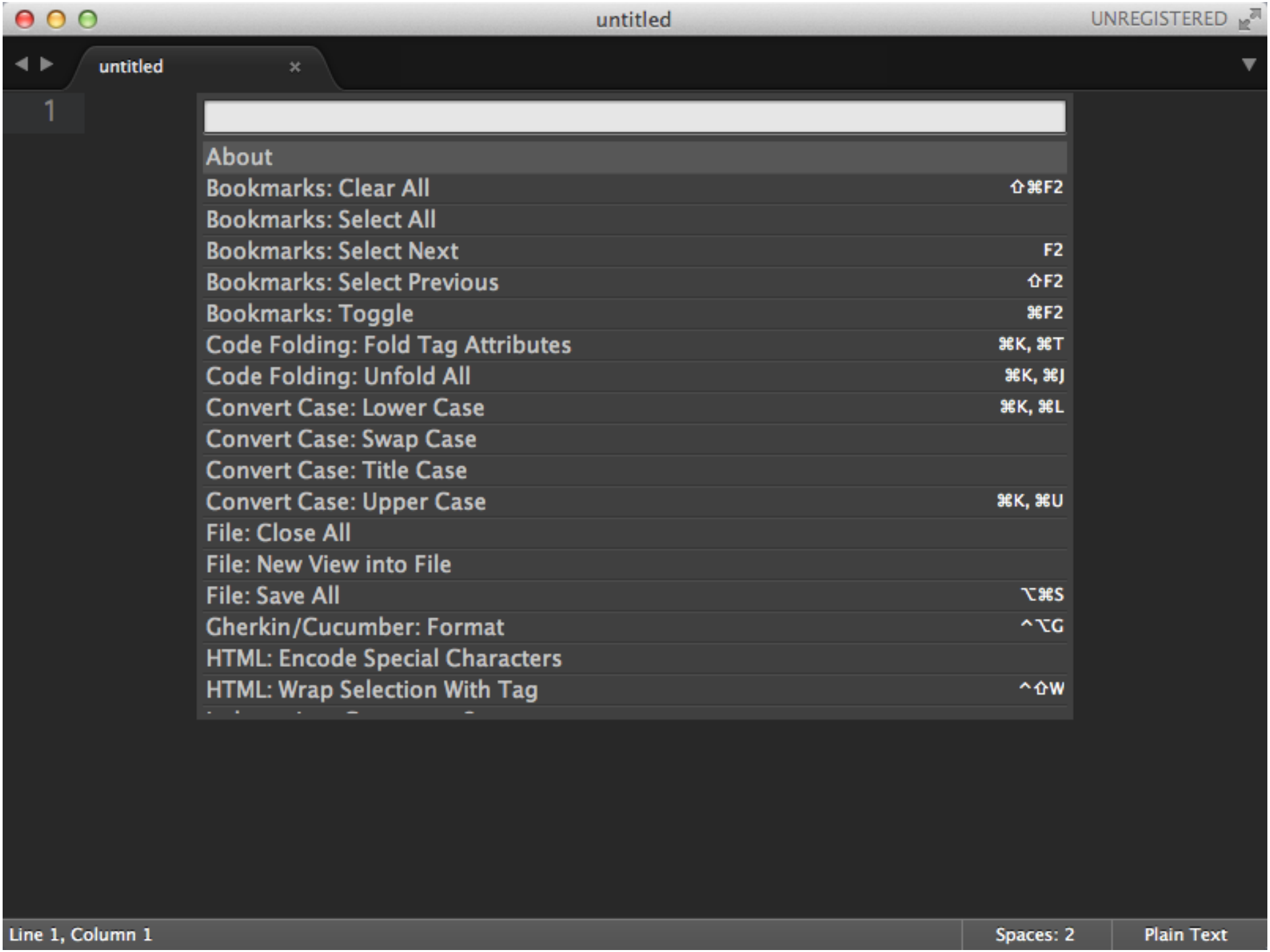
- **Control or Widget:** Specific components of a user interface.
- **Affordance:** The set of operations that the user can do at any given point in time.
- **State:** At any stage in the dialog, the system is displaying certain information in certain controls, and has a certain affordance.
- **Feedback:** The response from the system whenever the user does something, is called feedback.
- **Encoding techniques:** Ways of encoding information so as to communicate it to the user.

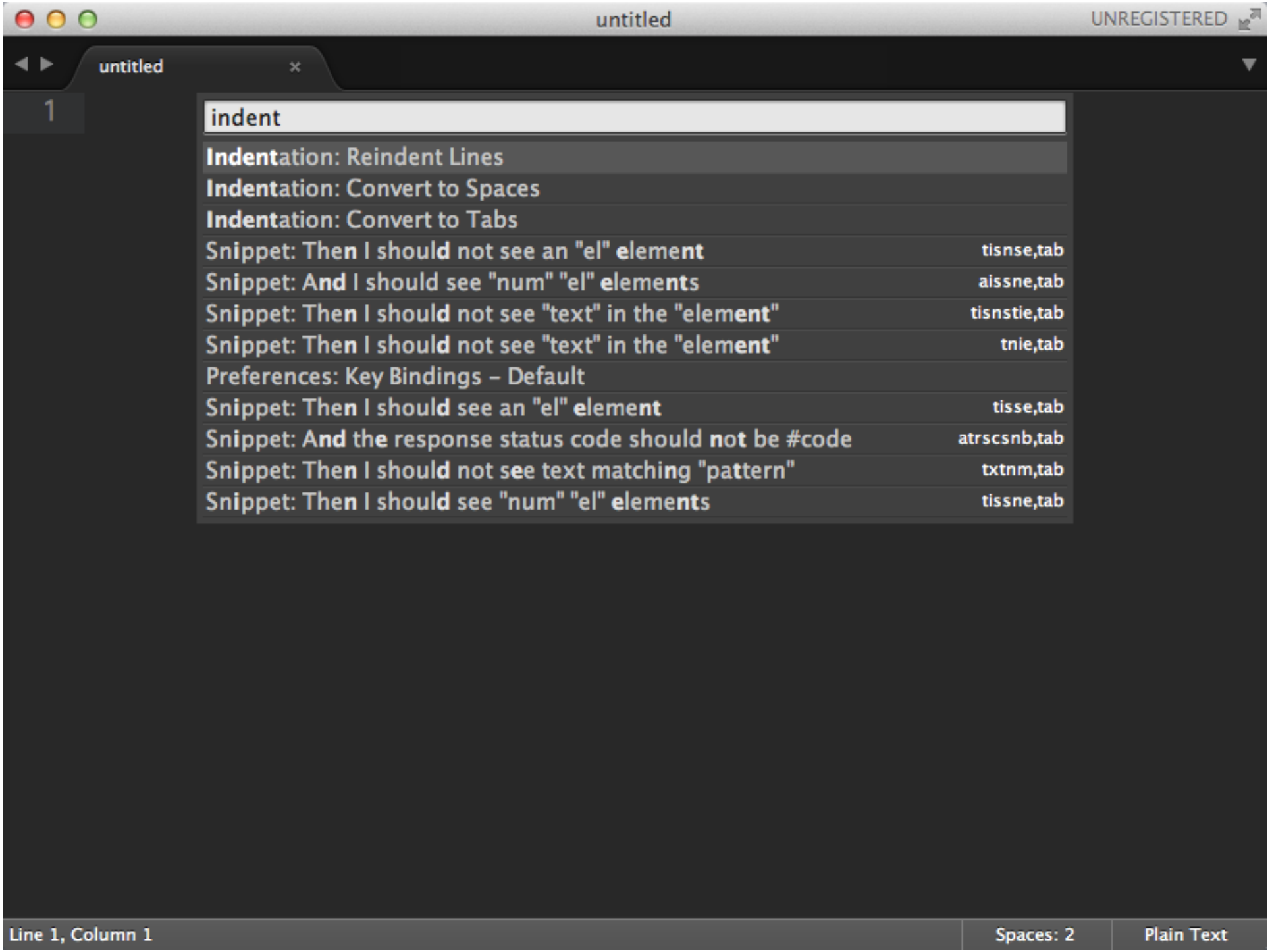
7.4 – Usability Principles

1. **Do not rely only on usability guidelines – always test with users.**
   - Usability guidelines have exceptions; you can only be confident that a UI is good if you test it successfully with users.
2. **Base UI designs on users' tasks.**
   - Perform use case analysis to structure the UI.

7.4 – Usability Principles

3. **Ensure that the sequences of actions to achieve a task are as simple as possible.**
   - Reduce the amount of reading and manipulation the user has to do.
   - Ensure the user does not have to navigate anywhere to do subsequent steps of a task.
4. **Ensure that the user always knows what he or she can and should do next.**
   - Ensure that the user can see *what commands are available* and are not available.
   - Make the *most important commands stand out*.

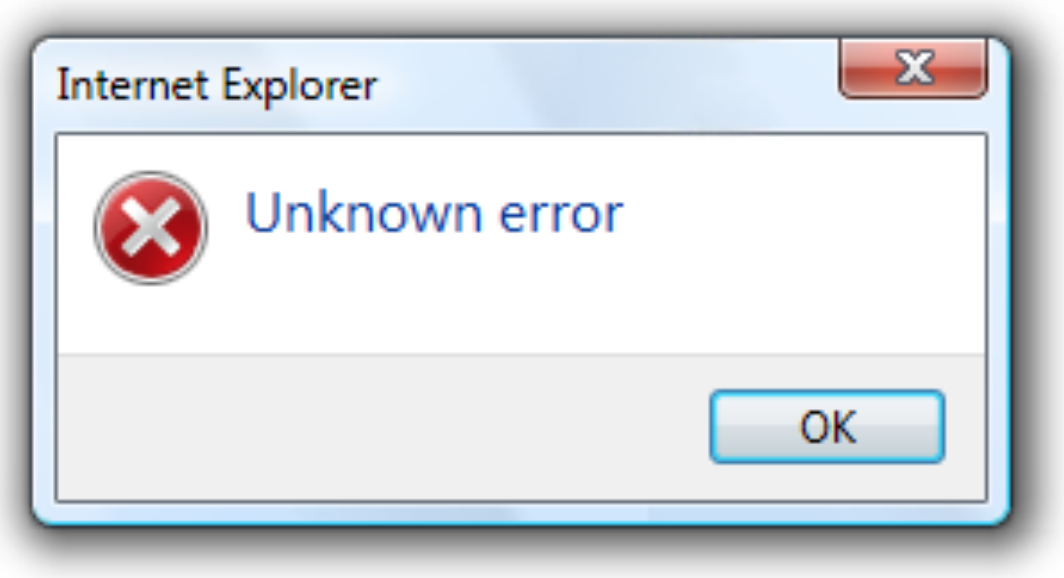

untitled
UNREGISTERED
untitled
1
About
Bookmarks: Clear All ⇧⌘F2
Bookmarks: Select All
Bookmarks: Select Next F2
Bookmarks: Select Previous ⇧F2
Bookmarks: Toggle ⌘F2
Code Folding: Fold Tag Attributes ⌘K, ⌘T
Code Folding: Unfold All ⌘K, ⌘J
Convert Case: Lower Case ⌘K, ⌘L
Convert Case: Swap Case
Convert Case: Title Case
Convert Case: Upper Case ⌘K, ⌘U
File: Close All
File: New View into File
File: Save All ⌥⌘S
Gherkin/Cucumber: Format ^⌥G
HTML: Encode Special Characters
HTML: Wrap Selection With Tag ^⇧W
Line 1, Column 1
Spaces: 2
Plain Text


untitled
UNREGISTERED
untitled
1
indent
Indentation: Reindent Lines
Indentation: Convert to Spaces
Indentation: Convert to Tabs
Snippet: Then I should not see an "el" element tisnse,tab
Snippet: And I should see "num" "el" elements aissne,tab
Snippet: Then I should not see "text" in the "element" tisnstie,tab
Snippet: Then I should not see "text" in the "element" tnie,tab
Preferences: Key Bindings – Default
Snippet: Then I should see an "el" element tisse,tab
Snippet: And the response status code should not be #code atrscsnb,tab
Snippet: Then I should not see text matching "pattern" txtnm,tab
Snippet: Then I should see "num" "el" elements tissne,tab
Line 1, Column 1
Spaces: 2
Plain Text

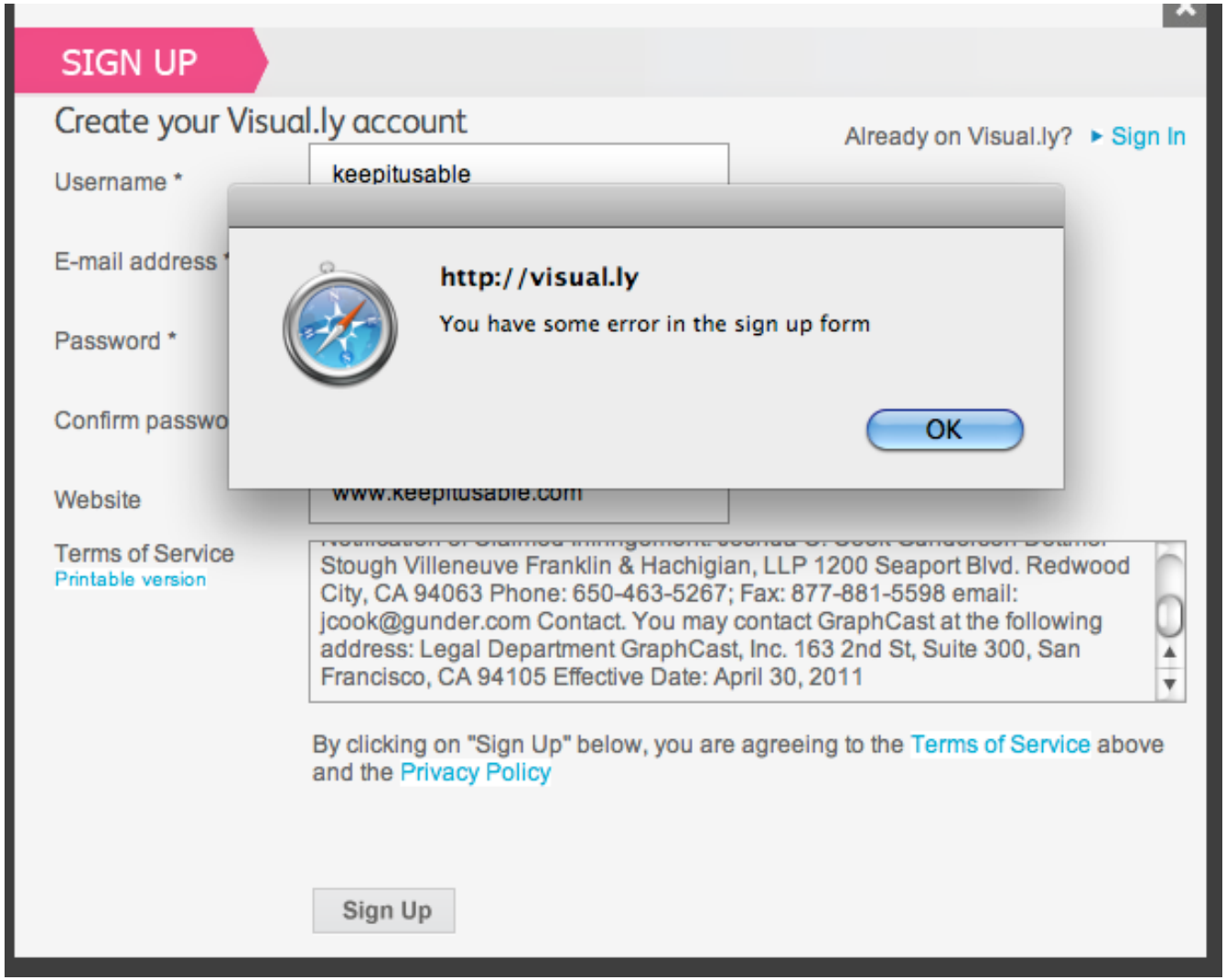
7.4 – Usability Principles

5. **Provide good feedback including effective error messages.**
   - Inform users of the progress of operations and of their location as they navigate.
   - When something goes wrong explain the situation in adequate detail and help the user to resolve the problem.
6. **Ensure that the user can always get out, go back or undo an action.**
   - Ensure that all operations can be undone.
   - Ensure it is easy to navigate back

Bad Error Messages



Bad Error Messages



Bad Error Messages

Bad Error Messages

A Typical User Reaction to a Bad Error Message

Contains coarse language; viewer discretion advised.

A Typical User After Years of Bad Error Messages

7.4 – Usability Principles

7. Ensure that response time is adequate.

- Users are very sensitive to slow response time
  - They compare your system to others.
- Keep response time less than a second for most operations.
- Warn users of longer delays and inform them of progress.

7.4 – Usability Principles

8. **Use understandable encoding techniques.**
   - Choose encoding techniques with care.
   - Use labels to ensure all encoding techniques are fully understood by users.
9. **Ensure that the UI's appearance is uncluttered.**
   - Avoid displaying too much information.
   - Organize the information effectively.

7.4 – Usability Principles

7.4 – Usability Principles

10. Consider the needs of different groups of users.

- Accommodate people from different locales and people with disabilities.
- Ensure that the system is usable by both beginners and experts.

11. Provide all necessary help.

- Organize help well.
- Integrate help with the application (context-sensitive help).
- Ensure that the help is accurate.

7.4 – Usability Principles

Address and geographical location

Address 1

Address 2

ZIP

City (Location)

Province (State)

District

Country

Latitude N 59° 11.153'

59.1987

9.61325

City or Location

Closest city/town or resort/island.

Latitude/Logitude

Possible input format:

- Dezimal: 8.57264
- GPS: E 8° 34.60'

Map

Move the red symbol more or less to the correct spot on the world map.
Drag the map in position to center the red symbol in the window.
Zoom the map. Repeat this, until you can move the symbol precisely to the desired position.

Find Adress

Write the address in the filed and click "Find Address".
If the address is not found or the position is not precise enough, please use the map.

7.4 – Usability Principles

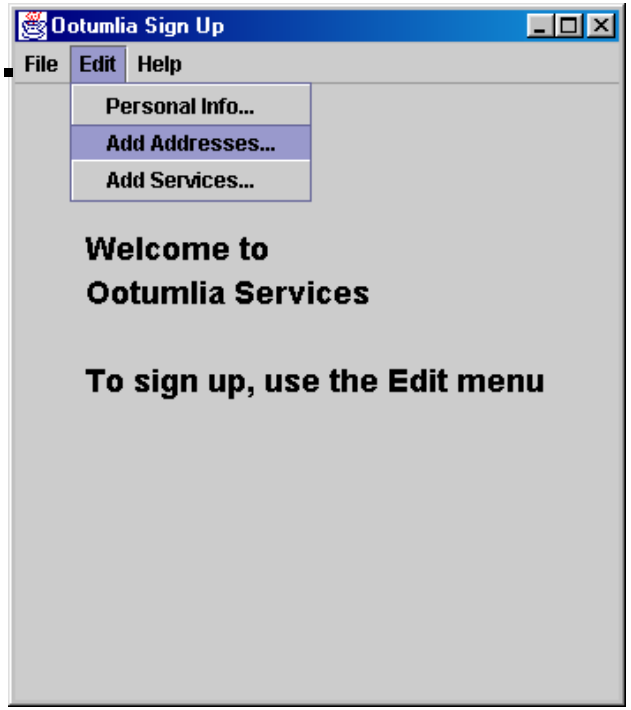
12. Be consistent.

- Use similar layouts and graphic designs throughout your application.
- Follow look-and-feel standards.
- Consider mimicking other applications.

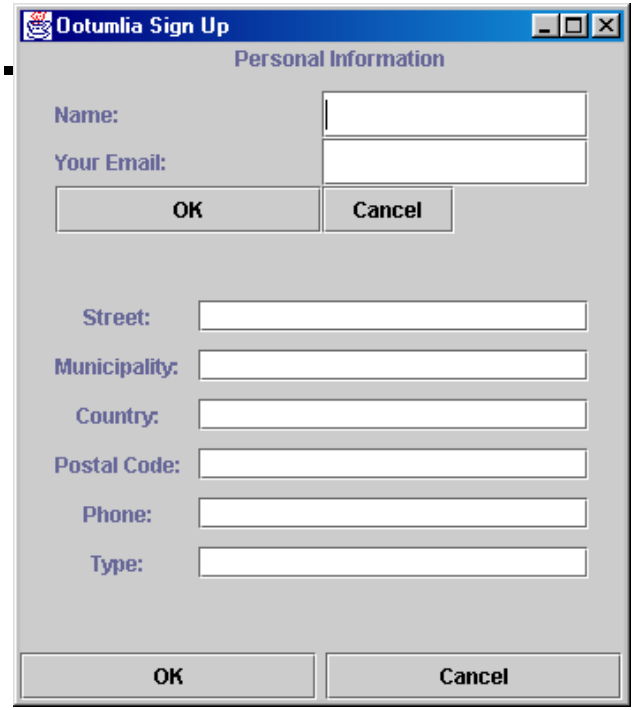
Example (Bad UI)

What are some of the problems with this sign-up wizard for an Internet Service Provider?

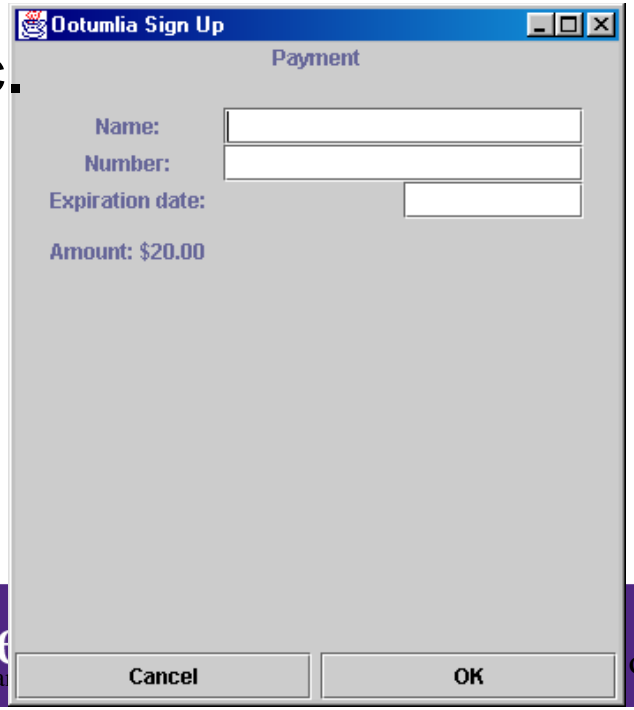
a.



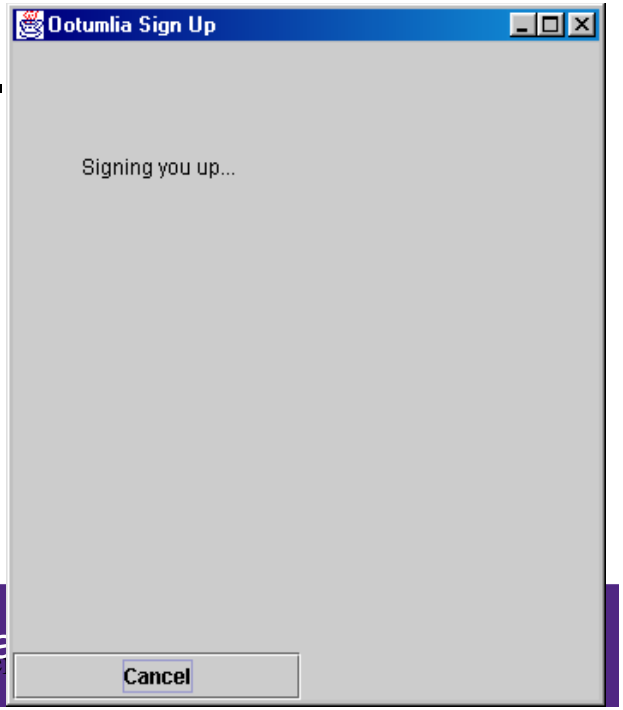
b.



c.



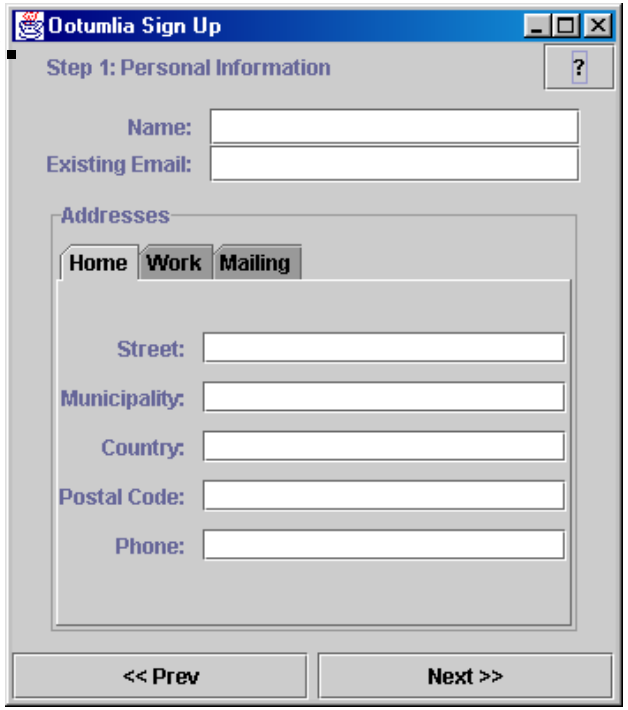
d.



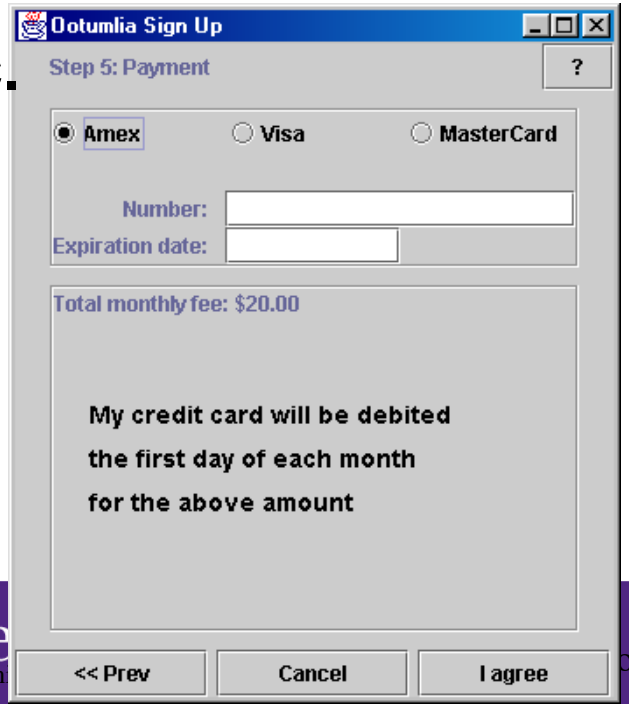
Example (Better UI)

a.

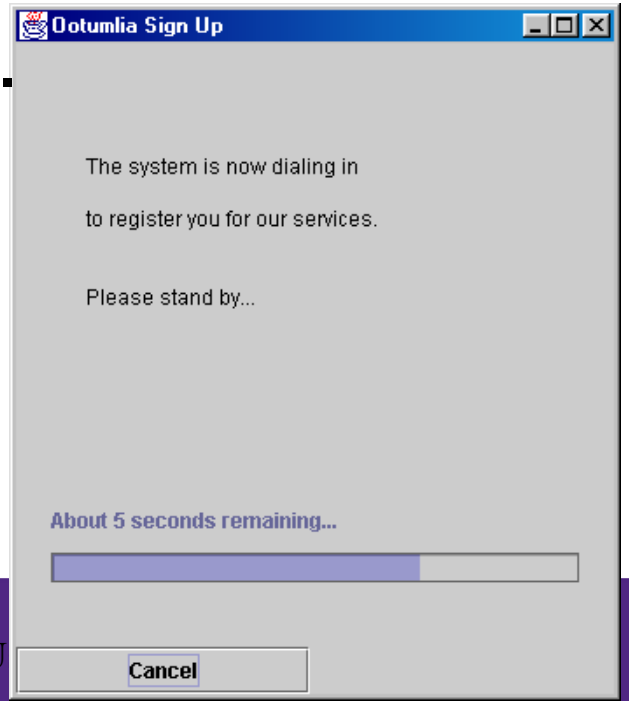
b.



c.



d.



7.5 – Evaluating User Interfaces

Heuristic evaluation

1. Pick some use cases to evaluate.
2. For each window, page or dialog that appears during the execution of the use case
   - Study it in detail to look for possible usability defects
3. When you discover a usability defect, write down the following information:
   - A short description of the defect.
   - Your ideas for how the defect might be fixed.

7.5 – Evaluating User Interfaces

Evaluation by observation of users

- Select users corresponding to each of the most important actors
- Select the most important use cases
- Write sufficient instructions about each of the scenarios
- Arrange evaluation sessions with users
- Explain the purpose of the evaluation
- Preferably videotape each session
- Converse with the users as they are performing the tasks
- When the users finish all the tasks, de-brief them
- Take note of any difficulties experienced by the users
- Formulate recommended changes

7.6 – Implementing a Simple GUI in Java

- See lab 4.
- http://docs.oracle.com/javase/tutorial/uiswing/
- http://docs.oracle.com/javase/tutorial/uiswing/components/componentlist.html
- http://zetcode.com/tutorials/javaswingtutorial/

7.7 – Difficulties / Risks in UI Design

Users differ widely

- Account for differences among users when you design the system.
- Design it for internationalization.
- When you perform usability studies, try the system with many different types of users.

User interface implementation technology changes rapidly

- Stick to simpler UI frameworks widely used by others.
- Avoid fancy and unusual UI designs involving specialized controls that will be hard to change.

7.7 – Difficulties / Risks in UI Design

User interface design and implementation can often take the majority of work in an application:

- Make UI design an integral part of the software engineering process.
- Allocate time for many iterations of prototyping and evaluation.

Developers often underestimate the weaknesses of a GUI

- Ensure all engineers have training in UI development.
- Always test with users.
- Study the UIs of other software.

Textual User Interface (TUI) Design

Provide reasonable defaults:

```
$ ssh-keygen -t dsa
Generating public/private dsa key pair.
Enter file in which to save the key (/home/jeff/.ssh/id_dsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
```

```
Do you wish to continue? [Y/n]:
```

Textual User Interface (TUI) Design

Provide short *and* long options:

```
# useradd -m -s /bin/bash -d /home/joe joe
```

```
# useradd --create-home --shell /bin/bash --home-dir /home/
joe joe
```

Textual User Interface (TUI) Design

Provide *standard* options:

```
-h  --help
-v  --verbose  (-v might be version, but generally means
                'verbose')
    --version
```

Textual User Interface (TUI) Design

Running a command without arguments should display its usage details:

```
$ useradd
Usage: useradd [options] LOGIN
       useradd -D
       useradd -D [options]

Options:
  -b, --base-dir BASE_DIR       base directory for the home
                                directory of the new account
  -c, --comment COMMENT         GECOS field of the new
                                account
  -d, --home-dir HOME_DIR       home directory of the new
                                account
  -D, --defaults                print or change default
                                useradd configuration
```

Textual User Interface (TUI) Design

Break up complex commands into sub-commands. Each sub-command can have its own options and arguments.

```
git init
git add .
git commit
git commit --amend
git mv file1.txt file2.txt
git commit –m “Renames file1”
git push –u origin master
```

Textual User Interface (TUI) Design

Break up complex commands into sub-commands. Each sub-command can have its own options and arguments.

```
$ dpkg -L git
.
.
/usr/lib/git-core/git-push
/usr/lib/git-core/git-commit
/usr/lib/git-core/git-mv
/usr/lib/git-core/git-init
/usr/lib/git-core/git-config
/usr/lib/git-core/git-ls-files
/usr/lib/git-core/git-unpack-file
/usr/lib/git-core/git-checkout
/usr/lib/git-core/git-peek-remote
```

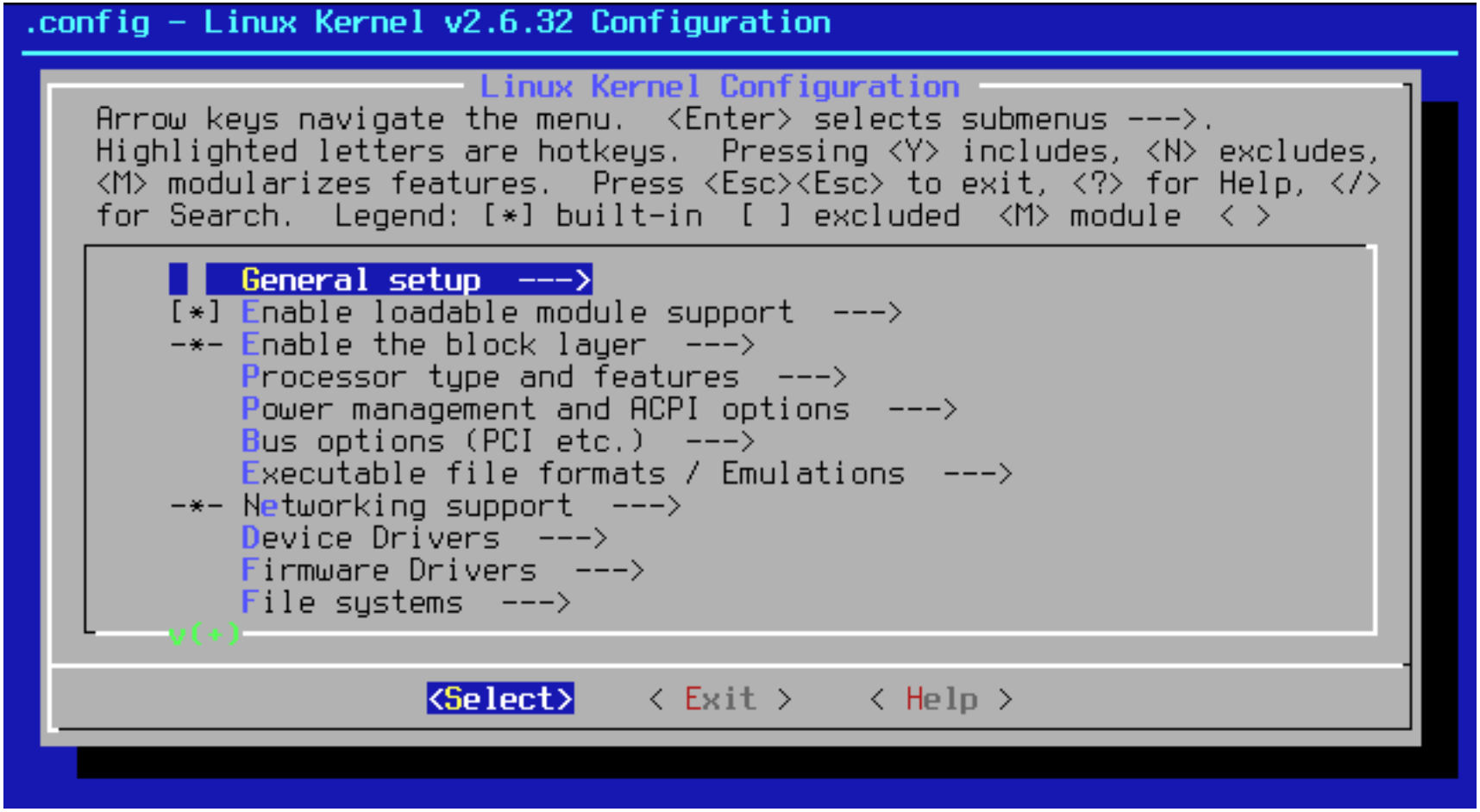
Textual User Interface (TUI) Design

Consider providing a shell for complex applications:

```
$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.

Type 'help;' or '\h' for help. Type '\c' to clear the current
input statement.

mysql> CREATE DATABASE cs2212;
Query OK, 1 row affected (0.04 sec)

mysql> USE cs2212;
Database changed

mysql> CREATE TABLE users ( user_id INTEGER NOT NULL
AUTO_INCREMENT, email VARCHAR(50), PRIMARY KEY(user_id) );
Query OK, 0 rows affected (0.70 sec)
```

Textual User Interface (TUI) Design

Consider providing a menu system for complex applications:



Textual User Interface (TUI) Design

Provide a man page so that users can RTFM!

```
NAME
       useradd - create a new user or update default new user
                 information

SYNOPSIS
       useradd [options] LOGIN

       useradd -D

       useradd -D [options]

DESCRIPTION
       useradd is a low level utility for adding users. On
       Debian, administrators should usually use adduser(8)
       instead.
```

Western

UNIVERSITY · CANADA