Resources and Internationalization

- Your code will run around the world on millions of machines, what do you do?
  - You cannot hardwire literals like “open” (or can you)
  - You should not hardwire text messages
  - What about international character sets/Unicode
  - Locales and Resource bundles can help, see Nutshell Examples, Chapter 14, also look at Formats

- ResourceBundles (in java.util)
  - can provide locale specific constants and objects that are used at class-loading time
  - ListResourceBundle: strings that map to objects
  - PropertyResourceBundle: (file-based) string properties
Resource Bundles

● The class PropertyResourceBundle
  ➤ see Toolbar.properties in ProtoHarp and DrawGui.java
  ➤ use static ResourceBundle.getBundle(filename) to read
  ➤ usually use buttons.properties, menu.properties, ...

● The class ListResourceBundle
  ➤ associate any objects with strings

```java
public class ProgramResource extends ListResourceBundle {
    public Object[][] getContents(){return myContents;}
    static final Object[][] myContents = {
        {"openbutton", new LoadCommand()},
        {"backgroundColor", Color.red},
        {"defaultSize", new int[]{100,200}}
    }
}
```
Resources and Reflection

- Resources, e.g., gifs, audiofiles, and classfiles, are searched for using the CLASSPATH environment variable
  - the program can search for resources this way as well including gif files, text files, class files, ....
  - To open a resource, use the `Class` methods `getResource` or the `getResourceAsStream` which return URL and `InputStream`, respectively --- see `DrawGui`
  - class method belongs to a class, not to an object, part of meta-object idea, also see `java.lang.reflection`

- Reflection allows program control over classes
  - can manipulate all the fields/methods of any class
  - can load a class given the class name, convert from string to class and back again
More Reflection

● To convert a name to a class use the static `Class.forName()` method
  ➤ `Class c = Class.forName("java.awt.Button");`
  ➤ what purpose does this have?
  ➤
  ➤ See also `newInstance()` to create instance of a class

● To manipulate innards of a class, for construction or for use, see `Class` methods, e.g., `getFields()`, `getMethods()`, ...
  ➤ used in ProtoHarp to load all colors into a menu and to select a color --- doesn’t matter if `java.awt.Color` changes

● Useful for loading tools/figures at runtime based on user preferences, for example