Outline

- Introduction
- Examples
- Checklist
- ResourceBundle
- Dealing with Compound Messages
- Formatting
- Working with Text
- Unicode and Chinese Coding

Introduction

- Internationalization (also called i18n):
  - With the addition of localized data, the same executable can run worldwide.
  - Textual elements, such as status messages and the GUI component labels, are not hardcoded in the program. Instead they are stored outside the source code and retrieved dynamically.
  - Support for new languages does not require recompilation.
  - Culturally-dependent data, such as dates and currencies, appear in formats that conform to the end user's region and language.
  - It can be localized quickly.
- Localization (also called l10n)

A Quick Example

- Before Internationalization

```
static public void main(String[] args) {
        System.out.println("Hello.");
        System.out.println("How are you?");
        System.out.println("Goodbye.");
}
```

A Quick Example (cont.)

- After Internationalization

```
static public void main(String[] args) {
        String language;
        String country;
        if (args.length != 2) {
            language = new String("en");
            country = new String("US");
        } else {
            language = new String(args[0]);
            country = new String(args[1]);
        }
        Locale currentLocale;
        ResourceBundle messages;
        currentLocale = new Locale(language, country);
        messages = ResourceBundle.getBundle
                    ("MessagesBundle", currentLocale);
        System.out.println(messages.getString("greetings"));
        System.out.println(messages.getString("inquiry"));
        System.out.println(messages.getString("farewell"));
    }
```

Internationalizing the Sample Program

- Create the Properties Files
  - Default file: “MessagesBundle.properties”

```
greetings = Hello
farewell = Goodbye
inquiry = How are you?
```

  - in file “MessagesBundle_en_US.properties”

```
farewell = Bye Bye
```

  - in file “MessagesBundle_fr_FR.properties”

```
greetings = Bonjour.
farewell = Au revoir.
inquiry = Comment allez-vous?
```

  - May be a class file.

Locale

- Define the Locale

```
frLocale = new Locale("fr","FR");
message = MesourceBundle.getBundle
        ("MessagesBundle",frLocale);
```

Checklist for I18n

- Identify Culturally Dependent Data
    - Messages
    - Labels on GUI components
    - Online help
    - Sounds
    - Colors
    - Graphics
    - Icons
    - Dates
    - Times
    - Numbers
    - Currencies
    - Measurements
    - Phone numbers
    - Honorifics and personal titles
    - Postal addresses
    - Page layouts

Checklist for I18n (cont)

- Isolate Translatable Text in Resource Bundles
- Deal with Compound Messages

```
String diskStatus = "The disk contains " + fileCount.toString() + "
  files.";
```

- Format Numbers and Currencies
- Format Dates and Times
- Use Unicode Character Properties
- Compare Strings Properly
- Convert Non-Unicode Text

ResourceBundle

- getBundle

```
Locale currentLocale = new Locale(”en", ”US", "UNIX");
ResourceBundle introLabels = ResourceBundle.getBundle("ButtonLabel",
    currentLocale);
```

- Search order for getBundle

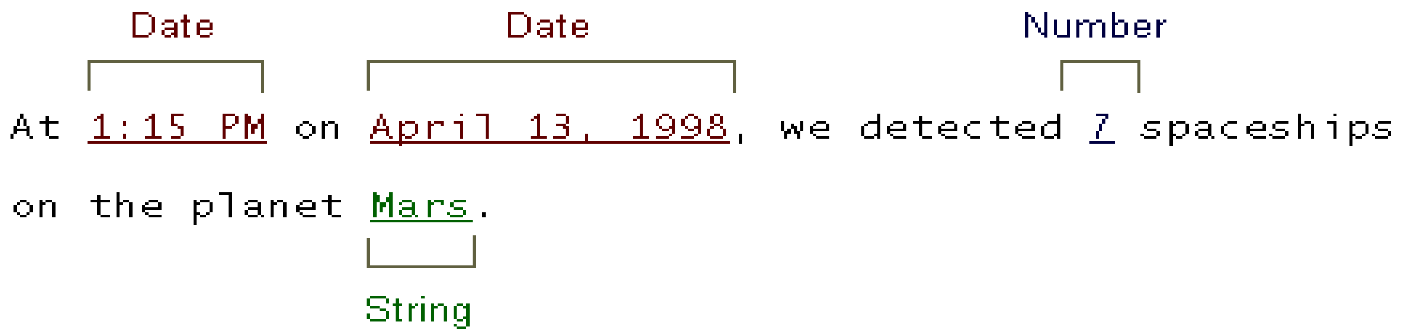
```
ButtonLabel_fr_CA_UNIX
ButtonLabel_fr_CA
ButtonLabel_fr
ButtonLabel_en_US
ButtonLabel_en
ButtonLabel
```

ResourceBundle (cont.)

- Class for ResourceBundle

```
class ButtonLabel_en extends ListResourceBundle {
   public Object[][] getContents() {
       return contents;
   }
   static final Object[][] contents = {
       {"OkKey", "OK"},
       {"CancelKey", "Cancel"},
   };
}
String okLabel = introLabels.getString("OkKey");
```

Dealing with Compound Messages



- Property file ("messages.property")

```
template = At {2,time,short} on {2,date,long}, we detected \
              {1,number,integer} spaceships on the planet {0}.
planet = Mars
```

- Example of using it.

```
Object[] messageArguments = {
   messages.getString("planet"),
   new Integer(7),
   new Date()
};
MessageFormat formatter = new MessageFormat("");
formatter.setLocale(currentLocale);
formatter.applyPattern(messages.getString("template"));
String output = formatter.format(messageArguments);
```

Other formatters

Other predefined formats:

- `ChoiceFormat`: handle plurals.
- `DateFormat`: handle date and time format

Customizing formats:

- Use `SimpleDateFormat` to define user’s own format.

Changing Date Format Symbols

- Change “Mon” to “週一”

Others:

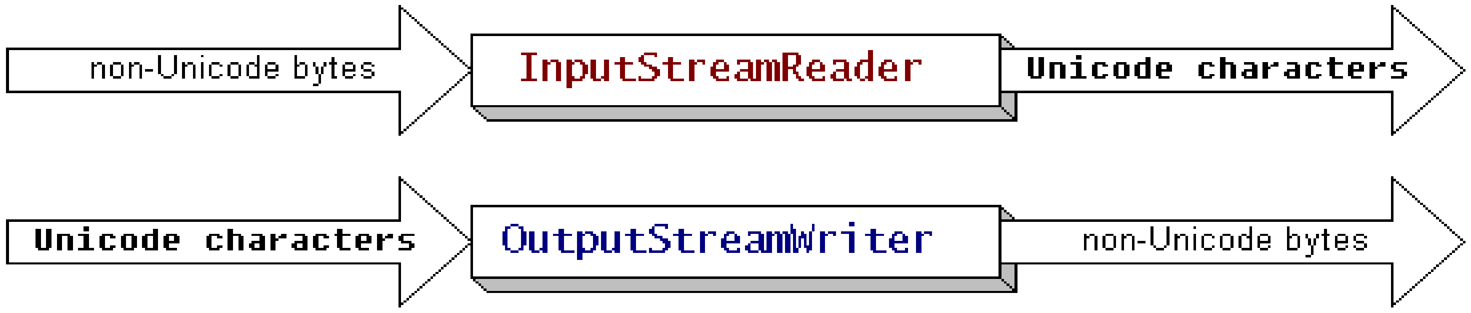
- Numbers (nnn,nnn.nnn)
- Currencies (NT$nnn)
- Percentages (nn%)
- Decimal Formatting (###,###.###)

Working with Text

- Character
    - isDigit
    - isLetter
    - isLetterOrDigit
    - isLowerCase
    - isUpperCase
    - isSpaceChar
    - isDefined
- Detecting Text Boundaries
    - About the BreakIterator Class
    - Character Boundaries
    - Word Boundaries
    - Sentence Boundaries
    - Line Boundaries
- Converting Non-Unicode Text

Converting Non-Unicode Text

- Byte Encodings and Strings
- Character and Byte Streams



Coding for Chinese

- BIG5 (Taiwan, HK)
  - The second character does not use 0x00 to 0x3f. (but, “\” may be used.)
  - 常用字：5401
  - 次常用字：7652
  - BIG5+: add 8124 words (total: 21585)
- GB (China, Singapore)
- UNICODE
  - Combine the common Chinese words for Chinese/Japan/Korean (CJK) coding systems.
  - E.g.
    4E00 - 9FFF CJK UNIFIED IDEOGRAPHS (20,902)

Unicode & UTF8

- **Table 1. UTF-8 encoding**

bytes	**bits**	**representation**
1	7	0bbbbbbb
2	11	110bbbbb 10bbbbbb
3	16	1110bbbb 10bbbbbb 10bbbbbb
4	21	11110bbb 10bbbbbb 10bbbbbb 10bbbbbb