12. Internationalisierung

Internationalisierung von Anwendungen

- Austausch länderspezifischer Merkmale ohne zu programmieren

Länderspezifische Merkmale

- Internationalisierte Versionen von GUIs (Texte etc.)
- Varianten in der Darstellung / Eingabe von Inhalten, z.B.
  - Zeitdarstellungen
  - Darstellung des Dezimalpunktes

Länderspezifische Systemeinstellungen

- Locale: beschreibt die eingestellten Ländereigenschaften

Beschreibung

A Locale object represents a specific geographical, political, or cultural region. An operation that requires a Locale to perform its task is called *locale-sensitive* and uses the Locale to tailor information for the user. For example, displaying a number is a locale-sensitive operation--the number should be formatted according to the customs/conventions of the user's native country, region, or culture.

Konstruktoren:

```
Locale ( String language, String country )
Locale ( String language, String country,
         String variant )
```

Language: ISO Language Code, definiert in der ISO-Norm 639:
http://www.ics.uci.edu/pub/ietf/http/related/iso639.txt

Country: ISO Country Code, definiert in der ISO-Norm 3166:
http://www.chemie.fu-berlin.de/diverse/doc/ISO_3166.html

Variant: Systemabhängig "WIN", "MAC" oder "POSIX"

Statische Informations-Methoden der Local-Klasse:

```
Liste verfügbarer Locales:
static Locale[] getAvailableLocales();

Aktuelles Standard-Locale:
static Locale getDefault();

ISO-Sprachen:
static String[] getISOLanguages();

ISO-Länder:
static String[] getISOCountries();
```

Auszug aus der Locale-Liste (deutsch):

```
de                      Languages: lower case
de_AT                   Countries: upper case
de_AT_EURO
de_CH
de_DE                   Default (DE): "de_DE"
de_DE_EURO
de_LU
de_LU_EURO
el
```

Eigenschaften: Country, Language

```
String getCountry()                    DE
String getLanguage()                   de
String getVariant()                    ""

String getDisplayCountry()             Deutschland
String getDisplayLanguage()            Deutsch
String getDisplayName()      Deutsch (Deutschland)
```

Vorgehensweise

- Alle länderspezifischen Inhalte werden aus dem Code ausgelagert (kein Hardcoding)
- Hierzu werden sogenannte Resources und Properties verwendet
- Resource-Klassen
  - java.util.ResourceBundle
  - Bevorzugte Verwendung: java.util.PropertyResourceBundle (verwendet Resource-Files)

12.3.1 Resource-Files

Aufbau der Resource-Files

```
key1=value1
key2=value2
...
key_n=value_n
```

keys sind beliebig

Benennung der Resource-Files

`MyResources.properties`	Default Resources
`MyResources_fr.properties`	Mit Sprachname
`MyResources_fr_FR.properties`	Mit Sprachname und Ländername

`MyResources` ist hier der **Basis-Filename**.

Lokalisierung der Resource-Files

- Suche erfolgt überall im CLASSPATH

Beispiel

FileManager_de_DE.properties

```
exit_label=Ende
open_label=Öffnen
ok_label=OK
```

Sinnvollerweise nennt man den Resource-File so wie die Anwendung, sofern man einen benutzt.

FileManager.properties

```
exit_label=Exit
open_label=Open
ok_label=OK
```

12.3.2 Class java.util.PropertyResourceBundle

Beispiel

```
RersourceBundle bundle =
  ResourceBundle.getBundle(„FileManager",
                           new Locale("fr", "FR"));
```

```
sucht den CLASSPATH nach einer Datei mit dem Namen:
  FileManager_fr_FR.properties          PRIO 1
  FileManager_fr.properties             PRIO 2
  FileManager.properties                PRIO 3
```

Abfrage der Resource

```
ok_button = new JButton(bundle.getString("ok_label"));
open_button = new JButton(bundle.getString("open_label"));
exit_button = new JButton(bundle.getString("exit_label"));
```