Locales

A **java.util.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 to **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.

Constructors:

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

Some methods:

- static Locale getDefault()
- static Locale[] getAvailableLocales()

Locales (cont.)

Examples:

- **new** `Locale("cs", "CZ")` – Czech, Czech Republic
- **new** `Locale("en", "US")` – English, United States
- **new** `Locale("en", "GB")` – English, United Kingdom
- **new** `Locale("fr", "FR")` – French, France
- **new** `Locale("de", "DE")` – German, Germany

Remark: Locales are often written as `"cs_CZ"`, `"en_US"`, `"en_GB"`, ...

Formatting Numbers

By invoking the methods provided by the **java.text.NumberFormat** we can format numbers, currencies, and percentages according to **Locale**.

Example:

```
NumberFormat f = NumberFormat.getNumberInstance(locale);
String s = f.format(345678.234);
System.out.println(s + " " + locale.toString());
```

We obtain:

```
345 678,234    cs_CZ
345,678.234    en_US
345.678,234    de_DE
```

Similarly we can use methods `getCurrencyInstance()` and `getPercentInstance()` to format currencies and percentages.

Formatting Numbers (cont.)

We can use the **java.util.DecimalFormat** class (it is a subclass of **NumberFormat**) to format decimal numbers.

The class allows to control display of leading and trailing zeros, prefixes and suffixes, grouping (thousands) separators, and the decimal separator. The format specified using pattern:

```
String pattern = ...
DecimalFormat f = new DecimalFormat(pattern);
String s = f.format(12345.6789);
System.out.println(pattern + "   " + s);
```

We obtain (using `"cs_CZ"` locale):

```
###,###.###     12 345,679
###.##          12345,68
000000.000      012345,679
```

It is possible to use the **DecimalFormatSymbols** class to change the symbols that appear in the formatted numbers produced by the `format()` method.