Localization

Where you are matters, because where you are or the region you're from, determines how you'll expect to see dates, currencies, and numbers formatted.

Even ignoring language differences, these formats can be different for different cultures and locations.

For example, Australia displays dates differently, than the United States.

Many countries use different currency symbols, and have different scales for the least significant currency digit.

Locale

Locale is an English word for a place where something happens.

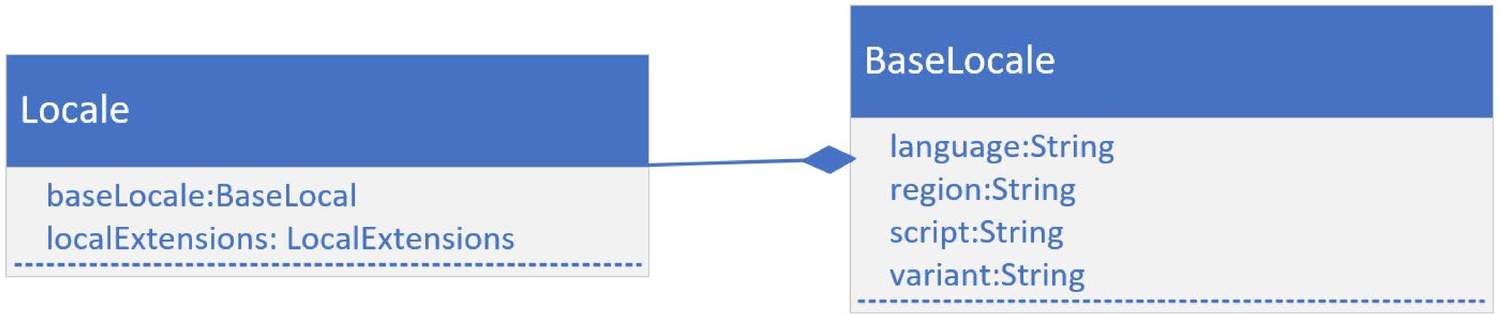
It's also the name of a class in the java.util package that underpins support, for both localization and internationalization.

Java has some built-in support for localization, with methods that let you pass a Locale instance to them.

Locale

A locale has five fields, a language, a country (or region), a variant, and less apparent, are script and extensions.

Underneath the covers, most of the fields are on a BaseLocale class as shown on this slide.



Java's localization support

Java provides localization support for dates, numbers, and currencies with no additional effort, other than defining a Locale and passing it to certain methods.

For language constructs, other than a month or a week day specified in a date, you'll need to do some additional work.

In this video, I'll be covering the built-in support.

Internationalization

Internationalization, or I18n for short, is a method of designing your application to allow language and regional elements, with the help of locales, to be plug and play.

Strings used in messages or user interface elements and images, are stored externally to the application.

These can be retrieved using a locale, and then displayed for a specific user.

Java uses the ResourceBundle class to support this feature.

Locale

On this slide, I'm showing you a useful link, which displays all of the jdk 17 supported locales.

https://www.oracle.com/java/technologies/javase/jdk17-suported-locales.html#compatlocales