**Capítulo TRINTA**

Localização (Localization)

**Objetivos do Exame**

* Ler e definir a localidade usando o objeto Locale.
* Criar e ler um arquivo de propriedades (Properties).
* Construir um pacote de recursos para cada localidade e carregar um pacote de recursos em uma aplicação.

**Localização**

Localização (abreviada como **l10N** por causa do número de caracteres entre a primeira e a última letra) é o mecanismo pelo qual uma aplicação é adaptada para um idioma e região específicos.

Está relacionada ao conceito de **internacionalização** (abreviada como **i18n** pela mesma razão que localização), que trata do design de uma aplicação capaz de lidar com diferentes idiomas e regiões.

As coisas mais comuns que podem ser personalizadas por idioma e/ou região são **mensagens, datas e números**.

Em Java, tudo começa com uma classe: java.util.Locale.

A classe Locale basicamente representa um idioma e um país embora, para ser preciso, uma localidade pode conter as seguintes informações:

* Um código de idioma ISO 639 alpha-2 ou alpha-3, como ja (Japonês)
* Um código de país ISO 3166 alpha-2 ou código de área numérico-3 UN M.49, como JP (Japão)
* Um nome de variante, geralmente vazio, mas pode ser qualquer string
* Um código de script ISO 15924 alpha-4, como Latn (Latim)
* Um conjunto de extensões representadas por caracteres únicos, como u.

Mas na maioria das vezes, apenas trabalhamos com **idiomas e países**.

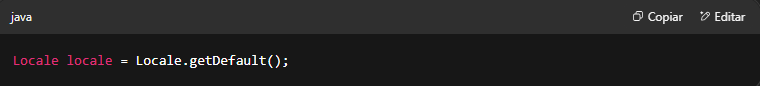
**Uma representação de Locale**

* A parte do idioma é **obrigatória**
* A parte do país é **opcional**

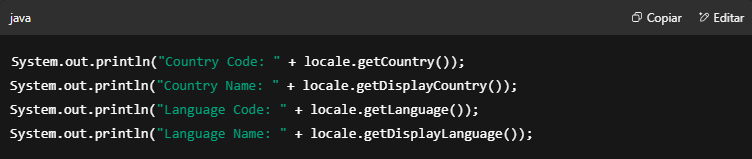
**Exemplo: fr\_CA**

* Observe as **letras minúsculas** na parte do idioma
* Em seguida, o **sublinhado** para separação
* Observe as **letras maiúsculas** na parte do país

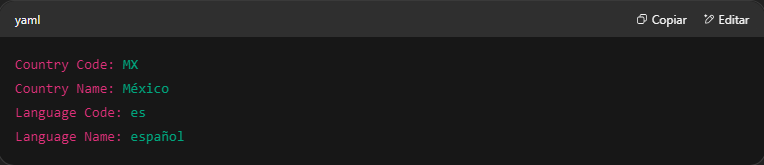
Você pode obter a localidade padrão da sua máquina com:



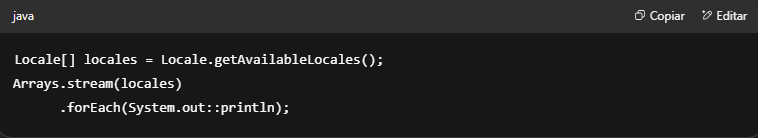
E obter informações como:



Saída (observe como os nomes reais são localizados em espanhol):



Você também pode obter todas as localidades suportadas pelo Java:



Isso imprimirá cerca de 160 localidades no formato idioma[\_país], por exemplo:

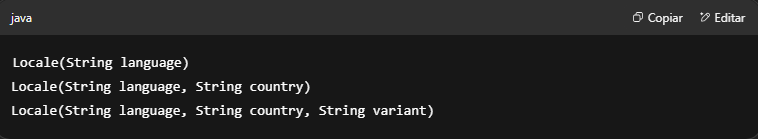


**Definindo a localidade**

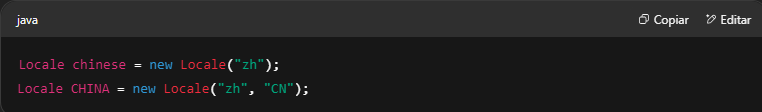
Há três formas diferentes de criar uma instância de Locale:

**1. Usando um construtor**

Existem três construtores:

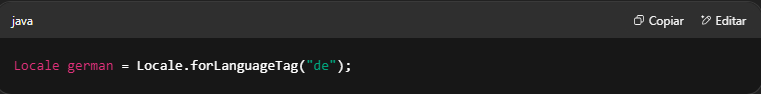


Exemplo:



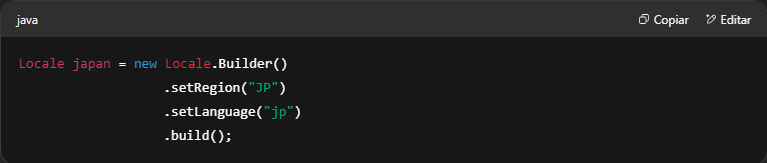
**2. Usando o método de fábrica forLanguageTag(String)**

Este método espera um código de idioma, por exemplo:

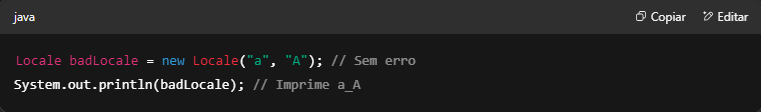


**3. Usando Locale.Builder**

Você pode definir as propriedades necessárias e construir o objeto no final, por exemplo:



Passar um argumento inválido para qualquer um dos três métodos acima **não lançará uma exceção**, apenas criará um objeto com opções inválidas que farão seu programa se comportar incorretamente:

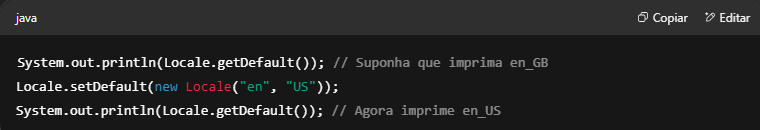


A classe Locale também fornece **constantes predefinidas** para alguns idiomas e países comuns, por exemplo:

* Locale.GERMAN
* Locale.KOREAN
* Locale.UK
* Locale.ITALY

Para o exame, você **não precisa conhecer todas essas constantes**, ou códigos de idioma e país obscuros, apenas que há quatro formas de começar a trabalhar com localidades.

Uma vez que você tem um objeto Locale, pode mudar a localidade do seu programa com o método setDefault(Locale):



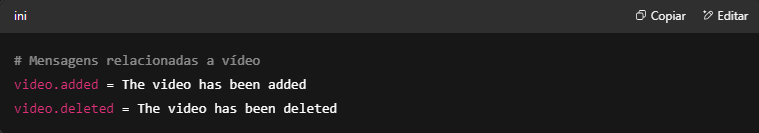
**Arquivos de Propriedades**

Arquivos de propriedades definem strings em pares **chave/valor** separados por linhas.

Há algumas regras, como:

* Espaços no início da linha (se houver) são ignorados.
* Qualquer linha que comece com # ou ! será tratada como **comentário**.
* Você pode quebrar uma linha para fins de legibilidade com uma **barra invertida \**.

Exemplo de arquivo:



Para lê-lo, crie uma instância de java.util.Properties e carregue-a com um java.io.Reader ou java.io.InputStream, por exemplo:



**Pacotes de Recursos (Resource Bundles)**

Para localizar uma aplicação, temos os **Resource Bundles (Pacotes de Recursos)**, que definem um conjunto de chaves com valores localizados. Pacotes de recursos podem ser arquivos de propriedades ou classes.

Para dar suporte a isso, temos uma classe abstrata java.util.ResourceBundle com duas subclasses:

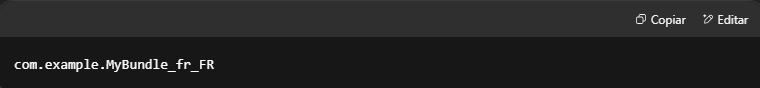
* java.util.PropertyResourceBundle  
  Cada localidade é representada por um arquivo de propriedades. As chaves e valores são do tipo String.
* java.util.ListResourceBundle  
  Cada localidade é representada por uma subclasse dessa classe que sobrescreve o método Object[][] getContents(). O array retornado representa as chaves e valores. As chaves devem ser do tipo String, mas os valores podem ser de qualquer objeto.

Em ambos os métodos, o nome (do arquivo ou da classe) segue uma convenção que permite ao Java procurar pacotes de recursos e associá-los às suas localidades correspondentes.

Essa convenção de nomes é:

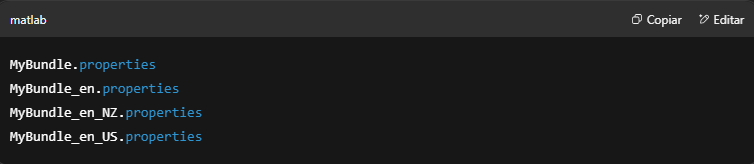


Por exemplo:



Apenas o nome do pacote de recurso é obrigatório (e o nome do pacote se não for o padrão).

Por exemplo, podemos ter pacotes com os seguintes nomes (assumindo que estamos trabalhando com arquivos de propriedades, embora seja o mesmo com classes):



Para determinar qual pacote pertence a uma localidade particular, o Java tenta encontrar o pacote mais específico que corresponde às propriedades da localidade.

Isso significa que:

1. O Java primeiro procura um pacote cujo nome corresponda à localidade completa:  
   package.bundle\_idioma\_pais\_variante
2. Se não encontrar um, ele remove o último componente do nome e repete a busca:  
   package.bundle\_idioma\_pais
3. Se ainda não encontrar, novamente remove o último componente e repete a busca:  
   package.bundle\_idioma
4. Se ainda não encontrar, o último componente é removido novamente, restando apenas o nome do pacote:  
   package.bundle

Se nada for encontrado, uma MissingBundleException é lançada.

Se uma **classe** e um **arquivo de propriedades** compartilharem o mesmo nome, o Java dá **prioridade à classe**.

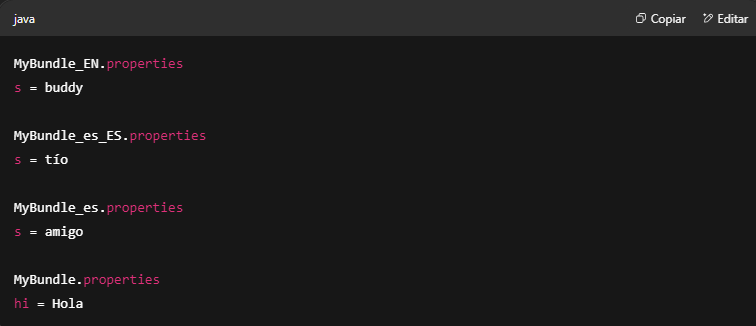
Mas há outro ponto importante.

No seu programa, você pode usar as chaves do pacote de recursos correspondente **e de QUALQUER UM de seus PAIS**.

Os pais de um pacote de recursos são os com o mesmo nome, mas com menos componentes. Por exemplo, os pais de MyBundle\_es\_ES são:

* MyBundle\_es
* MyBundle

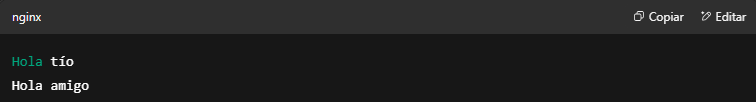
Por exemplo, vamos assumir a localidade padrão en\_US, e que seu programa está usando esses e outros arquivos de propriedades, todos no pacote padrão, com os valores:



Podemos criar um pacote de recursos assim:

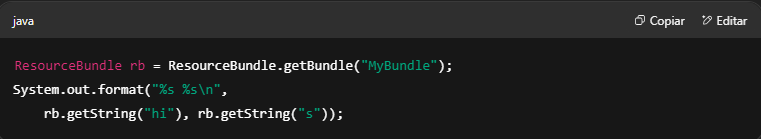


Saída:

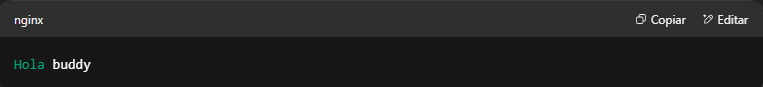


Como você pode ver, cada localidade escolhe valores diferentes para a chave s, mas ambas usam o mesmo valor para hi, já que essa chave está definida em seu pai.

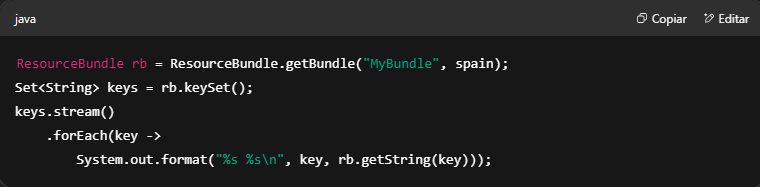
Se você **não especificar** uma localidade, a classe ResourceBundle usará a **localidade padrão do seu sistema**:



Como assumimos que a localidade padrão é en\_US, a saída será:



Também podemos obter todas as chaves de um pacote de recursos com o método keySet():

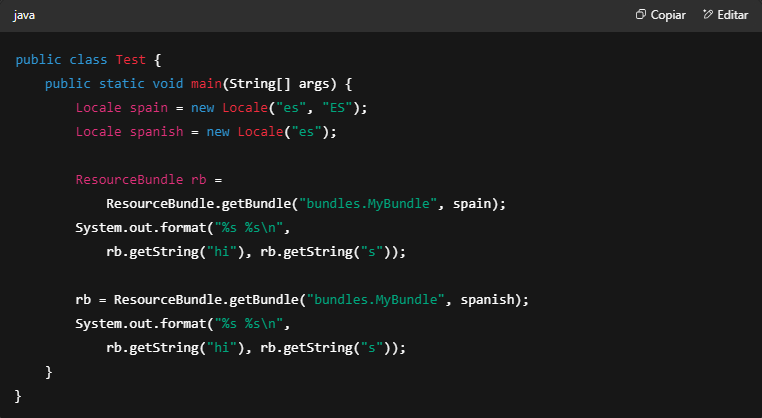


Saída (observe que também imprime a chave do pai):



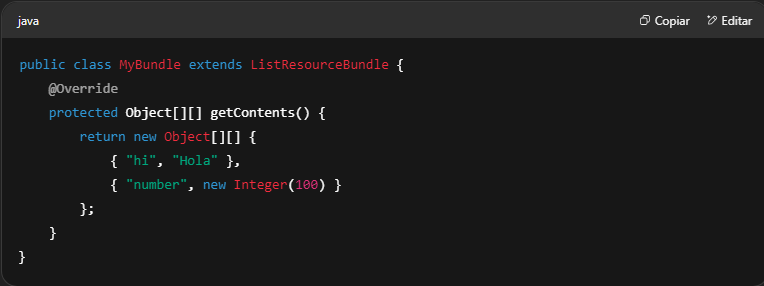
Se ao invés de usar arquivos de propriedades estivermos usando classes, o programa seria assim:



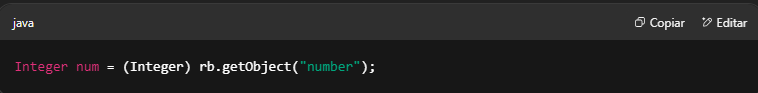


A única coisa que mudou na classe Test foi o nome do pacote (precisamos referenciar o pacote). Isso não deveria surpreender você, afinal, tanto PropertyResourceBundle quanto ListResourceBundle herdam da mesma classe.

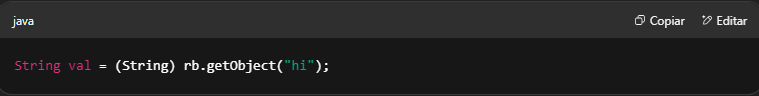
Lembre-se também, ao usar classes podemos ter valores de tipos diferentes de String, por exemplo:



Para obter um valor de objeto, usamos:

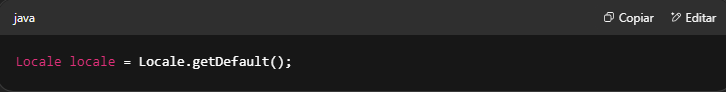


Ao invés de rb.getString(key). De fato, este método é apenas um atalho para:

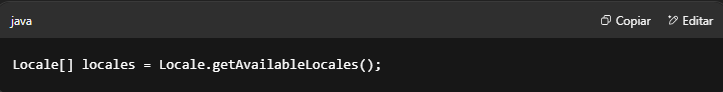


**Pontos-chave**

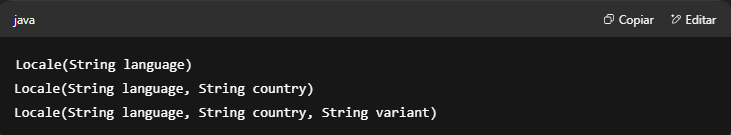
* **Localização** (abreviada como **l10N** por causa do número de caracteres entre a primeira e a última letra) é o mecanismo pelo qual uma aplicação é adaptada a um idioma e região específicos.
* A classe java.util.Locale basicamente representa um idioma e um país, e é o ponto de partida para a localização em Java.
* Você pode obter a localidade padrão da sua máquina com:



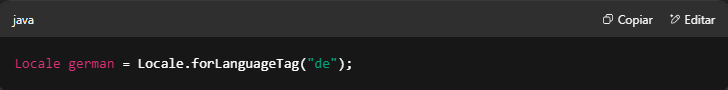
Você também pode obter todas as localidades suportadas pelo Java:



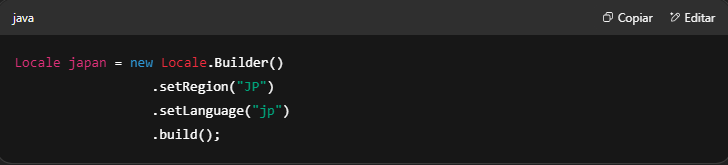
Você pode criar um objeto Locale usando um construtor:



Usando o método de fábrica forLanguageTag(String):



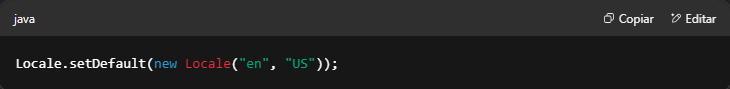
Usando Locale.Builder:



Usando constantes predefinidas para alguns idiomas e países comuns, por exemplo:



Uma vez que você tenha um objeto Locale, pode mudar a localidade do seu programa com o método setDefault(Locale):



 **Arquivos de propriedades** definem strings em pares **chave/valor** separados por linhas.

 Para localizar uma aplicação, temos **Resource Bundles**, que definem um conjunto de chaves com valores localizados. Resource Bundles podem ser **arquivos de propriedades** ou **classes**.

 java.util.PropertyResourceBundle: cada localidade é representada por um arquivo de propriedades. Chaves e valores são do tipo String.

 java.util.ListResourceBundle: cada localidade é representada por uma subclasse que sobrescreve o método Object[][] getContents(). O array retornado representa as chaves e valores. Chaves devem ser do tipo String, mas os valores podem ser qualquer objeto.

 Para determinar qual pacote pertence a uma localidade específica, o Java tenta encontrar o pacote mais específico que corresponda às propriedades da localidade.

 Se não conseguir localizar um, o último componente do nome é removido até restar apenas o nome do pacote.

 Se nada for encontrado, uma MissingBundleException é lançada.

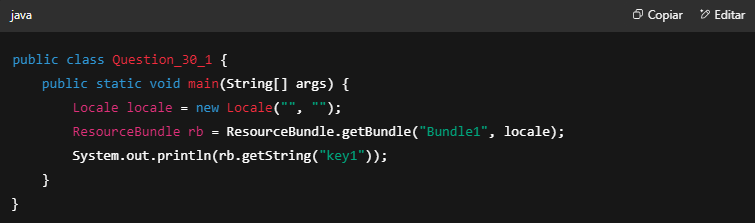
 Se uma classe e um arquivo de propriedades compartilharem o mesmo nome, o Java dá **prioridade à classe**.

 Você pode usar as chaves do pacote de recursos correspondente e **QUALQUER UM de seus PAIS**.

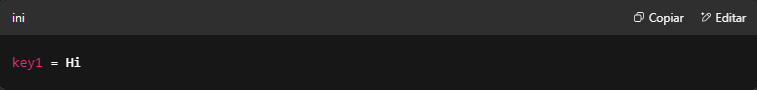
 Os pais de um pacote de recursos são os com o mesmo nome, mas com menos componentes.

**Autoavaliação**

**1. Dado:**



Bundle1.properties



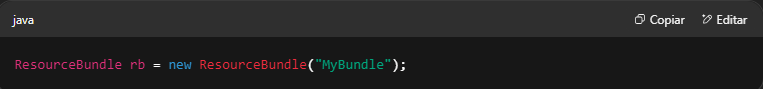
Qual é o resultado?

**A.** Hi  
**B.** null  
**C.** A compilação falha  
**D.** Uma exceção ocorre em tempo de execução

**2. Quais das seguintes são formas válidas de criar uma localidade?**

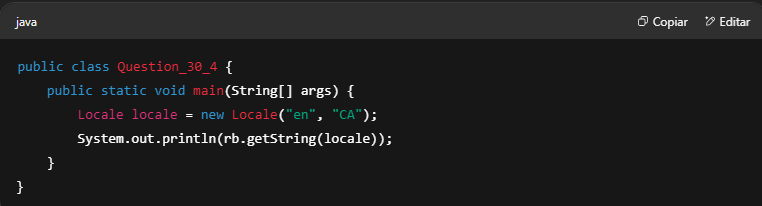
**A.** new Locale();  
**B.** Locale.Builder().setLanguage("de");  
**C.** new Locale.Builder().setRegion("DE").build();  
**D.** Locale.forRegionTag("it");

**3. Assumindo uma localidade padrão de\_DE, qual dos seguintes pacotes de recursos será carregado primeiro com:**



**A.** MyBundle.class  
**B.** MyBundle.properties  
**C.** MyBundle\_de.class  
**D.** MyBundle\_de.properties

**4. Dado:**



Qual é o resultado?

**A.** en  
**B.** en\_CA  
**C.** CA  
**D.** CA\_en

**5. Quais das seguintes são formas válidas de obter um valor dada sua chave de um pacote de recursos do tipo arquivo de propriedades rb?**

**A.** rb.getValue("key");  
**B.** rb.getProperty("key");  
**C.** rb.getObject("key");  
**D.** rb.get("key");