

1. Part 1: Internationalizing a program with PropertyResourceBundles

Consider the following class:

```
public class NotI18N {
    static public void main(String[] args) {
        System.out.println("Hello");
        System.out.println("Thank you");
    }
}
```

2. How many times would you need to edit and recompile this class if you were going to localize it for 70 regions? How would localization scale if you had had hundreds of classes and thousands of English strings to translate?
3. Create a properties file called "Labels.properties" containing entries for each String in NotI18N. Each entry should be on its own line and have the form "key = value".
4. Initialize a ResourceBundle using "Labels" as a base name and using the default locale.
5. Replace the hard-coded strings in NotI18N with calls to your Resource Bundle's `getString(key)` method.
6. Add a second .properties file for the locale of your choice. Your file should be named with the form "Labels_LC.properties" or "Labels_LC_CC.properties", where "LC" and "CC" are the two-letter language and country codes, respectively. A list of language codes is available at: <http://www.w3.org/WAI/ER/IG/ert/iso639.htm>
7. Be sure to translate the values in your new PropertyResourceBundle file to the appropriate language.
8. Add a method with the following signature to NotI18N:

```
static void printMessages(Locale currentLocale);
```

This new method "printMessages" should take a locale as input and print out "Hello" and "Thank You" in that locale's language.

9. Part 2: Using Unicode Characters

Modify the provided file "UnicodeDemo.java" so that instead of printing out "Copyright © 2005", it prints out "Yen ¥, Pound £, Euro €". Your version of Java must support Swing for this part. You will need to use the web to find the right character codes. Google or <http://www.unicode.org/> is a good place to start.