Currency Enhancements in Java 7

The `Currency` class in the `java.util` package exists since JDK 1.4. An instance of the `Currency` class represents a currency.

International Organization for Standardization (ISO) maintains the details of currencies in ISO 4217 standard. ISO 4217 currency codes are used in international businesses. The standard specifies three components of a currency.

- A 3-letter code
- A 3-digit code number
- A number that indicates the relationship between the minor unit of the currency and the currency itself

In the 3-letter currency code, the first two letters are the 2-letter country code specified in ISO 3166. The third letter is usually the first letter of the currency name. For example, the currency code for United States is USD, which contains the first two letters, US, from the country code for United States and the third letter, D, comes from its currency name dollar. Some more examples of currency codes are INR for Indian rupee, THB for Thai baht, JPY for Japanese yen, etc.

Each currency is also assigned a 3-digit code number. Usually, the code number for a currency is the same as the 3-digit country code number assigned to the country in ISO 3166. For example, the currency code number 840 is for USD, 356 for INR, 763 for THB, 392 for JPY, etc.

Usually, a country has a major currency unit and a minor currency unit. For example, in United States, dollar is the major currency unit and cent is the minor currency unit. ISO 4217 specifies the relationship between the major and minor currency units as an exponent of 10. For example, 100 US cents is 1 USD. In the exponent form, 100 can be expressed as 10^2. Therefore, the numeric value that expresses the relationship between USD and its minor current unit (cent) is 2. Some currencies do not have a minor currency. Some currency has so little value that its minor currency is not generally used, for example, Japanese yen has so little value that its minor currency Japanese *sen* (100 sen = 1 yen) is not used. In such cases, zero is used as the relationship between the major and minor currency units.

The `Currency` class does not provide a public constructor to make sure that the maximum of only one `Currency` instance is created for any currency. You can get an instance of the `Currency` class using one of the following three `static` methods. The `getAvailableCurrencies()` method was added in Java 7. It returns a `Set` of all available currencies.

```
public static Currency getInstance(Locale locale)
public static Currency getInstance(String currencyCode)
public static Set<Currency> getAvailableCurrencies()
```

For example, you can get the currency for United States, India, Thailand, and Japan using the following snippet of code.

```
Currency usCurrency = Currency.getInstance(Locale.US);

Locale indianLocale = new Locale("hin", "IN");
Currency indianCurrencycurr = Currency.getInstance(indianLocale);

Locale thaiLocale = new Locale("th", "TH");
Currency thaiCurrency = Currency.getInstance(thaiLocale);

Currency japaneseCurrency = Currency.getInstance(Locale.JAPAN);
```

The following three methods of the `Currency` class return the currency code, currency code number and the relationship between the major and minor units of the currency. The `getNumericCode()` method was added in Java 7.

```
public String getCurrencyCode()
public int getNumericCode()
public int getDefaultFractionDigits()
```

Java 7 added a `getDisplayName()` method to the `Currency` class to get the currency name suitable for display in the default locale and the specified locale. Two versions of the `getDisplayName()` method are as follows.

```
public String getDisplayName()
public String getDisplayName(Locale locale)
```

ISO 4217 currency codes are maintained by ISO independent of the Java releases. Starting from Java 7, you can work with new currencies without getting a newer version of Java. You can add a new currency details or override the details of an existing currency using a property file named `currency.properties`. The file should be placed in `JAVA_HOME\lib` directory, where `JAVA_HOME` is the Java runtime installation directory. This file consists of key/value pairs. Key and value are separated by an equal to (=) sign. A line in the file that begins with a hash sign (#) is treated as a comment line. The following is sample contents of the `currency.properties` file.

```
# Override the currency for US from USD to USE
US=USE,840,2

# Add a new currency ZZD for a country code ZZ
ZZ=ZZD,999,2
```

Listing 1 has the program to print the currency details for United States locale and a currency code ZZD. When you run this program without the `currency.properties` file with the above-mentioned contents, it will print the details for USD currency and throw an exception for ZZD. When

you run this program with `currency.properties` file, it will print the details for USE as well as ZZD.

Listing 1: Overriding the default current data using currency.properties file

```
// OverrideDefaultCurrency.java
package com.jdojo.chapter13;

import java.util.Currency;
import java.util.Locale;

public class OverrideDefaultCurrency {
     public static void main(String[] args) {
          Currency curr = Currency.getInstance(Locale.US);
          print(curr);

          curr = Currency.getInstance("ZZD");
          print(curr);
     }

     public static void print(Currency curr) {
          String code = curr.getCurrencyCode();
          String displayName = curr.getDisplayName();
          String symbol = curr.getSymbol();
          int numericCode = curr.getNumericCode();
          int fraction = curr.getDefaultFractionDigits();

          System.out.print("Code:" + code);
          System.out.print(", Numeric Code:" + numericCode);
          System.out.println(", Default fraction digits:" + fraction);
     }
}
```