Internationalization:
——————————————————————

Designing Java applications with respect to the local conventions is called Internationalization, in short I18N.

To provide INternationalization in Java applications, First we have to represent a group of local users in the program, for this we have to divide all the users into the groups.

To divide all the users into the number of groups we have to use the following parameters.

1. Language :
   It will be represented in the form of two lower case letters.
   EX: en, hi, fr, it....

2. Country :
   It will be represented in the form of two uppercase letters.
   EX: US, IN, IT, FR,....

3. System Variant[OS]:
   It will be represented in the form of three lower case letters.
   EX: win, lin, uni

To represent all the local users in java applications we have to use a predefined class in the form of “java.util.Locale”.

To create Objects for the Locale class we have to use the following constructors.

1. public Locale()
2. public Locale(String lang)
3. public Locale(String lang, String country)
4. public Locale(String lang, String country, String sysVariant)

```
EX:   Locale l1 = new Locale();
      Locale l2 = new Locale(“en”);
      Locale l3 = new Locale(“en”, “US”);
      Locale l4 = new Locale(“en”, “US”, “win”);
```

After representing all the local users we have to provide the following I18N Services.

1. Number Representations
2. Dates Representations
3. Messages Representations

Number Representations:

If we want to represent a number as per a particular Locale then we have to use a predefined class in the form of “java.text.NumberFormat”.

Steps:

1. Create Locale object:
   Locale locale = new Locale(“en”, “US”);

2. Create NumberFormat class object:
   NumberFormat nf = NumberFormat.getInstance(locale);

3. Represent a Number as per the Locale :
   System.out.println(nf.format(123456789.34567878);

EX:
```
import java.text.NumberFormat;
import java.util.Locale;

public class Main {
   public static void main(String[] args) {
        Locale locale = new Locale("en", "US");
        NumberFormat numberFormat = NumberFormat.getInstance(locale);
        System.out.println(numberFormat.format(123456789.23456787));

   }
}
```
```
123,456,789.235
```

EX:
```
import java.text.NumberFormat;
import java.util.Locale;

public class Main {
   public static void main(String[] args) {
        Locale locale = new Locale("it", "IT");
        NumberFormat numberFormat = NumberFormat.getInstance(locale);
        System.out.println(numberFormat.format(123456789.23456787));

    }
}
```

```
123.456.789,235
```

Date Representations:

To represent date with respect to a particular Local we have to use a predefined class in the form of “java.text.DateFormat” class.

Steps:

1. Create Locale class object:
   Locale locale = new Locale(“en”, “US”);

2. Create DateFormat class object:
   DateFormat df = DateFormat.getDateInstance(0, locale);
   Where DateStyles are 0,1,2,3

3. Represent date value as per the Locale:
   System.out.println(df.format(new Date());

EX:

```
import java.text.DateFormat;
import java.text.NumberFormat;
import java.util.Date;
import java.util.Locale;

public class Main {
   public static void main(String[] args) {
        Locale locale = new Locale("it", "IT");
        DateFormat dateFormat = DateFormat.getDateInstance(3,
locale);
        System.out.println(dateFormat.format(new Date()));
   }
}
```

Message Representations:

To represent messages with respect to a particular locale we have to use a predefined class in the form of “java.util.ResourceBundle”.

Steps:

1. Prepare properties files with the messages in the form of Key-Value pairs.
   IN general, the properties file name should be in the following format.

   BaseName_lang_country_sysVar.properties

   EX:
   abc_en_US.properties
   welcome = Welcome To en US Users.

   abc_it_IT.properties
   welcome = Welcome To it IT Users.

2. In Java class, Create Locale Object:
   Locale locale = new Locale(“en”, “US”);

3. Create ResourceBundle object:
   ResourceBundle rb = ResourceBundle.getBundle(“abc”, locale);

   It will search for the properties file on the basis of the provided properties file’s base name and the locale. If the properties file is identified then It will get the messages from the properties file.

4. Get the message from ResourceBundle object:
   System.out.println(rb.getString(“welcome”));

EX:
```
import java.text.DateFormat;
import java.text.NumberFormat;
import java.util.Date;
import java.util.Locale;
import java.util.ResourceBundle;

public class Main {
   public static void main(String[] args) {
       Locale locale = new Locale("en", "US");
       ResourceBundle resourceBundle =
ResourceBundle.getBundle("abc", locale);
       System.out.println(resourceBundle.getString("welcome"));
   }
}
```

abc_en_US.properties

```
welcome = Welcome to en US Users.
```

abc_it_IT.properties

```
welcome = Welcome to it IT Users.
```