Частное учреждение образования
Колледж бизнеса и права

УТВЕРЖДАЮ
Заведующий
методическим кабинетом
_____Е.В. Паскал
«___»_____2021

Специальность: 2-40 01 01 «Программное обеспечение информационных технологий»	Дисциплина: «Основы кроссплатформенного программирования»

ЛАБОРАТОРНАЯ РАБОТА № 13
Инструкционно-технологическая карта

Тема: «Локализация Java программ»

Цель: Научить локализировать программы

Время выполнения: 2 часа

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Изучить теоретические сведения;
2. Ответить на контрольные вопросы;
3. Откомпилировать примеры программ из раздела «Теоретические сведения»;
4. Выполнить ИДЗ.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ ДЛЯ ВЫПОЛНЕНИЯ РАБОТЫ

Локализация – это адаптация приложения к конкретному языку и региону путем перевода выводимых пользователю текстовых элементов и документации, а также определения данных времени, валют и др., согласно специфике данного региона.

Интернационализация – это процесс разработки приложения такой структуры, при которой дополнение нового языка не требует перестройки и перекомпиляции (сборки) всего приложения. Достигается это за счет отдельного хранения данных интернационализации в виде файлов свойств, загружаемых приложением динамически в процессе работы.

Для разделения ресурсов, специфичных для каждого языка, страны или региона, используется класс java.util.ResourceBundle или один из его потомков java.util.ListResourceBundle, java.util.PropertyResourceBundle.

Концептуально каждый java.util.ResourceBundle — это набор подклассов, которые используют одно и то же базовое имя. Например, пусть ButtonLabel — базовое имя. Символы, которые следуют за базовым именем указывают код языка, код страны и вариант Locale. Например, ButtonLabel_en_GB указывает на английский язык (en) и Великобританию (GB).

ButtonLabel
ButtonLabel_de
ButtonLabel_en_GB
ButtonLabel_fr_CA_UNIX

Чтобы выбрать подходящий ResourceBundle, вызовите метод ResourceBundle.getBundle. Следующий пример выбирает ButtonLabel ResourceBundle для Locale с французским языком, страной Канада и платформой UNIX:

Java

```
Locale currentLocale = new Locale("fr", "CA", "UNIX");
ResourceBundle introLabels = ResourceBundle.getBundle("ButtonLabel", currentLocale);
```

Если класса ResourceBundle, подходящего для указанной Locale нет, то getBundle пытается найти наиболее близкое совпадение. Например, если ищется ButtonLabel_fr_CA_UNIX, а Locale по умолчанию en_US, то getBundle ищет классы в следующем порядке:

ButtonLabel_fr_CA_UNIX
ButtonLabel_fr_CA
ButtonLabel_fr
ButtonLabel_en_US
ButtonLabel_en
ButtonLabel

Заметьте, что метод getBundle ищет классы, основанные на Locale по умолчанию перед поиском базового класса. Если getBundle не может найти совпадения в этом списке классов, то он бросает исключение java.util.MissingResourceException. Всегда создавайте базовый класс, чтобы избежать подобного исключения.

Абстрактный класс java.util.ResourceBundle имеет два дочерних класса: java.util.PropertyResourceBundle и java.util.ListResourceBundle.

Класс PropertyResourceBundle использует файлы настроек (properties) для хранения текста. Эти файлы не являются частью кода Java и они могут содержать только объекты String.

Класс ListResourceBundle использует список локализованных ресурсов, хранящихся в классах Java.

Класс java.util.ResourceBundle очень гибкий. Если вы сначала использовали PropertyResourceBundle, чтобы хранить локализованные строки в файлах properties, а позже решили использовать ListResourceBundle, то это не отразится на коде. Например, следующий getBundle получает

ResourceBundle для Locale независимо от способа хранения этого ResourceBundle.

ResourceBundle introLabels = ResourceBundle.getBundle("ButtonLabel", currentLocale);

Метод getBundle сначала ищет класс с указанным базовым именем, а затем, если его не находит, ищет файл properties.

Класс ResourceBundle содержит массив пар ключ-значение. Вы указываете ключ, который должен быть String, когда вам нужно достать значение из ResourceBundle. Пример:

Java

```
class ButtonLabel_en extends ListResourceBundle {
  // English version
  public Object[][] getContents() {
    return contents;
  }
  static final Object[][] contents = {
    {"OkKey", "OK"},
    {"CancelKey", "Cancel"},
  };
}
```

Получение значения:

Java

```
String okLabel = ButtonLabel.getString("OkKey");
```

Класс Locale

Класс Java java.util.Locale позволяет учесть особенности региональных представлений алфавита, символов, чисел и дат. Автоматически виртуальная машина использует текущие региональные установки операционной системы, но при необходимости их можно изменять.

Для некоторых стран региональные параметры устанавливаются с помощью констант, например: Locale.US, Locale.FRANCE.

Для всех остальных объект класса Locale нужно создавать с помощью конструктора, например: Locale rus = new Locale("ru", "RU");

Существует константа Locale.ROOT. Она представляет локаль, для которой язык, страна равны пустой строке(""). Эта локаль является базовой для всех остальных локалей. Используется для написания приложений, не зависящих от локали.

Определить текущий вариант региональных параметров можно следующим образом: Locale current = Locale.getDefault();

А можно и изменить для текущего экземпляра (instance) JVM: Locale.setDefault(Locale.CANADA)

Рассмотрим пример использования класса Locale:

```
import java.util.Locale;

public class LocaleDemo1 {
  public static void main(String[] args) {
    Locale fr = Locale.FRANCE;
    Locale us = Locale.US;
```

```
    Locale uk = new Locale("uk", "UA");
    Locale.setDefault(Locale.CANADA);
    Locale current = Locale.getDefault();
    getLocaleInfo(current);

    getLocaleInfo(fr);
    getLocaleInfo(us);
    getLocaleInfo(uk);
  }
  private static void getLocaleInfo(Locale current) {
    System.out.println("Код региона: " + current.getCountry());
    System.out.println("Название региона: " + current.getDisplayCountry());
    System.out.println("Код языка региона: " + current.getLanguage());
    System.out.println("Название языка региона: "
       + current.getDisplayLanguage());
    System.out.println();
  }
}
```

Класс ResourceBundle вместе с классом Locale лежит в основе процесса интернационализации в языке Java.

Класс ResourceBundle предназначен для чтения данных из текстовых файлов свойств (расширение - properties). Каждый отдельный файл с расширением properties содержит информацию для отдельной локали.

В файлах свойств (*.properties) информация должна быть организована по принципу:

#Комментарий
group1.key1 = value1
group1.key2 = value2
group2.key1 = value3…
Например:
label.button = submit
label.field = login
message.welcome = Welcome!
или
label.button = принять
label.field = логин
message.welcome = Добро пожаловать!

Рассмотрим правила выбора имени для properties файлов. Выбираем базовое имя для группы properties файлов. Например text. Добавляем к базовому имени через пробел код языка (uk) и код страны (UA).

Следующий список показывает возможный набор соответствующих ресурсов с базовым именем:

text.properties
text_ru_RU.properties
text_uk_UA.properties
text_fr_CA.properties

Чтобы выбрать определенный объект ResourceBundle, следует вызвать один из статических перегруженных методов getBundle(параметры).

Следующий фрагмент выбирает text объекта ResourceBundle для объекта Locale, который соответствует французскому языку и стране Канада:

Locale locale = new Locale("fr", "CA");

ResourceBundle rb = ResourceBundle.getBundle("text", locale);

Если объект ResourceBundle для заданного объекта Locale не существует, то метод getBundle() извлечет наиболее общий.

Если общее определение файла ресурсов не задано, то метод getBundle() генерирует исключительную ситуацию MissingResourceException. Чтобы это не произошло, необходимо обеспечить наличие базового файла ресурсов без суффиксов, а именно: text.properties в дополнение к частным случаям вида:

text_en_US.properties

text_ru_RU.properties.

Пусть базовое имя ресурса baseName, и ресурс нам нужен для fr, CA. Locale по умолчанию определена через uk, UA. Мы вызываем ResourceBundle.getBundle(baseName, canadaLocale). В каком порядке будет организован поиск в properties файлах? Формируется следующий список строк (так называемые кандидаты в имена пакетов):

baseName + "_"+ fr + "_" + CA

baseName + "_"+ fr

baseName + "_"+ uk + "_" + UA

baseName + "_"+ uk

baseName

Рассмотрим пример использования класса ResourceBundle:

```
import java.io.UnsupportedEncodingException;
import java.util.Locale;
import java.util.ResourceBundle;

public class ResourceBundleDemo1 {
   public static void main(String[] args) throws UnsupportedEncodingException {
      printInfo("", "");
      printInfo("en", "US");
      printInfo("uk", "UA");
   }
   private static void printInfo(String language, String country)
         throws UnsupportedEncodingException {
      Locale current = new Locale(language, country);
      ResourceBundle rb = ResourceBundle.getBundle("text", current);
      String s1 = rb.getString("str1");
      System.out.println(s1);
      String s2 = rb.getString("str2");
      System.out.println(s2);
      System.out.println();
   }
}
```

В классе ResourceBundle определен ряд полезных методов:

getKeys() - возвращает объект Enumeration, который применяется для последовательного обращения к элементам.

keySet() – возвращает множество Set всех ключей.

getString(String key) - извлекается конкретное значение по конкретному ключу.

boolean containsKey(String key) - проверить наличие ключа в файле.

В следующем примере используется метод keySet() класса ResourceBundle:

```
public class ResourceBundleDemo2 {
   public static void main(String[] args) throws UnsupportedEncodingException {
      printInfo("", "");
      printInfo("en", "US");
      printInfo("uk", "UA");
   }
   private static void printInfo(String language, String country)
         throws UnsupportedEncodingException {
      Locale current = new Locale(language, country);
      ResourceBundle rb = ResourceBundle.getBundle("text", current);
      for (String key : rb.keySet()) {
         String value = rb.getString(key);
         System.out.println(value);
      }
      System.out.println();
   }
}
```

Класс NumberFormat языка Java используется для форматирования чисел.

Чтобы получить объект класса для форматирования в национальном стандарте по умолчанию, используются следующие методы:

NumberFormat.getlnstance()

NumberFormat.getNumberlnstance() - идентичен getInstance()

NumberFormat.getCurrencylnstance()

NumberFormat.getPercentlnstance()

Чтобы получить объект класса для форматирования в других национальных стандартах используются следующие методы:

NumberFormat.getlnstance(Locale locale)

NumberFormat.getNumberlnstance(Locale locale) - идентичен getInstance(Locale locale)

NumberFormat.getCurrencylnstance(Locale locale)

NumberFormat.getPercentlnstance(Locale locale)

Рассмотрим пример использования класса NumberFormat:

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

public class NumberFormatDemo1 {
   public static void main(String[] args) {
```

```
        double number = 123.4567;
        Locale locFR = new Locale("fr");
        NumberFormat numberFormat1 = NumberFormat.getInstance();
        System.out.println("Число в текущей локали: " +
numberFormat1.format(number));
        NumberFormat numberFormat2 = NumberFormat.getInstance(locFR);
        System.out.println("Число во французской локали: " +
numberFormat2.format(number));
        NumberFormat numberFormat3 = NumberFormat.getCurrencyInstance();
        System.out.println("Денежная единица в текущей локали: " +
numberFormat3.format(number));
        NumberFormat numberFormat4 = NumberFormat.getCurrencyInstance(locFR);
        System.out.println("Денежная единица во французской локали: " +
numberFormat4.format(number));
        NumberFormat numberFormat5 = NumberFormat.getPercentInstance();
        System.out.println("Процент в текущей локали: " +
numberFormat5.format(number));
        NumberFormat numberFormat6 = NumberFormat.getPercentInstance(locFR);
        System.out.println("Процент во французской локали: " +
numberFormat6.format(number));
      }
    }
```

Другие полезные методы NumberFormat класса:

setMaximumFractionDigits(int digits) - устанавливает максимальное количество цифр после десятичной точки в форматируемом объекте. Последняя отображаемая цифра округляется.

setMaximumlntegerDigits(int digits) - устанавливает максимальное количество цифр перед десятичной точкой в форматируемом объекте. Используйте этот метод с предельной осторожностью. Если вы зададите слишком мало цифр, число будет просто усечено, и результат станет совершенно неправильным!

setMinimumFractionDigits(int digits) - устанавливает минимальное количество цифр после десятичной точки в форматируемом объекте. Если количество цифр в дробной части числа меньше минимального, то на экран выводятся замыкающие нули.

setMinimumlntegerDigits(int digits) - устанавливает минимальное количество цифр перед десятичной точкой в форматируемом объекте. Если количество цифр в целой части числа меньше минимального, то на экран выводятся ведущие нули.

getMaximumFractionDigits() - возвращает максимальное количество цифр после десятичной точки в форматируемом объекте.

getMinimumFractionDigits() - возвращает минимальное количество цифр после десятичной точки в форматируемом объекте.

Рассмотрим пример использования методов setMaximumFractionDigits(int digits), getMaximumFractionDigits(), setMaximumlntegerDigits(int digits):

```
import java.text.NumberFormat;
```

```
public class NumberFormatDemo2 {
   public static void main(String[] args) {
      double d = 123.45678;
      NumberFormat nf = NumberFormat.getInstance();
      System.out.print("Максимальное количество знаков в дробной части "
            + nf.getMaximumFractionDigits() + ": ");
      System.out.println(nf.format(d));
      nf.setMaximumFractionDigits(7);
      System.out.print("Максимальное количество знаков в дробной части 7: ");
      System.out.println(nf.format(d));
      nf.setMaximumIntegerDigits(2);
      System.out.print("Максимальное количество знаков в целой части 2: ");
      System.out.println(nf.format(d));
   }
}
```

Следующий пример демонстрирует использование методов getMinimumFractionDigits(), setMinimumlntegerDigits(int digits), setMinimumFractionDigits(int digits) класса NumberFormat:

```
import java.text.NumberFormat;

public class NumberFormatDemo3 {
   public static void main(String[] args) {
      double d = 123.45678;
      NumberFormat nf = NumberFormat.getInstance();
      System.out.print("Минимальное количество знаков в дробной части "
            + nf.getMinimumFractionDigits() + ": ");
      System.out.println(nf.format(d));
      nf.setMinimumFractionDigits(7);
      System.out.print("Минимальное количество знаков в дробной части 7: ");
      System.out.println(nf.format(d));
      nf.setMinimumIntegerDigits(5);
      System.out.print("Минимальное количество знаков в целой части 5: ");
      System.out.println(nf.format(d));
   }
}
```

Метод parse() класса NumberFormat преобразует строку к числу. Если перед вызовом метода parse() вызвать метод setParseIntegerOnly(true), как показано в следующем примере, то преобразовываться будет только целая часть числа.

```
import java.text.NumberFormat;
import java.text.ParseException;

public class NumberFormatDemo4 {
   public static void main(String[] args) throws ParseException {
      NumberFormat nf = NumberFormat.getInstance();
      System.out.println(nf.parse("1234,567"));
      nf.setParseIntegerOnly(true);
      System.out.println(nf.parse("1234,567"));
   }
```

}

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое локализация
2. Что такое интернационализация
3. Класс Locale
4. Класс ResourceBundle
5. Класс NumberFormat

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

Задание 1.

Создать properties файлы на русском и английским, содержащие названия изученных тем по Java. Создать метод, который получает в качестве параметра локаль и распечатывает всю информацию из properties файла. Используем метод ResourceBundle.keySet().

Задание 2.

Создать класс FullReport, который выводит ту же информацию, что и Report(задание 6, Лр 6), но заголовок отчета должен изменяться в зависимости от выбранной локали (используем класс ResourceBundle). Для форматирования чисел использовать класс NumberFormat.

ДОМАШНЕЕ ЗАДАНИЕ

Индивидуальное задание

ЛИТЕРАТУРА

Альфред В., Ахо Компиляторы. Принципы, технологии и инструментарий, Вильямс, 2017.

Преподаватель А.С.Кибисова

Рассмотрено на заседании цикловой комиссии программного обеспечения информационных технологий
Протокол № ____ от «___» _____ 2021
Председатель ЦК _____В.Ю.Михалевич