ИТМО ВТ

8. l10n и i18n

Разбор рубежек

- Лямбда-выражения

```
Arrays.sort(array, comparator)
Collections.sort(collection, comparator)

public interface Comparator<T> {
    public int compare(T object1, T object2); }

(object1, object2) ->
    object1.getX() == object2.getX() ? // equals
    object1.getY().compareTo(object2.getY()) :
    object1.getX() - object2.getX();
```

- У примитивных типов нет compareTo

Разбор рубежек (часть 2)

- Коллекции
  - .add (Object o) int → Integer (автоупаковка)
  - .remove (Object o)
- Set — множество неповторяющихся элементов
  - c.add(1); c.add(1) (уже не добавляется);
  - LinkedHashSet — в порядке добавления
  - TreeSet — в натуральном порядке сортировки (1,2,3,4)
  - c.remove(1) — удаляется элемент со значением 1
- List — индексированный список
  - .remove (Object o)
  - .remove (int index)
  - c.remove(1) — удаляется элемент с индексом 1
  - индексация с 0

Локализация и интернационализация

- Локализация — адаптация программных продуктов для определенной местности
  - Перевод текста
  - Использование соответствующих форматов данных
  - Замена звуковой и визуальной информации
  - localization = l10n
- Интернационализация — проектирование программ таким образом, чтобы их локализация была возможна без конструктивных изменений
  - выделение текстовых данных из кода
  - отображение данных с учетом местных форматов
  - internationalization = i18n

Локаль

- Локаль — совокупность характеристик, определяющих географический, политический или культурный регион
- Класс java.util.Locale
- Элементы локали:
  - язык — 2 строчные буквы (иногда 3) (ru)
  - страна (регион) — 2 заглавные буквы (RU) или 3 цифры
  - вариант (например, кодировка для русской локали)
  - письменность — 4 буквы, первая заглавная (Cyrl)
  - расширение

Создание локали

- Конструкторы класса Locale
  - new Locale(String lang)
  - new Locale(String lang, String country)
  - new Locale(String lang, String country, String variant)
- Класс Locale.Builder
  - new Locale.Builder().setLanguage("ru").setRegion("RU").build()
- Метод forLanguageTag()
  - Locale.forLanguageTag("ru-RU);
- Константы класса Locale
  - Locale.FRENCH
  - для русского константы нет

ИТМО ВТ

Методы для работы с локалью

- Получение списка локалей и локали по умолчанию
  - static Locale[] getAvailableLocales()
  - static Locale getDefault()
- Преобразование в строку
  - String toString() // ru_RU
  - String getDisplayName() // Russian (Russia)
- Класс LanguageRange — набор языков
  - *, en-*, *-CA
  - List<Locale> filter(List<LanguageRange>, Collection<Locale>)
  - Locale.lookup((List<LanguageRange>, Collection<Locale>)
  - List<LanguageRange> - осортированный по убыванию

Управление ресурсами

- Класс ResourceBundle
- Набор ресурсов вида "ключ-значение"

```
Locale loc = Locale.US; // Locale.getDefault() = ru_RU;
ResourceBundle rb = ResourceBundle.getBundle("GuiLabels", loc);
```

1) GuiLabels_en_US
2) GuiLabels_en
3) GuiLabels_ru_RU
4) GuiLabels_ru
5) GuiLabels

- MissingResourceException

Ресурсы в виде свойств - PropertyResourceBundle

- GuiLabels_en.properties
  s1 = Yes
  s2 = No

GuiLabels_ru_properties
  s1 = Да
  s2 = Нет

```
ResourceBundle r = ResourceBundle.getBundle("GuiLabels");
JButton b1 = new JButton(r.getString("s1");
JButton b2 = new JButton(r.getString("s2");
```

- \+ отдельные текстовые файлы
- \- только String
- Кодировка ISO-8859-1 — необходима обработка с помощью native2ascii

Ресурсы в виде списка - ListResourceBundle

```
public class GuiLabels_en extends ListResourceBundle {
    public Object[][] getContents() { return contents; }
    private Object[][] contents = { {"s1", "Yes"}, {"s2", "No"} };
}
public class GuiLabels_ru extends ListResourceBundle {
    public Object[][] getContents() { return contents; }
    private Object[][] contents = { {"s1", "Да"}, {"s2", "Нет"} };
}
ResourceBundle r = ResourceBundle.getBundle("GuiLabels");
JButton b1 = new JButton(r.getString("s1");
JButton b2 = new JButton(r.getString("s2");
```

- \+ любые типы объектов
- \- нужна компиляция файлов

Форматирование числовых данных

- Класс NumberFormat — абстрактный
  - NumberFormat nf = NumberFormat.getNumberInstance()
  - NumberFormat cf = NumberFormat.getCurrencyInstance()
  - NumberFormat pf = NumberFormat.getPercentInstance()
  - nf.format(new Float(999.8));
- Класс DecimalFormat
  - df = (DecimalFormat) nf;
  - df.applyPattern("##,#0.00");
  - df.format(new Float(888.7));
- Класс DecimalFormatSymbols
  - DecimalFormatSymbols ds = new DecimalFormatSymbols();
  - ds.setDecimalSeparator('=');
  - df.setDecimalFormatSymbols(ds);
  - df.format(new Float(777.6));

Символы шаблона

0 — цифра, 0 отображается

— цифра, 0 не отображается

. — разделитель десятичной дроби

, — разделитель групп разрядов

E — разделитель мантиссы и порядка

- — знак минус

; — разделитель подшаблонов

% — умножить на 100 и отобразить как процент

‰ — умножить на 100 и отобразить как промилле

¤ — символ валюты

Форматирование даты и времени

- Класс DateFormat — абстрактный
  - DateFormat df = DateFormat.getDateInstance(DateFormat.FULL)
  - DateFormat tf = DateFormat.getTimeInstance(DateFormat.LONG)
  - DateFormat dtf = DateFormat.getDateTimeInstance(DateFormat.SHORT)
  - df.format(new Date());
- Класс SimpleDateFormat
  - sdf = (SimpleDateFormat) df;
  - sdf.applyPattern("yyyy-MM-dd");
  - sdf.format(new Date());
- Класс DateFormatSymbols
  - DateFormatSymbols ds = new DateFormatSymbols();
  - ds.setShortWeekdays("пнд","втр","срд","чтв","птн","сбт","вск");
  - sdf.setDateFormatSymbols(ds);
  - sdf.format(new Date());

Символы шаблона

Более 4 символов — полный формат, 3 — сокращенный, 2 - число

G — эра

y — год

M — месяц после числа

L — месяц (название)

d — число

E — название дня недели

H — часы

m — минуты

s — секунды

S — миллисекунды

z — временная зона

Форматирование сообщений

- Класс MessageFormat
  - 9 марта 2016 в 1:58 будет полное солнечное затмение.
  - Total solar eclipse will be at 1:58 on March 9, 2016.

Solar_en.properties

```
msg = {0} solar eclipse will be at {1,time,short} on {1,date,short}.
full = Total
part = Partial
```

Solar_ru.properties

```
{1,date,short} в {1,time,short} будет {0} солнечное затмение
full = полное
part = частное
```

```
ResourceBundle r = ResourceBundle.getBundle("Solar");
MessageFormat mf = new MessageFormat(r.getString("msg");
Calendar x = new Calendar(); x.set(2016,3,9,1,58,0);
Object[] args = {r.getString("full", x.getTime()};
mf.format(args);
```

Формат с выбором

- Класс ChoiceFormat extends NumberFormat
  - 0 friends like it
  - 1 friend likes it
  - 1000 friends like it

Like_en.properties

```
msg = {0} it
one = {0,number} friend likes
many = {0,number} friends like
```

```
ResourceBundle r = ResourceBundle.getBundle("Like");
MessageFormat mf = new MessageFormat(r.getString("msg");
double[] lims = { 0, 1, 2 };
String o = r.getString("one");
String m = r.getString("many");
String[] msgs = { m, o, m };
ChoiceFormat cf = new ChoiceFormat(lims, msgs);
mf.setFormatByArguments(0, cf);
Object[] args = { new Integer(15) };
mf.format(args);
```

Сравнение строк

- Класс Collator - абстрактный
- Collator getInstance()
- int compare()

```
List<String> lst = Arrays.asList({"Fluor", "Chlor","Brom","Jod"});
Collator c1 = Collator.getInstance(Locale.EN);
Collator c2 = Collator.getInstance(new Locale("cz","CZ");
lst.sort(c1);       // Brom, Chlor, Flour, Jod
lst.sort(c2);       // Brom, Fluor, Chlor, Jod
// A, Á, B, C, Č, D, Ď, E, É, Ě, F, G, H, Ch, I, Í, J, K, L, M, N,
// Ň, O, Ó, P, Q, R, Ř, S, Š, T, Ť, U, Ú, Ů, V, W, X, Y, Ý, Z, Ž
```

- RuleBasedCollator

Разбор текста

- Класс BreakIterator — поиск границ
- Методы
  - getCharacterInstance()
  - getWordInstance()
  - getSentenceInstance()
  - getLineInstance()
  - int first()
  - int last()
  - int next()
  - int previous()
  - int following(int offset)
  - int preceding(int offset)
  - BreakIterator.DONE

Использование даты и времени

- Дата и время — 2 варианта представления:
  - Человеческое время — часы, минуты, дни, недели, месяцы
  - Машинное время — миллисекунды от нулевой точки отсчета
    - 1 января 1970 года, 00:00:00

Традиционные классы. Date

- Date
  - в версии 1.0 — единственный класс даты
  - человеческое и машинное представление
  - форматирование даты
  - версия 1.1 — Date — момент времени
  - почти все методы объявлены deprecated
- Конструкторы
  - Date
  - Date(long)
- Методы
  - long getTime()
  - boolean after(Date)
  - boolean before(Date)

TimeZone

- Временная зона — смещение от стандартного:
- до 1972 года - Гринвич (GMT)
- после 1972 — UTC — всемирное координированное
- Методы
  - getDefault()
  - getAvailableIDs()
  - getRawOffset() - смещение без учета летнего времени
  - getOffset(long date) — с учетом летнего времени

- Класс SimpleTimeZone — реализованный потомок

Calendar

- Абстрактный класс — преобразование из машинных в человеческие единицы
  - Calendar getInstance()
  - add(int field, int amount);
  - roll(int field, int amount);
  - set(int field, int value);
  - Date getTime()
  - setTime(Date)
- реализованный класс GregorianCalendar
  - сочетает 2 календаря (григорианский и юлианский)

Новые пакеты

- java.time — основной пакет для работы с датой и временем
- java.time.chrono — календарные системы
- java.time.format — классы для форматирования
- java.time.temporal — преобразования времени и вычисления
- java.time.zone — работа с поясным временем

- Все классы — неизменяемые, потокобезопасные
-

Соглашения по именам методов

- Статические
  - of — создает экземпляр на основе входных параметров
  - from — конвертирует экземпляр из другого типа
  - parse — создает экземпляр из строкового представления
- Методы экземпляра
  - format — форматирует объект в строку
  - get — возвращает частичное состояние объекта
  - is — проверяет параметр объекта
  - with — возвращает копию объекта с одним измененным элементом
  - plus — возвращает копию объекта с добавленным временем
  - minus — возвращает копию объекта с убавленным временем
  - to — преобразует объект в другой тип
  - at — комбинирует объект с другим

Перечисляемые типы — дни недели и месяцы

- enum DayOfWeek (1 (MONDAY) — 7 (SUNDAY))
- enum Month (1 (JANUARY) — 12 (DECEMBER))
- метод getDisplayName(style, locale)
- стиль — FULL, NARROW, SHORT / STANDALONE

Классы для представления даты и времени

	год	месяц	день	час	минута	секунда	нано
Year	x						
YearMonth	x	x					
MonthDay		x	x				
LocalDate	x	x	x				
LocalTime				x	x	x	x
LocalDateTime	x	x	x	x	x	x	x

.**of**(year,month,day,hour,min,sec,nano)
.**now**()

YearMonth Year.**at**Month(Month)
LocalDate MonthDay.**at**Year(Year)
LocalDate YearMonth.**at**Day(day)
LocalDate Year.**at**MonthDay(MonthDay)
LocalDateTime LocalDate.**at**Time(LocalTime)
LocalDateTime LocalTime.**at**Date(LocalDate)

.get
.getNano()
.getSecond()
.getMinute()
.getHour()
.getDayOfMonth()
.getDayOfWeek()
.getMonth()
.getYear()

.with
.plus
.plusNanos(nano)
.plusSeconds(sec)
.plusMinutes(min)
.plusHours(hour)
.plusDays(day)
.plusWeeks(week)
.plusMonths(month)
.plusYears(year)

.minus

Классы для представления временной зоны

- ZoneId — идентификатор зоны
  - Europe/Moscow
- ZoneOffset — разница со стандартным временем
  - UTC+01:00, GMT-2
- OffsetTime = LocalTime + ZoneOffset
- OffsetDateTime = LocalDateTime + ZoneOffset
- ZonedDateTime = LocalDateTime + ZoneId
  - использует java.time.zone.ZoneRules

Момент времени

- Класс Instant
- now()
- plusNanos()
- plusMillis()
- plusSeconds()
- minusNanos()
- minusMillis()
- minusSeconds()

Разбор и форматирование

- java.time.format.DateTimeFormatter
  - формат можно выбрать из констант:
    - BASIC_ISO_DATE
    - ISO_DATE/TIME/DATETIME
    - ISO_LOCAL_DATE/TIME/DATETIME
    - ISO_OFFSET_DATE/TIME/DATETIME
    - ISO_ZONED_DATETIME
    - ISO_INSTANT
  - задать шаблон
    - ofPattern()
  - методы format() и parse()

Периоды даты и времени

- Duration — продолжительность в часах и менее
  - toNanos(), toMillis(), toSeconds(), toMinutes(), toHours(), toDays()
- Period — период в днях и более
  - getDays(), getMonths(), getYears()
- .between()
- .plus
- .minus

Взаимодействие старых и новых классов

Соответствия:

- Date — Instant
- GregorianCalendar — ZonedDateTime
- TimeZone — ZoneId, ZoneOffset

- Методы:
  - Calendar.toInstant()
  - GregorianCalendar.toZonedDateTime()
  - GregorianCalendar.fromZonedDateTime()
  - Date.fromInstant()
  - Date.toInstant()
  - TimeZone.toZoneId()