Java.SE.03

INFORMATION HANDLING

Author: **Ihar Blinou**
Oracle Certified Java Instructor
ihar_blinou@epam.com

2011 © EPAM Systems, RD Dep.

Содержание

1. Класс String
2. Классы StringBuilder, StringBuffer
3. Форматирование строк. Класс Formatter
4. Интернационализация
5. ResourceBundle
6. Регулярные выражения
7. Pattern & Matcher
8. Кодировки

2011 © EPAM Systems, RD Dep.

КЛАСС STRING

2011 © EPAM Systems, RD Dep.

Класс String

Строка – объект класса **String.**

Строка является **неизменяемой** (**immutable**). Строковое значение не может быть изменено после создания объекта при помощи какого-либо метода.

Любое изменение приводит к созданию **нового** объекта.

Ссылку на объект класса **String** можно изменить так, чтобы она указывала на другой объект, т.е. на другое значение.

2011 © EPAM Systems, RD Dep.

Класс String

Ссылка типа String на строку-константу:

```
String s = "Это строка";
```

Замечание:

- Пустая строка `String s = "";` не содержит ни одного символа.
- Пустая ссылка `String s = null;` не указывает ни на какую строку и не является объектом.

2011 © EPAM Systems, RD Dep.

Класс String

Некоторые конструкторы класса String.

```
String()

String(String str)

String(char[] value)

String(char[] value, int offset, int count)

String(StringBuilder builder)

String(StringBuffer buffer)
```


2011 © EPAM Systems, RD Dep.

Класс String

Примеры создание строк

```
String str1 = new String();

char[] data1 = { 'a', 'b', 'c', 'd', 'e', 'f' };
String str2 = new String(data1, 2, 3);

char[] data2 = { '\u004A', '\u0062', 'V', 'A' };
String str3 = new String(data2);

byte ascii[] = { 65, 66, 67, 68, 69, 70 };
String str4 = new String(ascii); // ”ABCDEF”

byte[] data3 = { (byte) 0xE3, (byte) 0xEE };
String str5 = new String(data3, "CP1251");// ”го”
String str6 = new String(data3, "CP866");// ”ую”
```


2011 © EPAM Systems, RD Dep.

Класс String

Интерфейс **CharSequence** реализуют классы **String**, **StringBuilder**, **StringBuffer**. Методы интерфейса **CharSequence**:

Метод	Описание
public char charAt(int index)	Возвращает char-значение, находящееся в элементе с указанным индексом. Индекс находится в диапазоне от нуля до length() - 1.
public int length()	Возвращает длину данной последовательности символов. Длина - это количество 16-битных char-значений в последовательности.
public CharSequence subSequence(int start, int end)	Возвращает новый объект CharSequence, содержащий подпоследовательность данной последовательности. Подпоследовательность начинается с символа, находящегося под указанным стартовым индексом и заканчивается символом под индексом end - 1.
public String toString()	Возвращает строку, содержащую символы в данной последовательности и в том же порядке. Длина строки будет равна длине последовательности.

2011 © EPAM Systems, RD Dep.

Класс String

Методы чтения символов из строки:

- **char charAt(int index)** – возвращает символ по значению индекса;
- **void getChars(int srcBegin, int srcEnd, char[] dst, int dstBegin)** - возвращает символьное представление участка строки;
- **int length()** – возвращает длину строки;
- **boolean isEmpty()** – возращает true, если строки не содержит симоволов, и false – в противном случае;
- **int codePointAt(int index)** – возвращает кодовую точку для позиции в строке, заданной параметром index

Класс String

- **int codePointBefore(int index)** – возвращает кодовую точку для позиции в строке, предшествующей заданной параметром index;
- **int codePointCount(int beginIndex, int endIndex)** – возвращает количество кодовых точек в порции вызывающей строки, расположенной между символьными порциями beginIndex и endIndex-1;
- **byte[] getBytes()** – возвращает строку в виде последовательности байт, используя кодировку по умолчанию;
- **int offsetByCodePoints(int index, int codePointOffset)** – возвращает позицию в вызывающей строке, расположенную на расстоянии codePointOffset кодовых точек после начальной позиции, заданной параметром index;

Класс String

- **byte[]getBytes(Charset charset)** - возвращает строку в виде последовательности байт, используя указанную в параметре кодировку;

- **void getBytes(int srcBegin, int srcEnd, byte[] dst, int dstBegin)** - возвращает массив байт dst из подстроки с srcBegin до srcEnd индекса;

- **byte[]getBytes(String charsetName)** - возвращает строку в виде последовательности байт, используя название кодировки.

2011 © EPAM Systems, RD Dep.

Класс String

Методы сравнения строк:

- **boolean equals(Object obj)** - проверяет идентична ли строка указанному объекту;
- **boolean equalsIgnoreCase(String str2)** - если строки одинаковы, игнорируя строчные-прописные буквы, то true
- **int compareTo(String str2)** - лексиграфическое сравнение строк;

2011 © EPAM Systems, RD Dep.

Класс String

- **compareTolgnoreCase(String str)** - лексиграфическое сравнение строк без учета регистра символов;
- **boolean contentEquals(CharSequence cs)** – сравнивает строку с объектом типа CharSequence;
- **boolean contentEquals(StringBuffer sb)** – сравнивает строку с объектом типа StringBuffer;
- **String intern() -** занесение строки в пул литералов.

2011 © EPAM Systems, RD Dep.

Класс String

Работа с символами строки:

- **String toUpperCase()** - преобразует строку в верхний регистр;
- **String toUpperCase(Locale locale)** - преобразует строку в верхний регистр, используя указанную локализацию;
- **String toLowerCase()** - преобразует строку в нижний регистр;
- **String toLowerCase(Locale locale)** - преобразует строку в нижний регистр, используя указанную локализацию;
- **char[] toCharArray()** - преобразует строку с новый массив символов.

2011 © EPAM Systems, RD Dep.

Класс String. Example 1

Объединение строк:

- String concat(String str) ИЛИ +

```
package _java._se._03._string;
public class ConcatExample {
  public static void main(String[] args) {
    String attention = "Внимание: ";
    String s1 = attention.concat( "!!!");
    String s2 = attention + "неизвестный символ";
    System.out.println("s1 = " + s1);
    System.out.println("s2 = " + s2);
    String str1 = "2" + 2 + 2;
    String str2 = 2 + 2 + "2";
    String str3 = "2" + (2 + 2);
    System.out.println("str1=‟+str1+"; str2=‟+str2+";
str3=‟+str3);
  }
}
```

Результат:

```
s1 = Внимание: !!!
s2 = Внимание: неизвестный символ
str1=222; str2=42; str3=24
```

Класс String

Поиск символов и подстрок

- **int indexOf(int ch)** - поиск первого вхождения символа в строке;
- **int indexOf(int ch, int fromIndex)** - поиск первого вхождения символа в строке с указанной позиции;
- **int indexOf(String str)** - поиск первого вхождения указанной подстроки;
- **int indexOf(String str, int fromIndex)** - поиск первого вхождения указанной подстроки с указанной позиции;
- **int lastIndexOf(int ch)** - поиск последнего вхождения символа;

2011 © EPAM Systems, RD Dep.

Класс String

- **int lastIndexOf(int ch, int fromIndex)** - поиск последнего вхождения символа с указанной позиции;
- **int lastIndexOf(String str)** - поиск последнего вхождения строки;
- **int lastIndexOf(String str, int fromIndex)** - поиск последнего вхождения строки с указанной позиции;
- **String replace(char oldChar, char newChar)** - замена в строке одного символа на другой;
- **String replace(CharSequence target, CharSequence replacement)** - замена одной подстроки другой;

2011 © EPAM Systems, RD Dep.

Класс String

- **boolean contains(CharSequence cs)** - проверяет, входит ли указанная последовательность символов в строку;
- **static String copyValueOf(char[] data)** - возвращает строку, равную символам data;
- **static String copyValueOf(char[] data, int offset, int count)** - возвращает подстроку, равную части символов data;

2011 © EPAM Systems, RD Dep.

Класс String

- **boolean endsWith(String suffix)** - заканчивается ли String суффиксом suffix;
- **boolean startsWith(String prefix)** - начинается ли String с префикса prefix;
- **boolean startsWith(String prefix, int toffset)** - начинается ли String с префикса prefix учитывая смещение toffset.

2011 © EPAM Systems, RD Dep.

Класс String

Извлечение подстрок

- **String trim()** – отсекает на концах строки пустые символы;
- **String substring(int startIndex)** – возвращает подстроку, с startIndex до конца строки;
- **String substring(int startIndex, int endIndex)** – возвращает подстроку с beginIndex до endIndex;
- **CharSequence subSequence(int beginIndex, int endIndex)** – сокращает подпоследовательность типа CharSequence как подстроку с beginIndex до endIndex.

2011 © EPAM Systems, RD Dep.

Класс String

Приведение значений элементарных типов и объектов к строке

- **String toString()** - возвращает саму строку;
- **static String valueOf(Object obj)** - возвращает результат toString для объекта;
- **static String valueOf(char[] charArray)** - возвращает строку, из символов **charArray**;
- **static String valueOf(char[] data, int offset, int count)** - возвращает подстроку, из части символов data;

2011 © EPAM Systems, RD Dep.

Класс String

- **static String valueOf(boolean b)** -.возвращает строку “true” или “false”, в зависимости от b;
- **static String valueOf(char c)** - возвращает строку из символа c;
- **static String valueOf(int i)** - возвращает строку, полученную из i;
- **static String valueOf(long l)** - возвращает строку, полученную из l;
- **static String valueOf(float f)** - возвращает строку, полученную из f;
- **static String valueOf(double d)** - возвращает строку, полученную из d.

2011 © EPAM Systems, RD Dep.

Класс String

Форматирование строк

- **static String format(String format, Object... args)**
- **static String format(Locale l, String format, Object... args)**

(см. класс Formatter)

Класс String

Сопоставление с образцом

- **boolean regionMatches(boolean ignoreCase, int toffset, String ther, int ooffset, int len) -** сравнивает часть строки с другой строкой, если ignoreCase=true, то игнорирует строчные-прописные буквы;

- **boolean regionMatches(int toffset, String other, int ooffset, int len) -** сравнивает часть строки с другой строкой, len - сколько символов сравнивать;

- **String replace(char oldChar, char newChar) -** возвращает строку, где все символы oldChar заменены на newChar;

2011 © EPAM Systems, RD Dep.

Класс String

- **String replace(CharSequence target, CharSequence replacement)** - возвращает строку, заменяя элементы target на replacement.
- **boolean matches(String regexStr)** - удовлетворяет ли строка указанному регулярному выражению;
- **String replaceFirst(String regexStr, String replacement)** - заменяет первое вхождение строки, удовлетворяющей регулярному выражению, указанной строкой;
- **String replaceAll(String regexStr, String replacement)** - заменяет все вхождения строк, удовлетворяющих регулярному выражению, указанной строкой;

2011 © EPAM Systems, RD Dep.

Класс String

- **String[] split(String regexStr)** - разбивает строку на части, границами разбиения являются вхождения строк, удовлетворяющих регулярному выражению;

- **String[] split(String regexStr, int limit)** - аналогично предыдущему, но с ограничением применения регулярного выражения к строке значением limit. Если limit>0, то и размер возвращаемого массива строк не будет больше limit. Если limit<=0, то регулярное выражение применяется к строке неограниченное число раз.

2011 © EPAM Systems, RD Dep.

Класс String. Example 2

```
package _java._se._03._string;
public class StringReplaceFirst {
  public static void main(String[] args) {
    String str = "Her name is Tamana and Tamana is a good
girl.";
    String strreplace = "Sonia";
    String result = str.replaceFirst( "Tamana", strreplace);
    System.out.println(result);
  }
}
```

Результат:

```
Her name is Sonia and Tamana is a good
girl.
```


2011 © EPAM Systems, RD Dep.

Класс String. Example 3

```
package _java._se._03._string;
public class StringValueOf {
     public static void main(String[] args) {
          int i = 10;          float f = 10.0f;
          long l = 10;
          double d = 10.0d;          char c = 'a';
          boolean b = true;
          Object o = new String("Hello World");
          System.out.println(String.valueOf(i));
          System.out.println(String.valueOf(f));
          System.out.println(String.valueOf(l));
          System.out.println(String.valueOf(d));
          System.out.println(String.valueOf(c));
          System.out.println(String.valueOf(b));
          System.out.println(String.valueOf(o));
     }
}
```

Результат:

```
10
10.0
10
10.0
a
true
Hello World
```

Класс String. Example 4

```
package _java._se._03._string;
public class StringEquals {
    public static void main(String[] args) {
        String str1="Hello";
        String str2=new String("Hello");
        if(str1 == str2) System.out.println("Equal");
        else  System.out.println("Not Equal");

        str2 = str2.intern();
        if(str1 == str2) System.out.println("Equal");
        else  System.out.println("Not Equal");

        if(str1.equals(str2)) System.out.println("Equal");
        else  System.out.println("Not Equal");
    }
}
```

Результат:

```
Not Equal
Equal
Equal
```

КЛАССЫ STRINGBUILDER, STRINGBUFFER

2011 © EPAM Systems, RD Dep.

Классы StringBuilder, StringBuffer

Классы **StringBuilder** и **StringBuffer** по своему предназначению близки к классу **String.**

Но, содержимое и размеры объектов классов **StringBuilder** и **StringBuffer** можно изменять!!!

Основным и единственным отличием **StringBuilder** от **StringBuffer** является потокобезопасность последнего.

2011 © EPAM Systems, RD Dep.

Классы StringBuilder, StringBuffer

Конструкторы класса **StringBuilder**

- **StringBuilder(String str)** – создает **StringBuilder**, значение которого устанавливается в передаваемую строку, плюс дополнительные 16 пустых элементов в конце строки.
- **StringBuilder(CharSequence charSeq)** – строит **StringBuilder**, содержащий те же самые символы как указанно в CharSequence, плюс дополнительные 16 пустых элементов, конечных CharSequence.
- **StringBuilder(int length)** – создает пустой **StringBuilder** с указанной начальной вместимостью.
- **StringBuilder()** – создает пустой **StringBuilder** со способностью 16 (16 пустых элементов).

2011 © EPAM Systems, RD Dep.

Классы StringBuilder, StringBuffer

Чтение и изменение символов объекта **StringBuilder**

- **int length()** – созвращает количество символов в строке.
- **char charAt(int index)** – возвращает символьное значение, расположенное на месте index.
- **void setCharAt(int index, char ch)** – символ, расположенный на месте index заменяется символом ch.
- **CharSequence subSequence(int start, int end)** – возвращает новую подстроку.

2011 © EPAM Systems, RD Dep.

Классы StringBuilder, StringBuffer

ОТЛИЧИЕ объектов класса **String** от объектов классов **StringBuilder**, **StringBuffer**

Для класса **StringBuffer** не переопределены методы **equals()** и **hashCode()**, т.е. сравнить содержимое двух объектов невозможно, к тому же хэш-коды всех объектов этого типа вычисляются так же, как и для класса **Object**.

2011 © EPAM Systems, RD Dep.

Классы StringBuilder, StringBuffer

Добавление символов в объект класса **StringBuilder**. Добавляет аргумент этому **StringBuilder**. Данные преобразовываются в строку прежде, чем операция добавить будет иметь место.

- **StringBuilder append(Object obj)**
- **StringBuilder append(String str)**
- **StringBuilder append(CharSequence charSeq)**
- **StringBuilder append(CharSequence charSeq, int start, int end)**
- **StringBuilder append(char[] charArray)**
- **StringBuilder append(char[] charArray, int offset, int length)**
- **StringBuilder append(char c)**

2011 © EPAM Systems, RD Dep.

Классы StringBuilder, StringBuffer

Добавление символов в объект класса **StringBuilder** (продолжение).

- **StringBuilder append(boolean b)**
- **StringBuilder append(int i)**
- **StringBuilder append(long l)**
- **StringBuilder append(float f)**
- **StringBuilder append(double d)**
- **StringBuilder append(StringBuffer sb)**
- **StringBuilder appendCodePoint(int codePoint)**

Классы StringBuilder, StringBuffer

Вставка символов в объект **StringBuilder**. Вставляет второй аргумент в **StringBuilder**.. Первый аргумент целого числа указывает индекс, перед которым должны быть вставлены данные. Данные преобразовывают в строку прежде, чем операция вставки будет иметь место.

- **StringBuilder insert(int offset, Object obj)**
- **StringBuilder insert(int dstOffset, CharSequence seq)**
- **StringBuilder insert(int dstOffset, CharSequence seq, int start, int end)**
- **StringBuilder insert(int offset, String str)**
- **StringBuilder insert(int offset, char[] charArray)**
- **StringBuilder insert(int offset, char c)**

2011 © EPAM Systems, RD Dep.

Классы StringBuilder, StringBuffer

Вставка символов в объект **StringBuilder** (продолжение).

- **StringBuilder insert(int offset, boolean b)**
- **StringBuilder insert(int offset, int i)**
- **StringBuilder insert(int offset, long l)**
- **StringBuilder insert(int offset, float f)**
- **StringBuilder insert(int offset, double d)**
- **StringBuilder insert(int index, char[] str, int offset, int len)**

2011 © EPAM Systems, RD Dep.

Классы StringBuilder, StringBuffer

Удаление символов из объекта StringBuilder.

- **StringBuilder deleteCharAt(int index)** – удаляет символ, расположенный по index.
- **StringBuilder delete(int start, int end)** – удаляет подпоследовательность от start до end-1(включительно) в последовательности символов **StringBuilder's.**
- **StringBuilder reverse()** – полностью изменяет последовательность символов в этом **StringBuilder.**

2011 © EPAM Systems, RD Dep.

Классы StringBuilder, StringBuffer

Управление ёмкостью.

- **int capacity()** – возвращает текущую емкость.
- **void ensureCapacity(int minCapacity)** – гарантирует, что вместимость по крайней мере равна указанному минимуму.
- **void trimToSize()** – уменьшает емкость до величины хранимой последовательности.
- **void setLength(int newLength)** – устанавливает длину символьной последовательности. Если newLength - меньше чем length(), последние символы в символьной последовательности являются усеченными. Если newLength больше чем length(), нулевые символы добавляются в конце символьной последовательности.

2011 © EPAM Systems, RD Dep.

Классы StringBuilder, StringBuffer

В классе присутствуют также методы, аналогичные методам класса **String,** такие как:

replace(), **charAt**(), **length**(), **getChars**(), **codePointAt**(int index), **codePointBefore**(int index), **codePointCount**(int beginIndex, int endIndex), **getChars**(int srcBegin, int srcEnd, char[] dst, int dstBegin), **indexOf**(String str), **indexOf**(String str, int fromIndex), **lastIndexOf**(String str), **lastIndexOf**(String str, int fromIndex), **offsetByCodePoints**(int index, codePointOffset), **replace**(int start, int end, String str), **substring**(int start), **substring**(int start, int end) , **toString**().

2011 © EPAM Systems, RD Dep.

Классы StringBuilder, StringBuffer. Example 5

```
package _java._se._03._stringbuilder;
public class StringBuilderAppend {
    public static void main(String[] args) {
        StringBuilder sb = new StringBuilder();
        sb.append("Java StringBuilder");
        System.out.println("StringBuilder1 : " + sb);
        sb.append(" Example");
        System.out.println("StringBuilder2 : " + sb);
    }
}
```

Результат:

```
StringBuilder1 : Java StringBuilder
StringBuilder2 : Java StringBuilder
Example
```

Классы StringBuilder, StringBuffer. Example 6

```
package _java._se._03._stringbuilder;
public class StringBuilderInsert {
     public static void main(String[] args) {
          StringBuilder sb = new StringBuilder();
          sb.append("Java StringBuilder");
          sb.insert(5, "insert ");
          System.out.println("StringBuilder :" + sb);
     }
}
```

Результат:

```
StringBuilder :Java insert
StringBuilder
```


2011 © EPAM Systems, RD Dep.

Классы StringBuilder, StringBuffer. Example 7

```
package _java._se._03._stringbuilder;
public class StringBuilderSetcharat {
    public static void main(String[] args) {
        StringBuilder sb = new StringBuilder();
        sb.append("Java tringBuilder");
        sb.setCharAt(5, 'S');
        System.out.println("StringBuilder : "+sb);
    }
}
```

Результат:

```
StringBuilder : Java
SringBuilder
```


2011 © EPAM Systems, RD Dep.

ФОРМАТИРОВАНИЕ СТРОК. КЛАСС FORMATTER

2011 © EPAM Systems, RD Dep.

Форматирование строк. Класс Formatter

Класс **Formatter** (пакет java.util) - обеспечивает преобразование формата позволяющее выводить числа, строки, время и даты нужном формате

- **format(String format, Object... args)**
- **format(Locale l, String format, Object... args)**

2011 © EPAM Systems, RD Dep.

Форматирование строк. Класс Formatter

Для классов **PrintStream** и **PrintWriter** добавлен метод **printf()**. Метод **printf()** автоматически использует класс **Formatter**.

- **printf(String format, Object... args)**
- **printf(Locale l, String format, Object... args)**

2011 © EPAM Systems, RD Dep.

Форматирование строк. Класс Formatter

Спецификаторы формата. Общий синтаксис спецификатора формата следующий:

%[argument_index][flags][width][precision]conversion

Значение аргумента спецификатора формата **conversion** приведены в таблице далее. Кроме строчного написания значения **conversion** *можно использовать* следующие *значения*, *определяемые прописными буквами*: ‘B’, ‘H’, ‘S’, ‘C’, ‘X’, ‘E’, ‘G’, ‘A’, ‘T’.

2011 © EPAM Systems, RD Dep.

Форматирование строк. Класс Formatter

Параметр **conversion**

Спецификатор формата	Выполняемое форматирование
%a	Шестнадцатеричное значение с плавающей точкой
%b	Логическое (булево) значение аргумента
%c	Символьное представление аргумента
%d	Десятичное целое значение аргумента
%h	Хэш-код аргумента
%e	Экспоненциальное представление аргумента
%f	Десятичное значение с плавающей точкой

2011 © EPAM Systems, RD Dep.

Форматирование строк. Класс Formatter

Параметр **conversion**

Спецификатор формата	Выполняемое форматирование
%g	Выбирает более короткое представление из двух: %e или %f
%o	Восьмеричное целое значение аргумента
%n	Вставка символа новой строки
%s	Строковое представление аргумента
%t	Время и дата
%x	Шестнадцатеричное целое значение аргумента
%%	Вставка знака %

2011 © EPAM Systems, RD Dep.

Форматирование строк. Класс Formatter. Example 8

```
package _java._se._03._format;
import java.util.Formatter;
import java.util.Timer;
public class SimpleFormatExample {
  public static void main(String[] args) {
     Formatter formatter = new Formatter();
     boolean b1 = true;
     Boolean b2 = null;
     formatter.format("1. - %b, %b\n", b1, b2);
     System.out.println(formatter);
     System.out.println("-----------------------------------");
     Timer t = new Timer();
     formatter.format("2. - %h", t);
          // Integer.toHexString(t.hashCode)
     System.out.println(formatter);
     System.out.println(Integer.toHexString(t.hashCode()));
  }
}
```

Результат:

```
1. - true, false
-----------------------------------
1. - true, false
2. - 229ed927
229ed927
```

Форматирование строк. Класс Formatter

Аргумент спецификатора формата [**argument_index**] имеет два вида

i$ или **<**.

- **i$** – i (десятичное целое число) - указывает на положение аргумента во множестве параметров переменной длины **format(String format, Object... args)**, начинающемся с положения 1.

- **<** – указывает на тот же самый аргумент, который использовался в предыдущем спецификаторе формата в форматирующей последовательности, и не может поэтому быть первым в списке спецификаторов формата.

2011 © EPAM Systems, RD Dep.

Форматирование строк. Класс Formatter. Example 9

```
package _java._se._03._format;

import java.util.Formatter;

public class ArgumentIndexExample {
    public static void main(String[] args) {
        Formatter formatter = new Formatter();

        double d1 = 16.78967;
        formatter.format("%1$e, %<f, %<g, %<a\n", d1);
        System.out.println(formatter);
    }
}
```

Результат:

```
1.678967e+01, 16.789670, 16.7897,
0x1.0ca27d028a1ep4
```


2011 © EPAM Systems, RD Dep.

Форматирование строк. Класс Formatter

[**flag**] – указывает выравнивание форматируемого аргумента. Значение параметра **flag** приведены в таблице. Комбинация валидных флагов в спецификаторе формата зависит от преобразования.

Flag	Integrals			Floating-point				Description
	d	o	x X	e E	f	g G	a A	
‘-’	ok	ok	ok	ok	ok	ok	ok	Выравнивание по левому краю, требует положительного значения width (Также подходит для отображения символов, времени-даты)
‘#’	x	ok	ok	ok	ok	x	ok	Отображает в виде, применяемом для системы счисления и десятичную точку для вещественных
‘+’	ok	x	x	ok	ok	ok	ok	Отображает знак

2011 © EPAM Systems, RD Dep.

Форматирование строк. Класс Formatter

[**flag**] – указывает выравнивание форматируемого аргумента. Значение параметра **flag** приведены в таблице. Комбинация валидных флагов в спецификаторе формата зависит от преобразования.

Flag	Integrals			Floating-point				Description
	d	o	x X	e E	f	g G	a A	
‘ ‘	ok	x	x	ok	ok	ok	ok	Лидирующие пробелу для положительных значений
‘0’	ok	ok	ok	ok	ok	ok	ok	Отображает лидирующие пробелы, требует положительного значения width
‘,’	ok	x	x	x	ok	ok	x	Использует групповой разделитель, указываемый локалью
‘(‘	ok	x	x	ok	ok	ok	x	Отображает отрицательные числа в круглых скобках

2011 © EPAM Systems, RD Dep.

Форматирование строк. Класс Formatter. Example 10

```
package _java._se._03._format;
import java.util.Formatter;
public class FlagExample {
     public static void main(String[] args) {
          Formatter formatter = new Formatter();

          int i1 = 17;
          double d1 = 16.78967;
          formatter.format("1. (%%o) %o%n", i1);
          formatter.format("2. (%%a) %a%n", d1);
          formatter.format("3. (%%x) %x%n", i1);
          formatter.format("4. (%%#o) %#o%n", i1);
          formatter.format("5. (%%#a) %#a%n", d1);
          formatter.format("6. (%%#x) %#x%n", i1);
          System.out.println(formatter);
     }
}
```

Результат:

```
1. (%o) 21
2. (%a) 0x1.0ca27d028a1ep4
3. (%x) 11
4. (%#o) 021
5. (%#a) 0x1.0ca27d028a1ep4
6. (%#x) 0x11
```

Форматирование строк. Класс Formatter

Width – минимальное число символов, отводимое под представление форматируемого параметра.

Precision – имеет формат **.n**, где **n** – число символов в десятичной части числа. Особенности поведения зависят от преобразования.

Форматирование строк. Класс Formatter. Example 11

```
package _java._se._03._format;
import java.util.Formatter;
public class UseFormatterExample {
    public static void main(String[] args) {
        Formatter formatter = new Formatter();
        int i1 = 345;
        double d1 = 16.78967;
        formatter.format("- %-7dok%n", i1);
        formatter.format("- %+7dok%n", i1);
        formatter.format("- % 7dok%n", i1);
        formatter.format("- %07dok%n", i1);
        formatter.format("- %#fok%n", d1);
        formatter.format("- %.2fok%n", d1);
        System.out.println(formatter);
    }
}
```

Результат:

```
- 345    ok
-    +345ok
-     345ok
- 0000345ok
- 16.789670ok
- 16.79ok
```

Форматирование строк. Класс Formatter

Форматирование времени и даты.

Спецификатор формата	Выполняемое преобразование
%tH	Час (00 - 23)
%tI	Час (1 - 12)
%tM	Минуты как десятичное целое (00 - 59)
%tS	Секунды как десятичное целое (00 - 59)
%tL	Миллисекунды (000 - 999)
%tY	Год в четырехзначное формате
%ty	Год в двузначное формате (00 - 99)
%tB	Полное название месяца (“Январь”)
%tb или %th	Краткое название месяца (“янв”)

2011 © EPAM Systems, RD Dep.

Форматирование строк. Класс Formatter

Форматирование времени и даты.

Спецификатор формата	Выполняемое преобразование
%tm	Месяц в двузначном формате (1 - 12)
%tA	Полное название дня недели (“Пятница”)
%ta	Краткое название дня недели (“Пт”)
%td	День в двузначном формате (1 - 31)
%tR	То же что и “%tH:%tM”
%tT	То же что и “%tH:%tM:%tS”
%tr	То же что и “%tI:%tM:%tS %Tp” где %Tp = (AM или PM)
%tD	То же что и “%tm/%td/%ty”
%tF	То же что и “%tY-%tm-%td”
%tc	То же что и “%ta %tb %td %tT %tZ %tY”

2011 © EPAM Systems, RD Dep.

Форматирование строк. Класс Formatter. Example 12

```
package _java._se._03._format;
import java.util.Calendar;
import java.util.Formatter;
public class DateTimrFormatExample {

    public static void main(String[] args) {
        Formatter formatter = new Formatter();
        Calendar calendar = Calendar.getInstance();

        formatter.format("%tr", calendar);
        System.out.println(formatter);
    }
}
```

Результат:

```
09:20:30 AM
```

Форматирование строк. Класс Formatter

При работе с классом **Formatter** могут возникнуть следующие исключения. Данные классы исключений являются подклассами класса **IllegalFormatException**.

Форматирование строк. Класс Formatter

Format Exception	Meaning
DublicateFormatFlagsException	Flag используется более, чем один раз
FormatFlagsConversionMismatchException	Flag и conversion несовместимы
IllegalFormatConversionException	Тип аргумента несовместим с преобразованием
IllegalFormatFlagsException	Недействительная комбинация флагов
IllegalFormatPresionException	Точность неправильна или недопустима
IllegalFormatWidthException	Значение width недопустимо
MissingFormatArgumentException	Ошибка при передаче параметров
MissingFormatWidthException	Ошибка при задании ширины
UnknownFormatConversionException	Преобразование неизвестно
UnknownFormatFlagsException	Флаг неизвестен

2011 © EPAM Systems, RD Dep.

Форматирование строк. Класс Formatter

Метод **printf()** автоматически использует объект типа **Formatter** для создания форматированной строки. Она выводится как строка в стандартный поток вывода по умолчанию на консоль. Метод **printf()** определен в классах **PrintStream** и **PrintWriter**. В классе **PrintStream** у метода **printf()** две синтаксические формы записи:

- **PrintStream printf(String *fmtString,* Object...*args)**
- **PrintStream printf(Local *loc,* String *fmtString,* Object...*args)**

2011 © EPAM Systems, RD Dep.

Форматирование строк. Класс Formatter. Example 13

```
package _java._se._03._format;

import java.util.Calendar;
import java.util.Locale;

public class PringfExample {
  public static void main(String[] args) {
    Calendar cal = Calendar.getInstance();
    System.out.printf(Locale.FRANCE, "%1$tB %1$tA%n", cal);
    System.out.printf(Locale.getDefault(), "%1$tB %1$tA%n", cal);
  }
}
```

Результат:

```
septembre lundi
September
Monday
```


2011 © EPAM Systems, RD Dep.

ИНТЕРНАЦИОНАЛИЗАЦИЯ

2011 © EPAM Systems, RD Dep.

Интернационализация

Интернационализация программы (i18n) –

- Написание программы, работающей в различных языковых окружениях

Локализация программы (l10n) –

- Адаптация интернационализированной программы к конкретным языковым окружением

Пакеты

- java.util
- java.text

2011 © EPAM Systems, RD Dep.

Интернационализация

Класс **Locale**, (пакет java. util) идентифицирует используемое языковое окружение

Локаль определяется:

1) константами: **Locale.US, Locale.FRANCE**

2) конструкторами класса **Locale**

- Locale(language) – по языку
- Locale(language, country) – по языку и стране
- Locale(language, country, variant) – по языку стране и варианту

```
Locale l = new Locale("ru","RU");
Locale l = new Locale("en","US","WINDOWS");
```


2011 © EPAM Systems, RD Dep.

Интернационализация

Пример	
en_UK_windows	**en_UK_unix**
choose the folder containing colour information	choose the directory containing colour information
en_US	**ru_RU_unix**
choose the folder containing color information	Выберите каталог, содержащий цветовую информацию

Интернационализация

Методы класса **Locale**

- **getDefault()** возвращает текущую локаль, сконструированную на основе настроек операционной системы.
- **getLanguage()** – код языка региона
- **getDisplayLanguage** – название языка
- **getCountry()** – код региона
- **getDisplayCountry()** – название региона
- **getAvailableLocales()** – список доступных локалей

2011 © EPAM Systems, RD Dep.

Интернационализация. Example 14

```
package _java._se._03._locale.locale;

import java.util.Locale;

public class LocaleExample {
  public static void main(String[] args) {
    Locale defaultLocale = Locale.getDefault();
    Locale rusLocale = new Locale("ru","RU");
    Locale usLocale = new Locale("en", "US");
    Locale frLocale = new Locale("fr", "FR");

    System.out.println(defaultLocale.getDisplayCountry());
    System.out.println(defaultLocale
    .getDisplayCountry(Locale.FRENCH));
    System.out.println(frLocale.getDisplayCountry(defaultLocale));

    System.out.println(usLocale.getDisplayName());
    System.out.println(usLocale.getDisplayName(frLocale));
```

Интернационализация. Example 14

```
        System.out.println(rusLocale.getDisplayName(frLocale));

        System.out.println(defaultLocale.getCountry());
        System.out.println(defaultLocale.getLanguage());
        System.out.println(defaultLocale.getVariant());
    }
}
```

Результат:

```
Россия
Russie
Франция
английский (Соединенные
Штаты)
anglais (Etats-Unis)
russe (Russie)
RU
ru
```


2011 © EPAM Systems, RD Dep.

Интернационализация

Интернационализация чисел и дат - вывод данных в соответствии с языковым контекстом.

Типы данных

- Числа
- Время и дата
- Сообщения

Пакет

- java.text

Класс **NumberFormat**

Получение форматировщиков чисел

- getNumberInstance(locale) – обычные числа
- getIntegerIntance(locale) – целые числа (с округлением)
- getPercentInstance(locale) – проценты
- getCurrencyInstance(locale) – валюта

2011 © EPAM Systems, RD Dep.

Класс **NumberFormat**

Методы форматирования

- String format(long) – форматировать целое число
- String format(double) – форматировать число с плавающей точкой
- Number parse(String) – разобрать локализованное число

Выбрасываемое исключение

- ParseException – ошибка разбора

2011 © EPAM Systems, RD Dep.

Интернационализация. Example 15

```
package _java._se._03._locale.locale.number;

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

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

     Locale[] locales = { Locale.getDefault(), new Locale("no", "NO"),
               Locale.JAPAN };

     NumberFormat[] numFormatters = new NumberFormat[] {
          NumberFormat.getNumberInstance(),
          NumberFormat.getNumberInstance(locales[1]),
          NumberFormat.getNumberInstance(locales[2]) };

     NumberFormat[] currFormatters = new NumberFormat[] {
          NumberFormat.getCurrencyInstance(),
          NumberFormat.getCurrencyInstance(locales[1]),
          NumberFormat.getCurrencyInstance(locales[2]) };
```

Интернационализация. Example 15

```
    double number = 9876.598;
    System.out.println("Formatting the number: " + number);
    runFormatters(number, numFormatters, locales);

    for (NumberFormat nf : numFormatters) {
         nf.setMaximumFractionDigits(2);
    }

    System.out.println("\nFormatting the number " + number
                    + " (to 2 dec. places: ");
    runFormatters(number, numFormatters, locales);
    System.out.println("\nFormatting the currency amount: " + number);
    runFormatters(number, currFormatters, locales);
    runParsers("9876.598", numFormatters, locales);
    runParsers("9876,598", numFormatters, locales);
    runParsers("9876@598", numFormatters, locales);
    runParsers("@9876598", numFormatters, locales);
    runParsers("£9876.598", currFormatters, locales);
    runParsers("kr 9876,598", currFormatters, locales);
    runParsers("JPY 98@76598", currFormatters, locales);
    runParsers("@9876598", currFormatters, locales);
}
```

Интернационализация. Example 15

```
  static void runFormatters(double value, NumberFormat[] formatters,
                    Locale[] locales) {
     for (int i = 0; i < formatters.length; i++) {
          System.out.printf("%-24s: %s%n",
               locales[i].getDisplayName(),
               formatters[i].format(value));
     }
  }
  static void runParsers(String inputString, NumberFormat[] formatters,
                    Locale[] locales) {
     System.out.println("\nParsing: " + inputString);
     for (int i = 0; i < formatters.length; i++) {
       try {
           System.out.printf("%-24s: %s%n", locales[i].getDisplayName(),
               formatters[i].parse(inputString));
       } catch (ParseException e) {
           System.out.println(e);
       }
     }
  }
}
```

Интернационализация. Example 15

Результат:

```
Formatting the number: 9876.598
English (United Kingdom): 9,876.598
Norwegian (Norway)      : 9 876,598
Japanese (Japan)        : 9,876.598

Formatting the number 9876.598 (to 2 dec. places:
English (United Kingdom): 9,876.6
Norwegian (Norway)      : 9 876,6
Japanese (Japan)        : 9,876.6

Formatting the currency amount: 9876.598
English (United Kingdom): £9,876.60
Norwegian (Norway)      : kr 9 876,60
Japanese (Japan)        : ?9,877

Parsing: 9876.598
English (United Kingdom): 9876.598
Norwegian (Norway)      : 9876
Japanese (Japan)        : 9876.598

Parsing: 9876,598
English (United Kingdom): 9876598
Norwegian (Norway)      : 9876.598
Japanese (Japan)        : 9876598
```


2011 © EPAM Systems, RD Dep.

Интернационализация. Example 15

Результат:

```
Parsing: 9876@598
English (United Kingdom): 9876
Norwegian (Norway)      : 9876
Japanese (Japan)        : 9876

Parsing: @9876598
java.text.ParseException: Unparseable number: "@9876598"
java.text.ParseException: Unparseable number: "@9876598"
java.text.ParseException: Unparseable number: "@9876598"

Parsing: £9876.598
English (United Kingdom): 9876.598
java.text.ParseException: Unparseable number: "£9876.598"
java.text.ParseException: Unparseable number: "£9876.598"

Parsing: kr 9876,598
java.text.ParseException: Unparseable number: "kr 9876,598"
Norwegian (Norway)      : 9876.598
java.text.ParseException: Unparseable number: "kr 9876,598"
```


2011 © EPAM Systems, RD Dep.

Интернационализация. Example 15

Результат:

```
Parsing: JPY 98@76598
java.text.ParseException: Unparseable number: "JPY 98@76598"
java.text.ParseException: Unparseable number: "JPY 98@76598"
java.text.ParseException: Unparseable number: "JPY 98@76598"

Parsing: @9876598
java.text.ParseException: Unparseable number: "@9876598"
java.text.ParseException: Unparseable number: "@9876598"
java.text.ParseException: Unparseable number: "@9876598"
```


2011 © EPAM Systems, RD Dep.

Класс **DateFormat**

Получение форматировщиков времени и дат

- getDateInstance([dateStyle[, locale]]) – даты
- getTimeIntance([timeStyle[, locale]]) – времени
- getDateTimeIntance([dateStyle, timeStyle, [locale]]) – даты и времени

2011 © EPAM Systems, RD Dep.

Интернационализация

Класс **DateFormat**

Стили

- DEFAULT, FULL, LONG, MEDIUM, SHORT

Методы форматирования

- String format(date) – форматировать дату/время
- Date parse(String) – разобрать локализованную дату/время

Выбрасываемое исключение

- ParseException – ошибка разбора

2011 © EPAM Systems, RD Dep.

Интернационализация. Example 16

```
package _java._se._03._locale.locale.date;

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

public class DateFormatExample {
  public static void main(String[] args) {
     DateFormat[] dateTimeFormatters =  new DateFormat[] {
     DateFormat.getDateTimeInstance(DateFormat.FULL,
              DateFormat.FULL, Locale.US),
     DateFormat.getDateTimeInstance(DateFormat.LONG,
              DateFormat.LONG, Locale.US),
     DateFormat.getDateTimeInstance(DateFormat.MEDIUM,
              DateFormat.MEDIUM, Locale.US),
     DateFormat.getDateTimeInstance(DateFormat.SHORT,
              DateFormat.SHORT, Locale.US)
      };
```

```
FULL   : Tuesday, September 6, 2011 2:09:56 AM EEST
LONG   : September 6, 2011 2:09:56 AM EEST
MEDIUM: Sep 6, 2011 2:09:56 AM
SHORT : 9/6/11 2:09 AM
```

Интернационализация. Example 16

```
String[] styles = { "FULL", "LONG", "MEDIUM", "SHORT" };
    Date date = new Date();
    int i = 0;
    for (DateFormat dtf : dateTimeFormatters) {
        System.out.printf("%-6s: %s%n",
            styles[i++], dtf.format(date));
    }
  }
}
```

Результат:

```
FULL  : Tuesday, September 6, 2011 2:09:56 AM
EEST
LONG  : September 6, 2011 2:09:56 AM EEST
MEDIUM: Sep 6, 2011 2:09:56 AM
SHORT : 9/6/11 2:09 AM
```


2011 © EPAM Systems, RD Dep.

Интернационализация. Example 16

```
package _java._se._03._locale.locale.date;

import java.text.DateFormat;
import java.text.ParseException;
import java.util.Date;
import java.util.Locale;

public class DateFormatParseExample {
  public static void main(String[] args)  throws ParseException {
    Locale localeNOR =  new Locale("no", "NO");

    DateFormat[] dateFormatters =  new DateFormat[] {
    DateFormat.getDateInstance(DateFormat.SHORT, localeNOR),
    DateFormat.getDateInstance(DateFormat.MEDIUM, localeNOR),
    DateFormat.getDateInstance(DateFormat.LONG, localeNOR),
    DateFormat.getDateInstance(DateFormat.FULL, localeNOR),
    };
```

Интернационализация. Example 16

```
    System.out.println("Parsing: ");
    Date date = new Date();
    for(DateFormat df : dateFormatters){
    try{
        String strDate = df.format(date);
        Date parseDate = df.parse(strDate);
        System.out.println(strDate + "|"
                + df.format(parseDate));
    } catch(ParseException e){
        System.out.println(e);
    }

    System.out.println("Leniency: ");
    System.out.println("32.01.08|"
            + dateFormatters[0].parse("32.01.08"));
    }
  }
}
```

Интернационализация. Example 16

Результат:

```
Parsing:
06.09.11|06.09.11
Leniency:
32.01.08|Fri Feb 01 00:00:00 EET 2008
06.sep.2011|06.sep.2011
Leniency:
32.01.08|Fri Feb 01 00:00:00 EET 2008
6. september 2011|6. september 2011
Leniency:
32.01.08|Fri Feb 01 00:00:00 EET 2008
6. september 2011|6. september 2011
Leniency:
32.01.08|Fri Feb 01 00:00:00 EET 2008
```

RESOURCEBUNDLE

ResourceBundle

Управление набором ресурсов производится классом **ResourceBundle**, находящимся в пакете **java.util**.

Основой процесса работы с набором ресурсов является получение набора параметров “ключ-значение” при помощи метода **getBundle()** класса **ResourceBundle**.

2011 © EPAM Systems, RD Dep.

ResourceBundle. Example 17

Ресурс **ResourceExample** может быть представлен либо в виде класса унаследованного от **ListResourceBundle** либо в виде файла, именуемого **ResourceExample.properties**, содержащего пары ключ-значение.

```
package _java._se._03._resourcebundle.resources;

import java.util.ListResourceBundle;

public class ResourcesExample extends
ListResourceBundle{

    public Object[][] getContents() {
        return new Object[][] {
            { "my.key1", "value01" },
            { "my.key2", "value02" },
        };
    }
}
```

ResourceBundle. Example 17

```
package _java._se._03._resourcebundle.resources;

import java.util.ResourceBundle;

public class ResourcesBundle {

     ResourceBundle bundle;

     public ResourcesBundle() {
          bundle = ResourceBundle
                    .getBundle("_java._se._03
                    ._resourcebundle.resources
                    .ResourcesExample");
     }

     public String getValue(String key) {
          return bundle.getString(key);
     }
}
```

ResourceBundle. Example 17

```
package _java._se._03._resourcebundle.resources;

public class UsePropertiesFromClass {

     public static void main(String[] args){
          ResourcesBundle myBundle =  new ResourcesBundle();
          System.out.println(myBundle.getValue( "my.key1"));
     }
}
```

Результат:

```
value01
```

ResourceBundle. Example 18

2011 © EPAM Systems, RD Dep.

ResourceBundle. Example 18

prop.properties

```
my.key1 = "VALUE 1 default"
my.key2 = "VALUE 2 default"
```

prop_en.properties

```
my.key1 = "VALUE 1 en"
my.key2 = "VALUE 2 en"
```

prop_en_US.properties

```
my.key1 = "VALUE 1 en US"
my.key2 = "VALUE 2 en US"
```

prop_ru_RU.properties

```
my.key1 = "\u0417\u041d\u0410\u0427\u0415\u041d\u0418\u0415 1 ru RU"
my.key2 = "\u0417\u041d\u0410\u0427\u0415\u041d\u0418\u0415 2 ru RU"
```

Для корректного отображения нелатинских символов ознакомьтесь с работой утилиты native2ascii.

ResourceBundle. Example 18

```
package _java._se._03._resourcebundle.property.bundle;

import java.util.Locale;
import java.util.ResourceBundle;

public class ResourceProperty {
     ResourceBundle bundle;
     public ResourceProperty(Locale locale) {
          bundle = ResourceBundle
                   .getBundle("_java._se._03
                                       ._resourcebundle
                                       .property
                                       .property
                                       .prop", locale);
     }
     public String getValue(String key) {
          return bundle.getString(key);
     }
}
```


2011 © EPAM Systems, RD Dep.

ResourceBundle. Example 18

```
package _java._se._03._resourcebundle.property.use;

import java.util.Locale;
import _java._se._03._resourcebundle.property.bundle
                     .ResourceProperty;

public class UsePropertiesFromFile {
  public static void main(String[] args){
    ResourceProperty myBundle =  new ResourceProperty(
                 new Locale("en","US"));
    System.out.println(myBundle.getValue( "my.key1"));

    myBundle = new ResourceProperty(new Locale("en","UK"));
    System.out.println(myBundle.getValue( "my.key2"));

    myBundle = new ResourceProperty(new Locale("ru","BY"));
    System.out.println(myBundle.getValue( "my.key1"));
    myBundle = new ResourceProperty(new Locale("ru","RU"));
    System.out.println(myBundle.getValue( "my.key2"));
  }
}
```


2011 © EPAM Systems, RD Dep.

ResourceBundle. Example 18

Результат:

```
"VALUE 1 en US"
"VALUE 2 en"
"VALUE 1 en"
"ЗНАЧЕНИЕ 2 ru
RU"
```

Если вы хотите сохранять набор ресурсов в XML файле, то существует два способа загрузки файлов при помощи класса **java.util.Properties**. Первый способ позволяет читать файлы вида *key=value*, используя метод **load()**. Второй способ дает возможность чтения свойств из XML файла при помощи метода **loadFromXML()**.

Также благодаря наличию нового встроенного класса **Control** в наборе классов **ResourceBundle**, вы можете получать доступ к набору ресурсов, хранящемуся в виде XML файла.

РЕГУЛЯРНЫЕ ВЫРАЖЕНИЯ

2011 © EPAM Systems, RD Dep.

Регулярные выражения

Регулярные выражения (англ. regular expressions) — современная система поиска текстовых фрагментов в электронных документах, основанная на специальной системе записи образцов для поиска.

В стандартную библиотеку Java входит пакет, специально предназначенный для работы с регулярными выражениями – **java.util.regex**.

Эта библиотека может быть использована для выполнения таких задач:

- поиск данных;
- проверка данных;
- выборочное изменение данных;
- выделение фрагментов данных;
- и др.

2011 © EPAM Systems, RD Dep.

Регулярные выражения

Регулярное выражение представляет собой строку-образец (англ. Pattern), состоящую из символов и метасимволов и задающую правило поиска.

Метасимволы:

\	[]
^	-
$	.
?	*
+	()
|	

Регулярные выражения

Символы регулярных выражений

- **x** – неметасимвол
- **** - \ как неметасимвол
- **\t** – символ табуляции (‘\u009’)
- **\n** – символ новой строки (‘\u000A’)
- **\r** – символ возврата каретки (‘\u000D’)
- **\f** – символ перевода страницы (‘\u000C’)

2011 © EPAM Systems, RD Dep.

Регулярные выражения

Классы символов регулярных выражений

- **[abc]** – a, b, или c
- **[^abc]** – символ, исключая a, b или c
- **[a-zA-Z]** – символ от a до z или от A до Z, (диапазон)
- **[a-d[m-p]]** – от a до d или от m до p: [a-dm-p] (объединение)
- **[a-z&&[def]]** – d, e, или f (пересечение)
- **[a-z&&[^bc]]** – от a до z, исключая b и c: [ad-z] (вычитание)
- **[a-z&&[^m-p]]** – от a до z, не включая от m до p: [a-lq-z](вычитание)

Предопределенные классы символов

- **.** – любой символ
- **\d** – цифра [0-9]
- **\D** – не цифра: [^0-9]
- **\s** – [\t\n\x0B\f\r]
- **\S** – [^\s]
- **\w** – [a-zA-Z_0-9]
- **\W** – [^\w]

2011 © EPAM Systems, RD Dep.

Обнаружение совпадения вначале и в конце

- **^a** – якорь для обнаружения сначала строки
- **a$** – якорь на совпадение в конце строки

Логические операторы в регулярных выражениях

- **ab** – за a следует b
- **a|b** – a либо b
- **(a)** – a, для выделения групп

2011 © EPAM Systems, RD Dep.

Квантификаторы

- ***a*?** – *a* один раз или ни разу
- ***a**** – *a* ноль или более раз
- ***a*+** – *a* один или более раз
- ***a*{*n*}** – *a* *n* раз
- ***a*{*n*,}** – *a* *n или более раз*
- ***a*{*n*,*m*}** – *a* от *n* до *m* раз

2011 © EPAM Systems, RD Dep.

Регулярные выражения. Example 19

.+ – будет соответствовать любому тексту

A.+ – любое выражение, которое начинается на букву "A".

^\s+ – один или более пробелов вначале

\s+$ – один или более пробелов вконце

[\d\s()\-]+ – класс символов, в который входят все цифры \\d, все пробельные символы \\s, круглые скобки и дефис. Знак + в конце выражения означает, что любой из этих символов, может встречаться один или более раз

Регулярные выражения. Example 20

[a-zA-Z]{1}[a-zA-Z\d\u002E\u005F]+@([a-zA-Z]+\u002E){1,2}((net)|(com)|(org)) - последовательность вида [a-zA-Z] указывает на множество, {n} говорит о том, что некоторый символ должен встретится n раз, а {n,m} - от n до m раз, символ \d указывает на множество цифр, “\u002E” и “\u005F” - это символы точки и подчеркивания соответственно, знак плюс после некоторой последовательности говорит о том, что она должна встретится один или более раз, “|” - представление логического “или”.

([.[^@\s]]+)@([.[^@\s]]+)\.([a-z]+) – формат e-mail адреса

PATTERN & MATCHER

2011 © EPAM Systems, RD Dep.

Pattern & Matcher

Пакет java.util.regexсостоит всего из трех классов: **Matcher**, **Pattern**, **PatternSyntaxException**.

- **Pattern** - скомпилированное представление регулярного выражения.
- **Matcher** - движок, который производит операцию сравнения (match).
- **PatternSyntaxException** - указывает на синтаксическую ошибку в выражении.

2011 © EPAM Systems, RD Dep.

Pattern & Matcher

Последовательность вызова методов при работе с regexp:

```
Pattern p = Pattern.compile("1*0");

Matcher m = p.matcher("111110");

boolean b = m.matches();
```

Pattern & Matcher

Методы класса Pattern

- **Pattern compile(String regex)** - возвращает **Pattern,** который соответствует шаблону **regex.**
- **Matcher matcher(CharSequence input)** - возвращает **Matcher,** с помощью которого можно находить соответствия в строке **input.**
- **boolean matches(String regex, CharSequence input)** - проверяет на соответствие строки **input** шаблону **regex.**

2011 © EPAM Systems, RD Dep.

Методы класса Pattern

- **String pattern()** — возвращает строку, соответствующую шаблону
- **String [ ] split(CharSequence input)** - разбивает строку **input,** учитывая, что разделителем является шаблон.
- **String[] split(CharSequence input, int limit)** -разбивает строку **input** на не более чем **limit** частей.

2011 © EPAM Systems, RD Dep.

Pattern & Matcher. Example 21

```
package _java._se._03._patternmatcher;

import java.util.regex.Pattern;

public class PatternExample {

public static void main(String[] args) {
        String pattern01 = "<+";
        String pattern02 = "<?";
        String pattern03 = "<*";

        String str = "<body><h1> a<<<b </h1></body>" ;
        String[] result;

        Pattern p = Pattern.compile(pattern01);
        result = p.split(str);
        printTokens(result);

        p = Pattern.compile(pattern02);
        result = p.split(str);
        printTokens(result);
```

Pattern & Matcher. Example 21

```
        p = Pattern.compile(pattern03);
        result = p.split(str);
        printTokens(result);
    }

    public static void printTokens(String[] tokens) {
        for (String str : tokens) {
            if ("".equals(str)) {
                System.out.print("\"\"" + "|");
            } else {
                System.out.print(str + "|");
            }
        }
        System.out.println();
    }
}
```

Результат:

```
""|body>|h1> a|b |/h1>|/body>|
""|""|b|o|d|y|>|""|h|1|>| |a|""|""|""|b| |""|/|h|1|>|""|/|b|o|d|y|>|
""|""|b|o|d|y|>|""|h|1|>| |a|""|b| |""|/|h|1|>|""|/|b|o|d|y|>|
```

Методы класса Matcher

- Начальное состояние объекта типа **Matcher** неопределенно.
- **boolean matches**() — проверяет соответствует ли вся строка шаблону.
- **boolean lookingAt**() — пытается найти последовательность символов, начинающейся с начала строки и соответствующей шаблону.
- **boolean find()** или **boolean find(int start)** - пытается найти последовательность символов соответствующих шаблону в любом месте строки. Параметр **start** указывает на начальную позицию поиска.

2011 © EPAM Systems, RD Dep.

Методы класса Matcher

- **int end**() — возвращает индекс последнего символа подпоследовательности, удовлетворяющей шаблону.
- **reset()** или **reset(Char Sequence input)** - сбрасывает состояние **Matcher'a** в исходное,также устанавливает новую последовательность символов для поиска.
- **replaceAll(String replacement)** - замена всех подпоследовательностей символов, удовлетворяющих шаблону, на заданную строку.

2011 © EPAM Systems, RD Dep.

Выделение групп

Группы в шаблоне *обозначаются скобками* "(" и ")".

Номера групп начинаются *с единицы*. Нулевая группа совпадает со всей найденной подпоследовательностью.

((A)(B(C)))

- **1** ((A)(B(C)))
- **2** (A)
- **3** (B(C))
- **4** (C)

2011 © EPAM Systems, RD Dep.

Методы, для работы с группами

- **String group**() — возвращает всю подпоследовательность, удовлетворяющую шаблону.
- **String group(int group)** — возвращает конкретную группу.
- **int groupCount()** — возвращает количество групп.

2011 © EPAM Systems, RD Dep.

Методы, для работы с группами

- **int end()** — возвращает индекс последнего символа подпоследовательности, удовлетворяющей шаблону.
- **int end(int group)** — возвращает индекс последнего символа указанной группы.
- **int start()** — возвращает индекс первого символа подпоследовательности, удовлетворяющей шаблону.
- **int start(int group)** — возвращает индекс первого символа указанной группы.

2011 © EPAM Systems, RD Dep.

Pattern & Matcher. Example 22

```
package _java._se._03._patternmatcher;

import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class MatcherFindExample {
  public static void main(String args[]) {
    Pattern p = Pattern.compile("Java");
    String candidateString = "I love Java2s. Java2s is about
Java.";
    Matcher matcher = p.matcher(candidateString);
    while (matcher.find()) {
     System.out.println(matcher.group());
     }
  }
}
```

Результат:

```
Java
Java
Java
```

Pattern & Matcher. Example 23

```
package _java._se._03._patternmatcher;

import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class MatcherGroupExample {
  public static void main(String[] args) {
    Matcher m = Pattern.compile("\\w+").matcher("Today is
Sunday");
    while (m.find()){
     System.out.println(m.group());
     }
    int i = 0;
    while(m.find(i)){
     System.out.println(m.group() + " ");
     i++;
     }
  }
}
```

Pattern & Matcher. Example 23

Результат:

```
Today
is
Sunday
Today
oday
day
ay
y
is
is
s
Sunday
Sunday
unday
nday
day
ay
y
```

Pattern & Matcher. Example 24

```
package _java._se._03._patternmatcher;

import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class MatcherLookingAtExample {
  public static void main(String args[]) {
    Pattern p = Pattern.compile("J2SE");

    String candidateString_1 = "J2SE is the only one for me";
    String candidateString_2 = "For me, it's J2SE, or nothing at all";
    String candidateString_3 = "J2SEistheonlyoneforme";

    Matcher matcher = p.matcher(candidateString_1);
    String msg = ":" + candidateString_1 + ": matches?: ";
    System.out.println(msg + matcher.lookingAt());
```

Pattern & Matcher. Example 24

```
    matcher.reset(candidateString_2);
    msg = ":" + candidateString_2 + ": matches?: ";
    System.out.println(msg + matcher.lookingAt());

    matcher.reset(candidateString_3);
    msg = ":" + candidateString_3 + ": matches?: ";
    System.out.println(msg + matcher.lookingAt());
  }
}
```

Результат:

```
:J2SE is the only one for me: matches?: true
:For me, it's J2SE, or nothing at all: matches?:
false
:J2SEistheonlyoneforme: matches?: true
```

КОДИРОВКИ

2011 © EPAM Systems, RD Dep.

Кодировки

При создании строк конструктор **String(byte[] byteArray, String encoding)** создает **Unicode-строку** из массива байтовых **ASCII-кодировок** символов.

В самом простом случае компилятор для получения двубайтовых символов **Unicode** добавит к каждому байту сташий нулевой бит. Получится диапазон **'\u0000'-'\u04ff'** кодировки **Unicode**, соответствующий кодам **Latin1**.

Тексты на кириллице будут выведены неправильно.

2011 © EPAM Systems, RD Dep.

Кодировки

Если на компьютере установлена локаль, то компилятор создаст символы **Unicode** соответственно местной кодовой странице (в **Windows** обычно **CP1251**, в **DOS** - **CP866**, ***nix** - **KOI8-R**).

Если локальная кодировка совпадает с кодировкой выводимых символов, то строка будет верна.

Кодировки

Если исходный кириллический **ASCII-текст** был в одной кодировке, а местная - другая, то **Unicode-строки Java** не будут соответствовать кириллице.

В этих случаях используется этот конструктор, с параметром нужной кодировки. Таким образом, если поток байт выводится и на консоль (**CP866**) и в файл (**CP1251**) **Windows**, то результат будет отличаться.

Кодировки

Правильные символы Unicode-кириллицы получаются, если использовать ту же кодовую таблицу, в которой записан исходный массив байт.

При выводе же строки на консоль, в окно, в файл или при передаче по сети лучше преобразовать строку Java с символами Unicode по правилам выхода в нужное место.

2011 © EPAM Systems, RD Dep.

Кодировки. Example 25

```
package _java._se._03._charset;

import java.io.UnsupportedEncodingException;

public class Charsets {
  public static void main(String[] args)
        throws UnsupportedEncodingException {
    String str = "\u043A\u043E" +
"\u0434\u0438\u0440\u043E\u0432“
            + "\u043A\u0430";
    System.out.println(str);
    byte[] b = str.getBytes("Cp866");
    String str2 = new String(b, "Cp866");
    System.out.println(str2);
  }
}
```

Результат:

```
кодировка
кодировка
```

СПАСИБО ЗА ВНИМАНИЕ!

ВОПРОСЫ?

Java.SE.03

Information Handling

Author: **Ihar Blinou, PhD**
Oracle Certified Java Instructor
Ihar_blinou@epam.com