Java - P31

格式化日期

在Java中，SimpleDateFormat的基类DateFormat提供3解析字符串的方法

public Date parse(String source) throws ParseException

parse text from beginning of given string to produce date

method may not use the entire text of given string

source : String whose beginning should be parsed

ParseException : if beginning of specified string cannot be parsed

注意，由于parse()会抛出ParseException，所以必须显式地捕获

如 try {

Date date = new SimpleDateFormat(pattern).parse(source);

} catch (ParseException e) {

System.out.println("unparseable");

}

其中source从开头开始必须匹配pattern，但无需全部匹配

如 pattern："yyyy-MM-dd"，source："2019-08-18 AD Sun 15:59:28:239"

⟶ Sun Aug 18 00:00:00 MDT 2019

pattern："yyyy-MM-dd G EEE HH:mm:ss:SSS, z XXX"

source："2019-08-18 AD Sun 15:59:28:239, MDT -06:00"

⟶ Sun Aug 18 15:59:28 MDT 2019

pattern："yyyy-MM-dd G EEE HH:mm:ss:SSS, z XXX"

source："2019-08-18"

⟶ 抛出ParseException，输出"unparseable"

休眠

在Java中，java.lang.Thread类提供3 sleep()函数用于休眠

public static void sleep(long millis) throws InterruptedException

cause currently executing thread to sleep (temporarily cease execution)

for specified number of millisecond

subject to precision and accuracy of system timer and scheduler

thread not lose ownership of any monitor

public static void sleep(long millis, int nanos) throws InterruptedException

millis : length of time to sleep in millisecond

nanos : [0, 999999] additional nanosecond to sleep

InterruptedException : if any thread has interrupted current thread

interrupted status of current thread cleaned when exception thrown

与DateFormat.parse()类似，Exception需要被显式地捕获

Java - P32

计时

在Java中，提供了System.currentTimeMillis()函数
返回long类型表示的以毫秒计算的当前时间
如 System.currentTimeMillis() ⟶ 1566571559103
可以用于对语句块运行时间的计时
如
```
long start = System.currentTimeMillis();
<statements>
long end = System.currentTimeMillis();
long run_time = end - start;
```

Calendar

在Java中，提供了比Date类功能强大但实现复杂的Calendar类
Calendar类是抽象类，实际使用时实现特定子类的对象
创建对象的过程对外部是透明的，可以使用getInstance方法创建
如 Calendar c = Calendar.getInstance();
Calendar类中提供了对象字段(field)用于get，set

static int
- YEAR: indicating year
- ERA: indicating era, AD/BC
- MONTH: indicating month, 注意MONTH是0-based的.
- DATE: indicating day of month
- DAY_OF_MONTH: indicating day of month
- DAY_OF_WEEK: indicating day of week
- DAY_OF_YEAR: indicating day number within current year
- WEEK_OF_MONTH: indicating week number within current month
- WEEK_OF_YEAR: indicating week number within current year
- HOUR: indicating hour of morning / afternoon
- HOUR_OF_DAY: indicating hour of day
- AM_PM: indicating whether HOUR before/after noon
- MINUTE: indicating minute within hour
- SECOND: indicating second within minute
- MILLISECOND: indicating millisecond within second

protected long time: currently set time expressed in millisecond
protected boolean isTimeSet: True if value of time valid
protected boolean[] isSet: flags tell if specified calendar field is set

Java - P33

Calendar

在Java中，Calendar类通过get/set存取/修改

public static Calendar getInstance()
get calendar using default time zone and locale
public static Calendar getInstance(TimeZone zone, Locale aLocale)
zone : specified time zone to use
aLocale : specified locale for week data
Calendar returned based on current time in given time zone / FORMAT locale

public final void setTime(Date date)
Set Calendar's time with given Date
public final Date getTime()
return Date object representing this Calendar's time value

public void setTimeInMillis(long mills) from epoch
set Calendar's current time from given new time in UTC millisecond
public long getTimeInMillis()
return this Calendar's time value as UTC milliseconds from epoch

public int get(int field) : return value of given calendar field
lenient mode : all calendar fields normalized
non-lenient mode : all calendar fields validated
throw exception if any calendar field out-of-range
public void set(ind field, int value)
set given calendar field to given value
value not interpreted regardless of leniency mode
public void set(int year, int month, int date)
public void set(int year, int month, int date, int hourOfDay, int minute)
public void set(int year, int month, int date, int hourOfDay, int minute, int second)
year : used to set YEAR calendar field
month : used to set MONTH calendar field
date : used to set DAY_OF_MONTH calendar field
hourOfDay : used to set HOUR_OF_DAY calendar field
minute : used to set MINUTE calendar field
second : used to set SECOND calendar field

Java - P34

正则表达式 (regular expression)，在Java中，正则表达式定义了字符串的模式
可以用于搜索/编辑/处理文本
正则表达式不限于某一种语言，但在不同语言中有细微的差别
Java 正则表达式和 Perl 最为相似
在Java中提供了 java.util.regex 用于支持正则表达式
Pattern 类：正则表达式的编译表示 (compiled representation)
Pattern 类没有公共构造方法用于创建 Pattern 对象
需要先定义一个字符串表示的正则表达式
再调用公共静态编译方法，将字符串作为参数传入
public static Pattern compile(String regex)
如 String reg = "\\W+";
Pattern p = Pattern.compile(reg);

Matcher 类：对输入字符串进行解释和匹配操作的引擎
与Pattern类相似，没有公共构造方法用于创建对象
需要先定义一个用于搜索/匹配的字符串，
再调用公共[redacted]编译方法，将字符串作用参数传入
如 String line = "Hello World!";
Matcher m = p.matcher(line);
public Matcher Pattern.matcher(CharSequence input),

PatternSyntaxException：非强制异常类，用于表示正则表达式中的语法错误
invalid regular expression syntax

可以通过 Pattern [redacted]类提供的静态方法直接匹配 pattern 和 string
public static boolean matches(String regex, CharSequence input)
compile given regular expression and attempt to match given input
more efficient when just used one time and result
另外，Pattern.matches(regex, input);
等价于 Pattern.compile(regex).matcher(input).matches();
如 Pattern.matches("\\W+", "Hello World!"); ⟶ false
Pattern.compile("\\W+").matcher("Hello World!").matches();
⟶ false

Java - P35

正则表达式

反斜杠(\)：注意反斜杠在Java正则表达式中的意义与其他编程语言不同

通常在其他编程语言中，\\表示字面上的反斜杠

即在正则表达式中插入普通的反斜杠"\"，而不含有任何特殊意义

而在Java正则表达式中，\\表示正则表达式的反斜杠，其后字符具有特殊意义

而如Perl语言中，反斜杠本身即具有转义作用

如Perl：字面反斜杠 \\ ，数字字符 \d

Java：字面反斜杠 \\\\ ，数字字符 \\d

由于根据Java Language Specification的要求

Java源代码的字符串中的反斜杠被解释为Unicode转义或其他字符转义

因此必须在字符串字面值中使用\\，表示正则表达式受到保护

不被Java字节码编译器解释

如 当Java解释正则表达式时

字符串字面值 \b 匹配单个退格字符，\\b与单词边界匹配

模式	匹配
\	将下一个字符标记为特殊字符，文本，反向引用或八进制转义符
x	character x
\\	backslash character
\0n	character with octal value 0n , 0≤n≤7
\0nn	character with octal value 0nn, 0≤n≤7
\0mnn	character with octal value 0mnn, 0≤m≤3, 0≤n≤7
\xhh	character with hexadecimal value 0xhh
\xhhhh	character with hexadecimal value 0xhhhh
\x{h...h}	character with hexadecimal value 0xh...h Character.MIN_CODE_POINT ≤ 0xh...h ≤ Character.MAX_CODE_POINT
\t	tab character '\u0009'
\n	newline (line feed) character '\u000A'
\r	carriage-return character '\u000D'
\f	form-feed character '\u000C'
\a	alert (bell) character '\u0007'
\e	escape character '\u001B'
\cx	control character corresponding to x