Программирование на языке Java. Утилиты

Картузов А.В.

Библиотека классов языка включает в себя набор вспомогательных классов, широко используемых в других встроенных пакетах Java. Эти классы расположены в пакетах java.lang и java.util. Они используются для работы с наборов объектов, взаимодействия с системными функциями низкого уровня, для работы с математическими функциями, генерации случайных чисел и манипуляций с датами и временем.

1. Простые оболочки для типов.

Как вы уже знаете, Java использует встроенные примитивные типы данных, например, int и char ради обеспечения высокой производительности. Эти типы данных не принадлежат к классовой иерархии Java. Они передаются методам по значению, передать их по ссылке невозможно. По этой причине для каждого примитивного типа в Java реализован специальный класс.

1.1. Number

Абстрактный класс Number представляет собой интерфейс для работы со всеми стандартными скалярными типами:—long, int, float и double.

У этого класса есть методы доступа к содержимому объекта, которые возвращают (возможно округленное) значение объекта в виде значения каждого из примитивных типов:

- doubleValue() возвращает содержимое объекта в виде значения типа double.
- floatValue() возвращает значение типа float.
- intValue() возвращает значение типа int.
- longValue() возвращает значение типа long.

1.2. Double и Float

Double и Float—подклассы класса Number. В дополнение к четырем методам доступа, объявленным в суперклассе, эти классы содержат несколько сервисных функций, которые облегчают работу со значениями double и float. У каждого из классов есть конструкторы, позволяющие инициализировать объекты значениями типов double и float, кроме того, для удобства пользователя, эти объекты можно инициализировать и объектом String, содержащим текстовое представление вещественного числа. Приведенный ниже пример иллюстрирует создание представителей класса Double с помощью обоих конструкторов.

```
class DoubleDemo {
        public static void main(String args[]) {
                Double d1 = new Double(3.14159);
                Double d2 = new Double("314159E-5");
                System.out.println(d1+"="+d2+"->"+d1.equals(d2));
                }
        }
```

Как вы можете видеть из результата работы этой программы, метод equals возвращает значение true, а это означает, что оба использованных в примере конструктора создают идентичные объекты класса Double.

```
        C:\> java DoubleDemo
        3.14159 = 3.14159 -> true
```

1.3. Бесконечность и NaN

В спецификации IEEE для чисел с вещественной точкой есть два значения типа double, которые трактуются специальным образом: бесконечность и NaN (Not a Number—неопределенность). В классе Double есть тесты для проверки обоих этих условий, причем в двух формах—в виде методов (статических), которым значение double передается в качестве параметра, и в виде методов, проверяющих число, хранящееся в объекте класса Double.

- islnfinite(d) возвращает true, если абсолютное значение указанного числа типа double бесконечно велико.

- islnfinite() возвращает true, если абсолютное значение числа, хранящегося в данном объекте Double, бесконечно велико.
- isNaN(d) возвращает true, если значение указанного числа типа double неопределено.
- isNaN() возвращает true, если значение числа, хранящегося в данном объекте Double, неопределено.

Очередной наш пример создает два объекта Double, один с бесконечным, другой с неопределенным значением.

```
class InfNaN {
        public static void main(String args[]) {
                Double d1 = new Double(1/0.);
                Double d2 = new Double(0/0.);
                System.out.println(d1+":"+d1.isInfinite()+","+d1.isNaN());
                System.out.println(d2+":"+d2.isInfinite()+","+d2.isNaN());
                }
        }
```

Ниже приведен результат работы этой программы:

```
        C:\> java InfNaN
        Infinity: true, false
        NaN: false, true
```

1.4. Integer и Long

Класс Integer—класс-оболочка для чисел типов int, short и byte, а класс Long—соответственно для типа long. Помимо наследуемых методов своего суперкласса Number, классы Integer и Long содержат методы для разбора текстового представления чисел, и наоборот, для представления чисел в виде текстовых строк. Различные варианты этих методов позволяют указывать основание (систему счисления), используемую при преобразовании. Обычно используются двоичная, восьмеричная, десятичная и шестнадцатиричная системы счисления.

- parseInt(String) преобразует текстовое представление целого числа, содержащееся в переменной String, в значение типа int. Если строка не содержит представления целого числа, записанного в допустимом формате, вы получите исключение

NumberFormatException.
- parseInt(String, radix) выполняет ту же работу, что и предыдущий метод, но в отличие от него с помощью второго параметра вы можете указывать основание, отличное от 10.
- toString(int) преобразует переданное в качестве параметра целое число в текстовое представление в десятичной системе.
- toString(int, radix) преобразует переданное в качестве первого параметра целое число в текстовое представление в задаваемой вторым параметром системе счисления.

1.5. Character

Character—простой класс-оболочка типа char. У него есть несколько полезных статических методов, с помощью которых можно выполнять над символом различные проверки и преобразования.

- isLowerCase(char ch) возвращает true, если символ-параметр принадлежит нижнему регистру (имеется в виду не просто диапазон a-z, но и символы нижнего регистра в кодировках, отличных от ISO-Latin-1).
- isUpperCase(char ch) делает то же самое в случае символов верхнего регистра.
- isDigit(char ch) и isSpace(char ch) возвращают true для цифр и пробелов, соответственно.
- toLowerCase(char ch) и toupperCase(char ch) выполняют преобразования символов из верхнего в нижний регистр и обратно.

1.6. Boolean

Класс Boolean—это очень тонкая оболочка вокруг логических значений, она бывает полезна лишь в тех случаях, когда тип boolean требуется передавать по ссылке, а не по значению.

2. Перечисления

В Java для хранения групп однородных данных имеются массивы. Они очень полезны при использовании простых моделей доступа к данным. Перечисления же предлагают более совершенный объектно-ориентированный путь для хранения наборов данных

сходных типов. Перечисления используют свой собственный механизм резервирования памяти, и их размер может увеличиваться динамически. У них есть интерфейсные методы для выполнения итераций и для просмотра. Их можно индексировать чем-нибудь более полезным, нежели простыми целыми значениями.

2.1. Интерфейс Enumeration

Enumeration—простой интерфейс, позволяющий вам обрабатывать элементы любой коллекции объектов. В нем задается два метода. Первый из них—метод hasMoreElements, возвращающий значение типа boolean. Он возвращает значение true, если в перечислении еще остались элементы, и false, если у данного элемента нет следующего. Второй метод—nextElement—возвращает обобщенную ссылку на объект класса Object, которую, прежде чем использовать, нужно преобразовать к реальному типу содержащихся в коллекции объектов.

Ниже приведен пример, в котором используется класс Enum, реализующий перечисление объектов класса Integer, и класс EnumerateDemo, создающий объект типа Enum, выводящий все значения перечисления. Обратите внимание на то, что в объекте Enum не содержится реальных данных, он просто возвращает последовательность создаваемых им объектов Integer.

```
import java.util.Enumeration;
        class Enum implements Enumeration {
                private int count = 0;
                private boolean more = true;
                public boolean hasMoreElements() {
                return more;
                }
        public Object nextElement() {
                count++;
                if (count > 4) more = false;
                return new Integer(count);
                }
        }
class EnumerateDemo {
        public static void main(String args[]) {
                Enumeration enum = new Enum();
```

```
                while (enum.hasMoreElements()) {
                System.out.println(enum.nextElement());
                }
        }
}
```

Вот результат работы этой программы:

```
        C:\> java EnumerateDemo
        1
        2
        3
        4
        5
```

2.2. Vector

Vector—это способный увеличивать число своих элементов массив ссылок на объекты. Внутри себя Vector реализует стратегию динамического расширения, позволяющую минимизировать неиспользуемую память и количество операций по выделению памяти. Объекты можно либо записывать в конец объекта Vector с помощью метода addElement, либо вставлять в указанную индексом позицию методом insertElementAt. Вы можете также записать в Vector массив объектов, для этого нужно воспользоваться методом copyInto. После того, как в Vector записана коллекция объектов, можно найти в ней индивидуальные элементы с помощью методов Contains, indexOf и lastIndexOf. Кроме того методы elementAt, firstElement и lastElement позволяют извлекать объекты из нужного положения в объекте Vector.

2.3. Stack

Stack—подкласс класса Vector, который реализует простой механизм типа "первым вошел—первым вышел" (FIFO). В дополнение к стандартным методам своего родительского класса, Stack предлагает метод push для помещения элемента в вершину стека и pop для извлечения из него верхнего элемента. С помощью метода peek вы можете получить верхний элемент, не удаляя его из стека. Метод empty служит для проверки стека на наличие элементов—он возвращает true, если стек пуст. Метод search ищет заданный элемент в стеке, возвращая количество операция pop, которые

требуются для того чтобы перевести искомый элемент в вершину стека. Если заданный элемент в стеке отсутствует, этот метод возвращает -1.

Ниже приведен пример программы, которая создает стек, заносит в него несколько объектов типа Integer, а затем извлекает их.

```
import java.util.Stack;
import java.util.EmptyStackException;
        class StackDemo {
                static void showpush(Stack st, int a) {
                st.push(new Integer(a));
                System.out.println("push(" + a + ")");
                System.out.println("stack: " + st);
                }
        static void showpop(Stack st) {
                System.out.print("pop -> ");
                Integer a = (Integer) st.pop();
                System.out.println(a);
                System.out.println("stack: " + st);
                }
        public static void main(String args[]) {
                Stack st = new Stack();
                System.out.println("stack: " + st);
                showpush(st, 42);
                showpush(st, 66);
                showpush(st, 99);
                showpop(st);
                showpop(st);
                showpop(st);
                try {
                        showpop(st);
                }
                catch (EmptyStackException e) {
                        System.out.println("empty stack");
                }
        }
}
```

Ниже приведен результат, полученный при запуске этой программы. Обратите внимание на то, что обработчик исключений реагирует на попытку извлечь данные из

пустого стека. Благодаря этому мы можем аккуратно обрабатывать ошибки такого рода.

```
C:\> java StackDemo
stack: []
push(42)
stack: [42]
push(66)
stack: [42, 66]
push(99)
stack: [42, 66, 99]
pop -> 99
stack: [42, 66]
pop -> 66
stack: [42]
pop -> 42
stack: []
pop -> empty stack
```

3. Dictionary

Dictionary (словарь)—абстрактный класс, представляющий собой хранилище информации типа "ключ-значение". Ключ—это имя, по которому осуществляется доступ к значению. Имея ключ и значение, вы можете записать их в словарь методом put(key, value). Для получения значения по заданному ключу служит метод get(key). И ключи, и значения можно получить в форме перечисления (объект Enumeration) методами keys и elements. Метод size возвращает количество пар "ключ-значение", записанных в словаре, метод isEmpty возвращает true, если словарь пуст. Для удаления ключа и связанного с ним значения предусмотрен метод remove(key).

3.1. HashTable

HashTable—это подкласс Dictionary, являющийся конкретной реализацией словаря. Представителя класса HashTable можно использовать для хранения произвольных объектов, причем для индексации в этой коллекции также годятся любые объекты. Наиболее часто HashTable используется для хранения значений объектов, ключами которых служат строки (то есть объекты типа String). В очередном нашем примере в

HashTable хранится информация об этой книге.

```
import java.util.Dictionary;
import java.util.Hashtable;
class HTDemo {
        public static void main(String args[]) {
        Hashtable ht = new Hashtable();
        ht.put("title", "The Java Handbook");
        ht.put("author", "Patrick Naugnton");
        ht.put("email", "naughton@starwave.com");
        ht.put("age", new Integer(30));
        show(ht);
        }
static void show(Dictionary d) {
        System.out.println("Title: " + d.get("title"));
        System.out.println("Author: " + d.get("author"));
        System.out.println("Email: " + d.get("email"));
        System.out.println("Age: " + d.get("age"));
        }
}
```

Результат работы этого примера иллюстрирует тот факт, что метод show, параметром которого является абстрактный тип Dictionary, может извлечь все значения, которые мы занесли в ht внутри метода main.

```
C:\> java HTDemo
Title: The Java Handbook
Author: Patrick Naughton
Email: naughton@starwave.com
Age: 30
```

3.2. Properties

Properties—подкласс HashTable, в который для удобства использования добавлено несколько методов, позволяющих получать значения, которые, возможно, не определены в таблице. В методе getProperty вместе с именем можно указывать значение по умолчанию:

```
        getProperty("имя","значение_по_умолчанию");
```

При этом, если в таблице свойство "имя" отсутствует, метод вернет "значение_по_умолчанию". Кроме того, при создании нового объекта этого класса конструктору в качестве параметра можно передать другой объект Properties, при этом его содержимое будет использоваться в качестве значений по умолчанию для свойств нового объекта. Объект Properties в любой момент можно записать либо считать из потока—объекта Stream (потоки будут обсуждаться в главе 13). Ниже приведен пример, в котором создаются и впоследствии считываются некоторые свойства:

```
import java.util.Properties;
        class PropDemo {
                static Properties prop = new Properties();
                public static void main(String args[]) {
                        prop.put("Title", "put title here");
                        prop.put("Author", "put name here");
                        prop.put("isbn", "isbn not set");
                        Properties book = new Properties(prop);
                        book.put("Title", "The Java Handbook");
                        book.put("Author", "Patrick Naughton");
                        System.out.println("Title: " +
                        book.getProperty("Title"));
                        System.out.println("Author: " +
                        book.getProperty("Author"));
                        System.out.println("isbn: " +
                        book.getProperty("isbn"));
                        System.out.println("ean: " +
                        book.getProperty("ean", "???"));
                        }
                }
```

Здесь мы создали объект prop класса Properties, содержащий три значения по умолчанию для полей Title, Author и isbn. После этого мы создали еще один объект Properties с именем book, в который мы поместили реальные значения для полей Title и Author. В следующих трех строках примера мы вывели результат, возвращенный методом getProperty для всех трех имеющихся ключей. В четвертом вызове getProperty стоял несуществующий ключ "ean". Поскольку этот ключ отсутствовал в объекте book и в объекте по умолчанию prop, метод getProperty выдал нам указанное в его вызове значение по умолчанию, то есть "???":

```
C:\> java PropDemo
Title: The Java Handbook
Author: Patrick Naughton
isbn: isbn not set
ean: ???
```

4. StrinsTokenizer

Обработка текста часто подразумевает разбиение текста на последовательность лексем—слов (tokens). Класс StringTokenizer предназначен для такого разбиения, часто называемого лексическим анализом или сканированием. Для работы StringTokenizer требует входную строку и строку символов-разделителей. По умолчанию в качестве набора разделителей используются обычные символы-разделители: пробел, табуляция, перевод строки и возврат каретки. После того, как объект StringTokenizer создан, для последовательного извлечения лексем из входной строки используется его метод nextToken. Другой метод—hasMoreTokens—возвращает true в том случае, если в строке еще остались неизвлеченные лексемы. StringTokenizer также реализует интерфейс Enumeration, а это значит, что вместо методов hasMoreTokens и nextToken вы можете использовать методы hasMoreElements и nextElement, соответственно.

Ниже приведен пример, в котором для разбора строки вида "ключ=значение" создается и используется объект StringTokenizer. Пары "ключ=значение" разделяются во входной строке двоеточиями.

```
import java.util.StringTokenizer;
        class STDemo {
                static String in = "title=The Java Handbook:" +
                "author=Patrick Naughton:" + "isbn=0-07-882199-1:" +
                "ean=9 780078 821998:" + "email=naughton@starwave. corn";
                public static void main(String args[]) {
                StringTokenizer st = new StringTokenizer(in, "=:");
                while (st.hasMoreTokens()) {
                String key = st.nextToken();
                String val = st.nextToken();
                System.out.println(key + "\t" + val);
                }
        }
```

```
}
```

5. Runtime

Класс Runtime инкапсулирует интерпретатор Java. Вы не можете создать нового представителя этого класса, но можете, вызвав его статический метод, получить ссылку на работающий в данный момент объект Runtime. Обычно апплеты и другие непривелигированные программы не могут вызвать ни один из методов этого класса, не возбудив при этом исключения SecurityException. Одна из простых вещей, которую вы можете проделать с объектом Runtime—его останов, для этого достаточно вызвать метод exit(int code).

5.1. Управление памятью

Хотя Java и представляет собой систему с автоматической сборкой мусора, вы для проверки эффективности своего кода можете захотеть узнать, каков размер "кучи" и как много в ней осталось свободной памяти. Для получения этой информации нужно воспользоваться методами totalMemory и freeMemory.

Замечание:

При необходимости вы можете "вручную" запустить сборщик мусора, вызвав метод gc. Если вы хотите оценить, сколько памяти требуется для работы вашему коду, лучше всего сначала вызвать gc, затем free-Memory, получив тем самым оценку свободной памяти, доступной в системе. Запустив после этого свою программу и вызвав freeMemory внутри нее, вы увидите, сколько памяти использует ваша программа.

5.2. Выполнение других программ

В безопасных средах вы можете использовать Java для выполнения других полновесных процессов в своей многозадачной операционной системе. Несколько форм метода exec позволяют задавать имя программы и ее параметры.

В очередном примере используется специфичный для Windows вызов exec, запускающий процесс notepad—простой текстовый редактор. В качестве параметра редактору передается имя одного из исходных файлов Java. Обратите внимание—exec автоматически преобразует в строке-пути символы "/" в разделители пути в Windows—"\".

```
class ExecDemo {
        public static void main(String args[]) {
                Runtime r = Runtime. getRuntime();
                Process p = null;
                String cmd[] = { "notepad",
                 "/java/src/java/lang/Runtime.java" };
                try {
                        p = r.exec(cmd);
                } catch (Exception e) {
                System.out.println("error executing " + cmd[0]);
                }
        }
}
```

6. System

Класс System содержит любопытную коллекцию глобальных функций и переменных. В большинстве примеров этой книге для операций вывода мы использовали метод System.out.println(). В следующей главе будут детально рассмотрены потоки InputStream и OutputStream.

Метод currentTimeMillis возвращает текущее системное время в виде миллисекунд, прошедших с 1 января 1970 года.

Метод arraycopy можно использовать для быстрого копирования массива любого типа из одного места в памяти в другое. Ниже приведен пример копирования двух массивов с помощью этого метода.

```
class ACDemo {
        static byte a[] = { 65, 66, 67, 68, 69, 70, 71, 72, 73, 74 };
        static byte b[] = { 77, 77, 77, 77, 77, 77, 77, 77, 77, 77 };
        public static void main(
                String args[]) {
                System.out.println("a = " + new String(a, 0));
                System.out.println("b = " + new String(b, 0));
                System.arraycopy(a, 0, b, 0, a.length);
                System.out.println("a = " + new String(a, 0));
                System.out.println("b = " + new String(b, 0));
```

```
                System.arraycopy(a, 0, a, 1, a.length-1);
                System.arraycopy(b, 1, b, 0, b.length-1);
                System.out.println("a = " + new String(a, 0));
                System.out.println("b = " + new String(b, 0));
                }
        }
```

Как вы можете заключить из результата работы этой программы, копирование можно выполнять в любом направлении, используя в качестве источника и приемника один и тот же объект.

```
C:\> java ACDemo
a = ABCDEFGHIJ
b = MMMMMMMMM
a = ABCDEFGHIJ
b = ABCDEFGHIJ
a = AABCDEFGHI
b = BCDEFGHIJJ
```

6.1. Свойства окружения

Исполняющая среда Java предоставляет доступ к переменным окружения через представителя класса Properties (описанного ранее в этой главе), с которым можно работать с помощью метода System.getProperty. Для получения полного списка свойств можно вызвать метод System.getProperties().

7. Date

Класс Date используется для операций с датой и временем. Через него вы можете получить доступ к дате, месяцу, году, дню недели, часам, минутам, секундам. У объектов этого класса—несколько конструкторов. Самый простой—Date()—инициализирует объект текущими датой и временем. Три остальных конструктора предлагают дополнительные возможности задавать начальные значения для нового объекта.

- Date(year, month, date)—устанавливает указанную дату, при этом время устанавливается в 00:00:00 (полночь).
- Date(year, month, date, hours, minutes)—устанавливает указанные дату и время,

секунды устанавливаются в 0.
- Date(year, month, date, hours, minutes, seconds)—наиболее полное задание времени, в объекте устанавливаются указанные дата и время, в том числе и секунды.

7.1. get и set

Класс Date включает в себя набор методов для получения и установки отдельных атрибутов, хранящихся в объекте. Каждая из функций семейства get—getYear, getMonth, getDate, getDay, getHours, getMi-nutes и getSeconds—возвращает целое значение. Каждой из функций семейства set—setYear, setMonth, setDate, setHours, setMinutes и setSeconds—в качестве параметра передается целое значение. Вы также можете получить представление объекта Date в виде значения типа long с помощью метода getTime. Возвращаемое этим методом значение представляет собой число миллисекунд, прошедших после 1 января 1970 года.

7.2. Сравнение

Если у вас есть два объекта типа Date, и вы хотите их сравнить, то можете преобразовать хранящиеся в них даты в значения типа long, и сравнить полученные даты, выраженные в миллисекундах. Класс Date включает в себя три метода, которые можно использовать для прямого сравнения дат:—before, after и equals. Например, вызов

```
        new Date(96, 2, 18).before(new Date(96, 2, 12)
```

возвращает значение true, поскольку 12-й день месяца предшествует 18-му.

7.3. Строки и часовые пояса

Объекты Date можно конвертировать в текстовые строки различных форматов. Прежде всего, обычный метод toString преобразует объект Date в строку, которая выглядит, как "Thu Feb 15 22:42:04 1996". Метод toLocaleString преобразует дату в более короткую строку, выглядящую примерно так: "02/15/96 22:42:04". И, наконец, метод toGMTString возвращает дату в формате среднего времени по Гринвичу: "16 Feb 1996 06:42:04 GMT".

8. Math

Класс Math содержит функции с плавающей точкой, которые используются в геометрии и тригонометрии. Кроме того, в нем есть две константы, используемые в такого рода вычислениях:—E (приблизительно 2.72) и PI (приблизительно 3.14159).

8.1. Тригонометрические функции

Приведенные ниже три функции имеют один параметр типа double, представляющий собой угол в радианах, и возвращают значение соответствующей тригонометрической функции.

- sin(double a) возвращает синус угла a, заданного в радианах.
- cos(double a) возвращает косинус угла a, заданного в радианах.
- tan(double a) возвращает тангенс угла a, заданного в радианах.

Следующие четыре функции возвращают угол в радианах, соответствующий значению, переданному им в качестве параметра.

- asin(double r) возвращает угол, синус которого равен r.
- acos(double r) возвращает угол, косинус которого равен r.
- atan(double r) возвращает угол, тангенс которого равен r.
- atan2(double a, double b) возвращает угол, тангенс которого равен отношению a/b.

8.2. Степенные, показательные и логарифмические функции

- pow(double y, double x) возвращает y, возведенное в степень x. Так, например, pow(2.0, 3.0) равно 8.0.
- exp(double x) возвращает e в степени x.
- log(double x) возвращает натуральный логарифм x.
- sqrt(double x) возвращает квадратный корень x.

8.3. Округление

- ceil(double a) возвращает наименьшее целое число, значение которого больше или равно a.
- floor(double a) возвращает наибольшее целое число, значение которого меньше или равно a.

- rint(double a) возвращает в типе double значение a с отброшенной дробной частью.
- round(float a) возвращает округленное до ближайшего целого значение a.
- round(double a) возвращает округленное до ближайшего длинного целого значение a.

Кроме того, в классе Math имеются полиморфные версии методов для получения модуля, нахождения минимального и максимального значений, работающие с числами типов int, long, float и double:

- abs(a) возвращает модуль (абсолютное значение) a.
- max(a, b) возвращает наибольший из своих аргументов.
- min(a, b) возвращает наименьший из своих аргументов.

8.4. Random

Класс Random—это генератор псевдослучайных чисел. Используемый в нем алгоритм был взят из раздела 3.2.1 "Искусства программирования" Дональда Кнута. Обычно в качестве начального значения используется текущее время, что снижает вероятность получения повторяющихся последовательностей случайных чисел.

Из объекта класса Random можно извлекать 5 типов случайных чисел. Метод nextInt возвращает целое число, равномерно распределенное по всему диапазону этого типа. Аналогично, метод nextLong возвращает случайное число типа long. Методы nextFloat и nextDouble возвращают случайные числа соответственно типов float и double, равномерно распределенные на интервале 0.0..1.0. И, наконец, метод nextGaussian возвращает нормально распределенное случайное число со средним значением 0.0 и дисперсией 1.0.

9. Счет за услуги

В пакете java.util есть еще несколько классов по работе с битами, различными форматами дат и архивами (подкаталог zip). Структуры данных и системные интерфейсы, которые вы изучили в этой главе, окажут вам неоценимую помощь, когда вы начнете писать на Java более сложные программы. В следующих двух главах мы будем знакомиться с потоками ввода-вывода и сетевыми средствами.