



Самарский государственный аэрокосмический университет
имени академика С.П. Королёва

Объектно-ориентированное программирование

Базовые и утилитные классы API JAVA

Занятие 6

© Составление,
А.В. Гаврилов, 2014
А.П. Порфирьев, 2015

Самара
2015

План лекции

- Пакет `java.lang` и его структура
- Класс `Object` и его методы
- Класс `Class`
- Интерфейс `Comparable`
- Классы-обертки примитивных типов
- Класс `Math`
- Классы работы со строками
- Пакет `java.util` и его структура



Пакет java.lang

- Базовые классы
 - `Object`
 - `Class`
 - Обертки примитивных типов
 - `Math`
 - Классы работы со строками
 - Классы управления процессами и потоками
 - Средства рефлексии
 - И т.д.
- Базовые интерфейсы
 - `Cloneable`
 - `Comparable`
 - `Runnable`
 - И т.д.
- Основные исключения
 - `Exception`
 - `Error`
 - `RuntimeException`
 - `SecurityException`
 - И т.д.
- Этот пакет импортируется по умолчанию



Класс Object

- Является суперклассом для **всех** классов (включая массивы)
- Переменная этого типа может ссылаться на **любой** объект (но не на переменную примитивного типа)
- Его методы наследуются **всеми** классами
- Реализует базовые операции с объектами



Методы класса Object

- Получение строкового представления объекта
`String toString()`
- Получение ссылки на описание класса объекта
`final Class getClass()`
- Клонирование объекта (получение копии)
`protected Object clone()`
- Проверка равенства объектов
`boolean equals(Object obj)`
- Получение хэш-кода объекта
`int hashCode()`
- Метод завершения работы с объектом
`protected void finalize()`
- Методы обслуживания блокировок в многопоточных приложениях
`void wait(...)`, `void notify()`, `void notifyAll()`

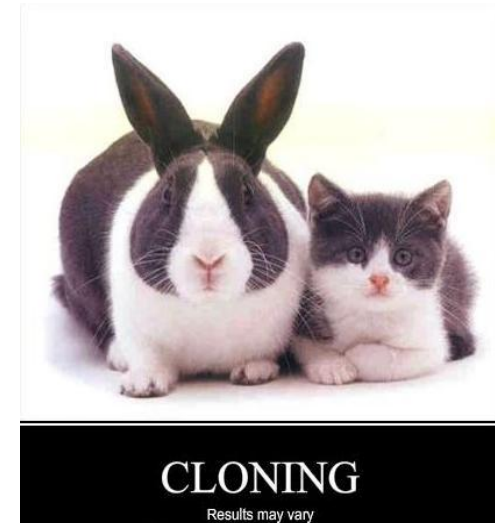


Клонирование объектов

- Считается, что результатом клонирования является копия объекта
- Массивы поддерживают операцию клонирования

```
int[] arrayCopy = (int []) array.clone();
```

- В классе `Object` метод `clone()` является защищенным
- Метод `clone()` реализуется в конкретном классе
- Никто не гарантирует того, что результатом его выполнения будет копия объекта, и даже того, что новый объект будет того же класса
- Однако существует ряд соглашений, регламентирующих реализацию метода `clone()`



Простое клонирование объектов

- Класс должен переопределять метод `clone()`
- Класс должен реализовывать интерфейс-маркер `Cloneable`
- Результат клонирования должен быть получен вызовом `super.clone()`
- Результатом работы метода `Object.clone()` является точная копия объекта

```
public Object clone() {
    Object result = null;
    try {
        result = super.clone();
    } catch (CloneNotSupportedException ex) {
        throw new InternalError();
    }
    return result;
}
```

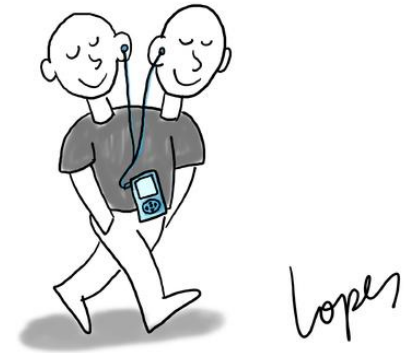


Особенности клонирования

```
int[][] a = {{1, 2, 3}, {4, 5, 6}};  
int[][] b = (int[][]) a.clone();  
System.out.println(a[0][0]);  
b[0][0] = 9;  
System.out.println(a[0][0]);
```

1
9

- В результате клонирования скопировался только сам объект **a**, но не объект, на который он ссылался
- При использовании результатов клонирования необходимо явное приведение типа
 - Начиная с Java5 для массивов можно не выполнять явное приведение типа, но только для массивов



Глубокое клонирование объектов

- Простого клонирования может быть недостаточно, если объект содержит ссылки на агрегированные объекты
- В этом случае после процедуры простого клонирования необходимо создать и их копии тоже

```
public Object clone() {
    Object result = null;
    try {
        result = super.clone();
        result.a = (...) a.clone();
        ...
    } catch (CloneNotSupportedException ex) { }
    return result;
}
```



Равенство объектов

- Простого сравнения ссылок недостаточно для сравнения содержимого объектов
- Для сравнения объектов по их содержимому применяется метод `equals (Object obj)`
- В классе `Object` метод реализован таким образом, что возвращает `true` только при сравнении с самим объектом
- Конкретный класс должен переопределять метод `equals (...)`



Равенство объектов

- Метод `equals(...)` должен проверять эквивалентность объектов с точки зрения бизнес-логики
- Отношение, задаваемое на множестве объектов этим методом, должно обладать следующими свойствами:
 - рефлексивность
 - симметричность
 - транзитивность
 - КОНСИСТЕНТНОСТЬ
 - сравнение с `null` должно приводить к результату `false`



Хэш-код объекта

- Метод `int hashCode ()` предназначен для получения хэш-кода – числа, используемого для быстрого сравнения объектов
- Если объект не изменял свое состояние, то значение хэш-кода не должно изменяться
- Если два объекта эквивалентны (с точки зрения метода `equals ()`), то хэш-коды объектов должны быть одинаковыми
- Если хэш-коды объектов одинаковы, то это еще не значит, что объекты эквивалентны
- Изменение реализации в классе метода `equals ()` влечет за собой изменение реализации метода `hashCode ()`



Класс Class

- Является метаклассом для всех классов Java
- Экземпляры содержат описания классов, загружаемых JVM
- Не имеет доступного конструктора
- Содержит методы для работы с классами и их методами
- Лежит в основе т.н. «рефлексии»



Интерфейс Comparable

- Реализация интерфейса означает введение отношения порядка на множестве объектов класса
- Метод `compareTo()`
 - `a.compareTo(b) < 0` $a < b$
 - `a.compareTo(b) > 0` $a > b$
 - `a.compareTo(b) = 0` $a = b$
- Настоятельно рекомендуется согласовывать работу методов `compareTo()` и `equals()`



Классы-обертки

ПРИМИТИВНЫХ ТИПОВ

- Значения примитивных типов не могут быть непосредственно использованы в контексте, где требуется ссылка
- Ссылочное представление значений примитивных типов является основной задачей т.н. классов-обертки
- Экземпляр такого класса хранит внутри значение примитивного типа и предоставляет доступ к этому значению



Классы-обертки ПРИМИТИВНЫХ ТИПОВ

- Boolean
- Byte
- Character
- Double
- Float
- Integer
- Long
- Number
- Short
- Void



Наполнение классов-оберток

- Константы типов
`Integer.MAX_VALUE`, `Double.NaN`
- Конструкторы: по значению и строке
`Float(float value)`, `Float(String s)`
- Методы получения значения
`Boolean.booleanValue()`, `Float.floatValue()`
- Методы преобразования типов
`Integer.parseInt(String s)`, `Float.byteValue()`
- Методы проверки состояния и вида значения
`compareTo(...)`, `Double.isInfinite()`
- Специальные методы, обусловленные спецификой типа
`Double.longBitsToDouble(...)`, `Integer.toHexString()`



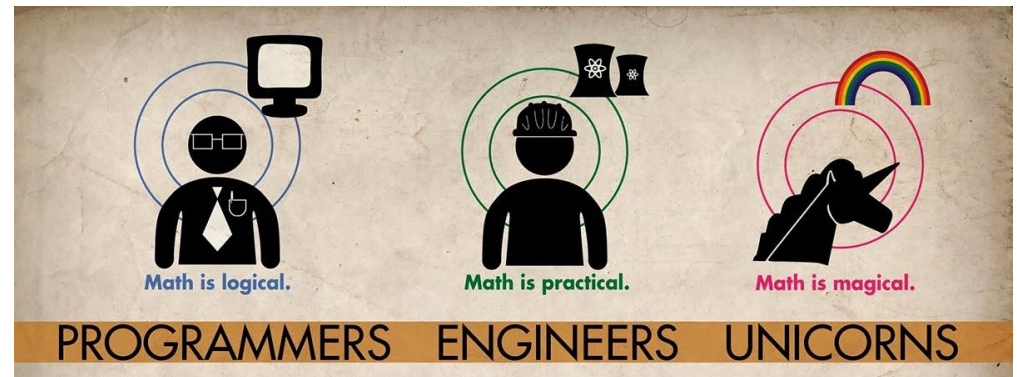
Задачи классов-оберток примитивных типов

- Ссылочное представление значений примитивных типов
- Хранение вспомогательных функций для работы со значениями примитивных типов
- Представление примитивных типов и их значений в механизмах рефлексии



Класс Math

- Предназначен для выполнения простых математических операций
- Не имеет явного конструктора
- Является `final`-классом
- Все методы являются статическими
- Не гарантирует повторяемости результатов на различных платформах (в отличие от класса `StrictMath`)



Наполнение класса Math

- Константы `E` и `PI`
- Функции взятия модуля `abs()`
- Функции максимума и минимума `max()`, `min()`
- Функции округления `round()`, `rint()`
- Функции ближайшего целого `ceil()`, `floor()`
- Тригонометрические функции `sin()`, `cos()`, `tan()`, `asin()`, `acos()`, `atan()`
- Функции перевода `toDegrees()`, `toRadians()`, `atan2()`
- Функции степени `pow()`, `exp()`, `log()`, `sqrt()`
- Случайное значение `random()` (см. класс `java.util.Random`)



Хранение строк

- `byte []`
Массив байт кодов
- `char []`
Массив Unicode-символов
- `String`
Неизменяемая строка
- `StringBuffer`
Изменяемая строка



Работа со строками

Класс `String`

- Значение объекта класса `String` не может быть изменено без порождения нового объекта
- Реализует операции для строки в целом
- Экземпляры этого класса можно создавать без ключевого слова `new`
- Каждый строковый литерал порождает экземпляр `String`
- Значение любого типа может быть приведено к строке



Наполнение класса String

- Строковое представление
`valueOf()`, `copyValueOf()`
- Преобразование типов
`getBytes()`, `getChars()`, `toCharArray()`, `toString()`
- Сравнение
`compareTo()`, `compareToIgnoreCase()`, `contentEquals()`,
`equals()`, `equalsIgnoreCase()`, `intern()`
- Выделение элементов
`charAt()`, `substring()`, `split()`
- Операции над всей строкой
`concat()`, `replace()`, `replaceAll()`, `replaceFirst()`,
`toLowerCase()`, `toUpperCase()`, `trim()`
- Проверка содержимого строки
`endsWith()`, `indexOf()`, `lastIndexOf()`, `length()`,
`matches()`, `regionMatches()`, `startsWith()`



Пакет `java.util.regex`

- Класс `Pattern`

Реализует шаблоны регулярных выражений. Позволяет составлять сложные шаблоны и разделять строки на элементы

- Класс `Matcher`

Реализует поиск элементов, соответствующих шаблону, в строках и проверку строк на соответствие шаблону

У вас есть проблема. Вы решили использовать регулярные выражения чтобы её решить. Теперь у вас две проблемы.



Работа со строками

Класс StringBuffer

- Реализует методы модификации строки без порождения нового объекта
- Реализует операции с элементами строки по отдельности
- Используется по умолчанию при конкатенации строк
- Для хранения строк использует буфер переменного объема



Наполнение класса StringBuffer

- Добавление фрагментов
`append(...)`, `insert(...)`
- Поиск вхождений
`indexOf()`, `lastIndexOf()`
- Извлечение фрагментов
`charAt()`, `getChars()`, `reverse()`, `substring()`
- Модификация строки
`delete()`, `deleteCharAt()`, `replace()`,
`setCharAt()`, `setLength()`
- Состояние буфера
`length()`, `capacity()`, `ensureCapacity()`,
`trimToSize()`



Конкатенация строк

```
System.out.println("a = " + a + ";" );
```

```
System.out.println(  
    (new StringBuffer("a = "))  
    .append(a)  
    .append(";")  
    .toString()  
);
```

- Не стоит злоупотреблять автоматической конкатенацией
- Особенно если для вас критична память и скорость выполнения программы



Пакет `java.util`

- Классы для работы со временем
- Классы для работы с локализацией
- Классы для работы с массивами
- Классы и интерфейсы коллекций
- Классы и интерфейсы для создания многопоточных приложений
- Прочие вспомогательные классы и интерфейсы



java.util.Arrays

Содержит статические методы для работы с массивами

- Представление массива списком
`List asList(Object[] a)`
- Поиск элемента в массиве
`int binarySearch(...[] a, ... key)`
- Сравнение массивов по элементам
`boolean equals(...[] a1, ...[] a2)`
- Заполнение массива элементами
`fill(...[] a, int from, int to, ... val)`
- Сортировка массива
`sort(...[] a, int from, int to)`



Классы работы со временем

- **Date**

Отражает дату и время с точностью до миллисекунд. Не рекомендуется к использованию

- **Calendar** и сопутствующие

Содержит константы и методы для работы с датой и временем с учетом особенностей локализации

- **Timer**

Позволяет создавать задания для более позднего запуска (с использованием потоков инструкций)



Классы для работы с локализацией

- **Locale**

Содержит константы и методы для работы с языками и особенностями регионов

- **TimeZone**

Содержит методы для работы с часовыми поясами

- **SimpleTimeZone**

Реализует **TimeZone** для Григорианского календаря



java.util.Random

- Экземпляр класса является отдельным генератором псевдослучайных чисел (ГПСЧ)
- Различные ГПСЧ позволяют формировать некоррелированные последовательности
- «Основание» имеет размерность 48bit
- Методы получения ПСЧ:
`nextBoolean()`, `nextByte()`,
`nextDouble()`, `nextFloat()`,
`nextInt()`, `nextLong()`,
`nextGaussian()`
- Метод настройки
`setSeed(long seed)`

```
int getRandomNumber()  
{  
    return 4; // chosen by fair dice roll.  
             // guaranteed to be random.  
}
```



Спасибо за внимание!

Дополнительные источники

- Арнолд, К. Язык программирования Java [Текст] / Кен Арнолд, Джеймс Гослинг, Дэвид Холмс. – М. : Издательский дом «Вильямс», 2001. – 624 с.
- Вязовик, Н.А. Программирование на Java. Курс лекций [Текст] / Н.А. Вязовик. – М. : Интернет-университет информационных технологий, 2003. – 592 с.
- Хорстманн, К. Java 2. Библиотека профессионала. Том 1. Основы [Текст] / Кей Хорстманн, Гари Корнелл. – М. : Издательский дом «Вильямс», 2010 г. – 816 с.
- Хорстманн, К. Java 2. Библиотека профессионала. Том 2. Тонкости программирования [Текст] / Кей Хорстманн, Гари Корнелл. – М. : Издательский дом «Вильямс», 2010 г. – 992 с.
- Эккель, Б. Философия Java [Текст] / Брюс Эккель. – СПб. : Питер, 2011. – 640 с.
- JavaSE at a Glance [Электронный ресурс]. – Режим доступа: <http://www.oracle.com/technetwork/java/javase/overview/index.html>, дата доступа: 21.10.2011.
- JavaSE APIs & Documentation [Электронный ресурс]. – Режим доступа: <http://www.oracle.com/technetwork/java/javase/documentation/api-jsp-136079.html>, дата доступа: 21.10.2011.

