Java-классы: взгляд изнутри

Алексей Владыкин

27 октября 2014

1. Reflection API
2. Расположение объекта в памяти
3. Байткод Java


Сэр, вы осознаете, что вы кот?
risovach.ru

- Reflection API — программный интерфейс для получения информации об объектах и классах во время исполнения программы
- Пакет `java.lang.reflect`
- Для каждого типа, в том числе примитивного, можно получить описывающий его экземпляр класса `Class`

Возможности Reflection API

- Получение списка конструкторов, методов и полей класса
- Создание экземпляров класса
- Вызов методов и чтение/запись полей, в том числе закрытых

Как получить Class

- Получение класса по объекту:
```
Class c1 = object.getClass();
```
- Получение класса через литерал:
```
Class c2 = String[].class;
Class c3 = int.class;
```
- Загрузка класса по имени:
```
Class c4 = Class.forName("java.lang.Integer");
```

Как загрузить класс с диска

```
URL jarFileURL =
    Paths.get("library.jar").toUri().toURL();

ClassLoader classLoader =
    new URLClassLoader(new URL[] {jarFileURL});

Class clazz = classLoader.loadClass(
    "ru.csc.java2014.DemoClass");
```

Имя класса

	int[]	Object[]	Foo.Bar
getName()	[I	[Ljava.lang.Object;	Foo$Bar
getCanonicalName()	int[]	java.lang.Object[]	Foo.Bar
getSimpleName()	int[]	Object[]	Bar

Типы классов

- `boolean isPrimitive()`
- `boolean isArray()`
- `boolean isEnum()`
- `boolean isInterface()`
- `boolean isAnnotation()`

Специфика массивов

```
if (clazz.isArray()) {
    System.out.println(
            "Array of " + c.getComponentType());
}
```

Специфика enum

```
if (clazz.isEnum()) {
    System.out.println("Enum of:");
    for (Object e : clazz.getEnumConstants()) {
        System.out.println(e);
    }
}
```

Конструкторы

- Открытые конструкторы:
  `Constructor getConstructor(Class... types)`
  `Constructor[] getConstructors()`

- Все конструкторы:
  `Constructor getDeclaredConstructor(Class... types)`
  `Constructor[] getDeclaredConstructors()`

Вызов конструктора

```
Constructor constructor =
        clazz.getConstructor(String.class);

Object instance =
        constructor.newInstance("Hello World!");
```

Методы

- Открытые методы, в том числе унаследованные:
  `Method getMethod(String name, Class... types)`
  `Method[] getMethods()`

- Все методы, но только из текущего класса:
  `Method getDeclaredMethod(String name, Class... types)`
  `Method[] getDeclaredMethods()`

Вызов метода

```
Method method =
        clazz.getMethod("doSomething", int.class);

Object result =
        method.invoke(instance, 42);
```

Поля

- Открытые поля, в том числе унаследованные:
  `Field getField(String name)`
  `Field[] getFields()`

- Все поля, но только из текущего класса:
  `Field getDeclaredField(String name)`
  `Field[] getDeclaredFields()`

Чтение/запись поля

```
Field field = clazz.getDeclaredField("x");
field.setAccessible(true);

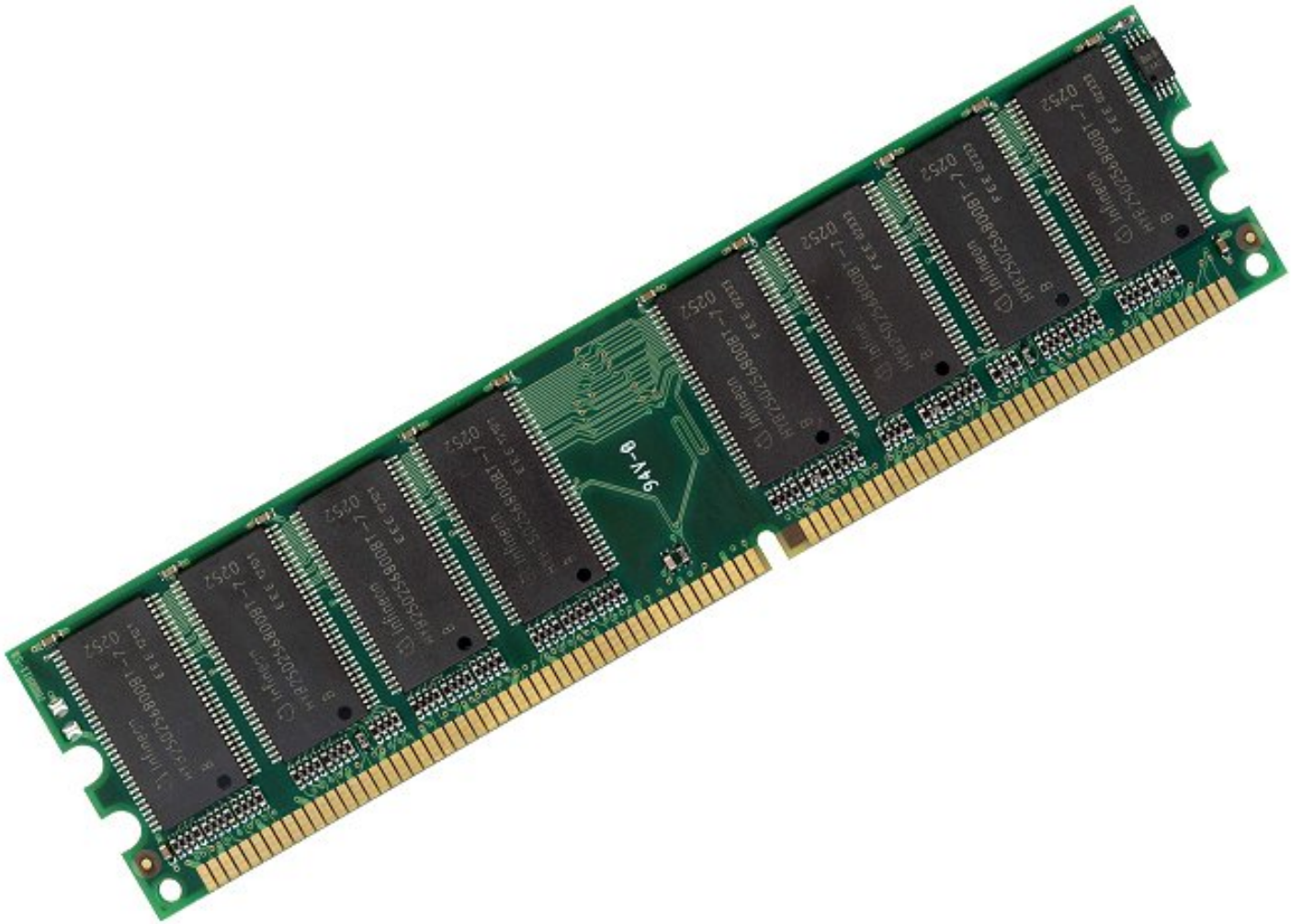
Object value = field.get(instance);

field.set(instance, null);
```

Аннотации

```
Version version =
        clazz.getAnnotation(Version.class);

if (version != null) {
    System.out.println(version.value());
    System.out.println(version.date());
}
```



- Инструмент JOL
- Демо


DJ BYTECODE
MEGAMIX 2013

Структура .class файла

- Заголовок (CAFEBABE, версия формата)
- Constant pool (числа, строки, имена классов, полей и методов)
- Объявление класса (модификаторы, имя класса, имя суперкласса, имена реализуемых интерфейсов)
- Поля класса
- Методы класса
- Атрибуты класса (аннотации, debug info, . . .)
  `javap -v -p ru.csc.java2014.DemoClass`

Имена в байткоде

- Никаких импортов, все имена полные
- Имена классов: `java/lang/String`
- Имена типов:
  - `B, C, D, F, I, J, S, Z`
  - `Ljava/lang/Object;`
  - `[[I`
- Имена методов:
  - `<init> ()V`
  - `<clinit> ()V`
  - `equals (Ljava/lang/Object;)Z`
  - `toString ()Ljava/lang/String;`
  - `sort ([III)V`
- Имена параметров и локальных переменных отсутствуют

Исполняемый код

- Состоит из простых инструкций (около 200)
- Работает в рамках одного фрейма стека вызовов
- Имеет локальный стек заданного размера, где и выполняет все вычисления
- Может обращаться к полям и методам объектов, а также к своим аргументам и локальным переменным

Стековая арифметика

- $2 + 3 \cdot 4$
- $2\ 3\ 4\ \cdot\ +$
- ```
  iconst_2
  iconst_3
  iconst_4
  imul
  iadd
  ```
  ```

Основные инструкции

- Значения в стеке и локальных переменных:
  const, ldc*, *load*, *store*
- Арифметика:
  *mul, *div, *add, *sub
- Работа с объектами:
  new, getfield, putfield, getstatic, putstatic
- Вызовы методов:
  invokestatic, invokevirtual, invokespecial, *return
- Проверки и переходы:
  cmp, if, goto*

- Библиотека ASM
- Демо

Зачем работать с байткодом?

- Альтернативные языки для JVM
- Статический и динамический анализ кода
- Enterprise фреймворки
- Библиотеки для mock'ов

Что сегодня узнали

- Что такое Reflection API и какие возможности он предоставляет
- Как узнать, сколько места занимает объект и как расположены его поля в памяти
- Как устроен байткод Java, и что это дает