

ORACLE®



ORACLE[®]

JavaME: введение

Борис Кварцхава
boris.kvartskhava@oracle.com

Что такое JavaME?

- JCP: Java Community Process
 - 2 комитета:
 - Standard/Enterprise
 - Micro Edition
- JSR: Java Specification Request («запрос на спецификацию Java»)
 - Спецификация Java
 - Их (JSR-ов) — много, пример:
 - JSR 118: MIDP 2.0/2.1
 - JSR 139: CLDC 1.1
 - Спецификация JSR содержит
 - спецификации API в виде .class файлов, с которыми можно компилировать приложения
 - Требования (например, реализация других JSR):
Например, на устройстве не может быть реализован MMAPI (JSR135), если не реализован CLDC или CDC

JSR состоит из

- Спецификации
- Набора тестов (ТСК=Technology Compatibility Kit)
- Эталонной реализации:
 - Выполняет все требования спецификации
 - «Проходит» весь набор тестов

Как скачать спецификацию конкретного JSR?

Просто! Идём

- <http://jcp.org/en/home/index> и вбиваем слева номер.

Либо,

<http://jcp.org/en/jsr/summary?id=XXX>

Например, MIDP: <http://jcp.org/en/jsr/summary?id=118>

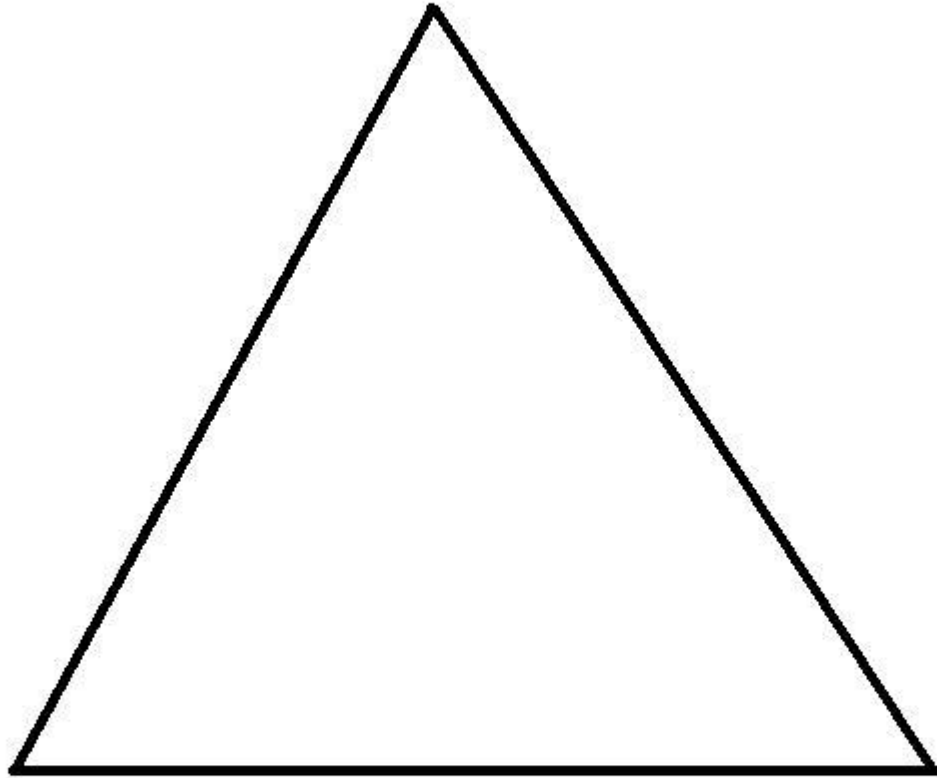
Пример1 — скачиваем спецификацию
MIDP 2.1

<http://jcp.org/en/jsr/summary?id=118>

Пример2 — скачиваем спецификацию
CLDC 1.1

<http://jcp.org/en/jsr/summary?id=139>

Спецификация



Тесты на соответствие
спецификации (ТСК)

Эталонная
реализация
спецификации

Немного в сторону - что такое сигнатуры API?

Что такое сигнатуры в нашем понимании:

Рассмотрим класс `java.lang.Object` (назовём его `O`) в спецификациях CLDC 1.1 и CDC 1.1:

- Обращаем внимание, что все методы `O` в CLDC 1.1 также присутствуют в CDC 1.1
- Есть методы у `O` в CDC 1.1, которых нет в CLDC 1.1
- Нет такого метода у `O` в CLDC 1.1, который имел бы другой возвращаемый тип.
- Спецификация любого метода `O` в CLDC 1.1 и CDC 1.1 - не противоречат.

Будем считать, что сигнатура метода, поля, Класса или интерфейса — строка, по которой этот элемент API однозначно идентифицируется.

Примеры:

- «public final void java.lang.Object.wait(long,int)»
(есть и в CLDC 1.1 и в CDC 1.1)
- «public class java.lang.Object»
- «public void java.lang.Runnable.run()»
- «public interface java.lang.Runnable»
- «public class java.io.File extends java.lang.Object implements java.io.Serializable, java.lang.Comparable»

Любое Java API можно рассматривать как множество сигнатур (строго сравнивать нельзя, есть нюансы!).

Два вида линковки классов:

Пример: если ваш код вызвал метод класса которого в runtime у него нет.

У платформы «жадная линковка»:

- Платформа не сможет загрузить ваш класс.

У платформы «ленивая линковка»:

- Ваш класс загрузится и запустится

- Проверка наличия вызываемых методов будет происходить по мере вызовов.

Попытка вызвать несуществующий метод и тп

Приведёт к возбуждению LinkageError-s.

JavaME:

Любая JavaME реализация

(другими словами, любая «Java на телефоне, телевизоре и тп.») всегда содержит:

- Конфигурацию (Configuration) — ровно одну!
(пример: CLDC или CDC)
- Профиль (ноль или один).
(пример: MIDP или PBP)
- Опциональные (необязательные) пакеты.
0 или более шт.
Примеры: JSR 135(MMAPI), JSR 211(CHAPI).

Зачем это нужно?

MMAPI (JSR135): расширенная поддержка видео/аудио

Есть телефоны, на которых «железо» не обладает
Необходимыми свойствами (скорость, наличие
видеопроцессора и тп).

На таких устройствах MMAPi реализовывать
бессмысленно.

Конфигурация (ровно одна на устройстве!):

Либо

- Connected Limited Device Configuration (CLDC) 1.1

- Описывает виртуальную машину, язык и минимальные требования к устройству, на котором можно её реализовать:

- 162 килобайта памяти минимум

- 16- или 32-битный процессор

- наличие какого-либо сетевого (например, COM) интерфейса.

Либо

- Connected Device Configuration (CDC)1.1:

- Файловая система

- Датаграммные сокетты.

- 4 мегабайта памяти

Имеет место следующее:

- CLDC 1.1 - подмножество CDC 1.1 (сигнатуры)
- CDC 1.1 — подмножество JavaSE specification 1.4.2

, другими словами, предполагается следующее:

- Программа, написанная по спецификации CLDC 1.1,
Будет компилироваться с классами CDC 1.1
- Программа, написанная по спецификации CLDC 1.1
Будет работать на и на CLDC 1.1, и на CDC 1.1
- Неверно, что программа написанная по спец. CDC 1.1
будет работать на CLDC 1.1.

Конфигурация описывает целые классы устройств, на которые её можно ставить.

Профиль - «уточнение этого класса».

Спецификация профиля JavaME:

- Включает требования к конфигурации платформы.
- Уточняет спецификацию конфигурации. Может, например, ослабить некоторые требования.
- Добавляет свои собственные требования к характеристикам устройств.
- Пример:
 - MIDP 1.0 требует только CLDC 1.0
 - MIDP 2.0 требует CLDC 1.0 или CLDC 1.1

Версии .class (класс) файлов. Имеет место быть следующее:

- Все java программы компилируются в .class файлы
- Виртуальная машина исполняет .class файлы
- В класс файле находится информация о его версии:

```
ClassFile {  
    u4 magic;  
    u2 minor_version;  
    u2 major_version;  
    ... }
```

- Каждая виртуальная машина поддерживает конкретный набор версий класс файлов.
- CLDC 1.1 и CDC 1.1 есть подмножества 1.4.2, Соответственно поддерживают версии ≤ 48.0 .
Если скомпилировать код с помощью JDK 1.5, то версия класс файлов будет 49.0.
Компилируйте с помощью **javac -target 1.4 Hello.java**

Опциональный пакет (0 или более):

- Содержат (как правило, минимальные) требования к реализации, на которую их можно поставить.
- Иногда уточняет требования конфигурации и профиля
Пример 1: Если JSR 75 FileConnection реализован, то
 - на устройстве есть файловая система
 - конфигурация - это «супер»множество CLDC 1.0 (например CLDC 1.1 или CDC 1.0 или 1.1)

Пример 2:

- MMAPI спецификация требует, чтобы конфигурация была CLDC 1.0 (или любая, которая её содержит)
- Требуется поддержка как минимум одного протокола и формата медиа (например, .mp3)

Профили (либо есть ровно один, либо нет)

Основные профили JavaME:

- MIDP 2.0/2.1 (JSR118): мобильные телефоны
 - 256+ kb памяти, кроме того, что требует CLDC
 - ЖК дисплей (можно сенсорный),
 - Характеристики: минимум 96x54, 1bpp (не менее чем монохромный),
Приблизительно квадратные пиксели.
 - Телефонная (ITU-T) клавиатура.
 - Поддерживается http
 - Частью спецификации является OTA/Provisioning:
(вольный перевод - «доставка приложений по воздуху»)
 - 128k памяти для работающего мидлета.

MIDP (MIDP 2.0/2.1=JSR 118, MIDP 3.0=JSR 271)

Мобильные телефоны

- Включает вспомогательные классы для:
 - работы с дисплеем
(классы в пакете `javax.microedition.lcdui`)
 - АВВ (`javax.microedition.media`).
Классы АВВ включены также и в JSR 135 MMAPI,
Так что АВВ можно называть
«пересечением» MIDP и MMAPI.
- RMS-помощники:
По-сути, Record Management System - это маленькое хранилище данных (`persistent storage`),
Выживет выключение телефона.
Размер 8192+ байтов.

- `javax.microedition.pki` -

Классы — помощники для установки безопасных соединений.

- `javax.microedition.midlet` — модель приложения для Вашей программы:

Вы написали MIDP приложение? Отлично, значит точка входа в него — написанный вами класс - наследник `javax.microedition.midlet.MIDlet`

Заметьте, «main» (`public static void main(String[])`)

модель в MIDP НЕ ПОДДЕРЖИВАЕТСЯ!

- Весь I/O (сеть, файловая система и тд)

`javax.microedition.io`.

Обратите внимание, класса `java.lang.Object` в спецификации MIDP — НЕТ! Он находится в спецификации CLDC.

Personal Basis Profile/Personal Profile (PBP/PP):

- Принтеры
- Наладонники
- Цифровые телеприёмники
- Blu-ray проигрыватели

Включает:

- Классы работы с графикой
- Своя модель приложения: `javax.microedition.xlet`
(все приложения реализуют интерфейс `javax.microedition.xlet.Xlet`)
- Сеть (`java.net.`, `javax.microedition.io.`, некоторая функциональность продублирована)
- Безопасность JavaSE (`java.security`)
- Есть РЕФЛЕКСИЯ! (`java.lang.reflect`)
- Работа с ZIP и JAR архивами

Устройства с MIDP на борту:

Сходите на:

- Платформы Sony Ericsson:

http://en.wikipedia.org/wiki/Sony_Ericsson_Java_Platform

http://dwgs3.sonyericsson.com/cws/download/1/724/181/1268293202/dw-101478-dg_java_me_s60_r3a.pdf

- Платформы Nokia: Series40 и Series 60:

http://wiki.forum.nokia.com/index.php/Java_ME_API_support_on_Nokia_devices

- Узнать, какие JSR-ы реализованы на каких устройствах

<http://devices.j2mepolish.org/interactivedb/searchdevices.faces>

Модели приложений

- **CLDC/CDC** — есть одна, неизвестно какая.

Причина — обычно спецификации не используются без профилей, в которых есть гарантированная своя

Обычно предполагается, что поддерживается точка

Входа как метод `public static void main(String[])`

- MIDP: `javax.microedition.lcdui.MIDlet` (класс)

- PBP: `javax.microedition.xlet.Xlet` (интерфейс)

- PP:

`javax.microedition.xlet.Xlet`

`java.applet.Applet` (класс)

- JavaTV («ставится» совместно с PBP):

`javax.tv.xlet.Xlet`

- JavaSE: «main», `java.applet.Applet`

AMS: Application Management System

По сути, это совокупность средств, предоставляемых платформой, описывающих механизм доставки и запуска программ (приложений) на устройство. Термин можно употреблять вне контекста Java.

Пример:

- Microsoft Windows: можно объединить группу операций:
 - «Установка/Удаление программ» в панели управления
 - Возможность запуска установленных приложенийи назвать объединение «Windows AMS»
- Linux: RPM + механизм запуска приложений.

- Обобщение понятия **AMS** (пример: Microsoft Office)
- Пакет приложения (Application Package (**AP**)): содержит несколько приложений плюс Дополнительные метаданные (ресурсы, библиотеки и тп)
 - Приложение (Application(**A**)): содержит «точку входа». То, что можно запустить.
- **AMS** умеет:
- Устанавливать («инсталлировать») **AP**
 - Удалять («деинсталлировать») **AP**
 - Найти **A** в **AP** и запустить
- Некоторые **AMS** умеют идентифицировать **AP** на устройстве и соответственно обновлять через установку **AP**.

MIDP AMS — введение

- .jar (java application archive): это специального вида .zip архив. Главное отличие — в нём всегда находится Файл META-INF\MANIFEST.MF. Будем называть его просто **MF**.
- **MF**: это текстовый файл с некоторыми ограничениями. (спецификация <http://download.oracle.com/javase/1.3/docs/guide/jar/jar.html>)
- **MF** надо написать самому, в текстовом редакторе
- .jar можно запаковать утилитой <JDK>/bin/jar.exe

MIDP AMS=OTA/Provisioning

1. MIDP AP: MIDlet suite (Мидлет сюита):

Состоит из двух файлов:

- (обязательный):

.jar файл () с класс файлами MIDlet-ов
и вспомогательные класс файлы.

По сути, .jar и содержит AP.

.jar — это такой «zip» архив
(есть маленькие различия)

MF должен содержать специальные атрибуты!

- (необязательный): .jad (java application descriptor):

Вспомогательный текстовый файл с описанием
и ссылкой на JAR.

MIDP A=MIDlet

- Каждый MIDlet в MIDP идентифицируется через MIDlet Suite ID (поговорим отдельно) и имя класса MIDlet-a.
- В runtime платформа создаёт экземпляр нашего класса, это и есть наш MIDlet (наше приложение)

Жизненный цикл MIDlet-a:

- Paused

Вход: только что создан, AMS вызвал `pauseApp()`, сам MIDlet известил AMS вызвав `NotifyPaused()`, `startApp()`, вызванный AMS, бросил `MIDletStateChangeException (MSCE)`

- Active

Вход: во входе в тело метода `startApp()`

- Destroyed

Вход: Метод `destroyApp(boolean)` вызванный AMS, успешно завершился (не учитываем случая когда `boolean=true` и `destroyApp()` бросил MSCE)

- MIDlet известил AMS о том, что он «умер», вызвав `notifyDestroyed()`

- Любое исключение из `pauseApp()` приводит к Destroyed!



ВОПРОСЫ

ORACLE®