JNode, an operating system based on Java

JNode.org

Fabien DUMINY

Contents

Introduction

- Simple to use & install operating system for personal use: written for and in Java
- Targets:
  - Modern devices
  - Desktop
  - Small servers
- Only actively developed pure Java OS in the open source world
  - 5 active developers
  - Release 0.2.8 : 22K downloads

History

- Original idea started in 1995
- First attempt: JBS (Java Bootable System)
  - Contained C code, did not work at all
- Second attempt: JBS2
  - Still did not work well, but was better
- Then: JNode
  - No C code anymore, Classpath class libraries (and now OpenJDK and IcedTea)
- Went public in May '03

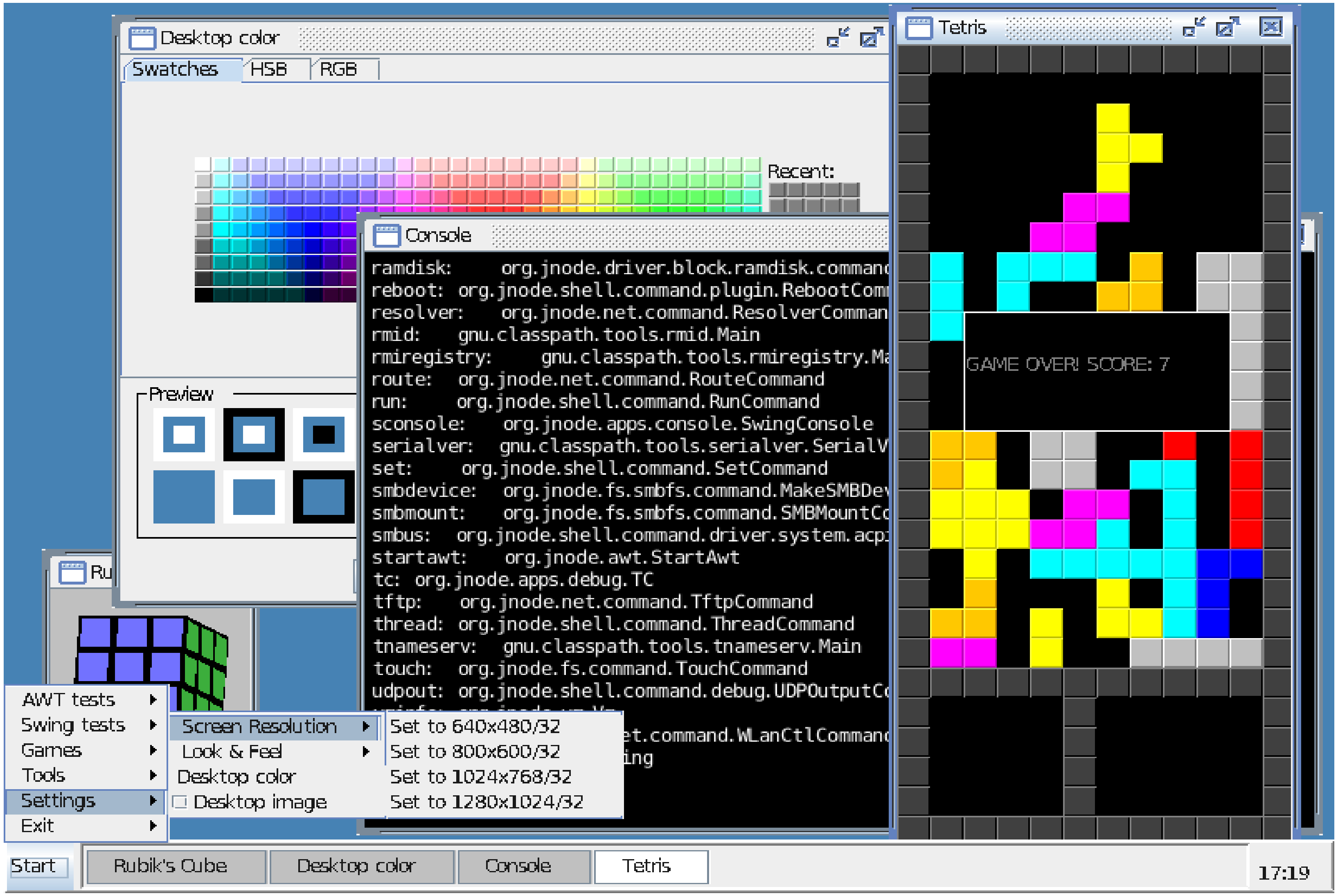
Characteristics

- All Java, minimal assembler, no C
- All java build system (almost)
- Extensible architecture
- Single flat memory address space, no virtual memory
- JVM written in Java
- All Java code is compiled on the fly, no interpreter
- Security is always on
- LGPL license

Status (1)

- Release 0.2.8
  - Support for isolate
  - Filesystems EXT2, FAT, NTFS, ISO9660, HFS+
  - Java 6 support
  - Simple heap managers & GC
    - MMTk based heap manager & GC in development
  - IA32 & AMD64 platform support
  - Bjorne shell
  - Graphical console and console in graphical mode

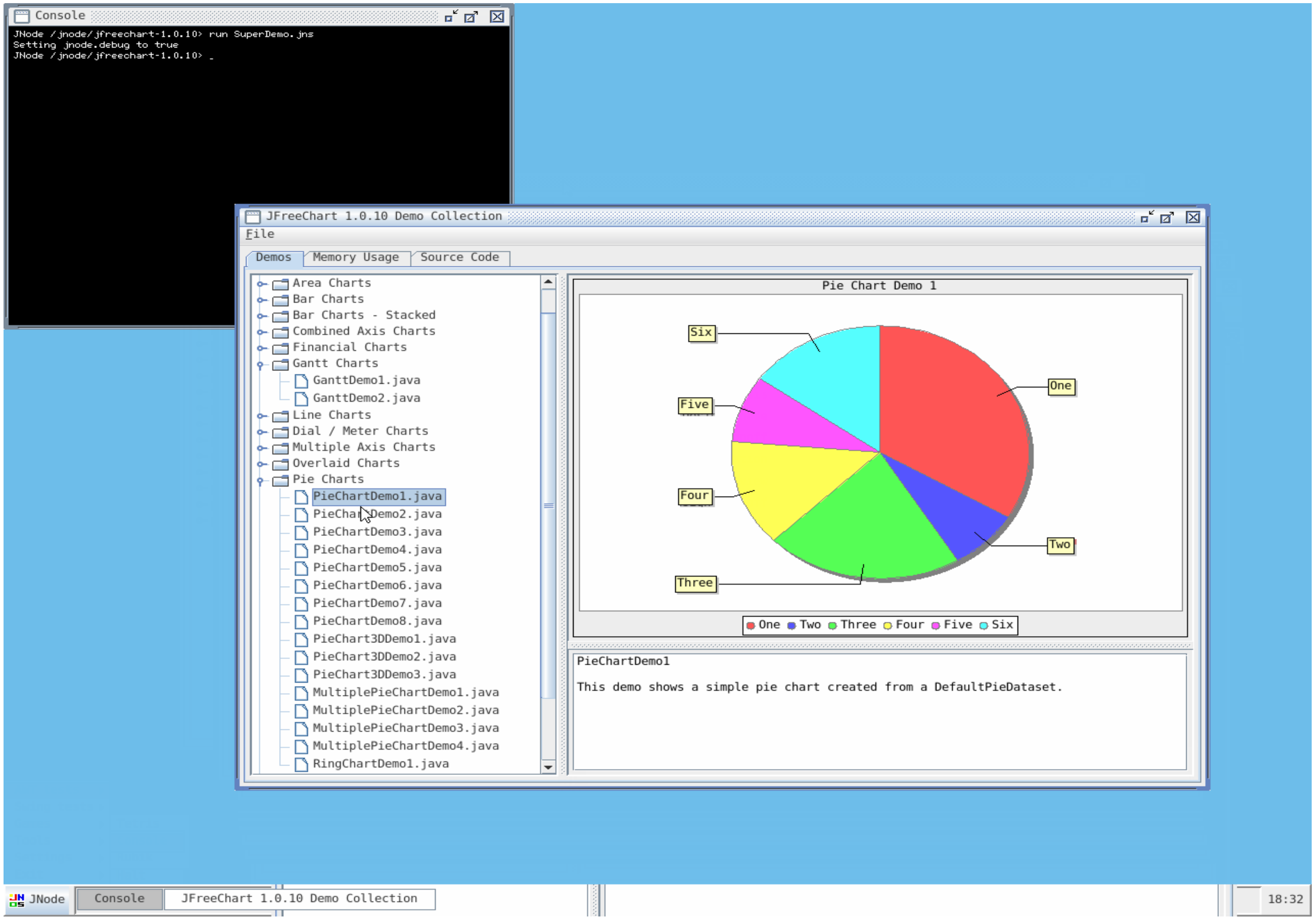
Status (2)



```
ramdisk:     org.jnode.driver.block.ramdisk.comman
reboot:  org.jnode.shell.command.plugin.RebootCom
resolver:    org.jnode.net.command.ResolverComman
rmid:    gnu.classpath.tools.rmid.Main
rmiregistry:     gnu.classpath.tools.rmiregistry.Ma
route:   org.jnode.net.command.RouteCommand
run:     org.jnode.shell.command.RunCommand
sconsole:    org.jnode.apps.console.SwingConsole
serialver:   gnu.classpath.tools.serialver.SerialV
set:     org.jnode.shell.command.SetCommand
smbdevice:   org.jnode.fs.smbfs.command.MakeSMBDe
smbmount:    org.jnode.fs.smbfs.command.SMBMountC
smbus:   org.jnode.shell.command.driver.system.acp
startawt:    org.jnode.awt.StartAwt
tc:  org.jnode.apps.debug.TC
tftp:    org.jnode.net.command.TftpCommand
thread:  org.jnode.shell.command.ThreadCommand
tnameserv:   gnu.classpath.tools.tnameserv.Main
touch:   org.jnode.fs.command.TouchCommand
udpout:  org.jnode.shell.command.debug.UDPOutputC
                                      et.command.WLanCtlComman
                                      ing
```

Playing Tetris on JNode

Status (3)



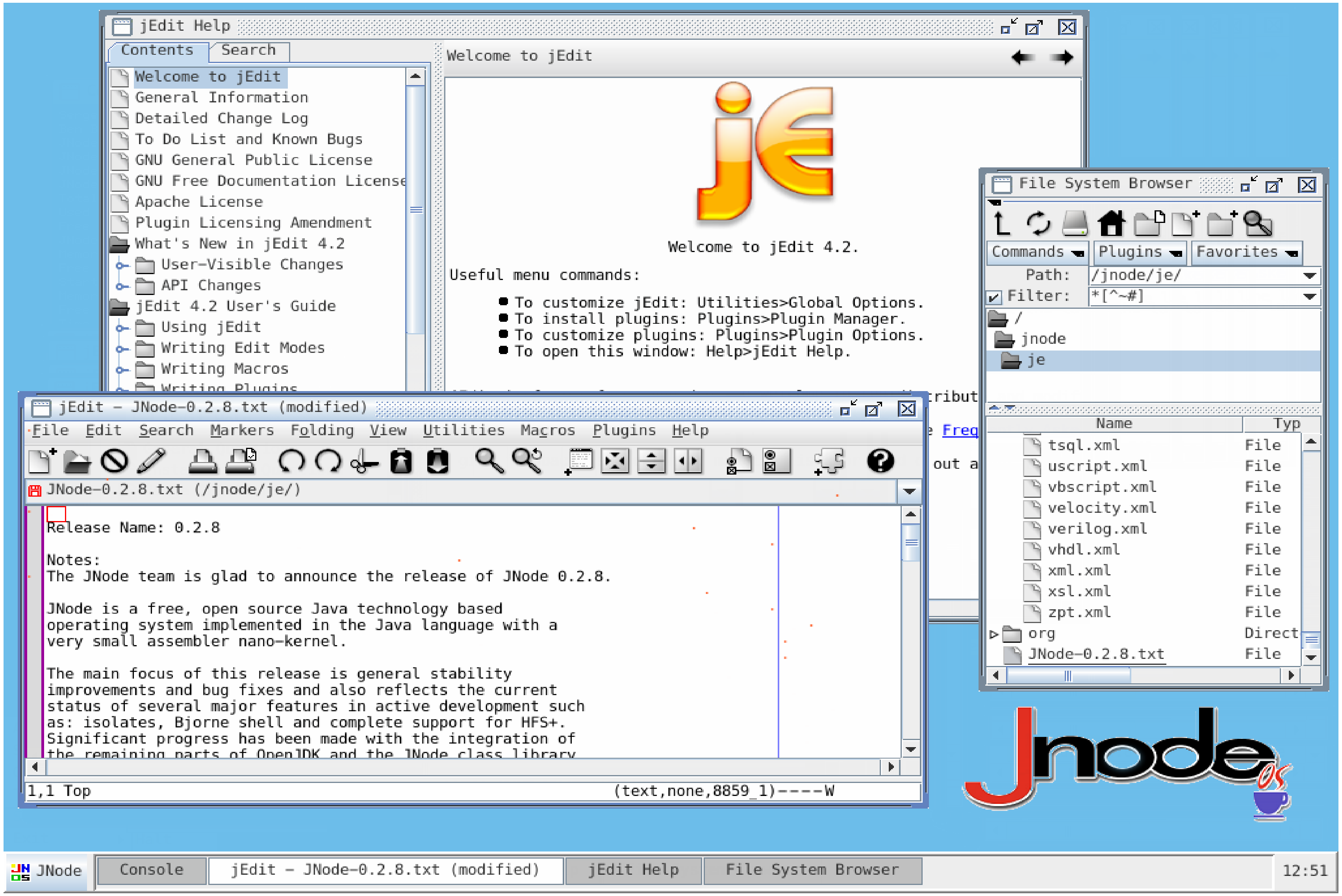
```
JNode /jnode/jfreechart-1.0.10> run SuperDemo.jns
Setting jnode.debug to true
JNode /jnode/jfreechart-1.0.10> _
```

PieChartDemo1

This demo shows a simple pie chart created from a DefaultPieDataset.

JFreeChart on JNode

Status (4)



jEdit Help

Contents | Search

Welcome to jEdit
General Information
Detailed Change Log
To Do List and Known Bugs
GNU General Public License
GNU Free Documentation License
Apache License
Plugin Licensing Amendment
What's New in jEdit 4.2
User-Visible Changes
API Changes
jEdit 4.2 User's Guide
Using jEdit
Writing Edit Modes
Writing Macros
Writing Plugins

Welcome to jEdit

Welcome to jEdit 4.2.

Useful menu commands:

- To customize jEdit: Utilities>Global Options.
- To install plugins: Plugins>Plugin Manager.
- To customize plugins: Plugins>Plugin Options.
- To open this window: Help>jEdit Help.

jEdit - JNode-0.2.8.txt (modified)

File Edit Search Markers Folding View Utilities Macros Plugins Help

JNode-0.2.8.txt (/jnode/je/)

```
Release Name: 0.2.8

Notes:
The JNode team is glad to announce the release of JNode 0.2.8.

JNode is a free, open source Java technology based
operating system implemented in the Java language with a
very small assembler nano-kernel.

The main focus of this release is general stability
improvements and bug fixes and also reflects the current
status of several major features in active development such
as: isolates, Bjorne shell and complete support for HFS+.
Significant progress has been made with the integration of
the remaining parts of OpenJDK and the JNode class library
```

1,1 Top (text,none,8859_1)----W

File System Browser

Commands | Plugins | Favorites

Path: /jnode/je/
Filter: *[^~#]

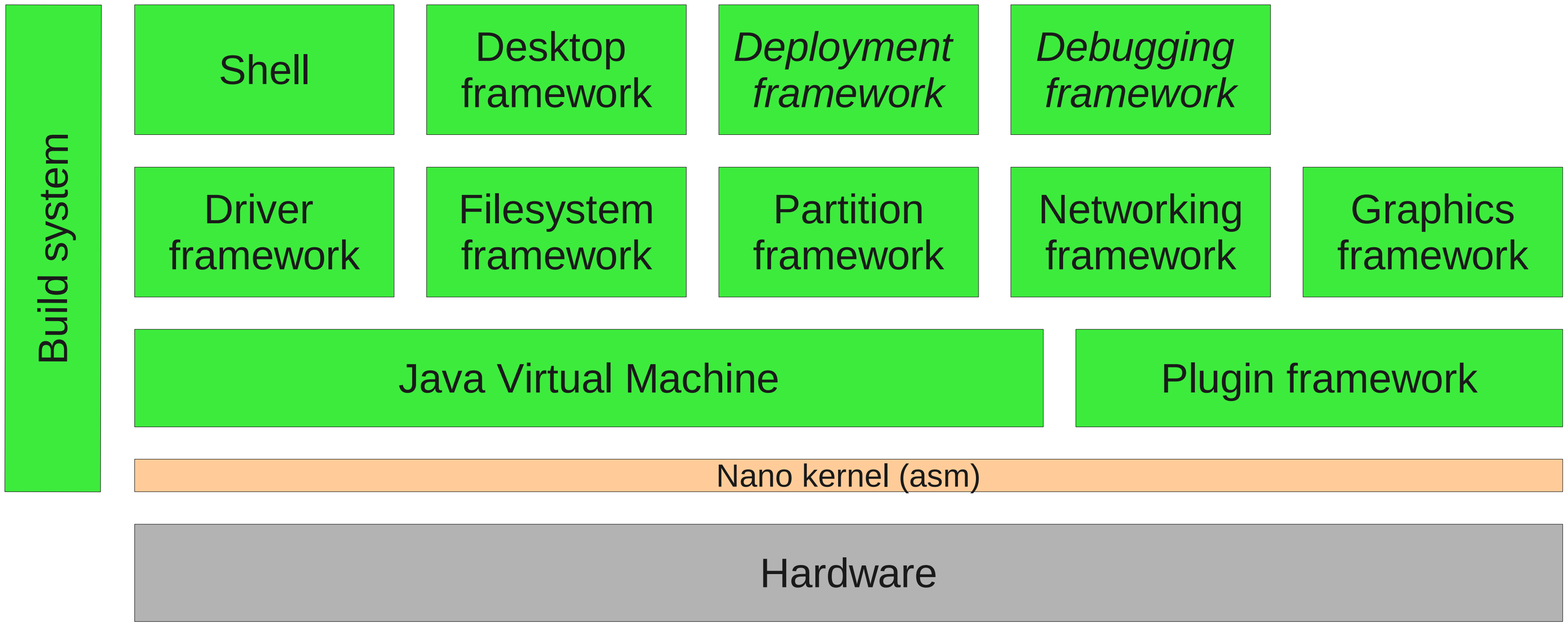
/
jnode
je

Name	Typ
tsql.xml	File
uscript.xml	File
vbscript.xml	File
velocity.xml	File
verilog.xml	File
vhdl.xml	File
xml.xml	File
xsl.xml	File
zpt.xml	File
org	Direct
JNode-0.2.8.txt	File

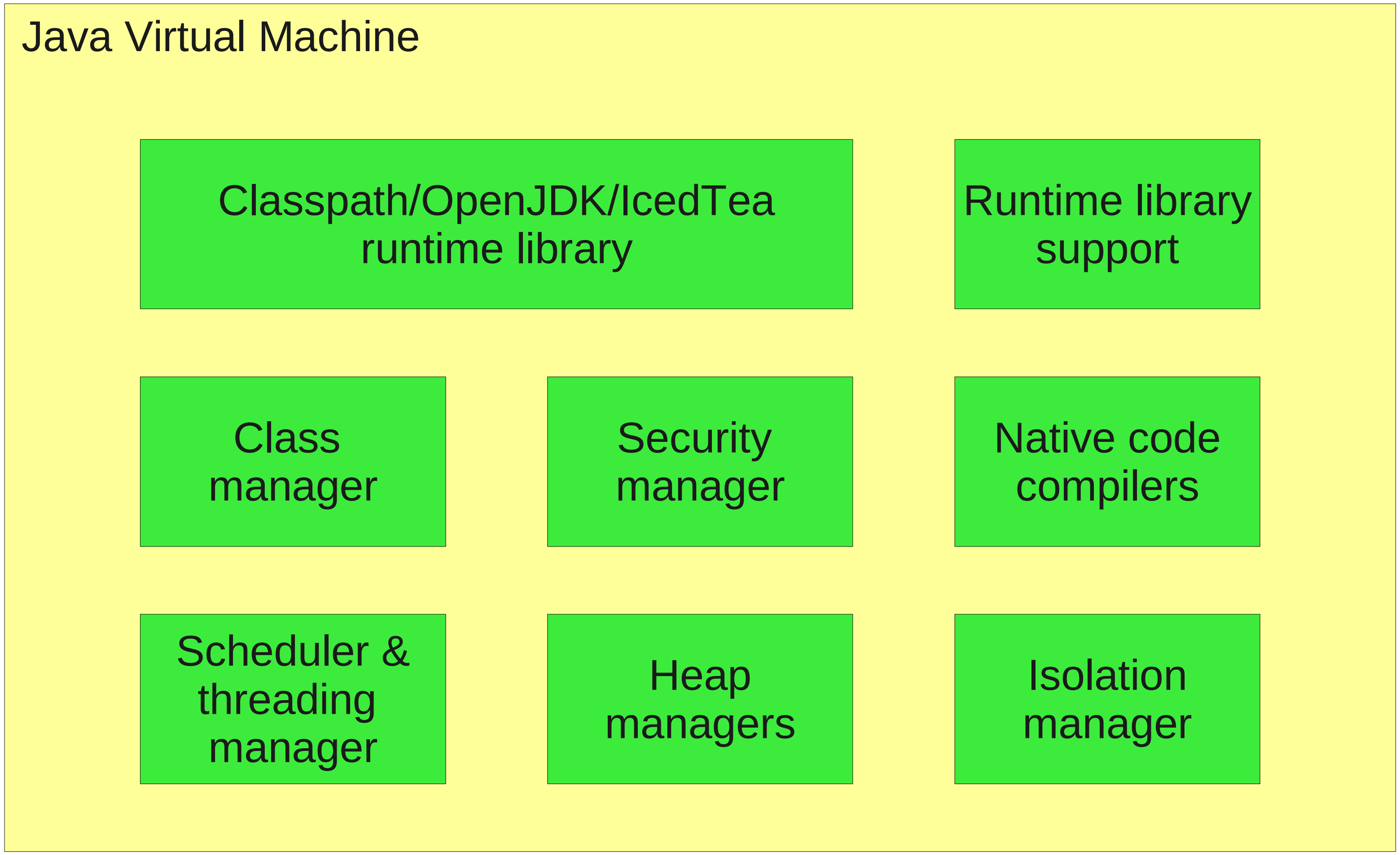
JNode | Console | jEdit - JNode-0.2.8.txt (modified) | jEdit Help | File System Browser | 12:51

JEdit
on JNode

Architecture (1)



Architecture (2)



Plugin framework (1)

- Everything is contained in a plugin
  - code, resources
  - even JVM & the plugin framework itself
- Plugins can:
  - be loaded, unloaded & reloaded (at runtime)
  - depend on other plugins
  - provide well known extension points
  - connect to well known extension points

Plugin framework (2)

- Plugins are:
  - described by a descriptor
    - descriptor also contains license info
  - JAR files
  - inspired by Eclipse plugins (before use of OSGi)

Plugin framework (3)

```
<plugin id="org.jnode.driver" name="JNode Driver Framework" version="@VERSION@"
        provider-name="JNode.org" license-name="lgpl" class="org.jnode.driver.DriverPlugin">

    <requires>
        <import plugin="org.jnode.work"/>
    </requires>

    <runtime>
        <library name="jnode-core.jar">
            <export name="org.jnode.driver.*"/>
            <export name="org.jnode.driver.util.*"/>
        </library>
    </runtime>

    <extension-point id="finders" name="System device finders"/>
    <extension-point id="mappers" name="Device to Driver mappers"/>

    <extension point="org.jnode.security.permissions">
        <permission class="java.util.PropertyPermission" name="jnode.cmdline"/>
    </extension>
</plugin>
```

General info

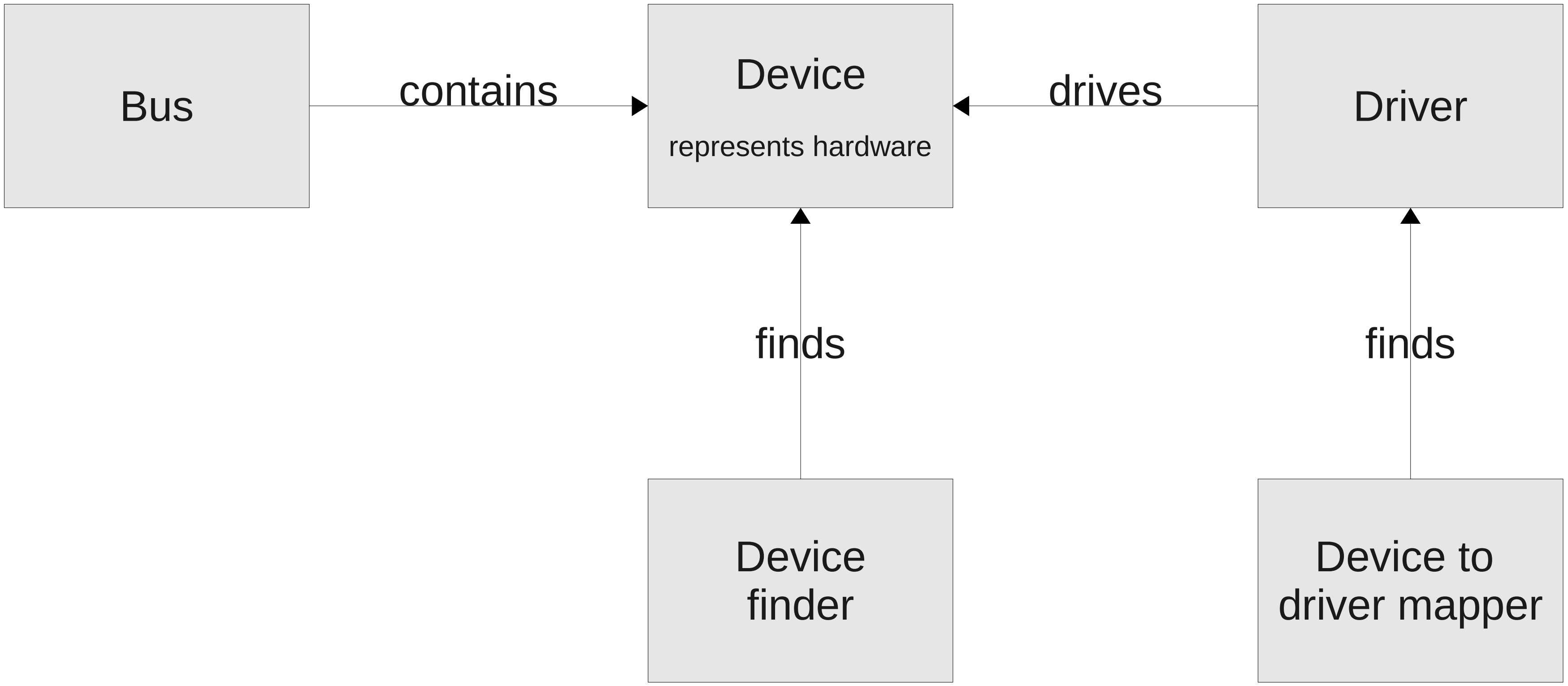
Dependencies

Code & resources

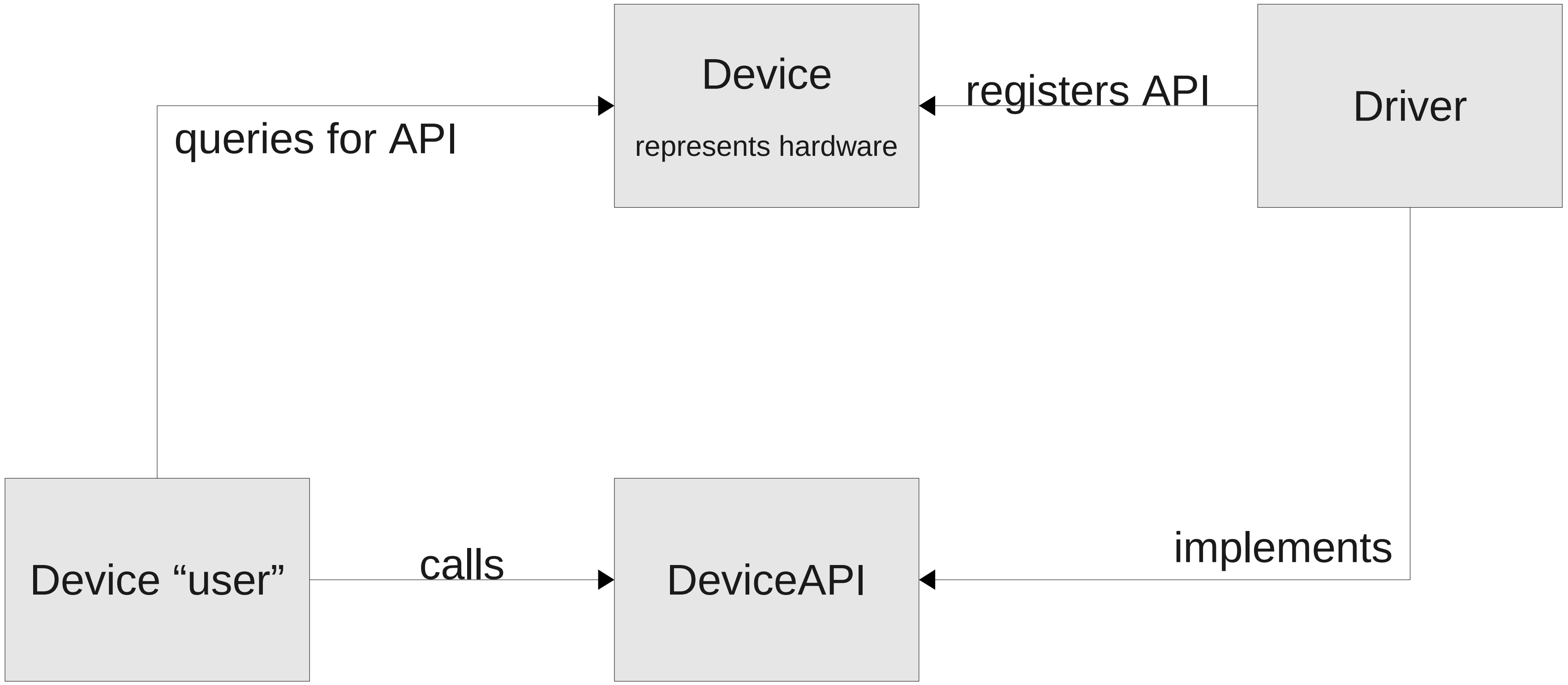
Well known extension points

Connection to well known extension point

Driver framework (1)

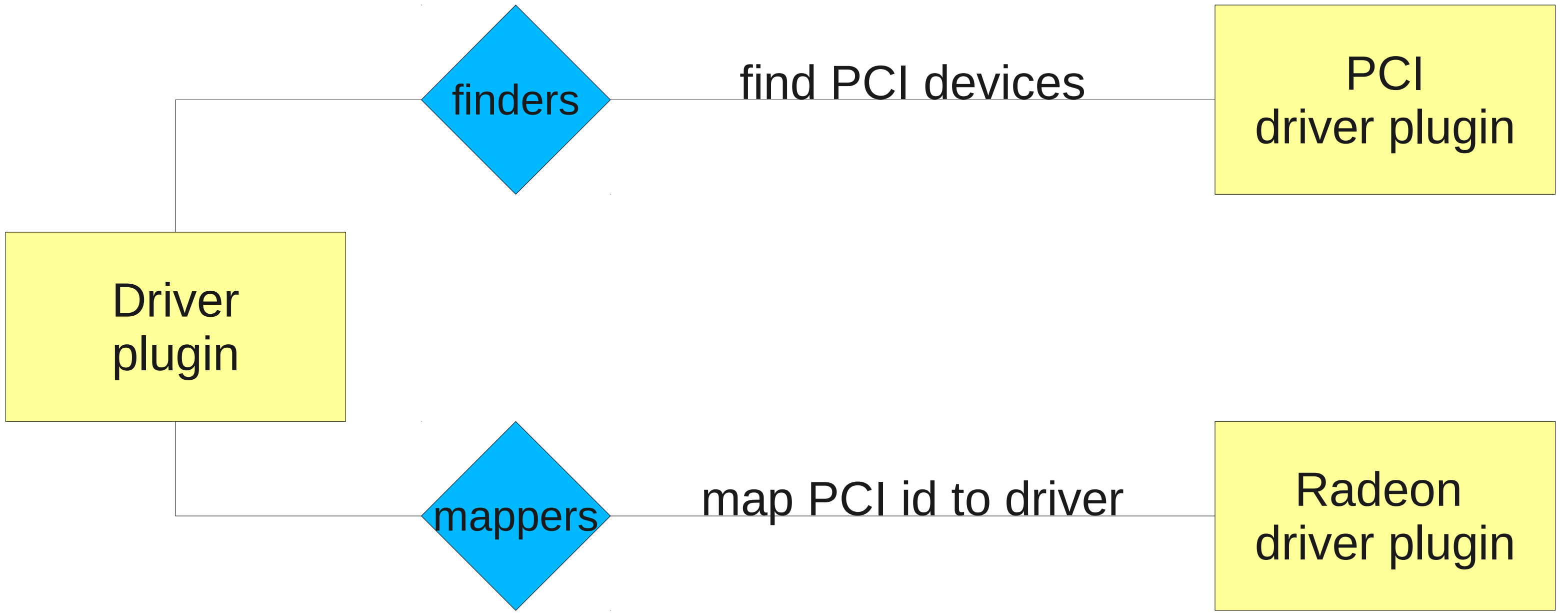


Driver framework (2)

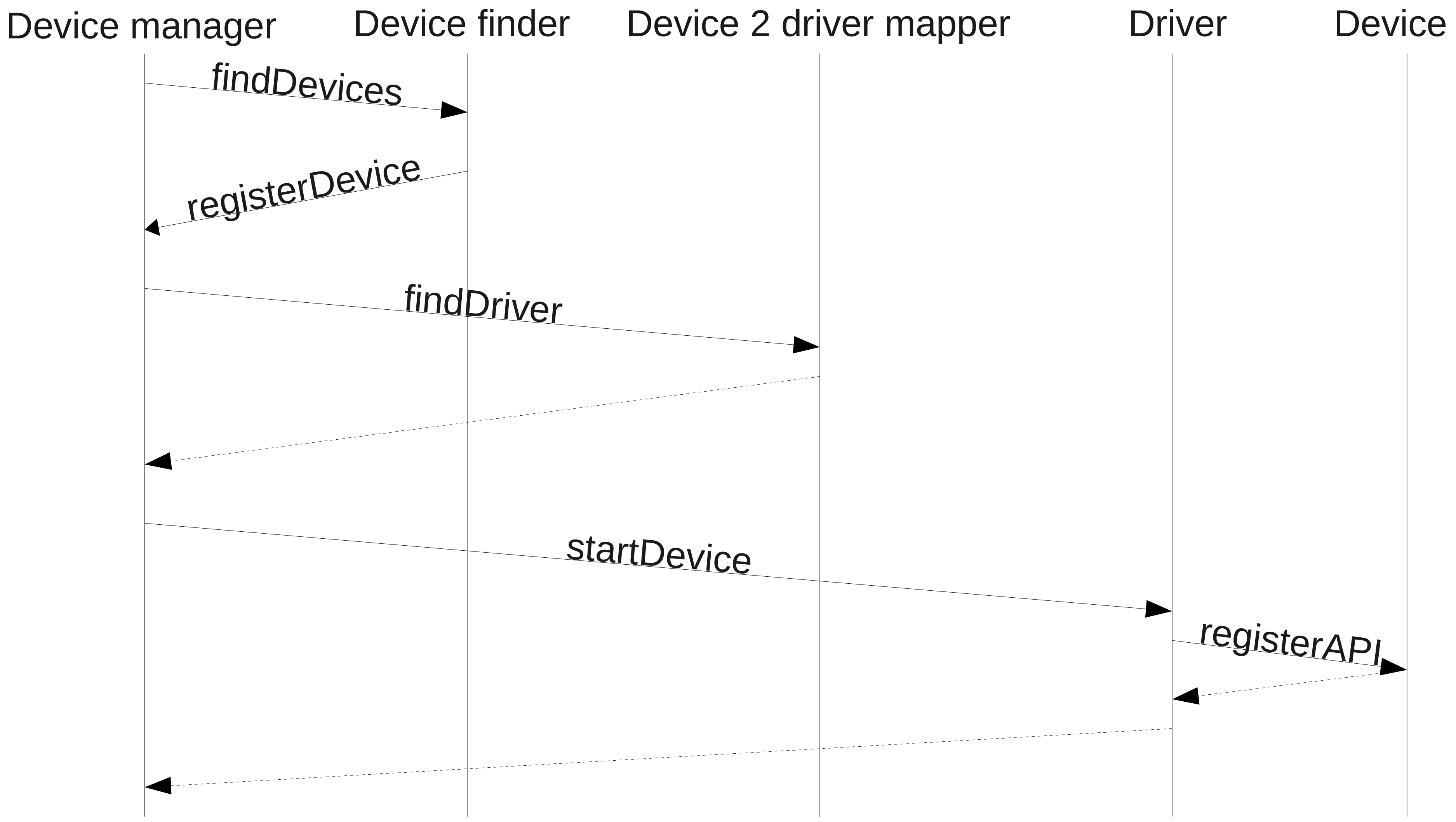


Driver framework (3)

Example: *Radeon Graphics card driver*



Driver framework (4)



Challenges (1/2)

- Make the people come
- Let the people stay
- Implement openjdk ↔ VM bindings
  - JNode VM : bindings in pure java !
  - Issues :
    - what is that native method supposed to do ?
    - Do not modify source code → easier maintenance
    - Add annotations for isolates

Challenges (2/2)

- Implement native methods in pure java
  - Needed by :
    - openjdk ↔ VM bindings
    - any application / library with native methods
  - Solution :
    - JarFile.class :

      private **native** String[] getMetaInfEntryNames();
    - Replaced by NativeJarFile.class :

      private **static** String[] getMetaInfEntryNames(**JarFile** instance)

Child Projects (1/4)

- Student projects
- Migration to maven
- JTestPlatform

Child Projects (2/4)

Student projects

- Goal : involve students in JNode development
- Source control : http://gitorious.org/jnode
- Inspired by Eric Bachard's conference at LSM 2009
- Results of 2009/2010 year :
  - Many (french) school were contacted
    - Only 3 replied positively
      - 2 went further and involved 5 students
  - 1 student from Latvia also involved

Child Projects (3/4)

Migration to Maven

- Goal : Use Maven instead of Ant
- Source control : http://gitorious.org/~fduminy/jnode/maven
- Expected benefits :
  - Standard layout instead of home made one => easier to learn for new developers
  - Architecture :
    - No cycle in dependencies between plugins
    - Promote modularity
  - Discover plugin dependency issues at build time
  - Easier to use existing tools (esp. QA ones)

Child Projects (4/4)

JTestPlatform

- Goal : Test any JVM implementation
- Source control : http://gitorious.org/jtestplatform
- Not only targeted at JNode !
- Supported test frameworks :
  - Actually : JUnit, Mauve
  - Future : jtreg (partially ?), your test framework ?
- Tests run in a cloud (libvirt + java binding)
- Tests run on a set of platforms (x86, x86_64 …)
- Generate reports : xml, text, html
- Future : use generated reports in QA tools

Future

- Short term:
  - Improved JVM performance (mm, compilers)
  - Deployment framework
  - Improved graphics
- Long term:
  - Simple to use desktop environment
    - Fully document oriented instead of app. oriented
  - Java powered servers
    - e.g. Cooperation with ApacheDS

Java benefits

- Dynamic linking
- Type safe language (even more since Java 5)
- Security
  - Security manager
  - No uncontrolled memory access
- Great development tools:
  - Eclipse, Ant

We need your help!

- Don't be scared by the codebase
  - Most of it is openjdk and classpath libraries
- Ask questions

- Visit http://www.jnode.org
- Contact me: fduminy@jnode.org

Demo

Questions