The 7th Generation
JSR 336

http://blog.eisele.net

http://twitter.com/myfear

Agenda

- Java – Introduction, History and Status
- Dynamic Languages in the JVM
- Small Language Enhancements
- NIO 2
- Other News
- Java Virtual Maschine

Introduction, History and Status

- History
- JSR 336
- OpenJDK
- Schedule

History

JDK/Year	Comment
JDK 1.1 (released on Feb 19,1997)	Major additions included an extensive retooling of the AWT event model, inner classes added to the language, JavaBeans and JDBC.
J2SE 1.2 (Dec 8, 1998) Codename Playground.	Reflection, a Collections framework, Java IDL (an IDL implementation for CORBA interoperability), Swing graphical API, a Java Plug-in, JIT compiler.
J2SE 1.3 (May 8, 2000) Codename Kestrel.	Bundling of the HotSpot JVM (the HotSpot JVM was first released in April, 1999 for the J2SE 1.2 JVM), JavaSound, JNDI and JPDA.
J2SE 1.4 (Feb 6, 2002) Codename Merlin.	Developed under the Java Community Process as JSR 59. Regular expressions modeled after Perl, exception chaining, an integrated XML parser and XSLT processor (JAXP), and Java Web Start.
J2SE 5.0 or 1.5 (Sep 30, 2004) Codename Tiger.	Developed as JSR 176, New language features including the for-each loop, generics, autoboxing and var-args.
Java SE 6 (Dec 11, 2006) Codename Mustang	Database manager, Use of scripting languages with the JVM, Visual Basic language support. Support for pluggable annotations (JSR 269), GUI improvements, including native UI enhancements, improvements to the JPDA & JVM Tool Interface
Java SE 7 — Codename Dolphin (Jul 28,2011)	The Dolphin Project started in August 2006

Plan A or Plan B – The Sundown

"It's been clear for some time that the most recent JDK 7 development schedule is, to put it mildly, unrealistic."
(Mark Reinhold, 2010/09/08)

Plan A:	JDK 7 (as defined by Sun)	Mid 2012
Plan B:	JDK 7 (minus Lambda, Jigsaw, and part of Coin)	Mid 2011
	JDK 8 (Lambda, Jigsaw, the rest of Coin, ++)	Late 2012

http://mreinhold.org/blog/rethinking-jdk7

Plan B! – Moving Java Foreward

Java SE 7 — Mid 2011

- JSR 292: Support for Dynamically-Typed Languages (“InvokeDynamic”)
- JSR 334: Small Language Enhancements (Project Coin)
- Upgrade Class-Loader Architecture
- Method to Close a URLClassLoader
- Concurrency and Collections Updates (including the Fork/Join Framework)
- Unicode 6.0
- Locale Enhancements (IETF BCP 47 & UTR 35)
- JSR 203: More New I/O APIs (“NIO.2”)
- TLS 1.2
- Elliptic-Curve Cryptography (ECC)
- JDBC 4.1
- Translucent & Shaped Windows
- Heavyweight/Lightweight Component Mixing
- Swing: Nimbus Look-and-Feel
- Swing: JLayer Component
- Update the XML Stack (JAXP, JAXB, & JAX-WS)

Java SE 8 — Late 2012

- JSR 294: Language and VM Support for Modular Programming
- JSR TBD: Platform Modularization
- JSR TBD: Lambda Expressions, Default Methods, & Bulk Data Operations (Project Lambda)
- JSR 308: Annotations on Java Types
- JSR TBD: More Small Language Enhancements (Project Coin part 2)

http://mreinhold.org/blog/plan-b-details

JSR #336, 06.06.2011

JSR #336
Java™ SE 7 Release Contents
Public Review Ballot

These are the final results of the Public Review Ballot for JSR #336. The Executive Committee for SE/EE has approved this ballot.

Votes

SE/EE

Credit Suisse ☐	Eclipse Foundation, Inc ✓	Ericsson AB ✓	Fujitsu Limited ✓
Goldman Sachs & Co. ✓	Google Inc. ✗	Hewlett-Packard ✓	IBM ✓
Intel Corp. ✓	Keil, Werner ▢	London Java Community ✓	Oracle ✓
RedHat ✓	SAP AG ✓	SouJava ✓	VMWare ✓

Icon Legend

- Yes ✓
- No ✗
- Abstain ▢
- Not voted ☐

http://jcp.org/en/jsr/results?id=5207

What is the OpenJDK?

- JDK7 is the second JDK done via the OpenJDK effort
- OpenJDK is free, open-source and GPL licensed (+ CP-Exception) (http://openjdk.java.net/legal/gplv2+ce.html)
- A lot of the improvements of JDK7 are separate projects on OpenJDK (http://openjdk.java.net/projects/)
- Some say that projects go under the cap of OpenJDK to avoid the cumbersome and slow JCP process

OpenJDK Schedule

Milestone	Date	# of Builds	Comment
M1	2009/01/02 – 2009/02/19 (b48)	7 builds	
M2	2009/02/20 – 2009/04/02 (b53)	5 builds	
M3	2009/04/03 – 2009/05/14 (b59)	6 builds	JavaOne Preview
M4	2009/06/05 – 2009/07/23 (b66)	7 builds	
M5	2009/07/24 – 2009/11/12 (b76)	10 builds	
M6	2009/11/27 – 2010/02/18 (b84)	8 builds	
M7	2010/02/26 – 2010/04/15 (b89)	5 builds	
M8	2010/04/16 – 2010/06/03 (b96)	7 builds	
M9	2010/06/04 – 2010/07/29 (b103)	7 builds	
M10	2010/07/30 – 2010/09/09 (b109)	6 builds	
M11	2010/09/10 – 2010/12/23 (b123)	14 builds	Feature Complete
M12	2010/12/31 – 2011/02/17 (b130)	7 builds	Developer Preview
	2011/04/12		Rampdown start: P1-P3 bugs only
	2011/04/28		API/interface changes: Showstoppers only
	2011/05/11		All targeted bugs addressed
	2011/05/18		Bug fixes: Showstoppers only
M13	2011/02/18 – 2011/06/02 (b145)	15 builds	Last scheduled build - Final test cycle starts
GA	**2011/07/28**		**General Availability**

Dynamic Languages in the JVM

- The Da Vinci Maschine Project
- JSR 292: Supporting Dynamically Typed Languages on the Java Plattform

The Da Vinci Maschine Project

a multi-language renaissance for the Java™ Virtual Machine architecture

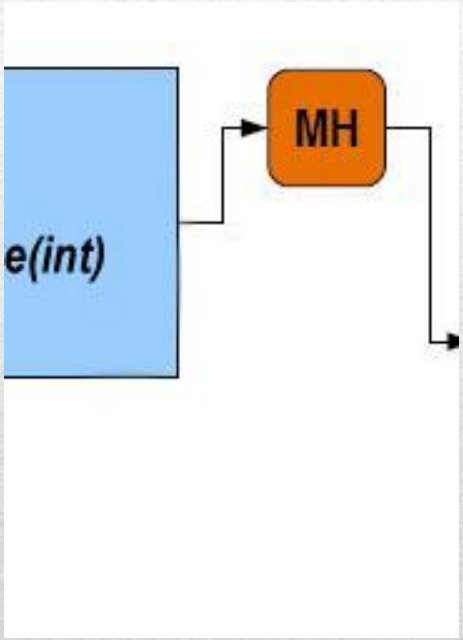
- Da Vinci Machine sub-projects include:
- Dynamic invocation
- Continuations
- Tail-calls
- And interface injection

- JVM has 4 bytecodes for method
  - invokevirtual - Invokes a method on a class. This is the typical type of method invocation.
  - invokeinterface - Invokes a method on an interface.
  - invokestatic - Invokes a static method on a class. This is the only kind of invocation that lacks a receiver argument.
  - invokespecial - Invokes a method without reference to the type of the receiver. Methods called this way can be constructors, superclass methods, or private methods.
- + one more since 1.7:
  - invokedynamic

Invokedynamic is the first new java bytecode since Java 1.0.

JSR 292: InvokeDynamic

- The standard behind Da Vinci Machine
- Natural continuation of JSR 223: Scripting for the Java Platform implemented in JDK
- New JVM instruction invokedynamic
  - Allows extremely fast dynamic method invocation through method handles
  - Will enable JRuby, Jython, Groovy and other dynamic and scripting languages to call dynamic methods natively at bytecode level
- Method handles
  - Lightweight references to a method – java.dyn.MethodHandle
  - Anonymous classes in the JVM
- Autonomous methods
  - Methods that can be dynamically attached to an existing class at runtime
- Interface injection
  - Acquiring base interfaces and method implementations at runtime
- Continuations and stack introspection
  - Suspend / resume thread's execution stack
- Tail calls and tail recursion



http://jcp.org/en/jsr/detail?id=292
http://blogs.oracle.com/gbracha/resource/JAOO2005.pdf
http://java.sun.com/developer/technicalArticles/DynTypeLang/index.html

Small Enhancements to the Java Programming Language

- Strings in switch
- Binary integral literals and underscore literals
- Multi-Catch and more precise rethrow
- Improved type interence for Gerneric instance creation (diamond)
- Try-with-resources statement
- Simplified varargs method invocation

Strings in Switch

```
String s = "";
switch (s) {
    case “doag":
        out.println("1");
        break;
    case “ukoug":
        out.println("2");
        break;
    default:
        out.println("Unknown UG.");
        break;
}
```

Underscores in Numbers

```
public static void main(String... args) {

  // THE OLD WAY
  int oldBillion = 1000000000;

  // THE NEW WAY
  int newBillion = 1_000_000_000;

}
```

Binary Literals

```
public static void main(String... args) {
  // THE OLD WAY
  int oldBinary1 = 153;
  int oldBinary2 = 128 ^ 0 ^ 0 ^ 16 ^ 8 ^ 0 ^ 0 ^ 1;

  // THE NEW WAY
  int newBinary = 0b1001_1001;
  out.println(oldBinary1);
  out.println(oldBinary2);
  out.println(format("[0b1001_1001] is {0}", newBinary));
}
```

Multi-Catch and more precise rethrow

```
try {
      throwAorB();
    } catch (ExceptionA | ExceptionB ex) {
          throw ex;
}
```

```
try {
      throwAorB();
    } catch (final Throwable t) {
          throw t; // ExceptionA or ExceptionB
}
```

Improved Type Inference for Generic Instance Creation

- This:

```
Map<String, List<String>> anagrams =
     new HashMap<String, List<String>>();
```

- ….becomes:

```
Map<String, List<String>> anagrams = new
HashMap<>();
```

- <> is called the „Diamond“ Operator

Try with resources

```
private static void close(Closeable closable) {
        if (closable != null) {
            try {
                closable.close();
            } catch (IOException e) { //optional
                System.err.println(e);
            }
        }
    }
```

```
try (InputStream fis = new FileInputStream(source);
                OutputStream fos = new
FileOutputStream(target)) {
            // do stuff
        } // closed here
```

More New I/O APIs for the Java Platform

- java.io.File
- java.nio.file.Path
- java.nio.file.FileSystem
- Other Usefull Methods

File

```
File file = new java.io.File(“mypath/subPath");
Path path = file.toPath();

System.out.println("Created path (%s) of type
%s%n”, path, path.getClass().getName());
```

Created path (myPath\subPath) of type sun.nio.fs.WindowsPath

http://download.oracle.com/javase/tutorial/essential/io/fileio.html

Path

```
Path path = java.nio.file.Paths.get("myPath",
"subPath");

System.out.println("Created path (%s) of type
%s%n", path, path.getClass().getName());
```

Created path (myPath\subPath) of type sun.nio.fs.WindowsPath

Filesystem

```
Path path =
FileSystems.getDefault().getPath("mypath",
"subPath");


System.out.println("Created path (%s) of type
%s%n", path, path.getClass().getName());
```

Created path (myPath\subPath) of type sun.nio.fs.WindowsPath

Determining MIME Types

```
Path path =
FileSystems.getDefault().getPath("java7.docx");

try {
    String type = Files.probeContentType(path);
    System.out.format("'%s' has the filetype
'%s'.%n", path, type);
    } catch (IOException x) {
      System.err.println(x);
    }
```

'java7.docx' has the filetype 'application/vnd.openxmlformats-officedocument.wordprocessingml.document'.

Other news

- Concurrency and collections updates (jsr166y)
- Security
- Internationalization
- SCTP, SDP, TLS
- JDBC 4.1 + RowSet 1.1
- Other other other

Concurrency and Collections Updates - JSR 166y

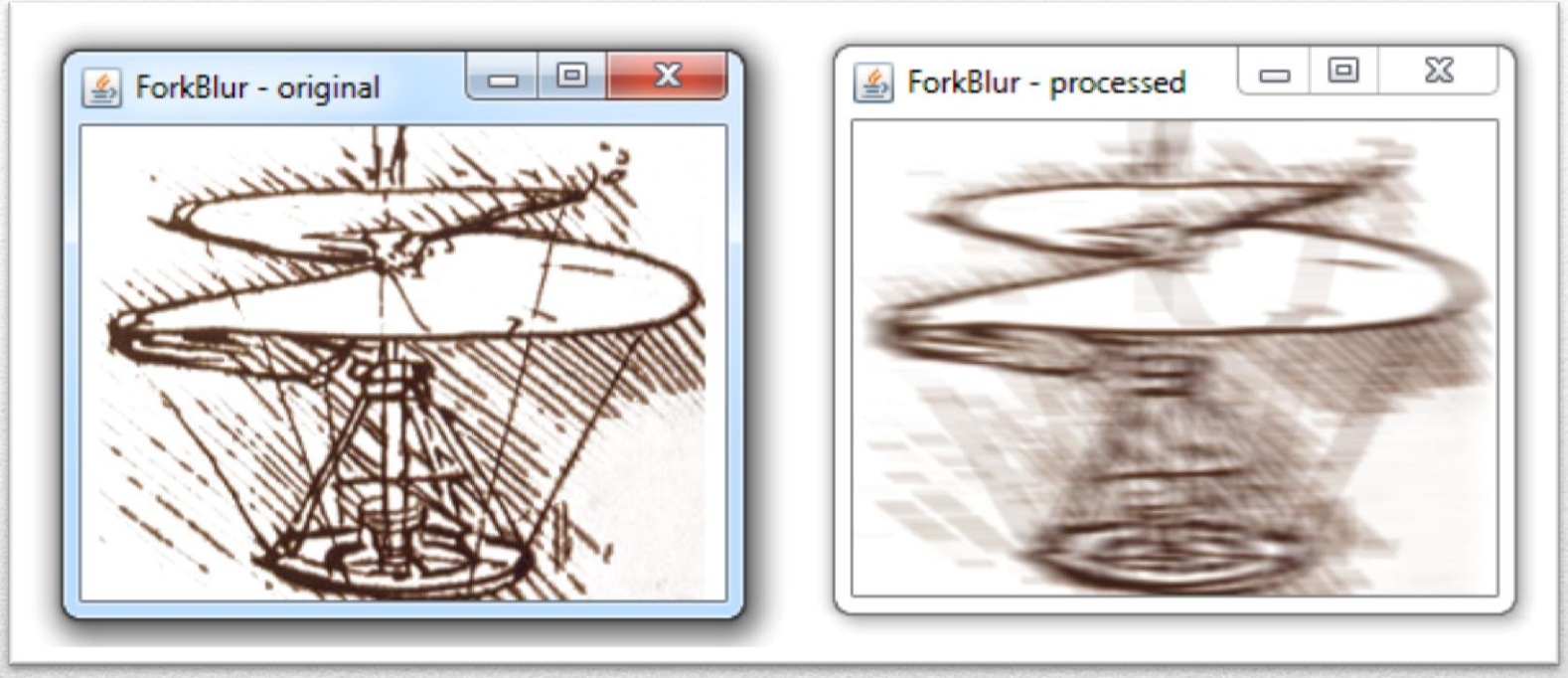
- Java threads on diet
- Thread class in Java is quite limited and doesn't support multi-cored hardware well enough. The fork/join framework is the answer.
  - introduces a layer of indirection between threads and tasks.
  - create a pool of threads the size of which corresponds to the number of processors/cores you have
  - and then start assigning tasks to the pool

```
if (my portion of the work is small enough)
  do the work directly
else
  split my work into two pieces
  invoke the two pieces and wait for the results
```

- Wrap this code as a ForkJoinTask subclass, typically as one of its more specialized types RecursiveTask (which can return a result) or RecursiveAction.
- After your ForkJoinTask is ready, create one that represents all the work to be done and pass it to the invoke() method of a ForkJoinPool instance.

http://download.oracle.com/javase/tutorial/essential/concurrency/forkjoin.html

Concurrency and Collections Updates - JSR 166y



run:
Array size is 59764
Threshold is 10000
8 processors are available
Image blur took 26 milliseconds.

http://download.oracle.com/javase/tutorial/essential/concurrency/examples/ForkBlur.java

Security

- Native ECC (Elliptic Curve Cryptography)
- TLS (Transport Layer Security)1.2
- Stronger pseudorandom functions, additional (stronger) hash/signature algorithms, enhanced key exchange
- ASLR (Address space layout randomization)
- DEP (Data Execution Prevention) – Windows Only

Internationalization

- Unicode 6.0
- IETF BCP47 and UTR35
  - One language many character codes (Kanji, Hiragana, Katakana, Romaji)
  - Three-letter base language codes; three-digit region codes
- Separate user locale and user interface locale
- Currency data enhancements

More Network Stuff

- SCTP (Stream Control Transmission Protocol)
  - used in Solaris
- SDP (Sockets Direct Protocol)
  - Infiniband protocol
- Redirection for Subprocess
  - You will be able to use Redirection for sub process, redirecting stdout and stderr for that sub process only.
- Vista IPv6 stack

JDBC 4.1

- Using try-with-resources Statements to Automatically Close JDBC Resources

```
public static void getCoffees(Connection con) throws SQLException {
        String query = "select COF_NAME, PRICE from COFFEES";
        try (Statement stmt = con.createStatement()) {

            ResultSet rs = stmt.executeQuery(query);

            while (rs.next()) {
                String coffeeName = rs.getString("COF_NAME");
                float price = rs.getFloat("PRICE");
                System.out.println(coffeeName + ", " + price);
            }
        }
    }
```

RowSet 1.1 - RowSetFactory

- You can use an instance of RowSetFactory to create a RowSet object.

```
public void testJdbcRowSet(String username, String password) throws SQLException {

        RowSetFactory myRowSetFactory = null;
        JdbcRowSet jdbcRs = null;
        ResultSet rs = null;
        Statement stmt = null;

        try {

            myRowSetFactory = RowSetProvider.newFactory();
            jdbcRs = myRowSetFactory.createJdbcRowSet();

            jdbcRs.setUrl("jdbc:myDriver:myAttribute");
            jdbcRs.setUsername(username);
            jdbcRs.setPassword(password);

            jdbcRs.setCommand("select COF_NAME, SUP_ID, PRICE, SALES, TOTAL from COFFEES");
            jdbcRs.execute();

            // ...

        } catch (Exception e) {
            // handle
        }
    }
```

http://download.oracle.com/javase/tutorial/jdbc/basics/jdbcrowset.html

RowSet 1.1 – FilteredRowSet

- A FilteredRowSet object lets you cut down the number of rows that are visible in a RowSet object so that you can work with only the data that is relevant to what you are doing.

```
FilteredRowSet fRs =
myRowSetFactory.createFilteredRowSet();
Range name = new Range(1, 10, "id");
fRs.setFilter(name);
fRs.next(); // only ids from 1 to 10 will be returned
```

http://download.oracle.com/javase/tutorial/jdbc/basics/filteredrowset.html

RowSet 1.1 – CachedRowSet

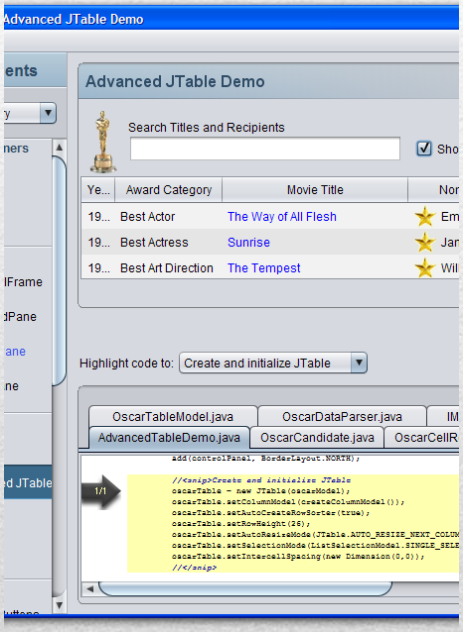
- A CachedRowSet object is special in that it can operate without being connected to its data source, that is, it is a disconnected RowSet object.

```
CachedRowSet cRs =
myRowSetFactory.createCachedRowSet();
cRs.setCommand("select * from COFFEES");
cRs.execute(); // RowSet gets populated
```

http://download.oracle.com/javase/tutorial/jdbc/basics/cachedrowset.html

Other other other

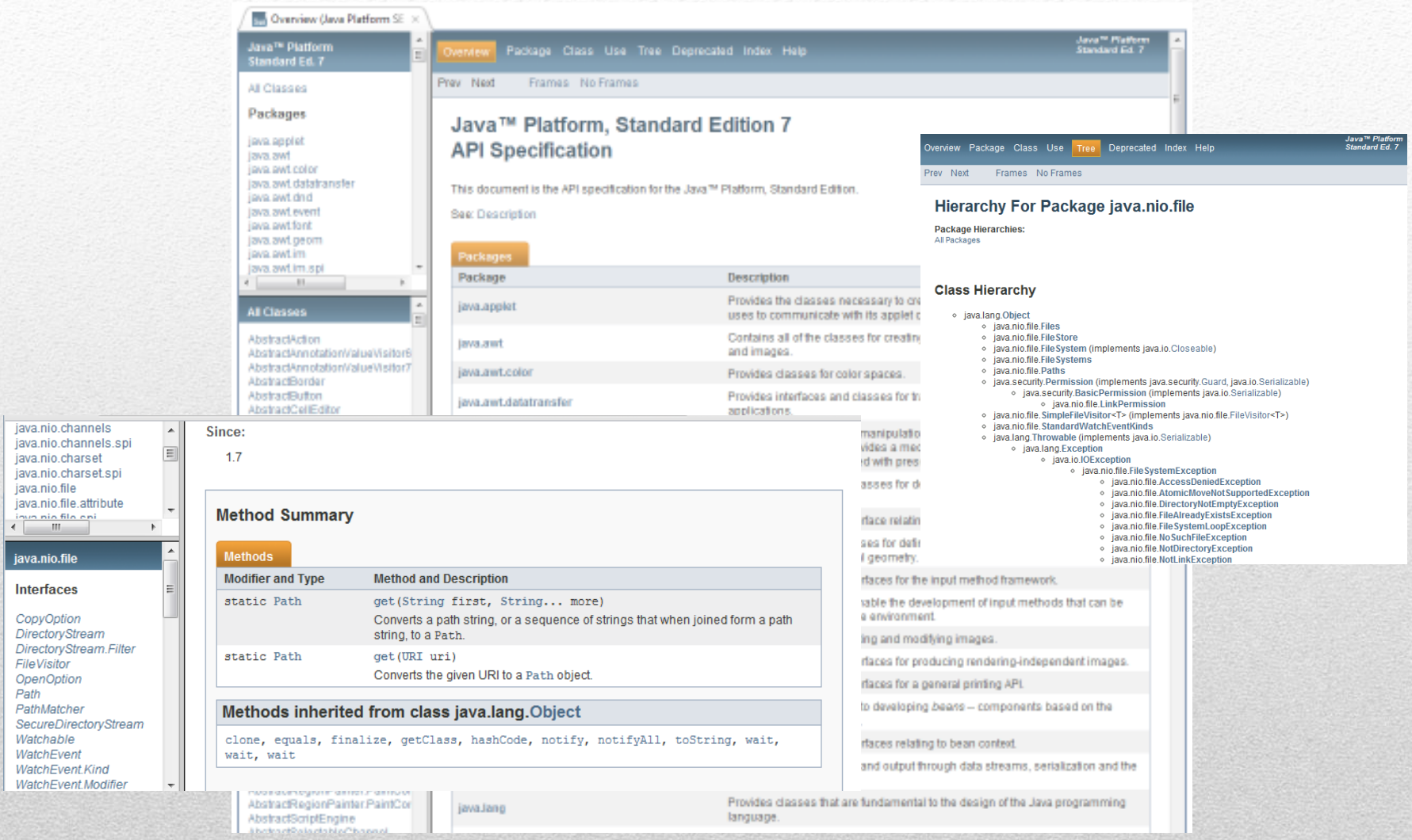
- Added Nimbus L&F to the standard
  - Much better –modern- look than what was previously available
- Platform APIs for Java 6u10 Graphics Features
  - Shaped and translucent windows
- JXLayer core included in the standard
  - Formerly SwingLabs component
- Allows easier layering inside of a component
- Optimized Java2D Rendering Pipeline for X-Windows
  - Allows hardware accelerated remote X



Other other other

- Better font configuration on Unix
  - Now uses standard Unix mechanism to find fonts.
- JAXP 1.4.4, JAXWS 2.2 and JAXB 2.2
- Upgrade class-loader architecture
  - Modifications to the ClassLoader API and implementation to avoid deadlocks in non-hierarchical class-loader topologies
- Close URLClassloaders
  - Allows applications to proactively clean up classloaders, freeing up native resources and unlocking JAR files
- Javadoc support for stylesheets

What‘s up JavaDoc?



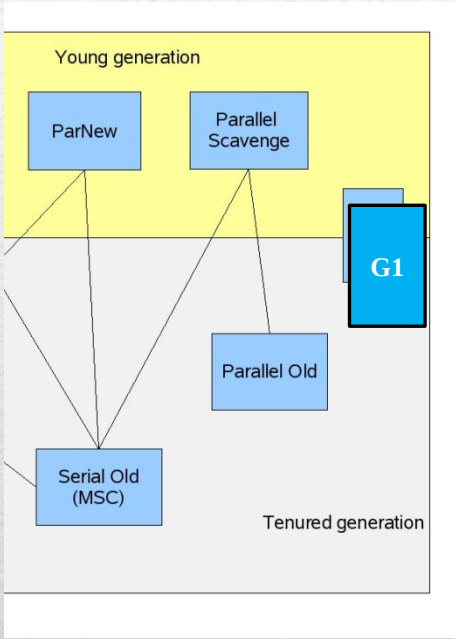
http://download.java.net/jdk7/docs/api/

Java Virtual Maschine

- Garbage-First (GC1)
- HotRockit

Garbage First (G1)

- "server-style" GC
- More predictable GC pauses
- Better GC ergonomics
- Low pauses without fragmentation
- Parallelism and concurrency in collections
- Better heap utilization



- -XX:+UseG1GC
- -XX:MaxGCPauseMillis=50
- -XX:GCPauseIntervalMillis=200

- G1 is planned to replace CMS in the Hotspot JVM

http://labs.oracle.com/jtech/pubs/04-g1-paper-ismm.pdf

JVM Convergence – Project HotRockit

- **2 teams enter, one team leave ….**
- **Hotspot**
  - Market share
  - Client and Server version
  - Quality
- **JRockit**
  - Value adds
  - JRockit Mission Control
  - JRockit Flight Recorder
  - JRockit Virtual Edition
  - Optimized for the Oracle stack
- **1+1 = 1.15**
  - Lot of common features in both the JVMs
  - converging them will add roughly 15% new features
- **First JRockit code check in to OpenJDK**
  - Wed Feb 02 13:23:17 2011 +0100
  - Changing 1652 files
  - Done by Stefan Karlsson

JVM Convergence – Project HotRockit

Hopefully, the convergence will be completed by JDK 8 GA, but **no promises as of today!**

JDK 7 GA

- Java SE 7 Support
- Rebranding
- No PermGen (stretch goal)
- Improved JMX Agent
- Command line servicability tool (jrcmd)
- ◉ JRockit Mission Control Console support

JDK 7 Update X

- Performance
- ◉ Initial JRockit Flight Recorder Support

JDK 7 Update Y

- More performance
- Improved command line servicability (jrcmd)
- G1 complete as CMS replacement
- Non-contiguous heaps
- ◉ Complete JRockit Flight Recorder Support
- ◉ JRockit Virtual Edition Support
- ◉ Soft Real Time GC

JDK 8 GA

- Java SE 8 Support
- All performance features from JRockit ported
  - I/O Performance
  - JIT Optimizations
- All servicability features from JRockit ported
  - Compiler controls
  - Verbose logging
- ◉ JRockit Mission Control Memleak Tool Support

◉ = Commercial Add-On

Hate Java? You‘re fighting the wrong battle.

- Java has been widely successful for a number of reasons:
- It’s widely accepted in the established companies.
- It’s one of the fastest languages.
- It’s one of the most secure languages.
- Synchronization primitives are built into the language.
- It’s platform independent.
- Hotspot is open source.
- Thousands of vendors exist for a multitude of Java products.
- Thousands of open source libraries exist for Java.
- Community governance via that JCP (pre-Oracle).

- Tell me, what do the following have in common?
- Paying with a credit card.
- Going to the emergency room.
- Adjusting your 401k.
- Using your insurance card at the dentist.
- Shopping around for the best car insurance.
- A BNSF train pulling a Union Pacific coal car.
- Transferring money between banks.
- Filling a prescription.

http://www.javacodegeeks.com/2011/06/hate-java-fight-wrong-battle.html

Thank your for your attention!