New Features in OpenJDK 7
JDK and OpenJDK (r)evolution
JDK (r)evolution (1)

- **1996 - JDK 1.0**
  - Initial release, but the first stable version was JDK 1.0.2
- **1997 - JDK 1.1**
  - Inner classes, AWT rewritten, JDBC, remote method invocation (RMI)
  - Syntax changed due to inner classes
- **1998 - J2SE 1.2**
  - Swing, JIT, collections
  - Syntax changed due to the new `strictfp` keyword
JDK (r)evolution (2)

- 2000 - J2SE 1.3
  - JNDI, JPDA, Hotspot JVM
  - Synthetic proxy classes

- 2002 - J2SE 1.4
  - regexp, JAXP, JWS, NIO, assert
  - Syntax changed due to the new assert keyword
New features in OpenJDK 7

JDK (r)evolution (3)

- 2004 - J2SE 5.0
  - Generics, annotations, enumerations, varargs, autoboxing, unboxing, for-each loop
  - Syntax changed, but no new keywords has been added
- 2006 - Java SE 6
  - JDBC 4.0, JAXB 2.0, scripting language support
- 20?? - Java SE 7 (I hope ?? means 11 :-)
OpenJDK evolution

New features in OpenJDK 7

Pavel Tišnovský
Java language basic principles

- Program code should do what it seems to do
- **Reading** is more important than writing
  - X TMTOWTDI preferred by Perl community
- Code should be a joy to read
  - Well, sometimes...
- Clear relationship between syntax and semantics
  - Absence of “magic“ variables like $$, $$_, etc.
(OpenJDK) (r)evolution priorities

- Productivity
  - Get more done with less code
- Performance
  - Multi-core processors are the next target
- Universality
  - More languages at top of JVM
- Modularity
  - CLASSPATH hell
- Integration
- Serviceability
  - Monitoring, diagnosing problems...
JDK 7 feature list

- **Plan A:**
  - JDK 7 with all previously planned features
  - Delivery in mid 2012

- **Plan B:**
  - JDK 7 (without Lambda, Jigsaw and some features from the Coin project)
  - Delivery in mid 2011
  - JDK 8 (Lambda, Jigsaw, the rest of the Coin project, ...)
  - End of year 2012
... and the winner is ...
Plan B
Schedule

- According to Mark Reinhold (Oracle)
  - He talked about OpenJDK plans on FOSDEM 2011 conference
Schedule

- Last full test cycle:

2011-06-08
Final release: 2011-07-28
New OpenJDK7 features included in Plan B (1)

- **JSR 292**
  - Supporting Dynamically Typed Languages
- **Project Coin**
  - Small Language Changes in JDK 7
- **JSR 166y**
  - Concurrency Utilities (Collections API changed)
- **JSR 203**
  - More New I/O APIs for the Java platform (NIO.2)
New OpenJDK7 features included in Plan B (2)

- JSR 203: SCTP
  - Stream Control Transmission Protocol
- JSR 203: SDP
  - Sockets Direct Protocol
- ECC
  - Elliptic curve cryptography
- XRender pipeline support for Java 2D
- Nimbus look-and-feel
  - for Swing applications
- Swing JLayer component
New OpenJDK 7 features included in Plan B (3)

- TLS 1.2
- JDBC 4.1
- Unicode 6.0
- Locale
  - Distinguishing between user locale settings and GUI settings
- NIO.2 filesystem
  - Support for zip and jar archives
- Windows Vista IPv6 stack
  - (system dependent)
Features postponed to JDK 8 (or 9?...)

- JSR 294
  - Language and VM support for modular programming
- JSR 308
  - Annotations on Java types
- JSR TBD
  - Language support for collections
- JSR TBD
  - Project Lambda
Features postponed to JDK 8 (or 9?...)

- Modularization
  - Project Jigsaw
- JSR 296
  - Swing application framework
- New Swing component
  - JDatePicker
New features in OpenJDK 7
Unicode 6.0

- Unicode 5.2
  - October 2009
  - ISO/IEC 10646:2003 + amendments 1, 2, 3, 4, 5 and 6
  - 90 scripts
  - 107,361 characters

- Unicode 6.0
  - October 2010
  - ISO/IEC 10646:2010 + Indian rupee sign
  - 93 scripts
  - 109,449 characters
Unicode in Java language :-)
Support for XRender Graphics Pipeline

- X Rendering Extension
  - Raster operations (rops)
  - Including various alpha channel operations
- Font antialiasing
- Graphics accelerator support
- Significant speed-up of some operations
Support for XRender Graphics Pipeline

MigLayout Swing Benchmark

- X11
- Xrender

<table>
<thead>
<tr>
<th>Hardware</th>
<th>benchmark</th>
</tr>
</thead>
<tbody>
<tr>
<td>945GM-XAA</td>
<td></td>
</tr>
<tr>
<td>945GM-EXA</td>
<td></td>
</tr>
<tr>
<td>NV17-XAA</td>
<td></td>
</tr>
<tr>
<td>NV17-EXA</td>
<td></td>
</tr>
<tr>
<td>6600-EXA</td>
<td></td>
</tr>
</tbody>
</table>

New features in OpenJDK 7
Support for XRender Graphics Pipeline in IcedTea6?

- XRender code has been backported into IcedTea6!
  - Fedora 12-rawhide, RHEL 6, Ubuntu...
- backport in form of one big patch
  - 340 kB :-)
  - 6307603-xrender-01.patch
Project COIN
Small Language Changes for JDK 7
“Making things programmers do every day easier.”
Project Coin

- Small changes in semantics and syntax of Java programming language
  - Syntactic sugar in some cases
  - To speed up day-to-day programming
  - Inspiration taken from other languages
- No new opcodes in the bytecode instruction set
- No new keywords!
Project Coin

- Small language changes:
  - Understandable specification
  - Quite easy implementation
  - Easier testing
- But not only `java/javac` affected!
  - NetBeans
  - Eclipse
  - ECJ/GCJ...
import java.util.List;

public class Test {
    List<String> list = new String<>(());

    static {
        String test = "qwe";
        switch (test) {
            case "aaa": break;
            case "bbb": break;
        }
    }
}
NetBeans
Text editors (Vim in this case)

```java
import java.util.ArrayList;
import java.util.List;

public class Test {
    List<String> list = new ArrayList<>();

    static {
        String test = "qwe";
        switch (test) {
            case "aaa": break;
            case "bbb": break;
        }
    }
}
```
Other compilers (ecj)

dhcp-lab-190:~/workspace_javatests/ProjectCoin/src/$ ecj -1.7 Test.java
----------
1. ERROR in Test.java (at line 6)
List<String> list = new ArrayList<>();
  ^
Syntax error on token "<", ? expected after this token
----------
1 problem (1 error)

1. ERROR in Test.java (at line 10)
switch (test) {
  ^^^^  
Cannot switch on a value of type String. Only convertible int values or enum constants are permitted
----------
1 problem (1 error)
Project Coin

• Binary integer constants
• Optional underscore (_) in numerical constants
• Strings in the `switch` statement
• “Diamond“ operator
• Automatic Resource Management (ARM)
• Better exception handling
Project Coin

Some examples shown on the following slides are based on source codes prepared by Joe Darcy (Oracle).

- published with his permission -
Binary integer constants

```java
int binary = 0b11001001001;
```
Optional underscore (_) in numerical constants

double amount = 1_999_888_777.25;

int color = 0xdd_dd_dd;

int binary = 0b110_0100_1001;

int octal = 0_12_34_56_77
Switch statement in Java

- Before JDK 5
  - Only integer constants allowed
Switch statement in Java

- Before JDK 5
  - Only integer constants allowed
- JDK 5
  - + enumeration types
Switch statement in Java

- **Before JDK 5**
  - Only integer constants allowed
- **JDK 5**
  - + enumeration types
- **JDK 7**
  - + strings (string literals)
Strings in switch statement

```java
switch (language) {
    case "cz":
        value = "Ahoj";   break;
    case "en":
        value = "Hi";     break;
    case "de":
        value = "Guten tag"; break;
    default:
        value = "Hello";  break;
}
```
int monthNameToDays(String s, int year) {
    if(s.equals("April") ||
        s.equals("June") ||
        s.equals("September") ||
        s.equals("November"))
        return 30;
    if(s.equals("January") ||
        s.equals("March") ||
        s.equals("May") ||
        s.equals("July") ||
        s.equals("August") ||
        s.equals("December"))
        return 31;
    if(s.equals("February"))
        ...
    else
        ...
}
int monthNameToDays(String s, int year) {
    switch(s) {
    case "April":
    case "June":
    case "September":
    case "November":
        return 30;
    case "January":
    case "March":
    case "May":
    case "July":
    case "August":
    case "December":
        return 31;
    case "February":
        ...
    }
“Diamond“ operator
Pre-generics style of Java code

List list = new ArrayList();
JDK 5 with generics support

List<String> list = new ArrayList<String>;;
JDK 7 and “diamond“ support

```java
List<String> list = new ArrayList<>();
```
JDK 7 and “diamond“ support

```java
List<List<List<String>>> list = new ArrayList<>();
```
"Diamond" operator

```java
Map<String, Collection<Integer>> map = new LinkedHashMap<>();
```

```java
Map<String, Collection<Integer>> map = new LinkedHashMap<String, Collection<Integer>>() {
    @Override
    public Collection<Integer> get(String key) {
        return super.get(key);
    }
};
```
Automatic Resource Management (ARM)

FileOutputStream fos = new FileOutputStream(file);
InputStream is = url.openStream();

byte[] buf = new byte[2048];
int len;
while ((len = is.read(buf)) > 0) {
    fos.write(buf, 0, len);
}
Incorrect solution

```java
public void write(URL url, File file)
{
    try {
        FileOutputStream fos = new FileOutputStream(file);
        InputStream is = url.openStream();

        byte[] buf = new byte[2048];
        int len;
        while ((len = is.read(buf)) > 0) {
            fos.write(buf, 0, len);
        }
        is.close();
        fos.close();
    } catch (IOException e) {} 
}
```
Incorrect solution

```java
public void write(URL url, File file) {
    try {
        FileOutputStream fos = new FileOutputStream(file);
        InputStream is = url.openStream();

        byte[] buf = new byte[2048];
        int len;
        while ((len = is.read(buf)) > 0) {
            fos.write(buf, 0, len);
        }
        is.close();
        fos.close();
    }
    catch (IOException e) {}
}
```
Correct solution

- Implemented using more nested try-catch-finally blocks
- `is.close()` and `fos.close()` can throw exception
  - And they are in `finally` blocks!
- Solution is more than 3 slides long :-)
- Exception from close methods should propagate
public void write(URL url, File file) throws IOException {
    try (
        FileOutputStream fos = new FileOutputStream(file);
        InputStream is = url.openStream() )
    {
        byte[] buf = new byte[2048];
        int len;
        while ((len = is.read(buf)) > 0) {
            fos.write(buf, 0, len);
        }
    }
}
Automatic Resource Management (ARM)

- After the previous method finishes:
  - FileInputStream is automagically closed
  - InputStream is automagically closed
  - all system resources are released
  - … and (almost) nothing can be forgotten

- New interface java.lang.AutoCloseable
Better form of exception handling

• Earlier:
  
```java
try {
    // ...program block...
} catch (FirstException ex) {
    logger.error(ex);
    throw ex;
} catch (SecondException ex) {
    logger.error(ex);
    throw ex;
}
```

Better form of exception handling

- In OpenJDK7:
  ```java
  try {
      ...program block...
  } catch (FirstException | SecondException ex) {
      logger.error(ex);
      throw ex;
  }
  ```
More information in backtrace

```
java.io.IOException
   at Suppress.write(Suppress.java:19)
   at Suppress.main(Suppress.java:8)
Suppressed: java.io.IOException
   at Suppress.close(Suppress.java:24)
   at Suppress.main(Suppress.java:9)
Suppressed: java.io.IOException
   at Suppress.close(Suppress.java:24)
   at Suppress.main(Suppress.java:9)
```

java.lang.AutoCloseable
java.lang.AutoCloseable

- Interface with a single method signature
  - `close()`
- For objects which have to be “closed” when not needed
- This method is called automatically from an extended `try` statement
  - See previous slides
- Also available for JDBC 4.1!
  - `java.sql.Connection`
  - `java.sql.ResultSet`
  - `java.sql.Statement`
java.lang.AutoCloseable (1)

java.lang.AutoCloseable (2)

java.lang.AutoCloseable (3)

Project Coin

- What did **not** make it into JDK7
  - Collection declaration
  - Access to collection items using brackets
  - Extended annotation options (for methods, parameters...)

:-(
Collection declarations - Lists

List<Integer> numbers = [ 1, 2, 4, 8, 16, 32, 64, 128 ];
Collection declarations - Sets

Set<Integer> numbers = { 256, 512, 1024, 2048, 4096 };
Collection declarations - Maps

Map<String, String> translations = {
    "Hi" : "Bonjour",
    "Goodbye" : "Au revoir",
    "Thanks" : "Merci"
};
Collections and brackets

```java
List<String> list = ...;
Map<String, String> map = ...;

String firstValue = list[0];
map["Test"] = firstValue;
String valueFromMap = map["Test"];```
Extended annotation options

Current use:

```java
@Override
int overriddenMethod() {}
```

New options:

```java
// List of non-null strings
List<@NonNull String> stringList;
```
Extended annotation options

// Non-empty list of strings
@NonEmpty List<String> stringList;

// Non-empty list of non-null strings
@NonEmpty List:@NonNull String stringList;

// Method with a read-only parameter which cannot change its object's attribute(s)
void marshall(@Readonly Object arg)
@Readonly
Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".
Useful links

http://openjdk.java.net/projects/jdk7/features/
http://blogs.sun.com/darcy
http://blogs.sun.com/mcimadamore
http://www.baptiste-wicht.com/2010/04/java-7-more-dyr
http://www.baptiste-wicht.com/2010/04/java-7-updates-
http://gee.cs.oswego.edu/dl/concurrency-interest/index.l
http://gee.cs.oswego.edu/dl/jsr166/dist/jsr166ydocs/
The purpose of this License is to make a manual, textbook, or other document “free” in the sense of freedom; to permit its free copying, redistribution and modification by others. To achieve this, this License expressly disclaims the warranty of merchantability, and does not allow the owner of this license to assert or defend any such warranty, either orally or in writing in any application. To the extent that applicable jurisdictions permit warranties or disclaimers of warranties, such warranties or disclaimers shall govern this License, rather than the words of this License. As a consequence, the licensee has the function of inspecting the material to determine whether it is usable and fits the user’s needs. This license discloses the function of the author and publisher in a way to get credit for their work, while not being considered responsible for modifications made by others.

This is a kind of “copyleft”, which means that derivatives of the work must also be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software. We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the source code and explanations so that it can be used for any purpose. We believe the best model for free documentation is one that licensing allows it to be used for any text work, regardless of subject matter, or whether it is published as a printed book. We recommend this license principally for works whose purpose is instruction or reference.

### I. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. This notice grants a worldwide, royalty-free, sublicenseable, unlimited, to use the work with the conditions stated herein. The “Document”, below, refers to any such manual or work. Any part of the work containing the notice as given in the Document (all of its principal authors, if it has fewer than five), and its copyright notices, a license notice saying that it is under the terms of this License, the in the History section of the license notice, and that you preserve all their Warranty Disclaimers.

#### A. Modified Version

The “Original Author” of the Document means any work containing the notices given in this License. “Contributor” means any person or entity that Distributes or modifies a Modified Version. ‘ ‘ means any person or entity that Distributes, translated, or otherwise modifies a work under this License. A “Cover Text” means any public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below. A cover text may then be added to the list of Invariant Sections in the notice which gives the Public Permission to the Document under this License. A cover notice may then be added to the beginning of each section.

#### B. COPYING IN QUANTITY

If you publish printed copies (or copies in media commonly known to be storage devices) of the Document, including a cover text, you must include the full text of Creative Commons Attribution-ShareAlike 4.0 International License, and all the license notices in the original material, and if you modify or derive another work, in or from the Document, or if you use a Cover Text, you must use the same license notice as the Document, in the same form, as the license notice in the Document. You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Invariant Sections in the license notice.

### II. MODIFICATIONS

If you distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release this Cover Text, then the document as it would have been produced by a faithful implementation of the Modified Version under this License, is a “Work on the Original Work”, and is also Part of the Original Work.

#### A. Subsequent Modifications

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Invariant Sections in the license notice, and that you preserve all their Warranty Disclaimers.

#### B. Translation

A Translation is a work produced by translating an existing Work into another language or by adapting a work for use in a particular country or region. Translations must be published under the same terms of this License, and the new license notice must include explicit permission under that license to publish this translation.

#### C. OBITUARY NOTICE

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Invariant Sections in the license notice, and that you preserve all their Warranty Disclaimers.

### III. COPYING DOCUMENTS

You may make a copy consisting of the Document and all the added or changed materials that applies this License to the Document itself, and all the added or changed materials that appears in the Document as part of the Document, and that you add no other license notices or instructions to the original version. You may extract a single document from such a collection, and distribute it individually under this License, provided you follow the rules of this License for copying a single work.

### IV. AGREEMENT WITH INDEPENDENT WORKS

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding derivative copying of that document.

### V. COLLECTING DOCUMENTS

You may make a collection consisting of the Document and all the other documents distributed under this License, which is a “collective work”, and distribute these in all media and formats that do not infringe this License. You must obey the copyright notices of others concerning their contributions, and such notices must appear in all copies distributed under this License.

### VI. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but other changes can be made free of copyright license restrictions. If you make a translation of this License, you may distribute both the original and your translation, under the same terms, provided that the translation well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.
Thank you for your attention