



What's New in Java 7?

A Glance at the Future



What's New in Java 7?

- Java 7.
- Modularization
- JSR 308.
- Library Changes.
- Language Changes.
- Enhancements to the JVM.
- What's Not Included.

What's new in Java 7

- Scheduled release date: **early 2010.**
- No platform JSR yet.
- Many exciting new features.

- Lets go over them...



Changes in Java 7

- Modularization.
- Annotations on Types.
- Fork/Join Library.
- NIO2.
- Safe Re-throw.
- Null Dereference.
- Multi-catch.
- JVM Enhancements.
- *...and more...*



What's New in Java 7?

- Java 7.
- Modularization.
- JSR 308.
- Library Changes.
- Language Changes.
- Enhancements to the JVM.
- What's Not Included.

The JAR Hell

- JARs are archaic beasts headed for extinction.
- No version management.
- No dependencies mechanism.
- Don't plug well into existing module-frameworks (e.g. OSGi).
- Don't support encapsulation.



Introducing Modules

- The *module* keyword provides a new accessibility level to the standard levels (private, default/package, protected, public).
- Here is an example of using module:

```
module com.trainologic.services@4.1;
package com.trainologic.modules.example;

public class InnerService {
    //...
}
```

Accessibility

- In the previous example, the *InnerService* class belongs to the **module** *com.trainologic.services* version 4.1
- This class is accessible from everywhere, it is public!.
- If the class or one of its members/constructors is declared *module*, it will **only be accessible from a type that belongs to the same module.**

Module Meta-data

- With Java 5, we were able to provide package meta-data (annotations) by the use of the *package-info.java* file.
- With Java 7 we will be able to provide module meta-data with the *module-info.java* file.

Example

- module-info.java:

```
module core provides infra {  
    // Must have this module visible  
    requires anotherModule;  
    // Only this module will be able to require me.  
    permits extensions;  
}
```

Project Jigsaw

- Project Jigsaw breaks the JDK into modules.
- This will enable using smaller JDK Jars in applications (based on needs).
- This will also be an excellent opportunity to get rid of deprecated APIs.





What's New in Java 7?

- Java 7.
- Modularization
- JSR 308.
- Library Changes.
- Language Changes.
- Enhancements to the JVM.
- What's Not Included.

JSR 308

- JSR 308 will enhance the Java Annotations framework and will enable placing annotations in more places than available in Java 6.
- You can now place annotations on Types:

```
List<@NonNull Object>
```

More Examples

```
Map<@NonNull String, @NonEmpty List<@ReadOnly Document>> files;  
  
void monitorTemperature() throws @Critical TemperatureException  
{  
    myString = (@NonNull String) myObject;  
  
    class Folder<F extends @Existing File> { ... }  
  
    Collection<? super @Existing File>
```

Method Receivers

- You can also apply annotations to the object on which the method is called:

```
// The read method can be invoked only on an Opened object!  
public byte[] read() @Open { ... }  
  
// The configure method can be invoked  
// only on an object which is in Configuration state!  
public void configure(Map props) @Configuration { ... }
```

Annotations on Types

- This new feature will enable “Design by Contract”.
- It will enable new opportunities for static syntax checkers to find bugs.
- One of the exciting new features for Java 7!



What's New in Java 7?

- Java 7.
- Modularization
- JSR 308.
- Library Changes.
- Language Changes.
- Enhancements to the JVM.
- What's Not Included.

Concurrency Utils

- Doug Lea, the founder of the excellent *java.util.concurrent* introduces (through JSR 166) the following new features:
 - Fork/Join framework.
 - TransferQueue.
 - Phasers.

Fork/Join

- The basic idea of the Fork/Join framework is that many tasks can be split to several concurrent threads, and the result should be merged back.
- Let's take a look at an example...

Fork/Join

- Instead of doing it single-threaded, the idea of fork/join is to split the operation to several (depends on the # of cores) concurrent threads.

```
public class Fibonacci extends RecursiveTask<Integer> {
    private final int n;

    public Fibonacci(int n) { this.n = n; }
    protected Integer compute() {
        System.out.println(Thread.currentThread().getName());
        if (n <= 1)
            return n;
        Fibonacci f1 = new Fibonacci(n - 1);
        f1.fork();
        Fibonacci f2 = new Fibonacci(n - 2);
        return f2.compute() + f1.join();
    }
}
```

Example

- The *Main* class:

```
public class Main {  
    public static void main(String[] args) {  
        ForkJoinPool pool = new ForkJoinPool(3);  
        Fibonacci fibonacci = new Fibonacci(20);  
        pool.execute(fibonacci);  
        System.out.println(fibonacci.join());  
    }  
}
```

TransferQueue

- A *BlockingQueue* on which the producers await for a consumer to take their elements.
- Usage scenarios are typically message passing applications.



Phasers

- **"Beam me up, Scotty!"**
- A Phaser is quite similar to *CyclicBarrier* and *CountDownLatch* but is more powerful and flexible.
- A *Phaser* has an associated phase-number which is of type *int*.
- A *Phaser* has a number of **unarrived parties**. Unlike *CyclicBarrier* and *CountDownLatch*, this number is dynamic.

Phasers

- A *Phaser* supports operations for arriving, awaiting, termination, deregistration and registration.
- When all the parties arrive, the *Phaser* advances (increments its phase-number).
- Phasers also supports *ForkJoinTasks*!



NIO.2

- JSR 203 adds new APIs to the NIO package.
- Main features:
 - BigBuffers – Buffers that support more than *Integer.MAX_VALUE* elements.
 - Filesystem API.
 - Asynchronous Channels.

Filesystem API

- Here is an example of using events on filesystem:

```
FileSystem fs = ...
FileReference file = ...
WatchService watcher = fs.newWatchService();
file.register(watcher, WatchEvent.MODIFY_EVENT);
for (;;) {
    WatchKey key = watcher.take(); // wait for the next key
    List<WatchEvent> events = key.takeEvents();
    for (WatchEvent event: events) {
        if ((event & WatchEvent.MODIFY_EVENT) > 0) {
            FileChannel fc = FileChannel.open(file,
                OpenFlag.READ);
            FileLock lock = fc.lock();
            // do something
            lock.unlock();
            fc.close();
        }
    }
    key.reset();
}
```



What's New in Java 7?

- Java 7.
- Modularization
- JSR 308.
- Library Changes.
- Language Changes.
- Enhancements to the JVM.
- What's Not Included.

Multi-Catch

- Finally, we can treat several *Exception* types in **one catch block**:

```
try {
    Class.forName("Kuku").newInstance();
} catch (InstantiationException, IllegalAccessException e) {
    e.printStackTrace();
} catch (ClassNotFoundException e) {
    // do something else
}
```

Safe Re-throw

- Currently the following code will not compile:

```
public void foo() throws IOException, SQLException {
    try {
        // do something that may throw IOException or
        // SQLException
    } catch (Exception e) {
        // do something
        throw(e);
    }
}
```

Safe Re-throw

- The proposed syntax is with the ***final*** keyword:

```
public void foo() throws IOException, SQLException {  
    try {  
        // do something that may throw IOException or  
        // SQLException  
    } catch (final Exception e) {  
        // do something  
        throw(e);  
    }  
}
```

Null-safe Dereference

- A new operator will be introduced:

```
currentCompany?.getCEO()?.getAddress()?.getStreet()
```

COOL

Type Inference

- How about easing the cumbersome generic-types initialization?

```
Map<String, Integer> map = new HashMap<String, Integer>();
```

- Now it will be:

```
Map<String, Integer> map = new HashMap<>();
```



What's New in Java 7?

- Java 7.
- Modularization
- JSR 308.
- Library Changes.
- Language Changes.
- Enhancements to the JVM.
- What's Not Included.

Garbage First (G1)

- A new Garbage Collector is developed by Sun for Java 7.
- It will be called G1.
- This GC will split the memory into multiple regions (unlike 2 in the current version) and will (most likely) perform faster than the current parallel collectors.



Compressed 64-bit Pointers

- One of the downsides of a 64-bit JVM is that low level pointers take a lot of space.
- Java7 will include a “compressed 64-bit pointers” that will consume less space and perform just like 32-bit pointers.

InvokeDynamic

- A new bytecode instruction will (finally) be added to the JVM bytecode.
- This new instruction will greatly simplify dynamic languages implementations on the JVM.
- Dynamic Language?? – Watch for the Groovy presentation!!



What's New in Java 7?

- Java 7.
- Modularization
- JSR 308.
- Library Changes.
- Language Changes.
- Enhancements to the JVM.
- What's Not Included.

Not Included

- The following (promised) features will **not** be included in Java 7:
 - **Closures.**
 - **Refined Generics.**
 - Properties.
 - Operator overloading.
 - BigDecimal syntax.
 - Language level XML support.



Thank You

Q&A

Shimi Bandiel,
VP R&D, Trainologic LTD