Java 7 - Peeks & Pokes

09.12.2011
Presentation for anderScore Goldschmiede
Cologne

About Author

- Maz Rashid

- Freelancer
- Senior Java Architect
- Certified Project Manager (IPMA-Level C, PMP)

- Devoted to Object Oriented Software Design
- Working in long term projects
- high performance, high concurrency environment

- Finance, Telecom, Logistics, Energy Trading, Workforce Management

- More on http://www.mazcity.de

Performance

- Garbage-First-Collector (G1)
  ➔ experimental, propably not ready/stable for production

- Java-Hotspot Improvements
  ➔Further development of the hotspot compiler
  ➔ Performance-Improvements

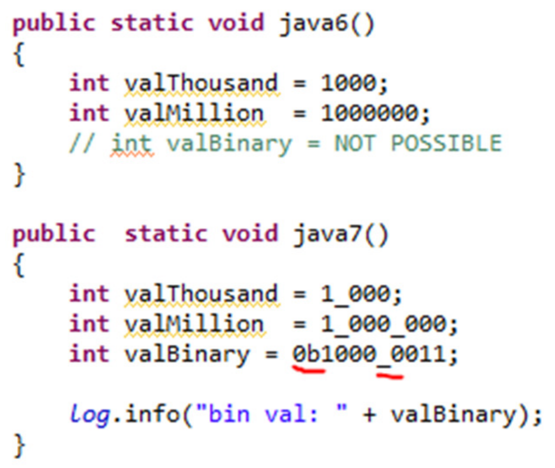
Swing Enhancements

- Support for Non-Rectangular window forms.
- Better support for TrayIcon
- Some cosmetic changes in the AWT-API / Implementation.

mazcity Consulting Design Development Let‘s do IT better!

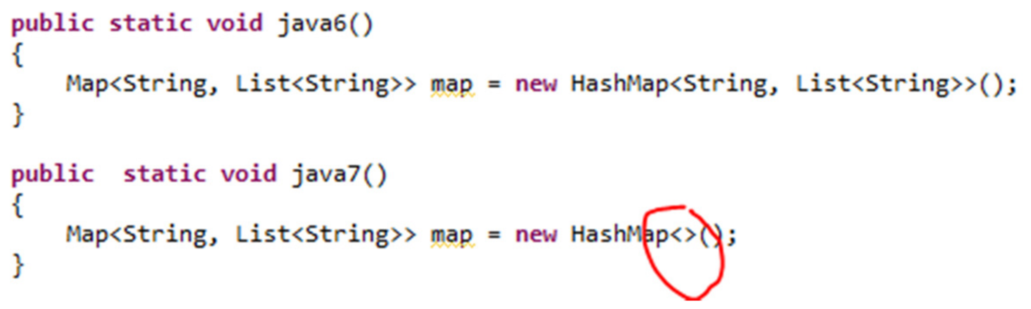
Literals

- An underscore can be used as delimiter for numeric literals
- Binary literals by prefix ‚0b‘
- Syntactical sugar
- Userfull to enhance readability



```
public static void java6()
{
    int valThousand = 1000;
    int valMillion  = 1000000;
    // int valBinary = NOT POSSIBLE
}

public  static void java7()
{
    int valThousand = 1_000;
    int valMillion  = 1_000_000;
    int valBinary = 0b1000_0011;

    log.info("bin val: " + valBinary);
}
```

Diamond Operator

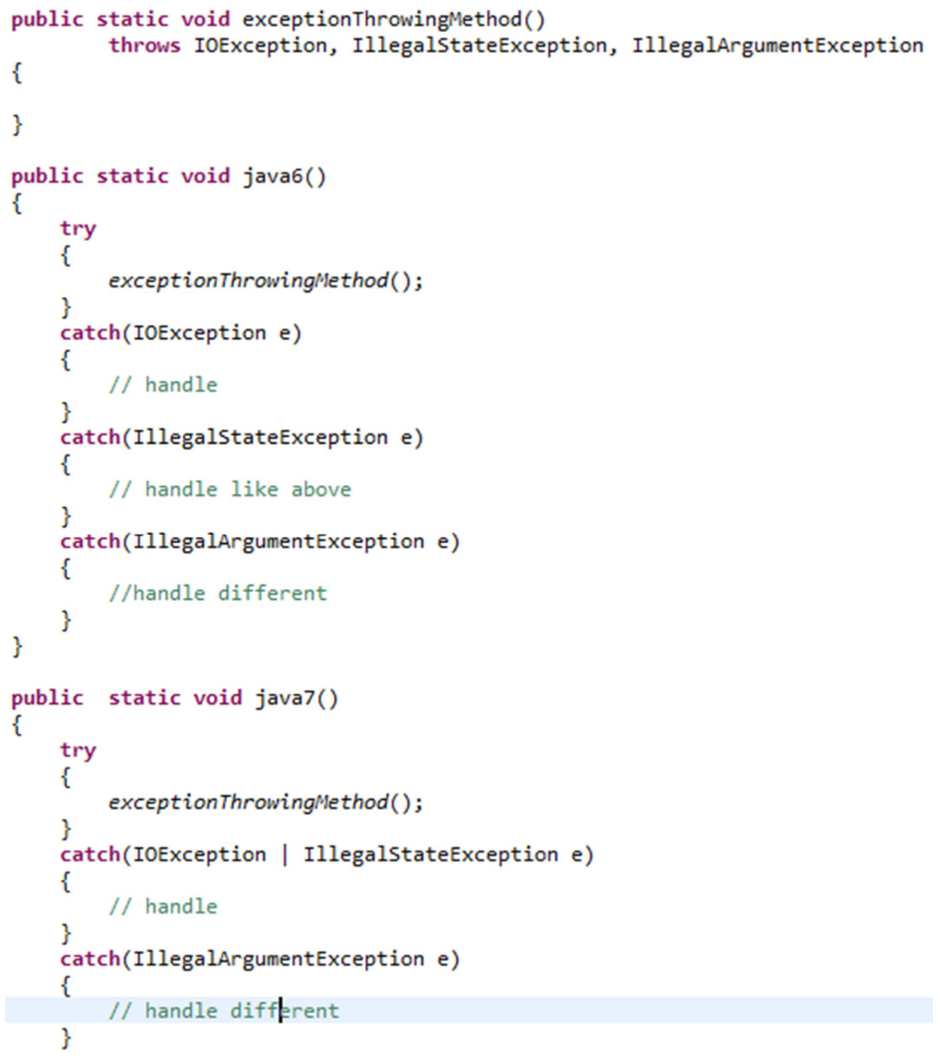


```
public static void java6()
{
    Map<String, List<String>> map = new HashMap<String, List<String>>();
}

public  static void java7()
{
    Map<String, List<String>> map = new HashMap<>();
}
```

- Syntactical sugar
- Already given by modern IDEs, so no real advantage.

Multiple Catch (Exception)

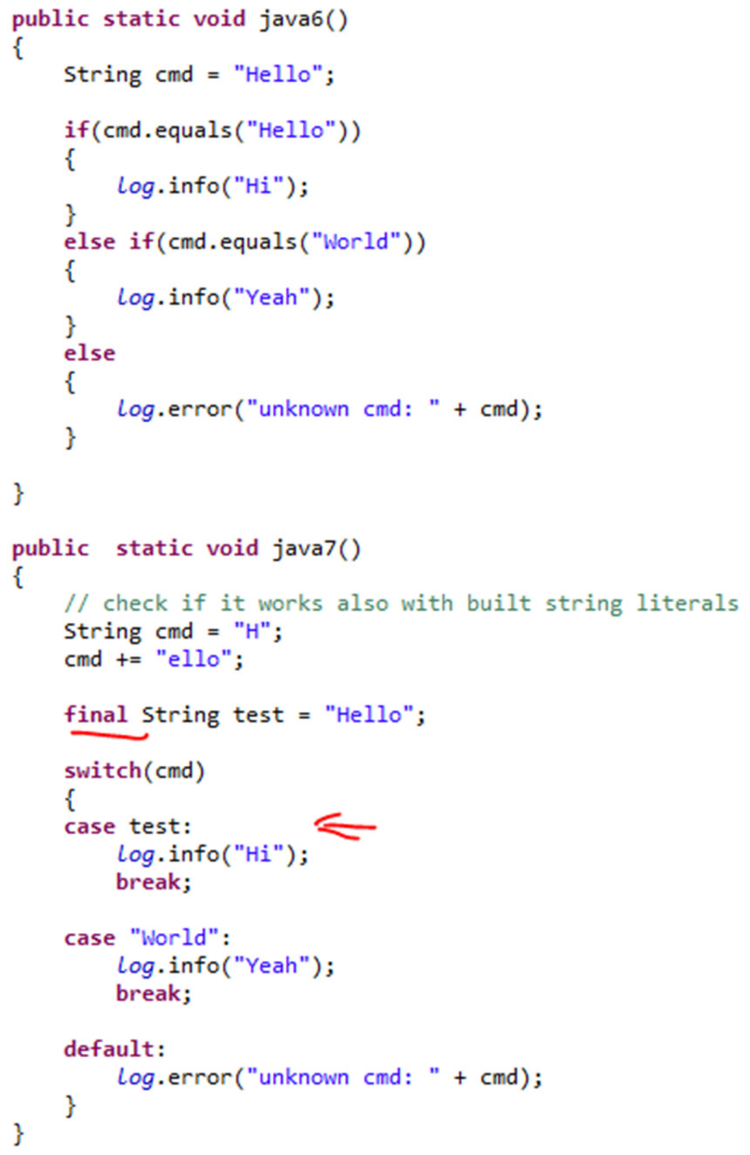
- Multiple Exceptions can be caught and handled in one block.
- The bar ‚|' is used as list seperator in the catch statement.
- Any combination (catching one Exception, catching list of exceptions) is allowed.



```
public static void exceptionThrowingMethod()
        throws IOException, IllegalStateException, IllegalArgumentException
{

}

public static void java6()
{
    try
    {
        exceptionThrowingMethod();
    }
    catch(IOException e)
    {
        // handle
    }
    catch(IllegalStateException e)
    {
        // handle like above
    }
    catch(IllegalArgumentException e)
    {
        //handle different
    }
}

public  static void java7()
{
    try
    {
        exceptionThrowingMethod();
    }
    catch(IOException | IllegalStateException e)
    {
        // handle
    }
    catch(IllegalArgumentException e)
    {
        // handle different
    }
```

String in switch-case

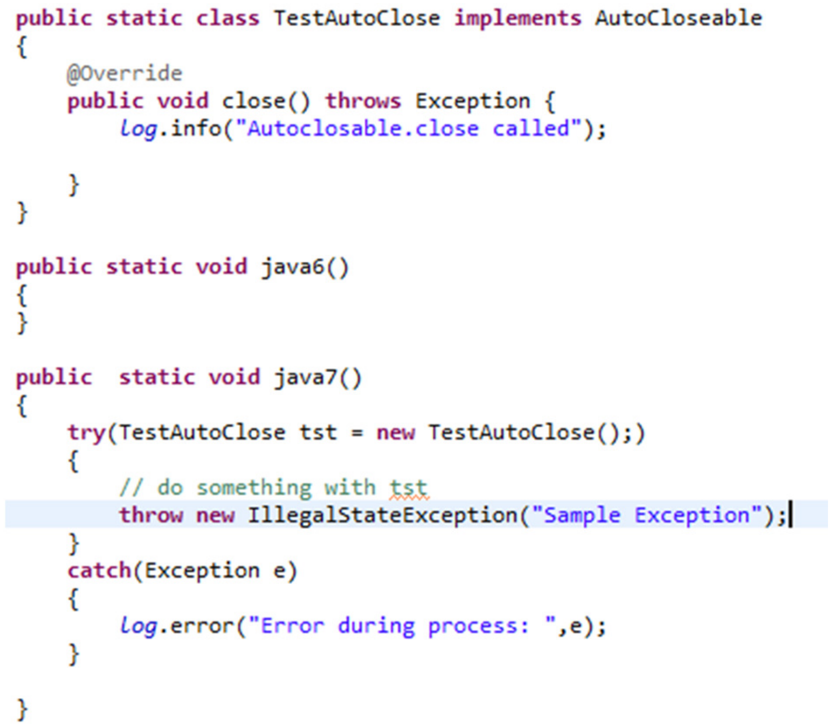
- Before Java7 only numerical values or enumerations could be used in switch statements
- For String dispatching an ugly chain of String.equals in if-else statements were to be used.
- With Java7 finaly String literals are allowed in switch statements.
- Only constant values allowed in case (final variables, so no variables)



```
public static void java6()
{
    String cmd = "Hello";

    if(cmd.equals("Hello"))
    {
        log.info("Hi");
    }
    else if(cmd.equals("World"))
    {
        log.info("Yeah");
    }
    else
    {
        log.error("unknown cmd: " + cmd);
    }

}

public  static void java7()
{
    // check if it works also with built string literals
    String cmd = "H";
    cmd += "ello";

    final String test = "Hello";

    switch(cmd)
    {
    case test:
        log.info("Hi");
        break;

    case "World":
        log.info("Yeah");
        break;

    default:
        log.error("unknown cmd: " + cmd);
    }
}
```

Automatic Resource Closing I

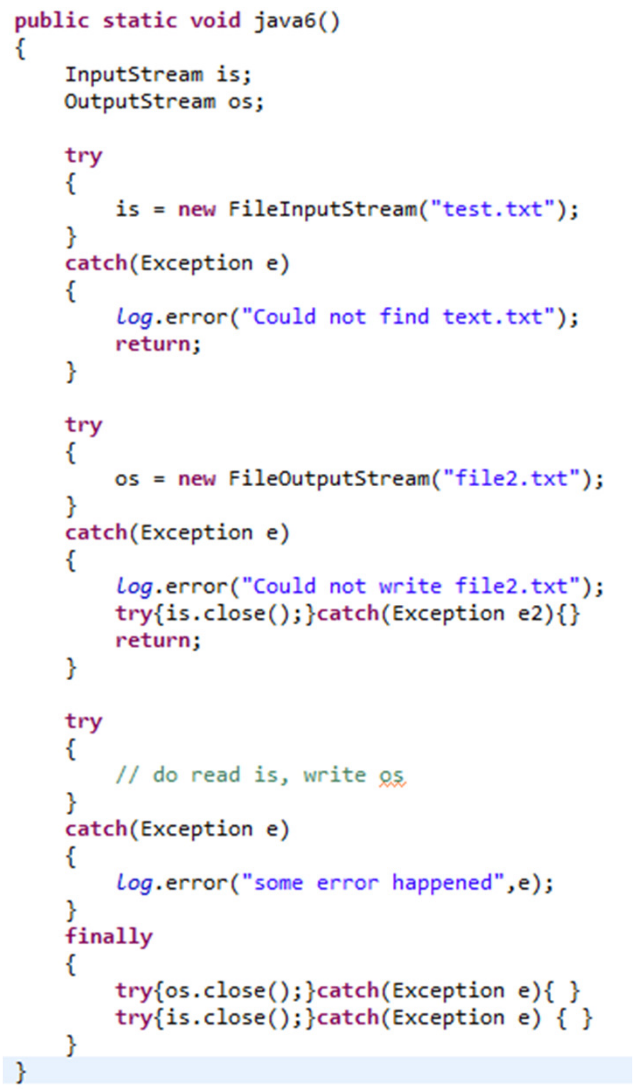
- Introduction of AutoClosable interface
- Introduction of automatic resource handling in try-catch
- JVM ensures that the autoclosable resources will be closed when the control block is left.
- ➔No forgotten open resources anymore
- ➔Could be used as „desctructor“ or finalizer with a defined call time.



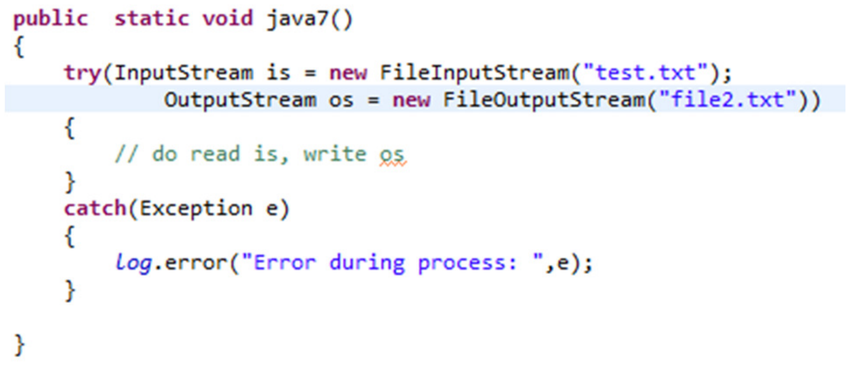
```java
public static class TestAutoClose implements AutoCloseable
{
    @Override
    public void close() throws Exception {
        log.info("Autoclosable.close called");

    }
}

public static void java6()
{
}

public  static void java7()
{
    try(TestAutoClose tst = new TestAutoClose();)
    {
        // do something with tst
        throw new IllegalStateException("Sample Exception");
    }
    catch(Exception e)
    {
        log.error("Error during process: ",e);
    }

}
```

Automatic Resource Closing II

Consulting
Design
Development
Let‘s do IT better!



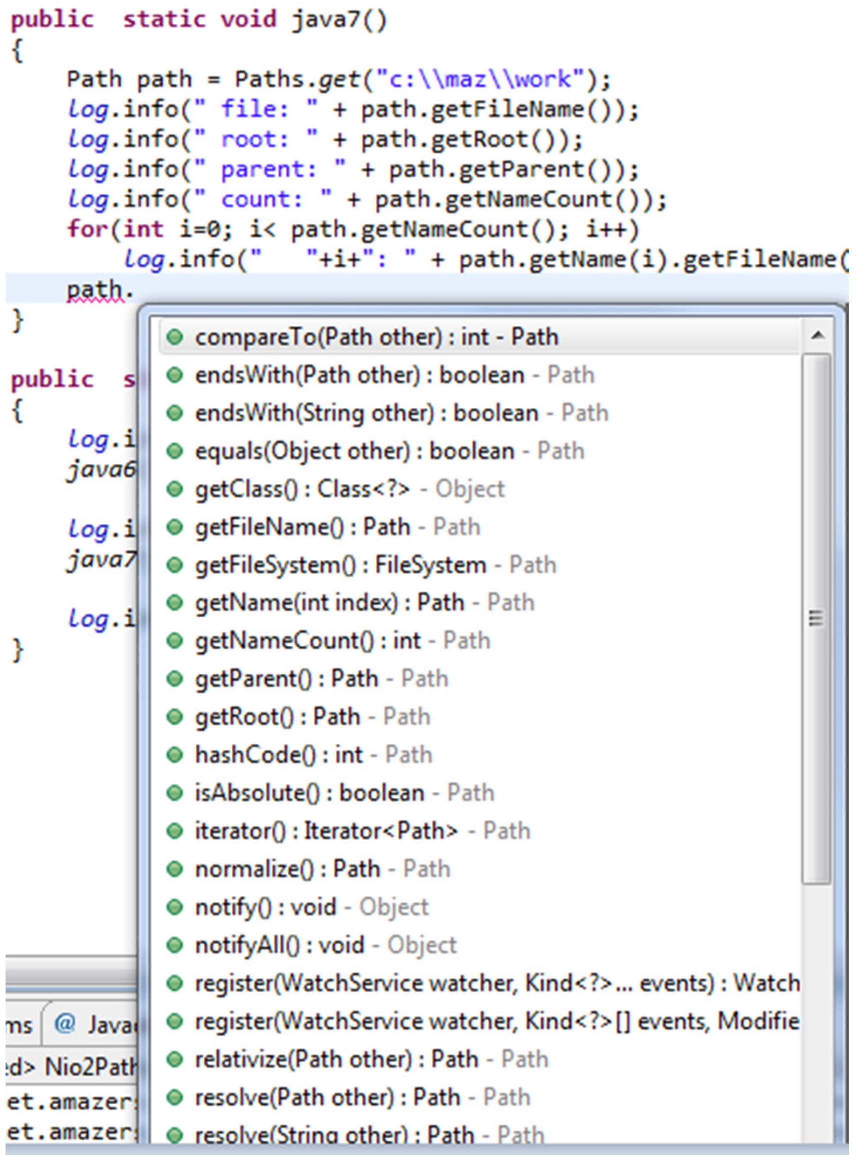
```
public static void java6()
{
    InputStream is;
    OutputStream os;

    try
    {
        is = new FileInputStream("test.txt");
    }
    catch(Exception e)
    {
        log.error("Could not find text.txt");
        return;
    }

    try
    {
        os = new FileOutputStream("file2.txt");
    }
    catch(Exception e)
    {
        log.error("Could not write file2.txt");
        try{is.close();}catch(Exception e2){}
        return;
    }

    try
    {
        // do read is, write os
    }
    catch(Exception e)
    {
        log.error("some error happened",e);
    }
    finally
    {
        try{os.close();}catch(Exception e){ }
        try{is.close();}catch(Exception e) { }
    }
}
```



```
public  static void java7()
{
    try(InputStream is = new FileInputStream("test.txt");
            OutputStream os = new FileOutputStream("file2.txt"))
    {
        // do read is, write os
    }
    catch(Exception e)
    {
        log.error("Error during process: ",e);
    }

}
```

NIO2 API – java.nio.file.*

- New NIO package for File/Path handling
- FileSystems, Paths, Files
- ZipFileSystem-Provider
- NFS-Sample



```
public  static void java7()
{
    Path path = Paths.get("c:\\maz\\work");
    log.info(" file: " + path.getFileName());
    log.info(" root: " + path.getRoot());
    log.info(" parent: " + path.getParent());
    log.info(" count: " + path.getNameCount());
    for(int i=0; i< path.getNameCount(); i++)
        log.info("   "+i+": " + path.getName(i).getFileName(
    path.
}

public  s
{
    log.i
    java6

    log.i
    java7

    log.i
}
```

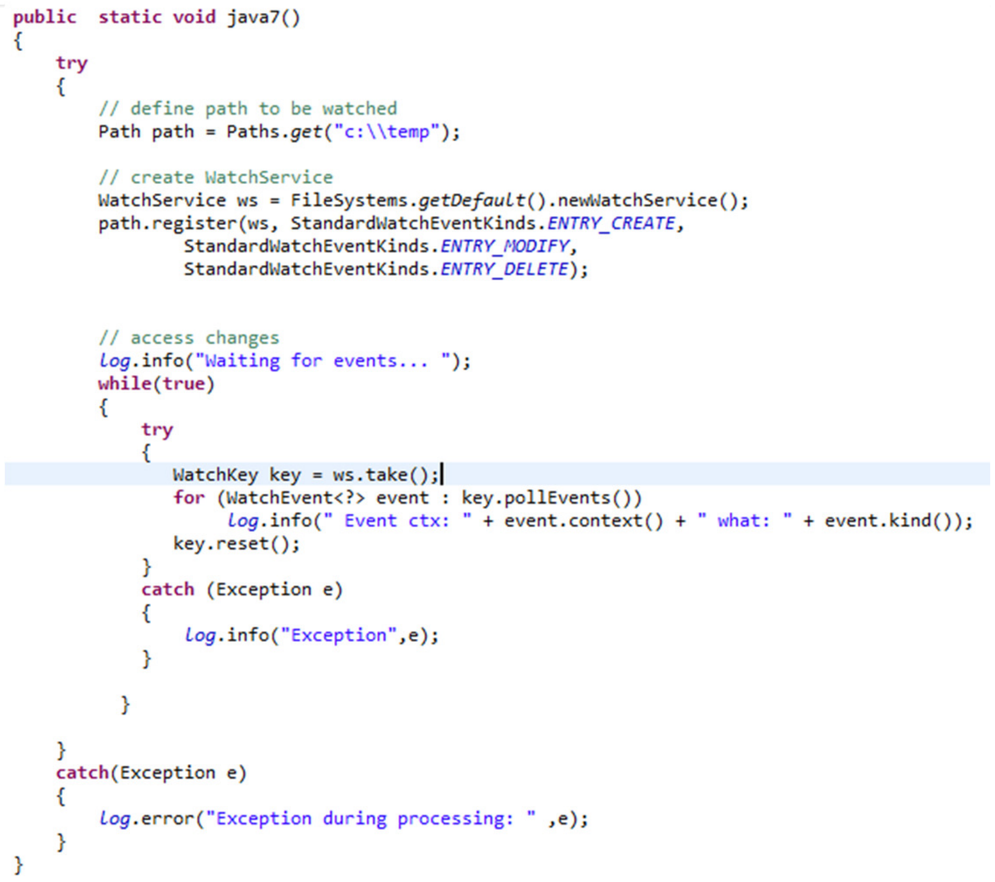
```
compareTo(Path other) : int - Path
endsWith(Path other) : boolean - Path
endsWith(String other) : boolean - Path
equals(Object other) : boolean - Path
getClass() : Class<?> - Object
getFileName() : Path - Path
getFileSystem() : FileSystem - Path
getName(int index) : Path - Path
getNameCount() : int - Path
getParent() : Path - Path
getRoot() : Path - Path
hashCode() : int - Path
isAbsolute() : boolean - Path
iterator() : Iterator<Path> - Path
normalize() : Path - Path
notify() : void - Object
notifyAll() : void - Object
register(WatchService watcher, Kind<?>... events) : Watch
register(WatchService watcher, Kind<?>[] events, Modifie
relativize(Path other) : Path - Path
resolve(Path other) : Path - Path
resolve(String other) : Path - Path
```

```
ms | @ Java
d> Nio2Path
et.amazer
et.amazer
```

NIO2 API - WatchService

Consulting
Design
Development
Let's do IT better!

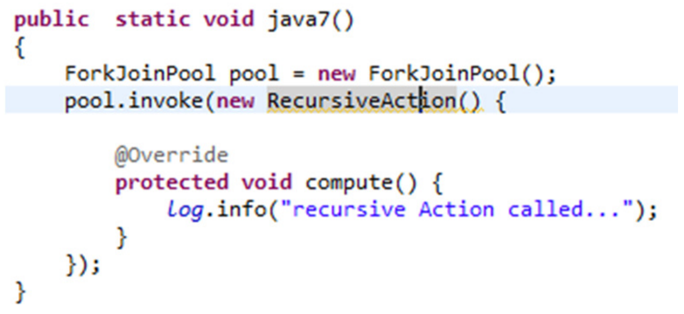
- Subsribe for notifications of changes on folders
- A Queue will contain all Events (Create, Modified, Deleted)



```
public  static void java7()
{
    try
    {
        // define path to be watched
        Path path = Paths.get("c:\\temp");

        // create WatchService
        WatchService ws = FileSystems.getDefault().newWatchService();
        path.register(ws, StandardWatchEventKinds.ENTRY_CREATE,
                StandardWatchEventKinds.ENTRY_MODIFY,
                StandardWatchEventKinds.ENTRY_DELETE);


        // access changes
        log.info("Waiting for events... ");
        while(true)
        {
            try
            {
               WatchKey key = ws.take();
               for (WatchEvent<?> event : key.pollEvents())
                    log.info(" Event ctx: " + event.context() + " what: " + event.kind());
               key.reset();
            }
            catch (Exception e)
            {
                log.info("Exception",e);
            }

          }

    }
    catch(Exception e)
    {
        log.error("Exception during processing: " ,e);
    }
}
```

ForkJoin Framework

- Main classes: ForkJoinPool and ForkJoinTask
- Framework for use of parallel cores
- Divide and Conquer
- Workers work on basis of „Steal-Work“, so idle workers will take work from busy workers.



```
public  static void java7()
{
    ForkJoinPool pool = new ForkJoinPool();
    pool.invoke(new RecursiveAction() {

        @Override
        protected void compute() {
            log.info("recursive Action called...");
        }
    });
}
```

- Need more insight:
  - What is the difference to Executer-Service?
  - How is the whole „Thing“ working.

Invoke Dynamic

mazcity Consulting Design Development Let‘s do IT better!

- For integration of languages with dynamic type system (i.e. Ruby, Python, etc.)
- Package: java.lang.invoke
- Main classes: MethodHandle and CallSite
- Allows invocation of dynamic methods. Not easy to write a sample code, as Java do not allow creating dynamic types/methods.

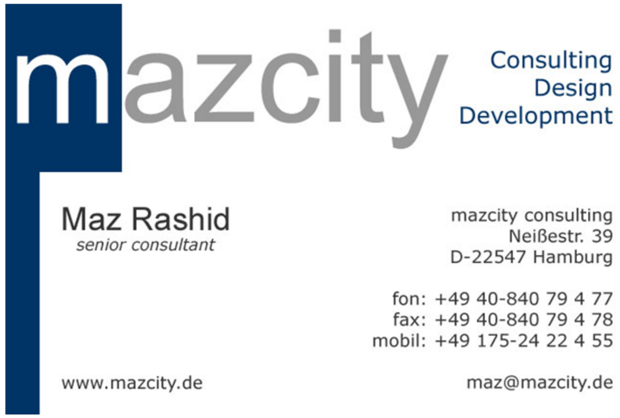
  Marc Hoffmann has created a small example, how Java bytecode with an *invokedynamic* instruction can be created using the ASM library. As the *invokedynamic* instruction has been created for new script language, the normal Java compiler will not emit such instructions. The example is available for download (http://download.eclipselab.org/jacoco/docs/20110912-invoke-dynamic-example.zip)

mazcity Consulting Design Development
Let‘s do IT better!

Thank You

- Further Infos see Oracle Release-Notes: http://www.oracle.com/technetwork/java/javase/jdk7-relnotes-418459.html

- Comments, questions, suggestions?
  Contact me on:



mazcity Consulting Design Development

Maz Rashid
senior consultant

mazcity consulting
Neißestr. 39
D-22547 Hamburg

fon: +49 40-840 79 4 77
fax: +49 40-840 79 4 78
mobil: +49 175-24 22 4 55

www.mazcity.de
maz@mazcity.de