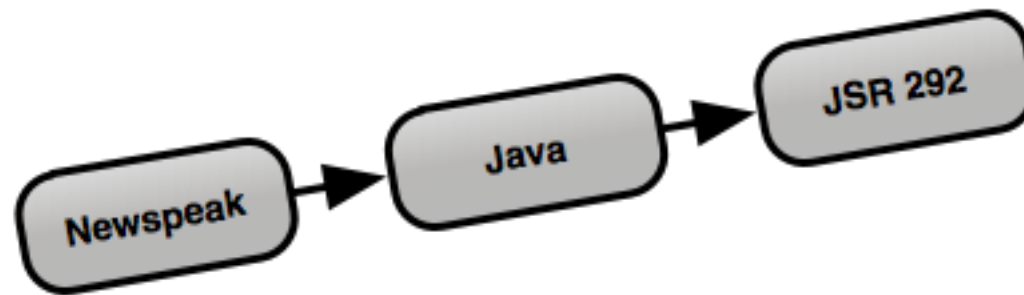


Metaprogramming

Newspeak on the JVM

Hasso-Plattner-Institut Potsdam
Software Architecture Group
Prof. Dr. Robert Hirschfeld
Konstantin Haase, Tim Felgentreff
<http://www.hpi.uni-potsdam.de/swa/>
09.02.2011

Outline



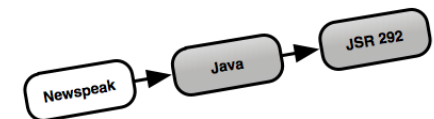
Newspeak

- Gilad Bracha
- Derived from Squeak/Smalltalk and inspired by Self and Beta
- Focus on increased modularity and security

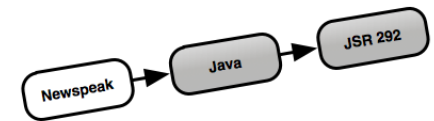
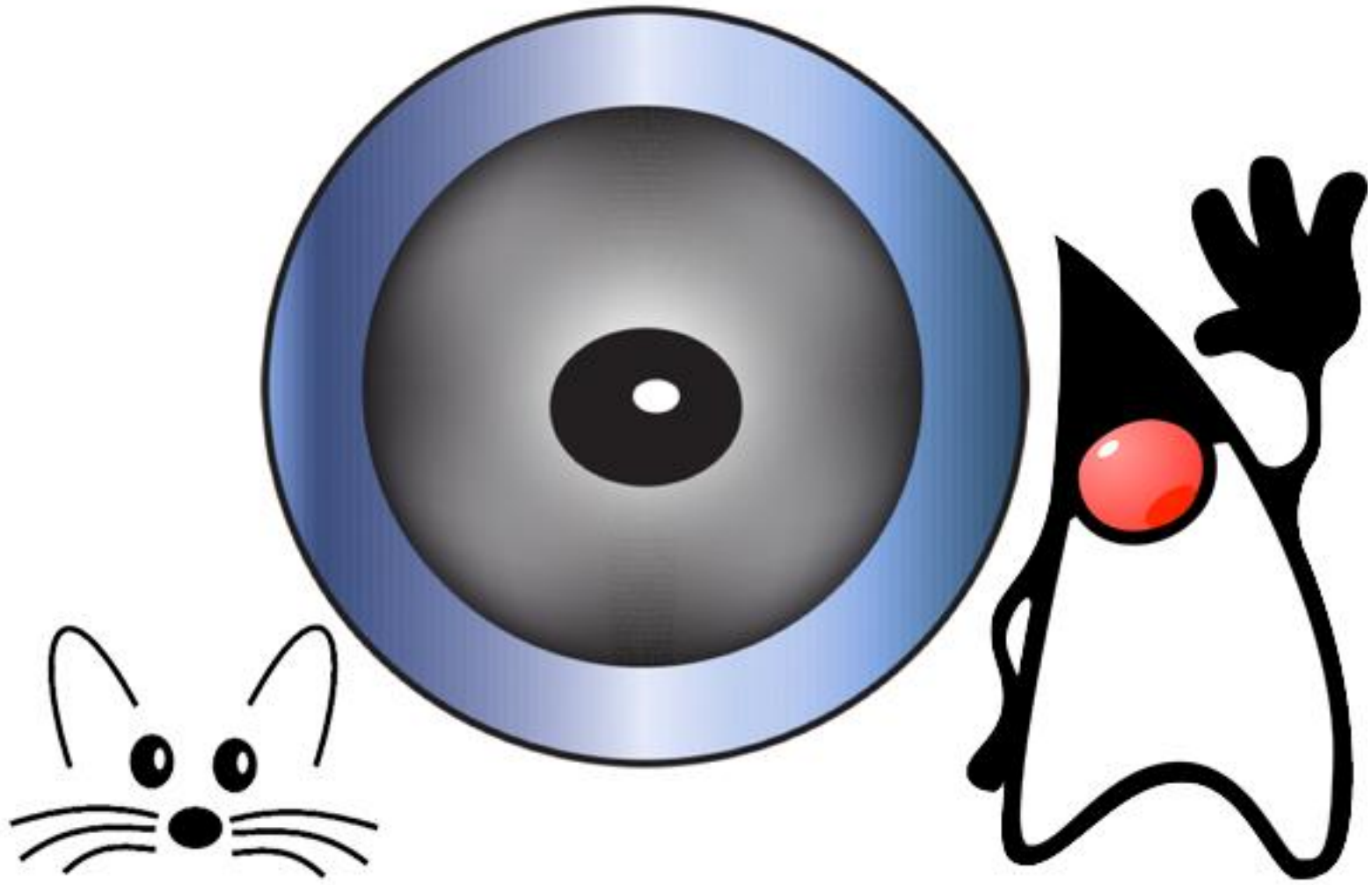


Photo: Dennis Hamilton

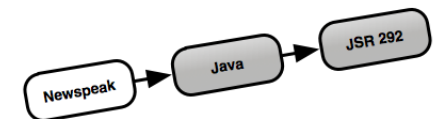
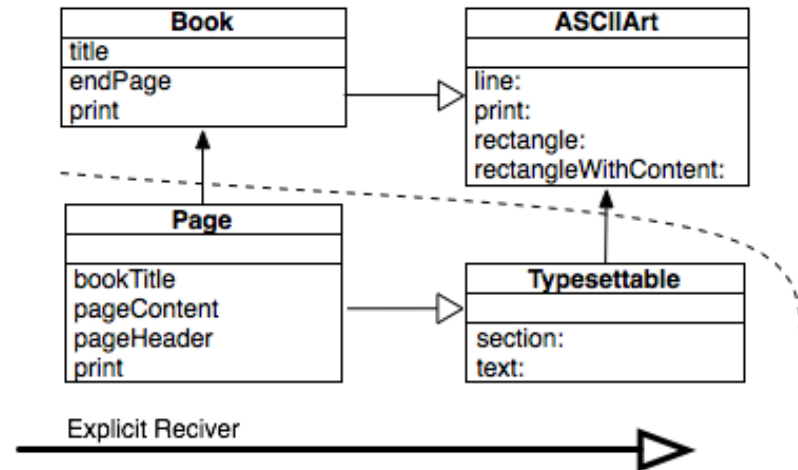
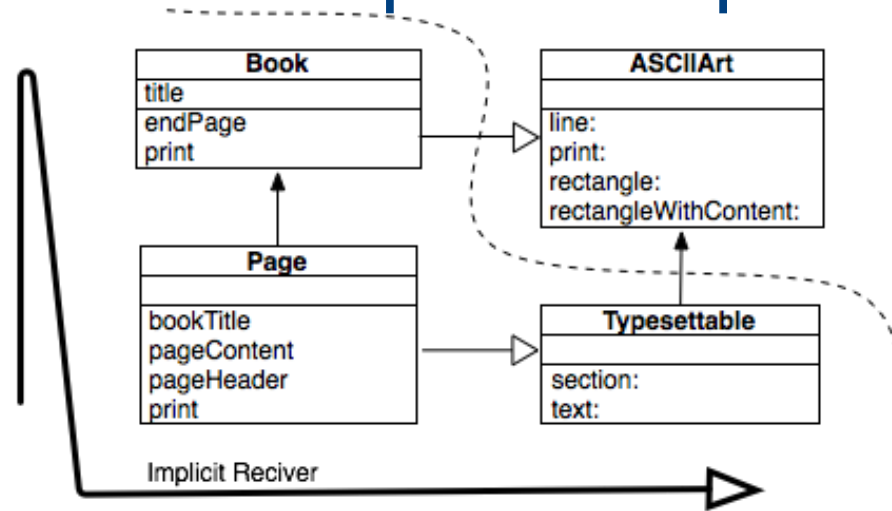
Late bound, scoped symbol resolution



Demo

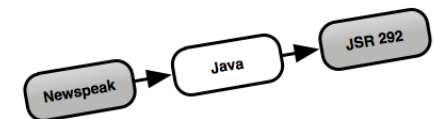


The Newspak Dispatch

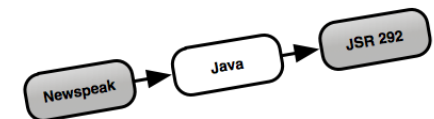
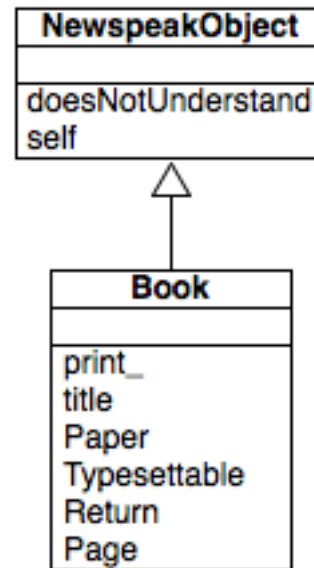


Newspeak Java Runtime Library

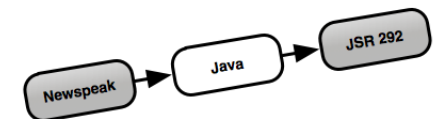
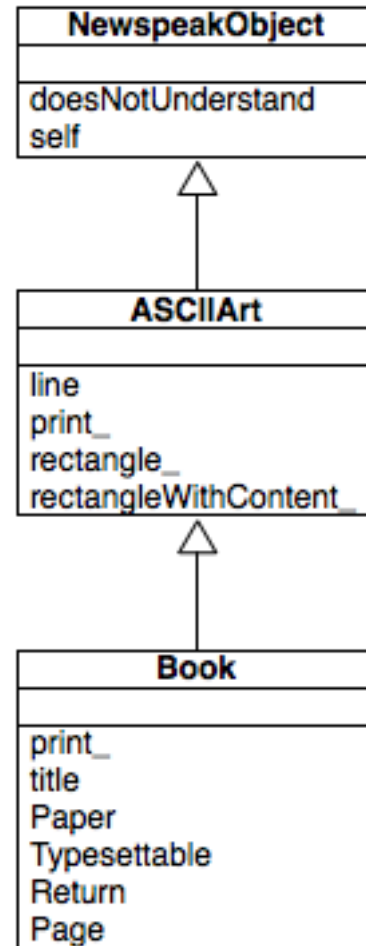
Book
print_ title Paper Typesettable Return Page



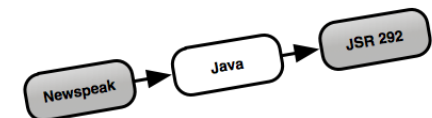
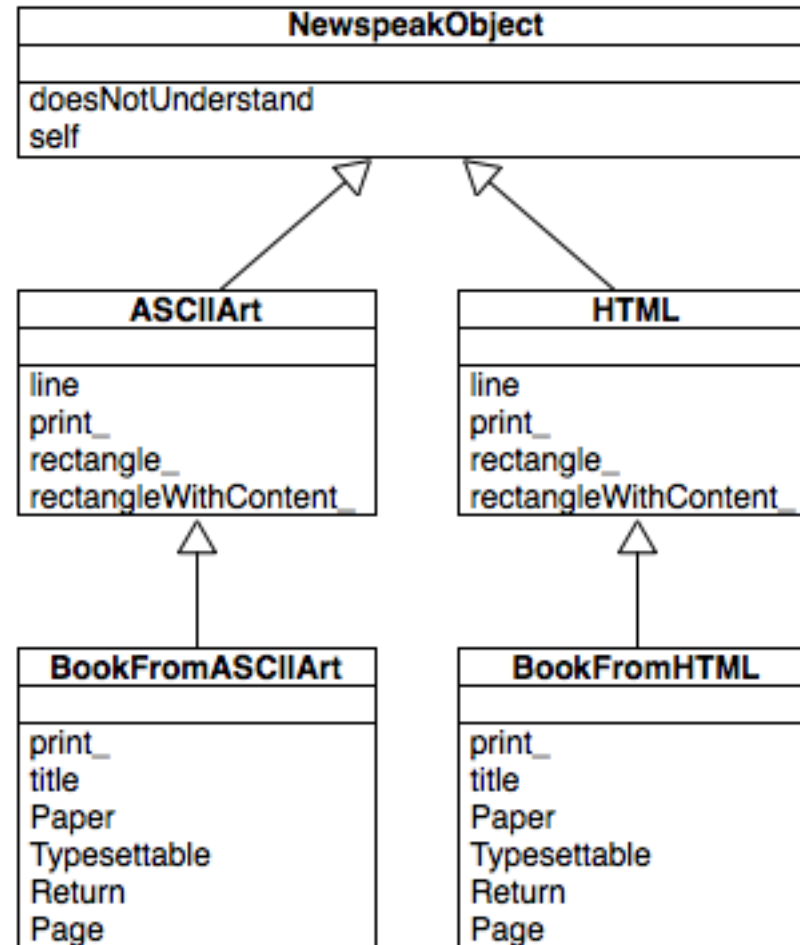
Newspeak Java Runtime Library



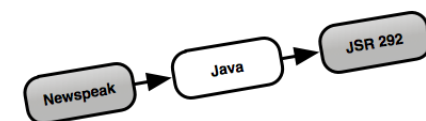
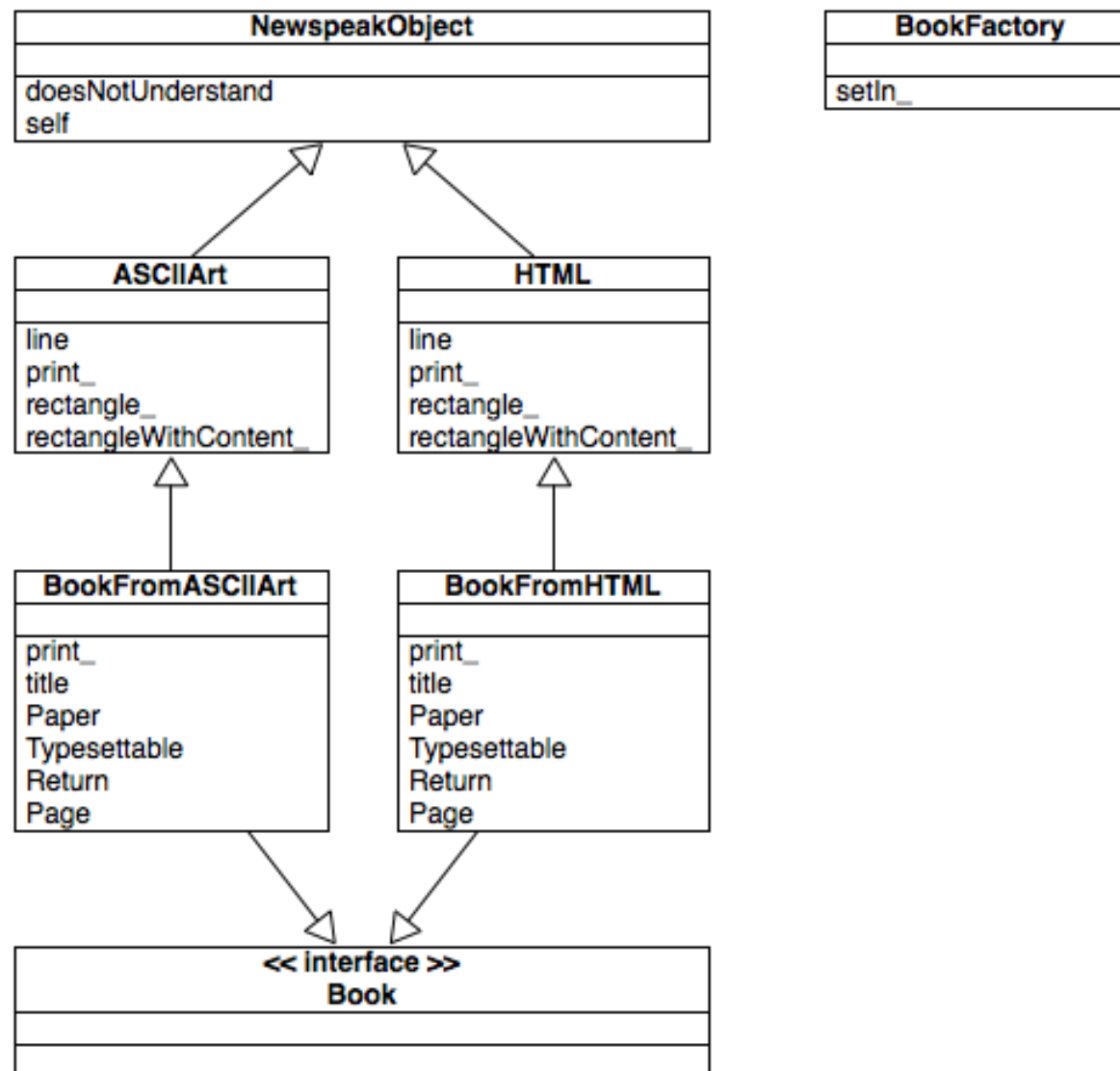
Newspeak Java Runtime Library



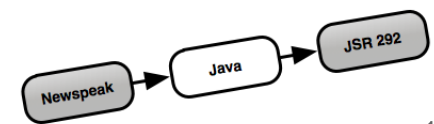
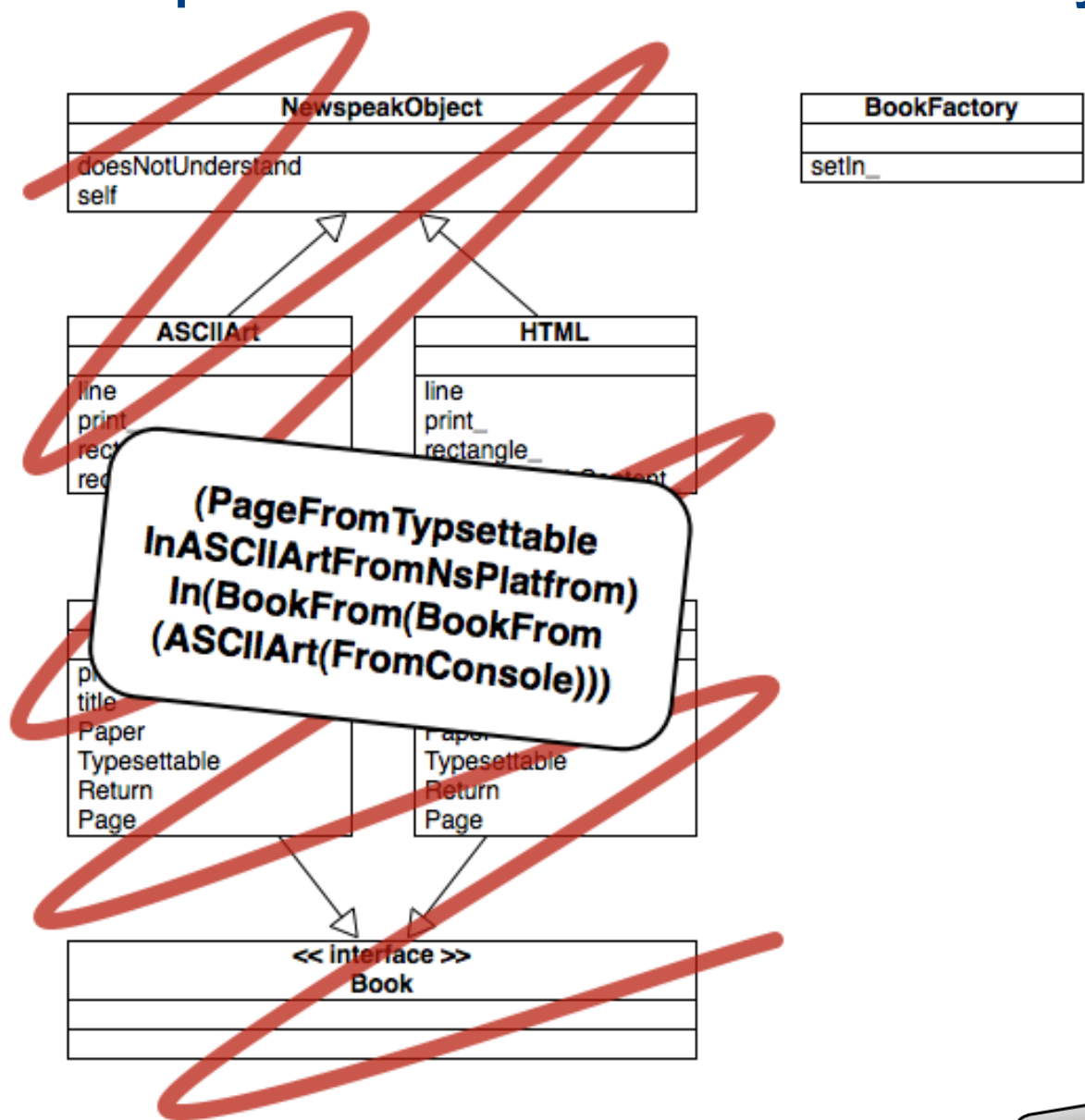
Newspeak Java Runtime Library



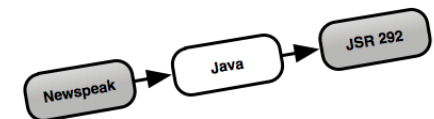
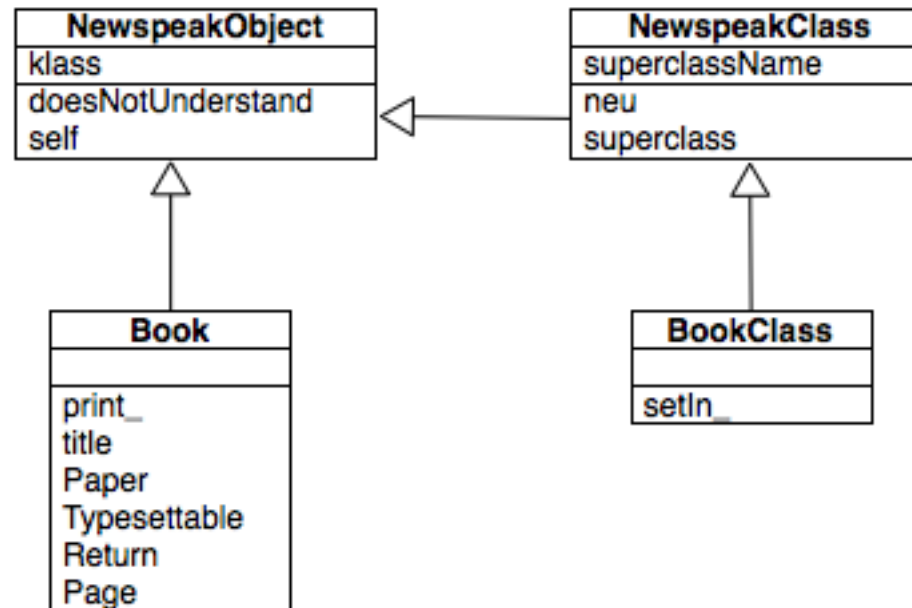
Newspeak Java Runtime Library



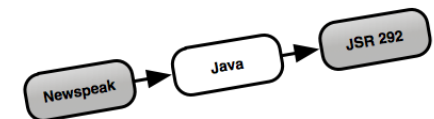
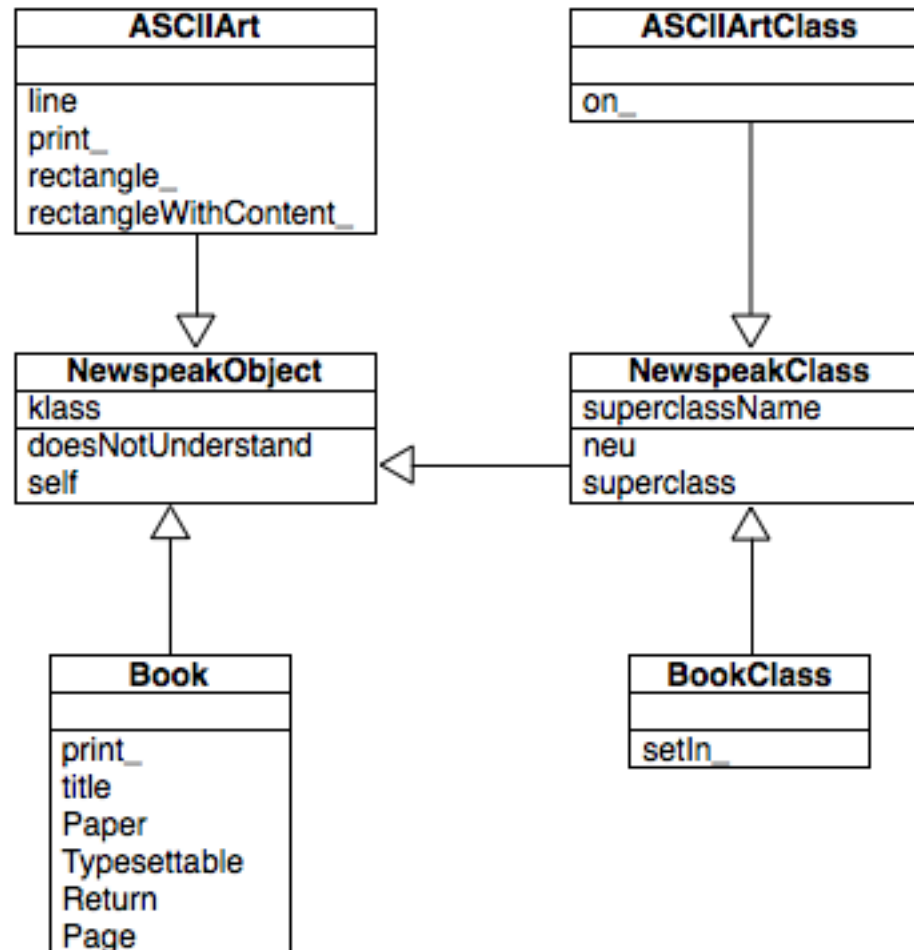
Newspeak Java Runtime Library



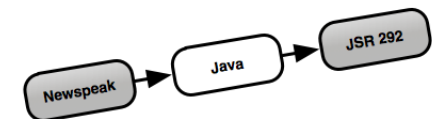
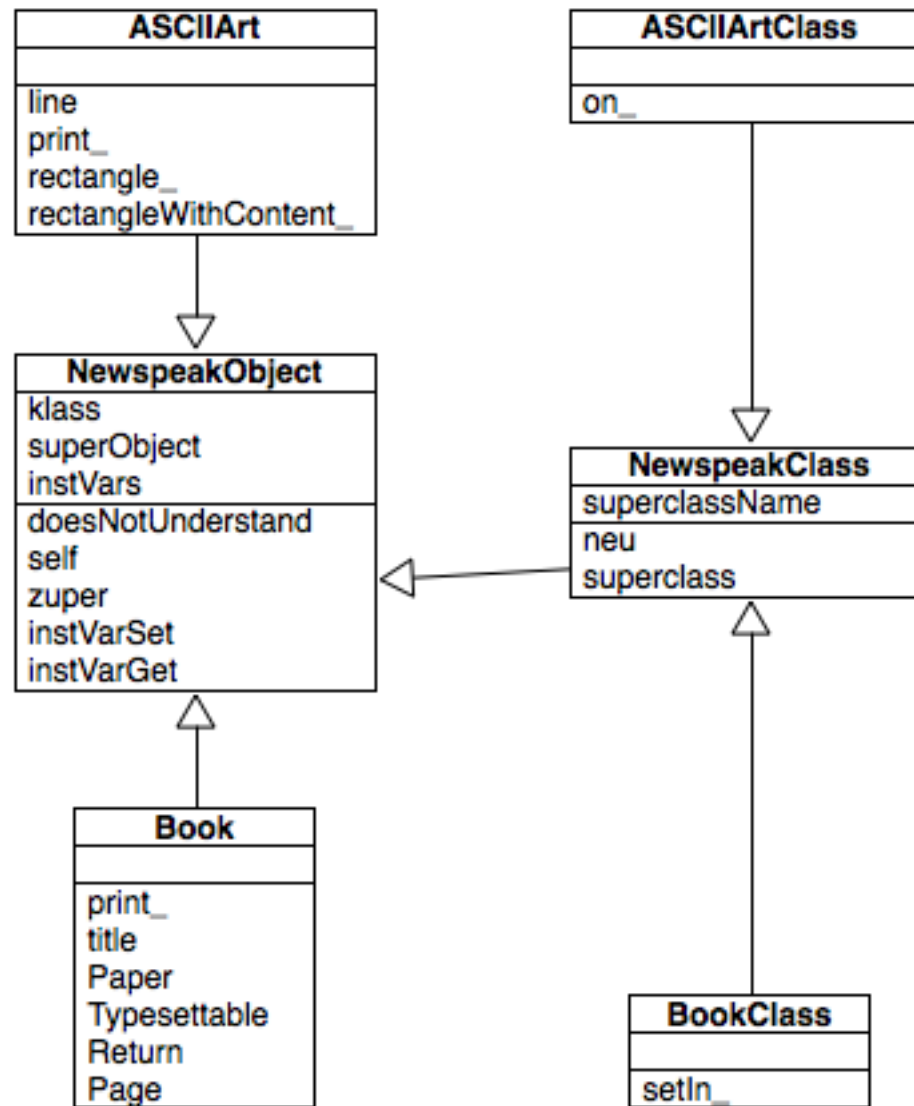
Newspeak Java Runtime Library



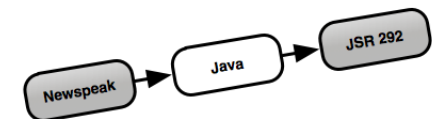
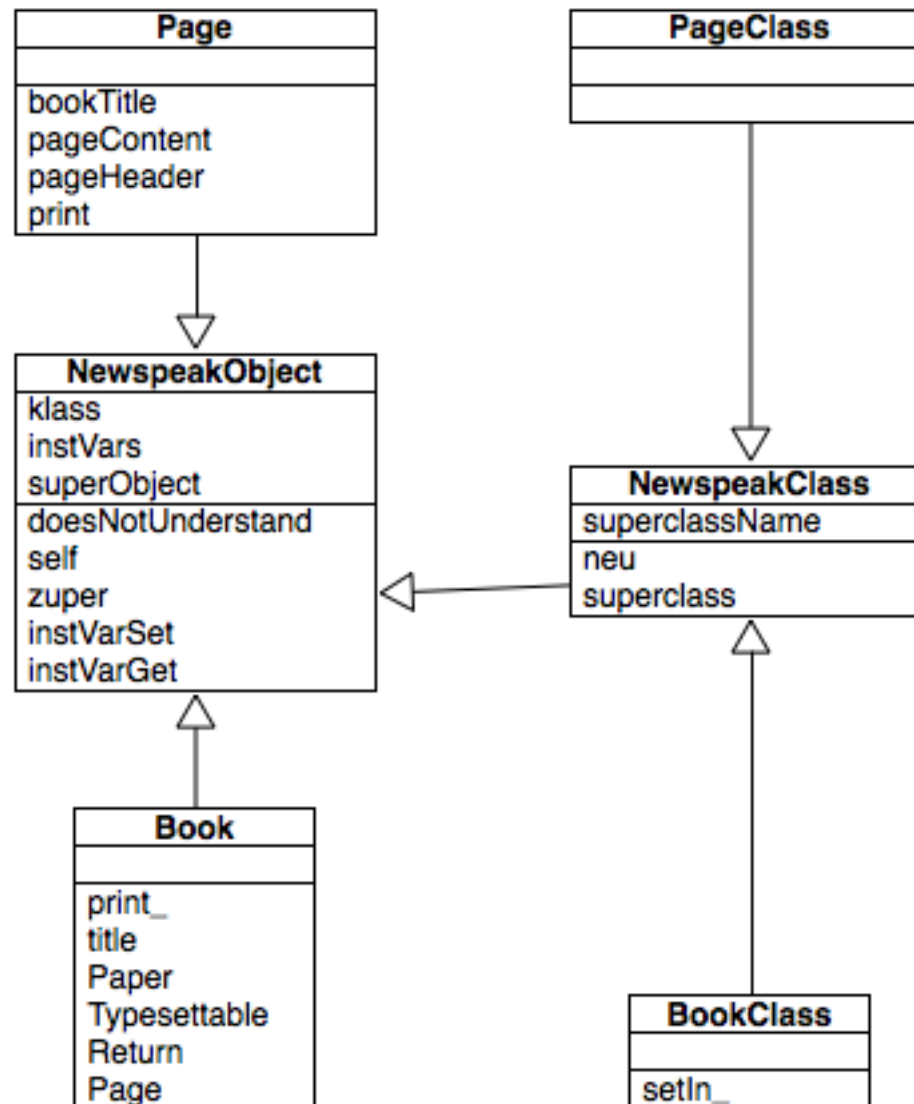
Newspeak Java Runtime Library



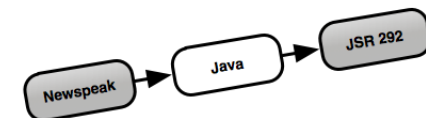
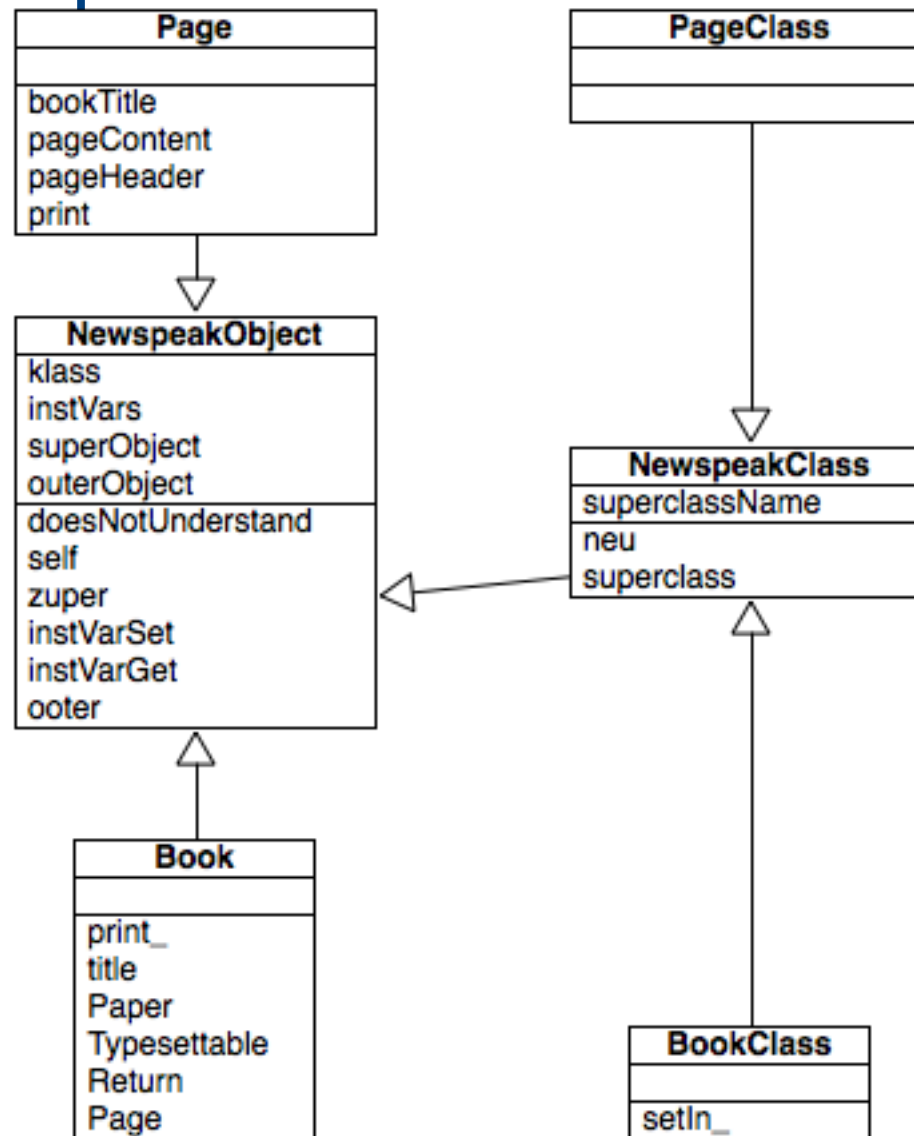
Newspeak Java Runtime Library



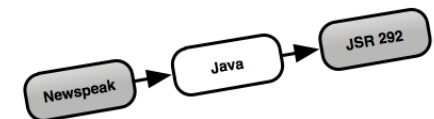
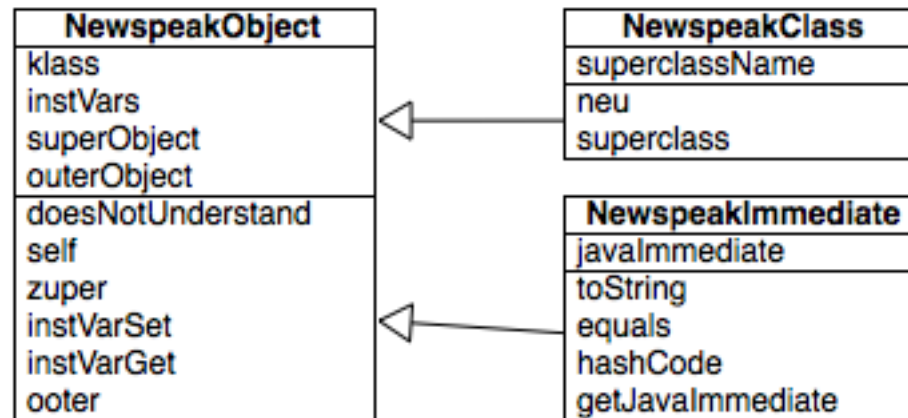
Newspeak Java Runtime Library



Newspeak Java Runtime Library



Newspeak Java Runtime Library



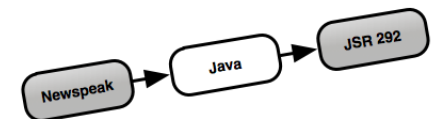
Newspeak Java Runtime Library

doesNotUnderstand: aMessage

```
((outerObject respondsTo: aMessage) and: [ aMessage isImplicitCall ])  
  ifTrue: [ ↑ outerObject perform: aMessage ].
```

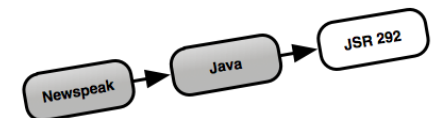
```
(superObject respondsTo: aMessage)  
  ifTrue: [ ↑ superObject perform: aMessage ].
```

```
↑ super doesNotUnderstand: aMessage ]
```

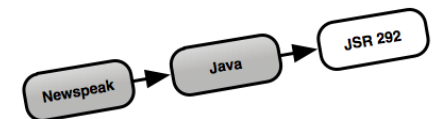
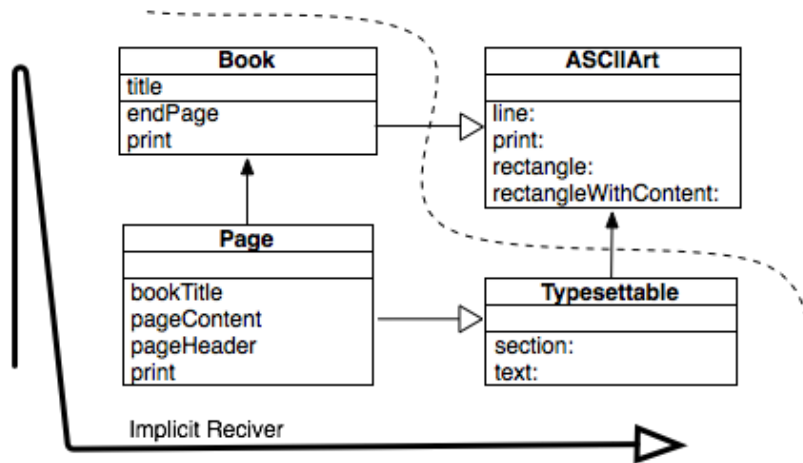


JSR-292 aka InDy

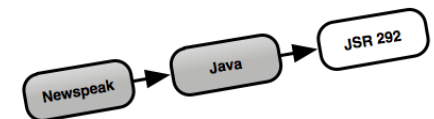
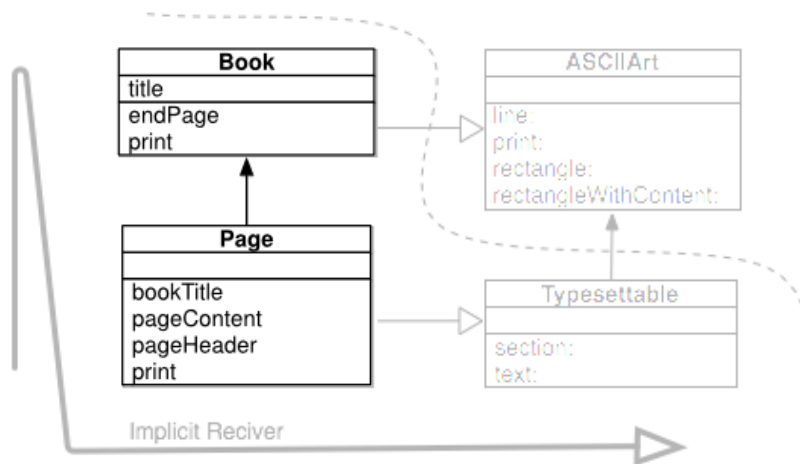
- JSR-292
 - Bytecode for dynamic invokation (*invokedynamic*)
 - Lightweight method objects and reflection
 - Parameter conversions at VM level



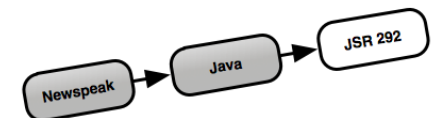
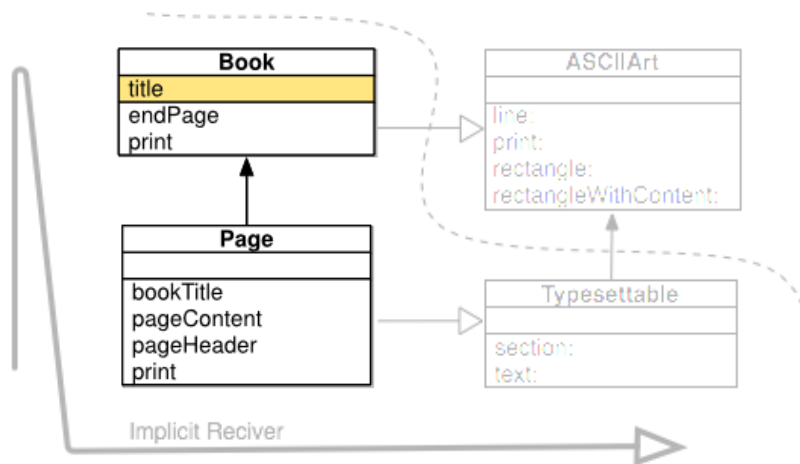
Invoke Dynamic



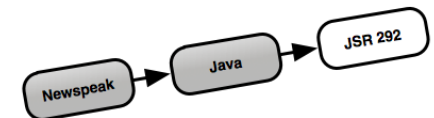
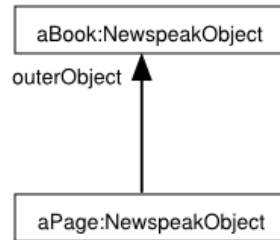
Invoke Dynamic



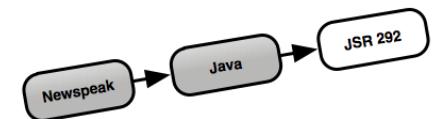
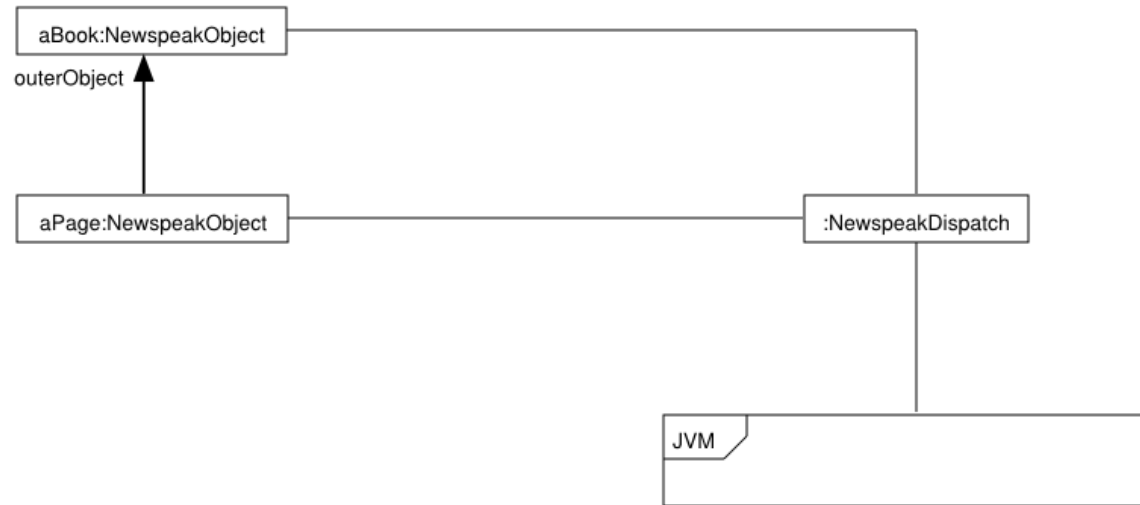
Invoke Dynamic



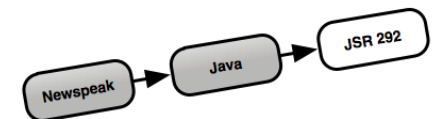
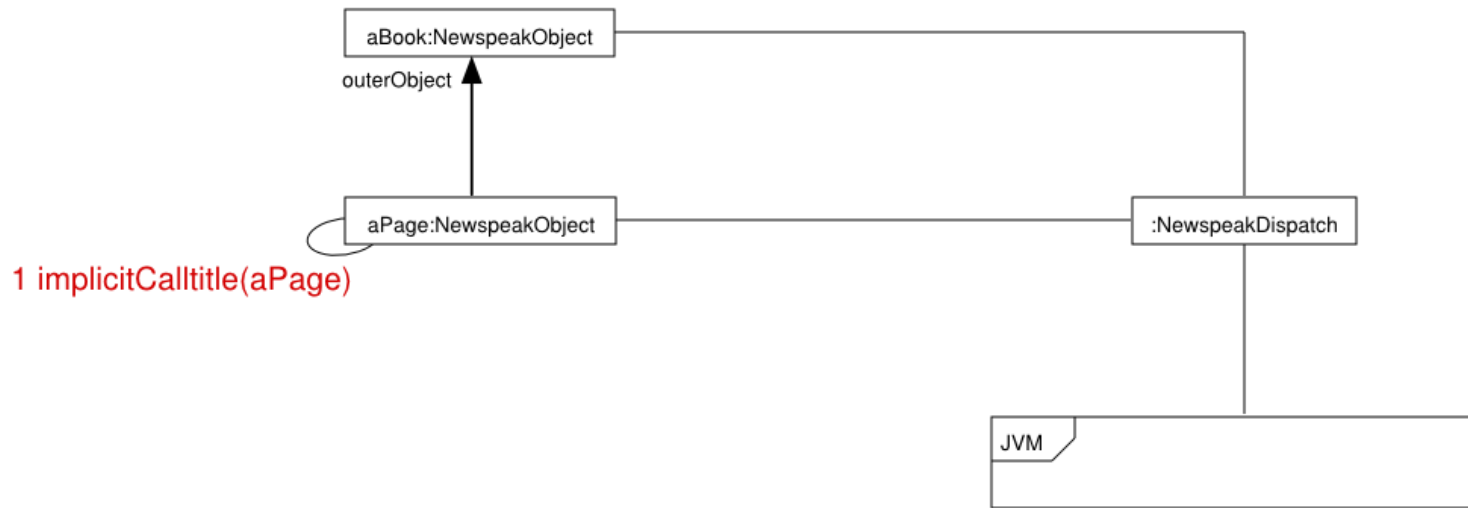
Invoke Dynamic



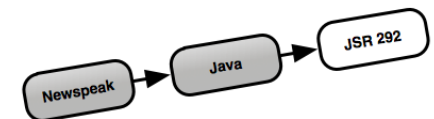
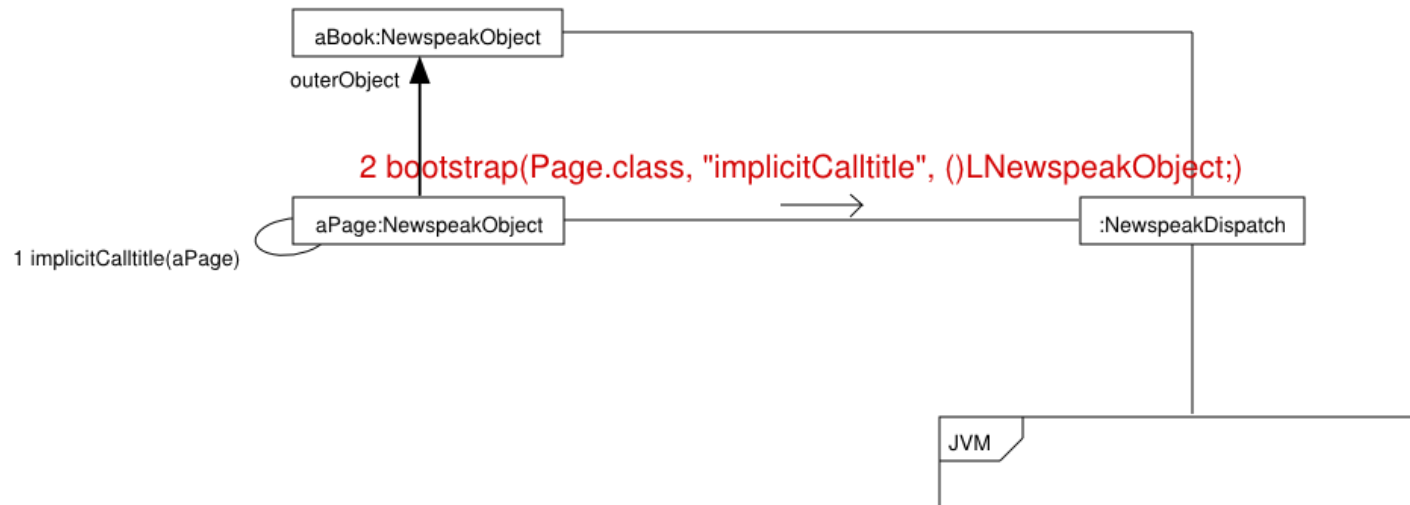
Invoke Dynamic



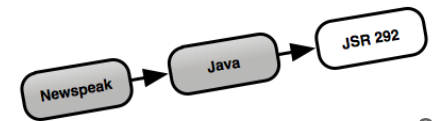
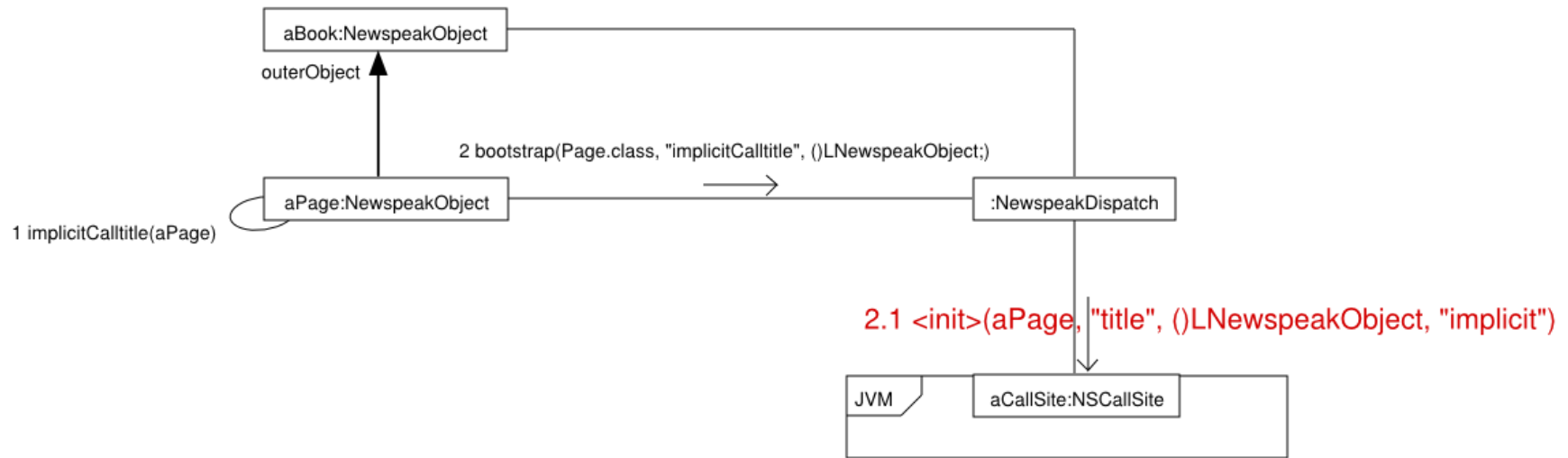
Invoke Dynamic



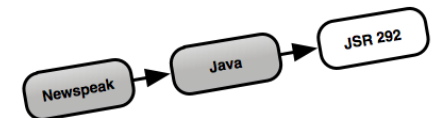
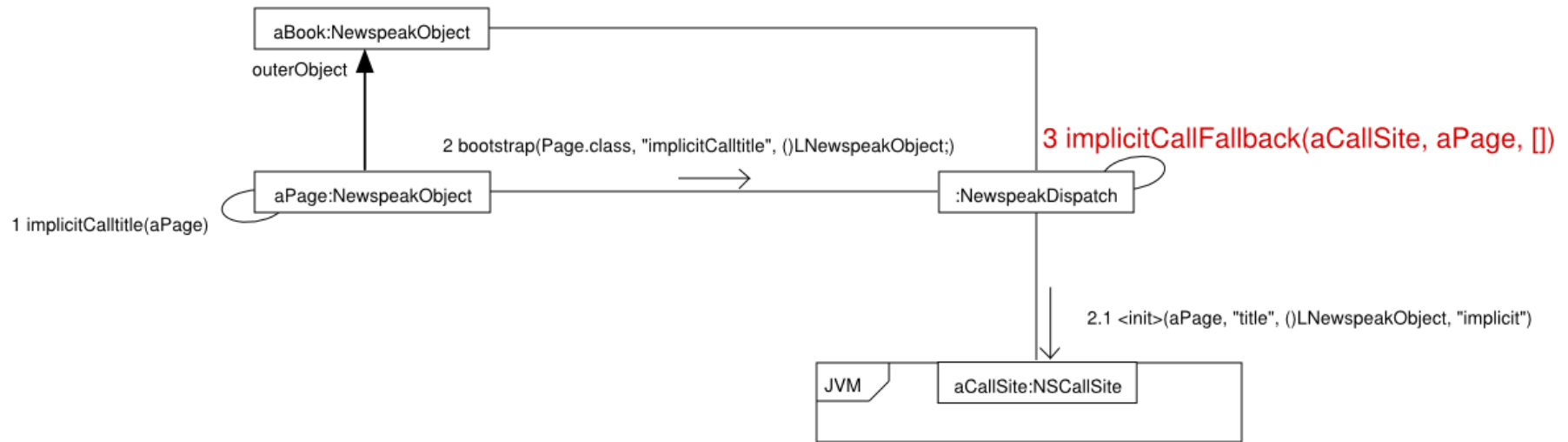
Invoke Dynamic



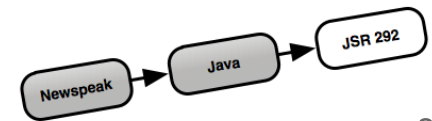
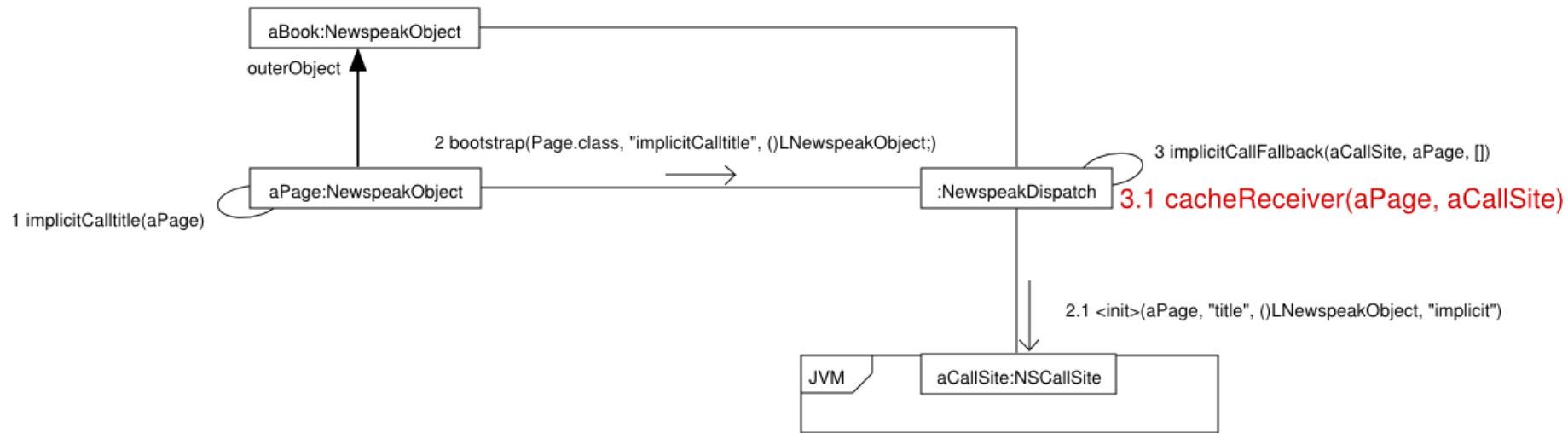
Invoke Dynamic



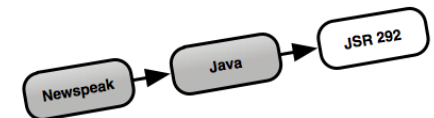
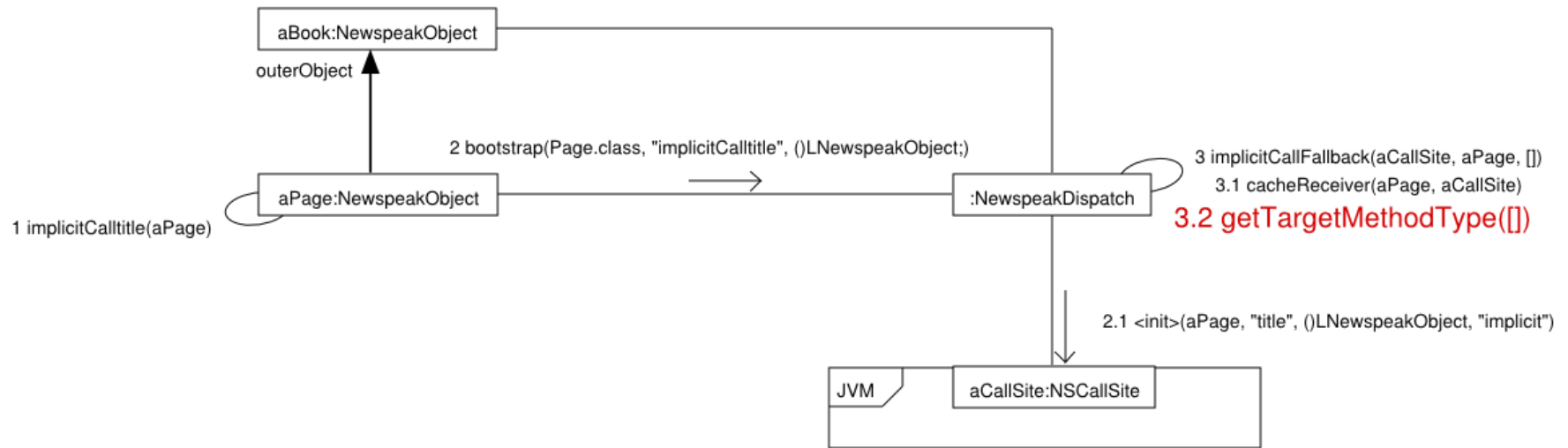
Invoke Dynamic



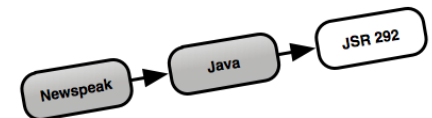
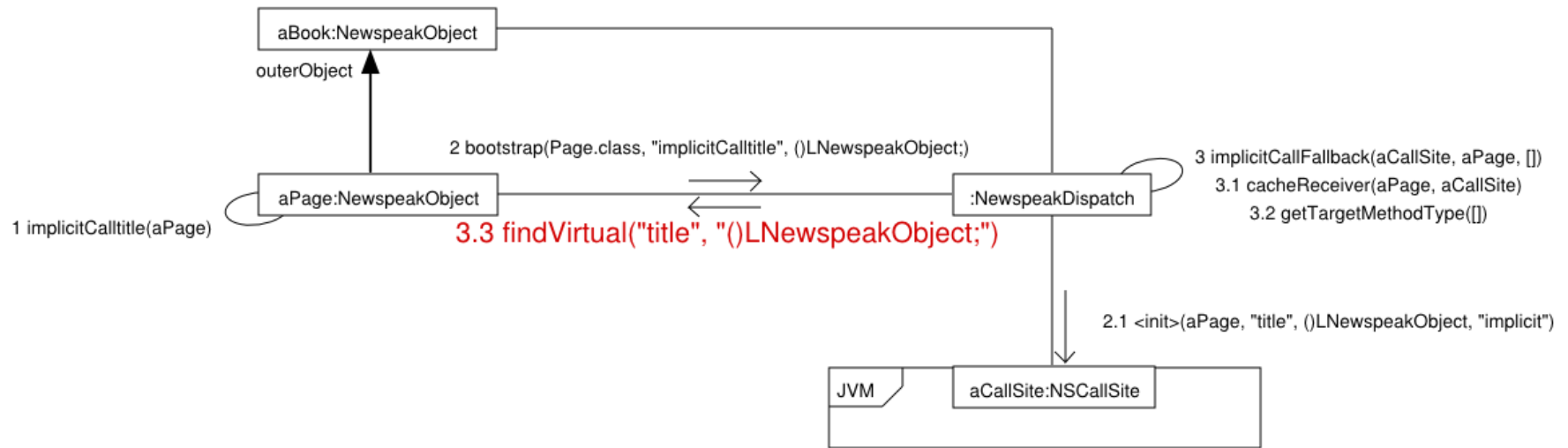
Invoke Dynamic



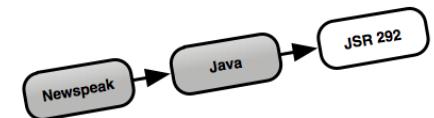
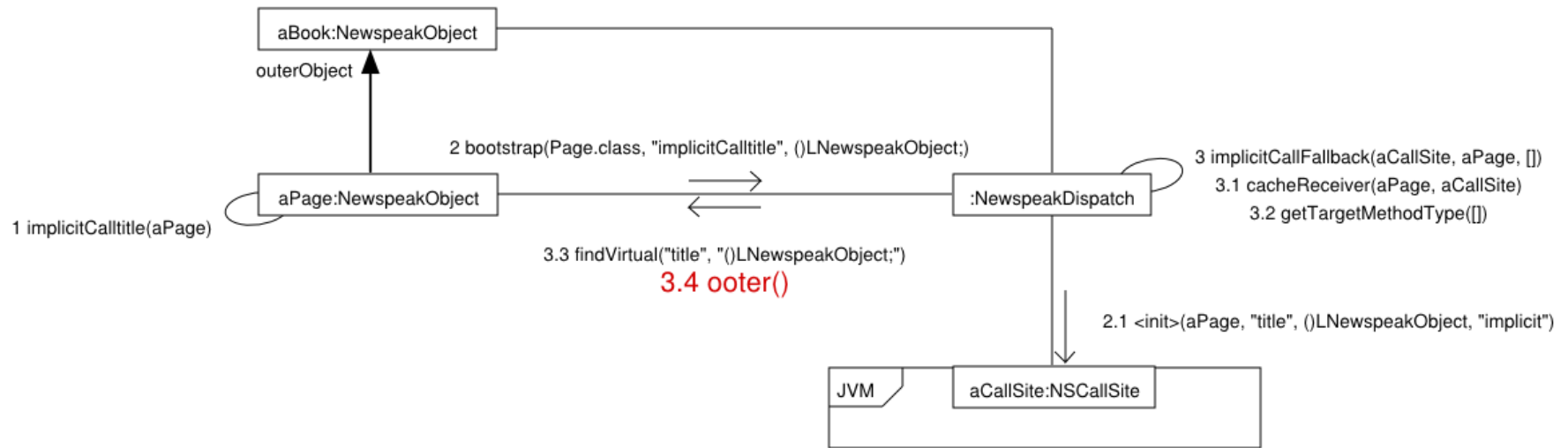
Invoke Dynamic



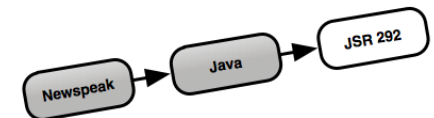
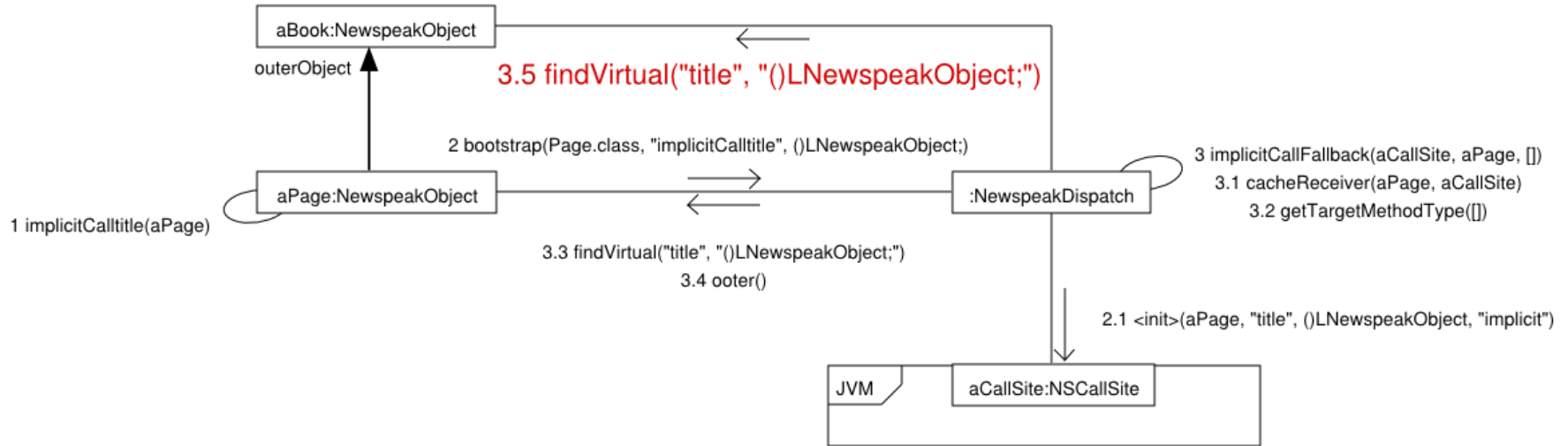
Invoke Dynamic



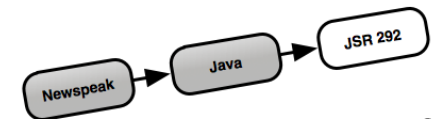
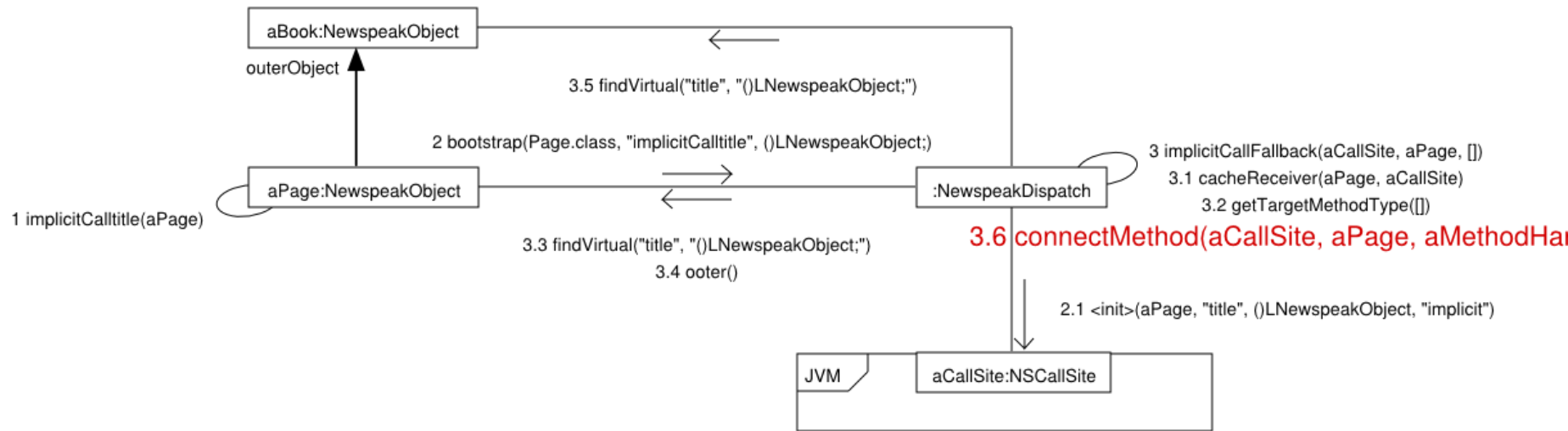
Invoke Dynamic



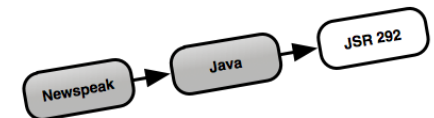
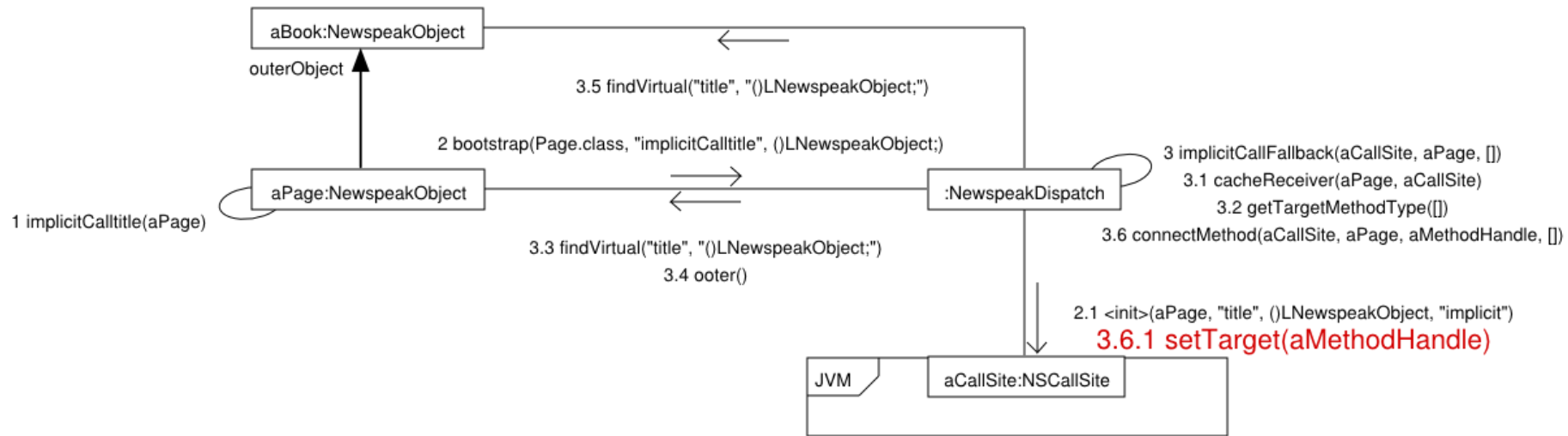
Invoke Dynamic



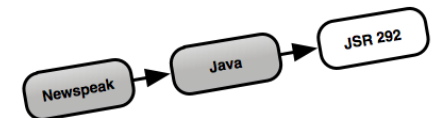
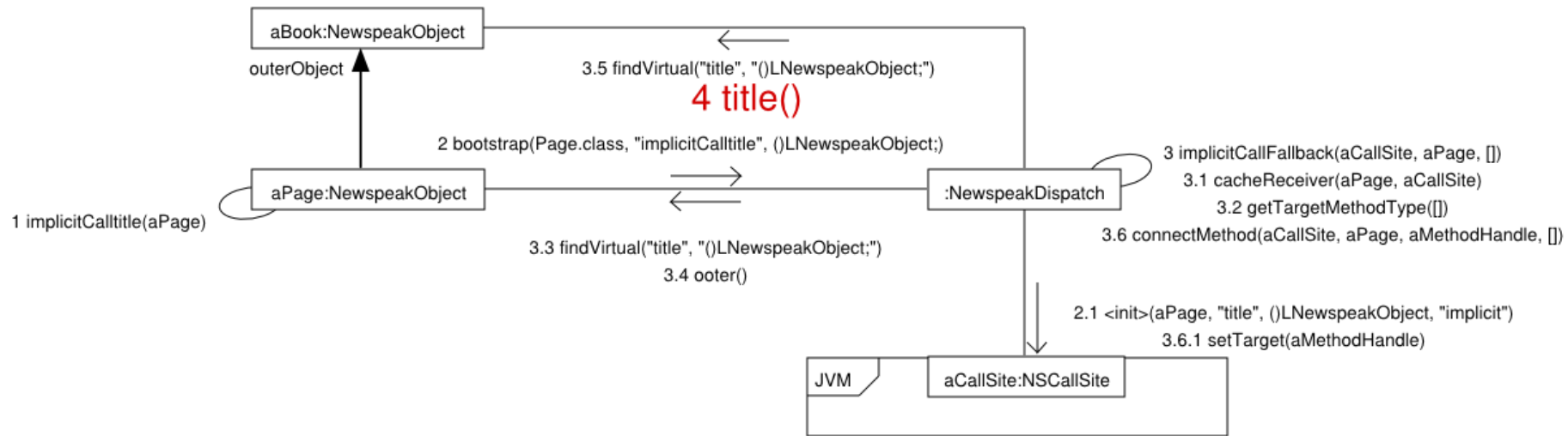
Invoke Dynamic



Invoke Dynamic



Invoke Dynamic

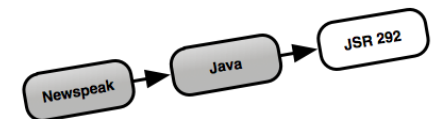


Bytecode

Naïve code

```
public class PageClass extends NewspeakClass {  
    public PageClass(NewspeakObject scope) {  
        super(scope);  
        this.superclass = (NewspeakClass) scope.Typesettable();  
    }  
}
```

The method Typesettable() is undefined for the type NewspeakObject

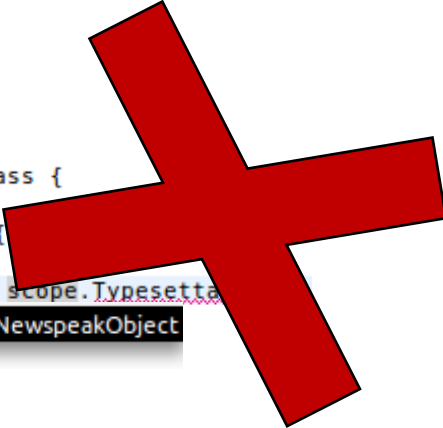


Bytecode

Naïve code

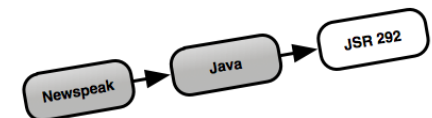
```
public class PageClass extends NewspeakClass {  
    public PageClass(NewspeakObject scope) {  
        super(scope);  
        this.superclass = (NewspeakClass) scope.Typesetta  
    }  
}
```

The method Typesettable() is undefined for the type NewspeakObject



Actual Java source

```
public class PageClass extends NewspeakClass {  
    public PageClass(NewspeakObject scope) {  
        super(scope);  
        this.superclass = (NewspeakClass) scope.call("Typesettable");  
    }  
    public Page neu() { return new Page(this); }  
}
```



Bytecode

Before

The screenshot shows the bytecode for the <init> method in the class Lim/rkh/newspack/rtl/NewspeakClass. The code is as follows:

```

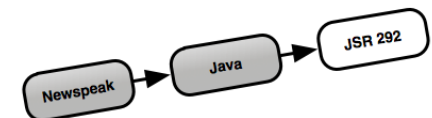
1  0  aload 0
2  1  aload_1
3  2  invokespecial #8 <im/rkh/newspack/rtl/NewspeakClass/<init>(Lim/rkh/newspack/rtl/NewspeakObject;)V>
4  5  aload 0
5  6  aload_1
6  7  ldc #10 <Typesetable>
7  9  invokevirtual #12 <im/rkh/newspack/rtl/NewspeakObject/call(Ljava/lang/String;)Lim/rkh/newspack/rtl/NewspeakObject;>
8 12  checkcast #3 <im/rkh/newspack/rtl/NewspeakClass>
9 15  putfield #18 <NS2Java/Book/Page/PageClass/superclass Lim/rkh/newspack/rtl/NewspeakClass;>
10 18  return
  
```

After

The screenshot shows the bytecode for the <init> method in the class Lim/rkh/newspack/rtl/NewspeakClass after optimization. The code is as follows:

```

1  0  aload 0
2  1  aload_1
3  2  invokespecial #9 <im/rkh/newspack/rtl/NewspeakClass/<init>(Lim/rkh/newspack/rtl/NewspeakObject;)V>
4  5  aload 0
5  6  aload_1
6  7  xxxunusedxxx
7  8  nop
8  9  fconst_1
9 10  nop
10 11  nop
11 12  checkcast #4 <im/rkh/newspack/rtl/NewspeakClass>
12 15  putfield #16 <NS2Java/Book/Page/PageClass/superclass Lim/rkh/newspack/rtl/NewspeakClass;>
13 18  return
  
```



Bytecode

Before

```

Bytecode
1  0 aload 0
2  1 aload 1
3  2 invokespecial #8 <im/rkh/newspeak/rtl/NewspeakClass/<init>(Lim/rkh/newspeak/rtl/NewspeakObject;)V>
4  5 aload 0
5  6 aload 1
6  7 ldc #10 <Typesetable>
7  9 invokevirtual #12 <im/rkh/newspeak/rtl/NewspeakObject/call(Ljava/lang/String;)Lim/rkh/newspeak/rtl/NewspeakObject;>
8 12 checkcast #3 <im/rkh/newspeak/rtl/NewspeakClass>
9 15 putfield #18 <NS2Java/Book/Page/PageClass/superclass Lim/rkh/newspeak/rtl/NewspeakClass;>
10 18 return

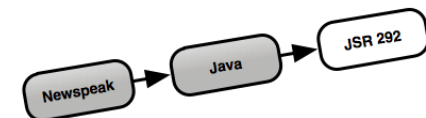
```

After

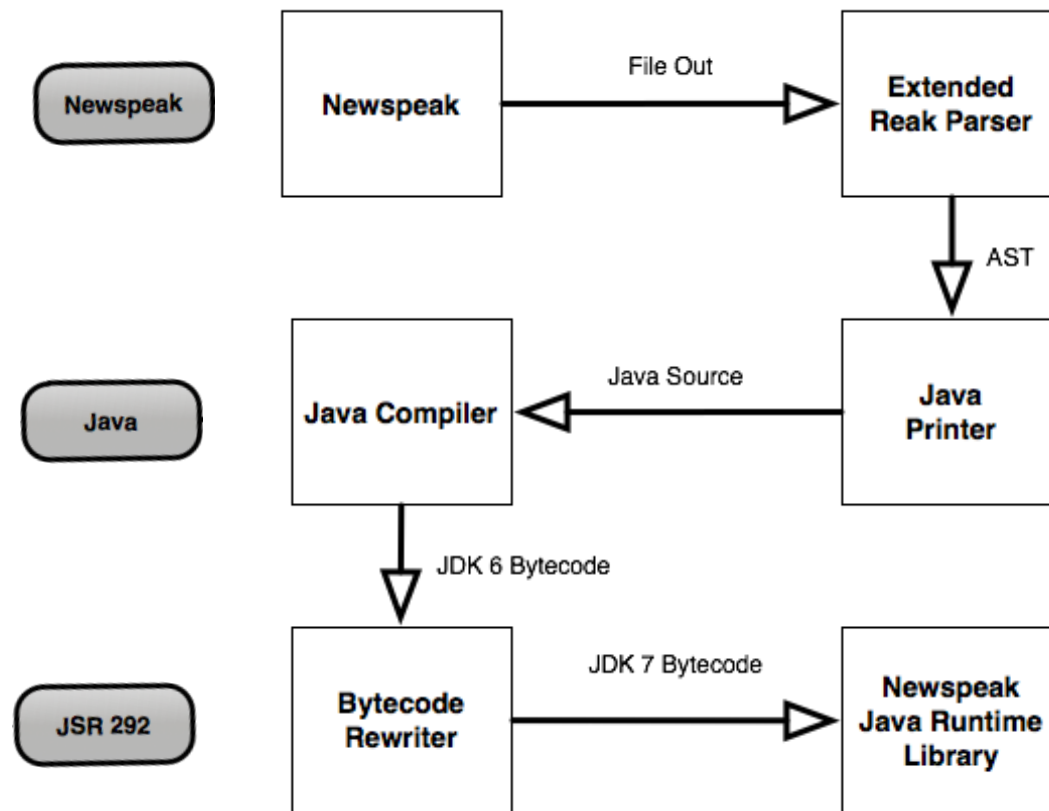
```

Bytecode
1  0 aload 0
2  1 aload 1
3  2 invokespecial #9 <im/rkh/newspeak/rtl/NewspeakClass/<init>(Lim/rkh/newspeak/rtl/NewspeakObject;)V>
4  5 aload 0
5  6 aload 1
6  7 xxxunusedxxx
7  8 nop
8  9 fconst 1
9 10 nop
10 11 nop
11 12 checkcast #4 <im/rkh/newspeak/rtl/NewspeakClass>
12 15 putfield #16 <NS2Java/Book/Page/PageClass/superclass Lim/rkh/newspeak/rtl/NewspeakClass;>
13 18 return

```



Toolchain



Lessons Learned

“Look at Java, it’s so staticky and rigid!”

- The JVM is more than Java
 - Bytecode oblivious to variable types
 - Stack operations are abundant
 - Reflective API is powerful (albeit slow)
- But: Actual method invocation **is** static
 - ... target types
 - ... need known parameter
 - ... and return known types

Lessons Learned

- Invokedynamic

`bootstrap(callSite, methodName, methodParameters)`

- Looks a lot like `doesNotUnderstand: aMessage`
- After the initial reflective lookup, effectively statically bound (until invalidated)

- But: Java syntax for `invokedynamic` missing in JDK7 (will be part of JDK8)

Sources

- Jon Rose Blog (<http://blogs.sun.com/jrose/>)
- The Java Virtual Machine Specification, 2nd Ed.
- Charles Nutter Blog (<http://headius.blogspot.com/2008/05/road-to-babel.html>)
- Bytecodes meet Combinators: invokedynamic on the JVM (J.Rose)

- Newspeak on Squeak (G. Bracha, P. Ahe, V. Bykov)
- Executable Grammars in Newspeak (G.Bracha)
- The Newspeak Programming Platform (G.Bracha, P.Ahe, V.Bykov, E.Miranda, Y.Kashai)
- Modules as Objects in Newspeak (G.Bracha, P.Ahe, V.Bykov, E.Miranda, Y.Kashai, W.Maddox)