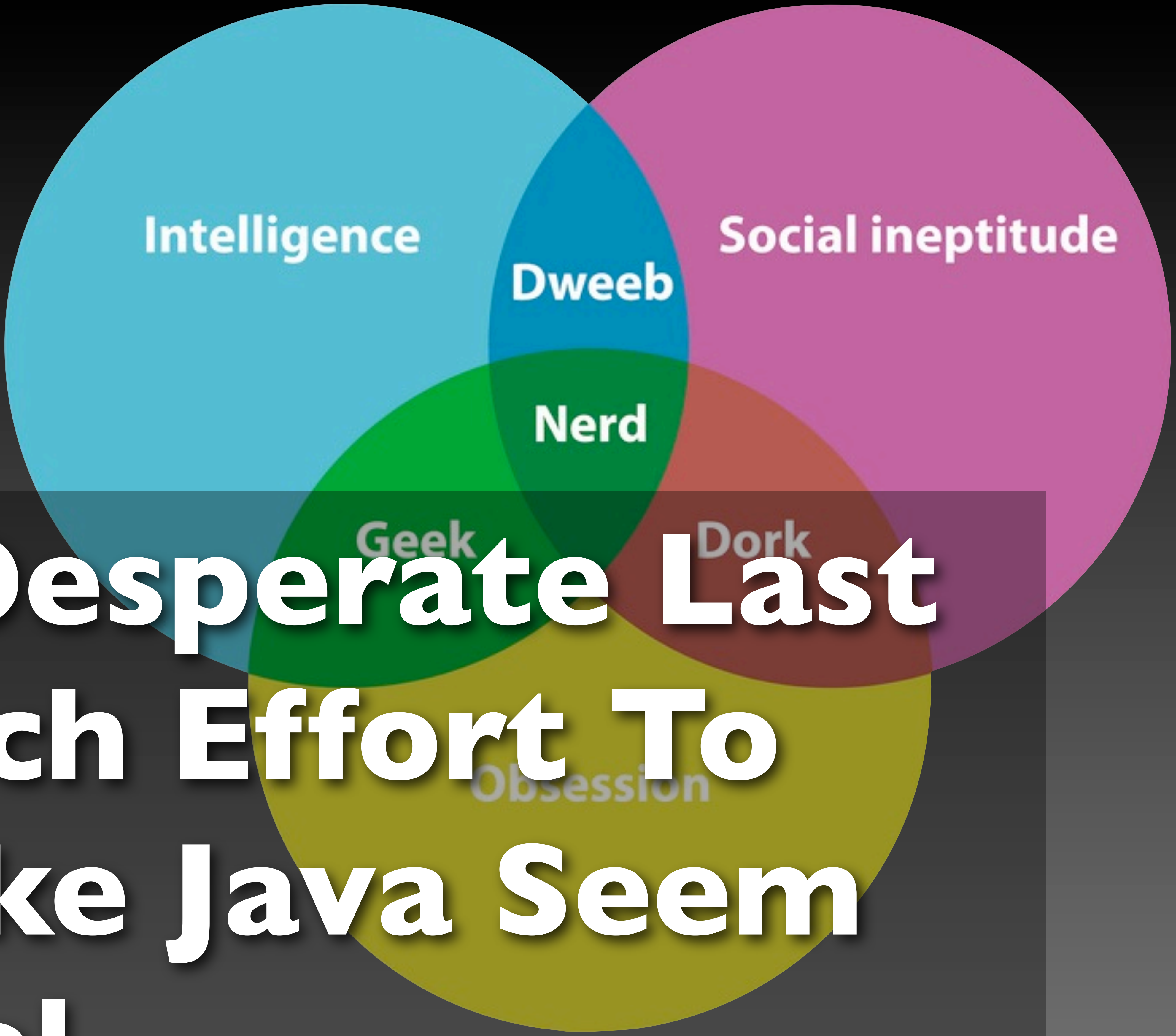
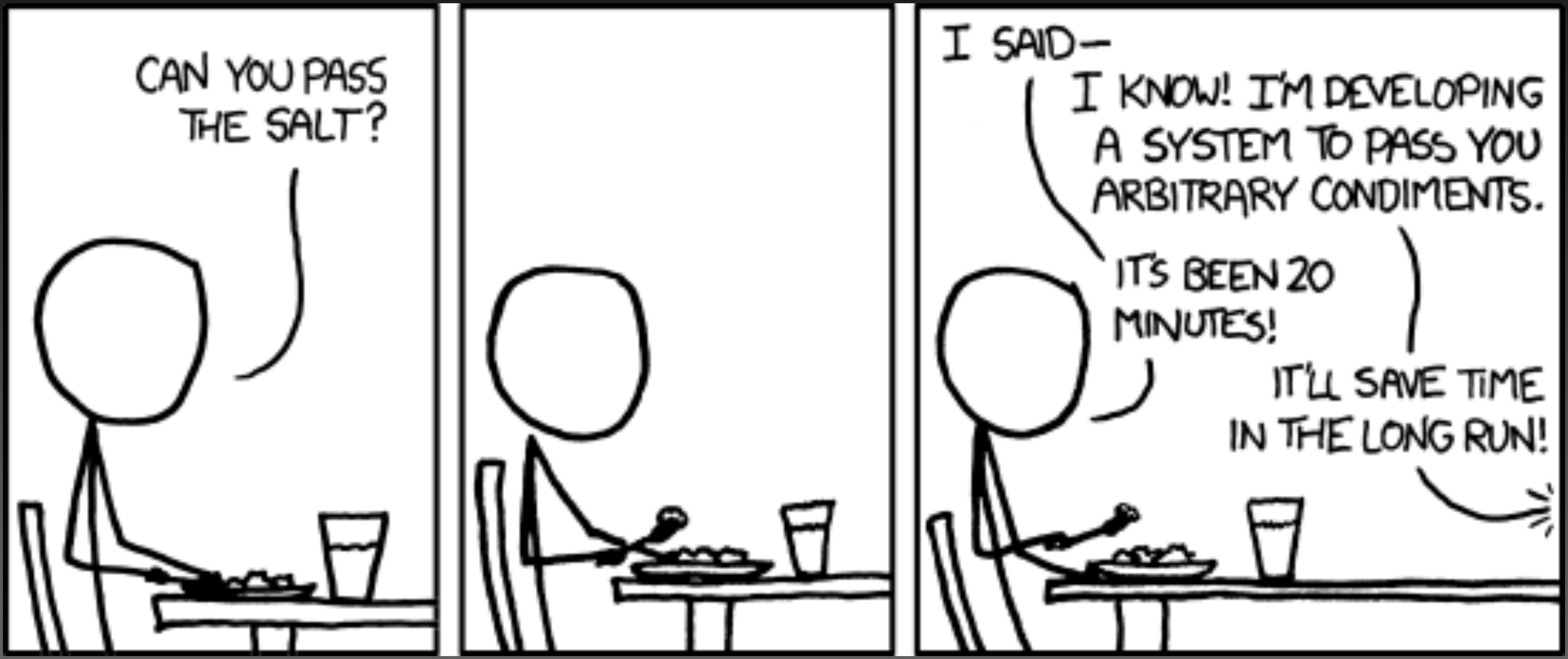
Have Your Cake And Eat It Too: Meta-Programming Java

Howard M. Lewis Ship

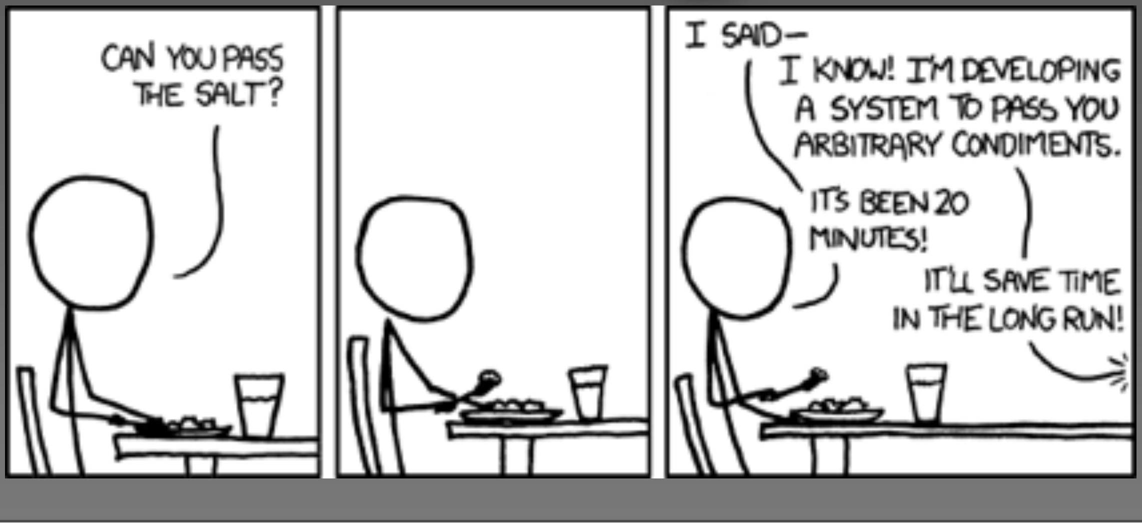


A Desperate Last Ditch Effort To Make Java Seem Cool


CAN YOU PASS THE SALT?
I SAID—
I KNOW! I'M DEVELOPING A SYSTEM TO PASS YOU ARBITRARY CONDIMENTS.
IT'S BEEN 20 MINUTES!
IT'LL SAVE TIME IN THE LONG RUN!

“I find that when someone's taking time to do something right in the present, they're a perfectionist with no ability to prioritize, whereas when someone took time to do something right in the past, they're a master artisan of great foresight.”

Randall Munroe


CAN YOU PASS THE SALT?
I SAID—
I KNOW! I'M DEVELOPING A SYSTEM TO PASS YOU ARBITRARY CONDIMENTS.
IT'S BEEN 20 MINUTES!
IT'LL SAVE TIME IN THE LONG RUN!

Ease of Meta Programming

What is Meta-Programming?

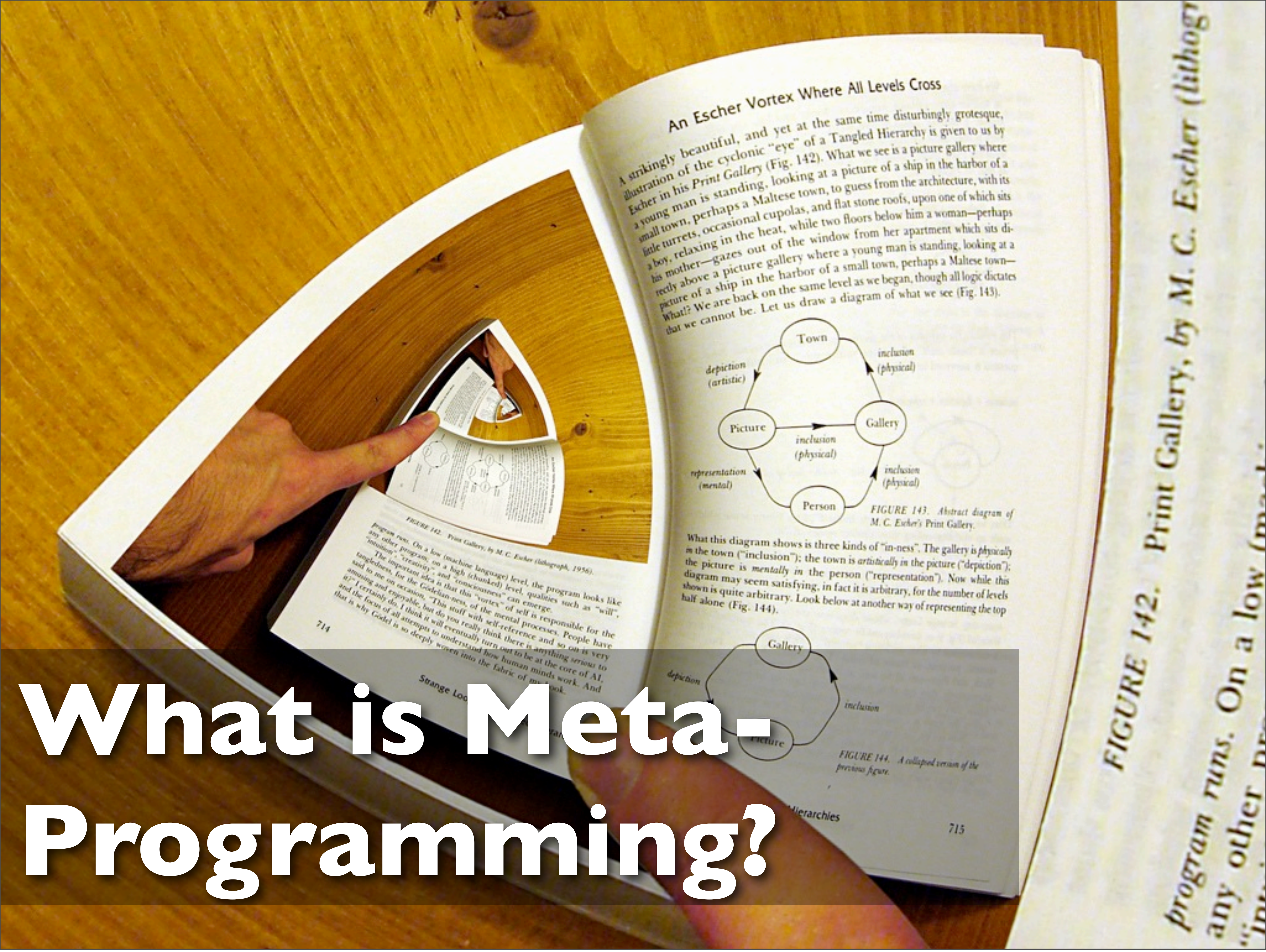


FIGURE 142. Print Gallery, by M. C. Escher (lithograph, 1956).

program runs. On a low (machine language) level, the program looks like any other program; on a high (chunked) level, qualities such as "will", "intuition", "creativity", and "consciousness" can emerge.

The important idea is that this "vortex" of self is responsible for the tangledness, for the Gödelian-ness, of the mental processes. People have said to me on occasion, "This stuff with self-reference and so on is very amusing and enjoyable, but do you really think there is anything serious to it?" I certainly do. I think it will eventually turn out to be at the core of AI, and the focus of all attempts to understand how human minds work. And that is why Gödel is so deeply woven into the fabric of my book.

714 Strange Loo

An Escher Vortex Where All Levels Cross

A strikingly beautiful, and yet at the same time disturbingly grotesque, illustration of the cyclonic "eye" of a Tangled Hierarchy is given to us by Escher in his *Print Gallery* (Fig. 142). What we see is a picture gallery where a young man is standing, looking at a picture of a ship in the harbor of a small town, perhaps a Maltese town, to guess from the architecture, with its little turrets, occasional cupolas, and flat stone roofs, upon one of which sits a boy, relaxing in the heat, while two floors below him a woman—perhaps his mother—gazes out of the window from her apartment which sits directly above a picture gallery where a young man is standing, looking at a picture of a ship in the harbor of a small town, perhaps a Maltese town—What!? We are back on the same level as we began, though all logic dictates that we cannot be. Let us draw a diagram of what we see (Fig. 143).

FIGURE 143. Abstract diagram of M. C. Escher's Print Gallery.

What this diagram shows is three kinds of "in-ness". The gallery is *physically* in the town ("inclusion"); the town is *artistically in* the picture ("depiction"); the picture is *mentally in* the person ("representation"). Now while this diagram may seem satisfying, in fact it is arbitrary, for the number of levels shown is quite arbitrary. Look below at another way of representing the top half alone (Fig. 144).

FIGURE 144. A collapsed version of the previous figure.

Hierarchies 715

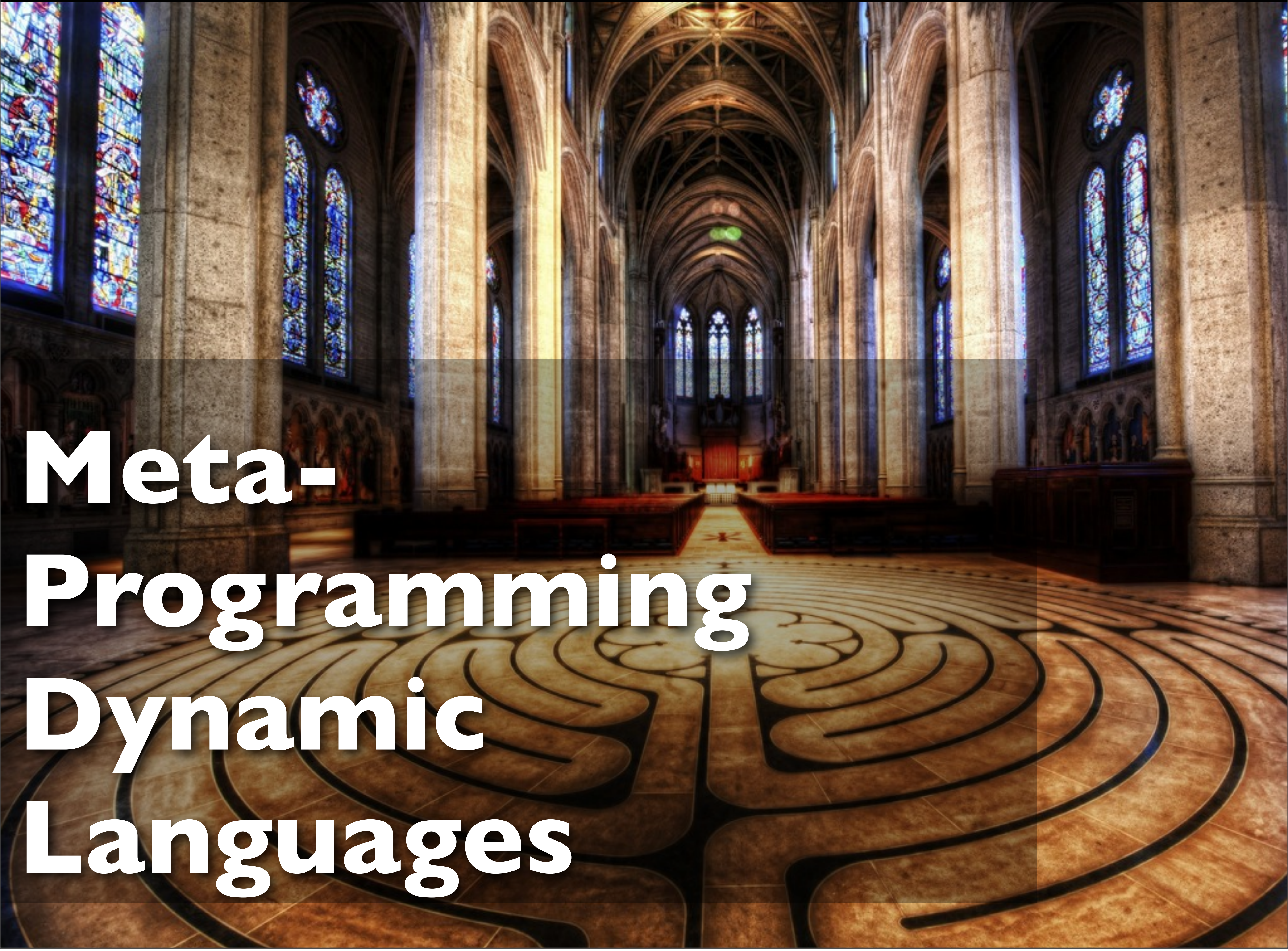
FIGURE 142. Print Gallery, by M. C. Escher (lithogr

program runs. On a low (machi
any other pr

Code Reuse

Without Inheritance

~~Boilerplate~~

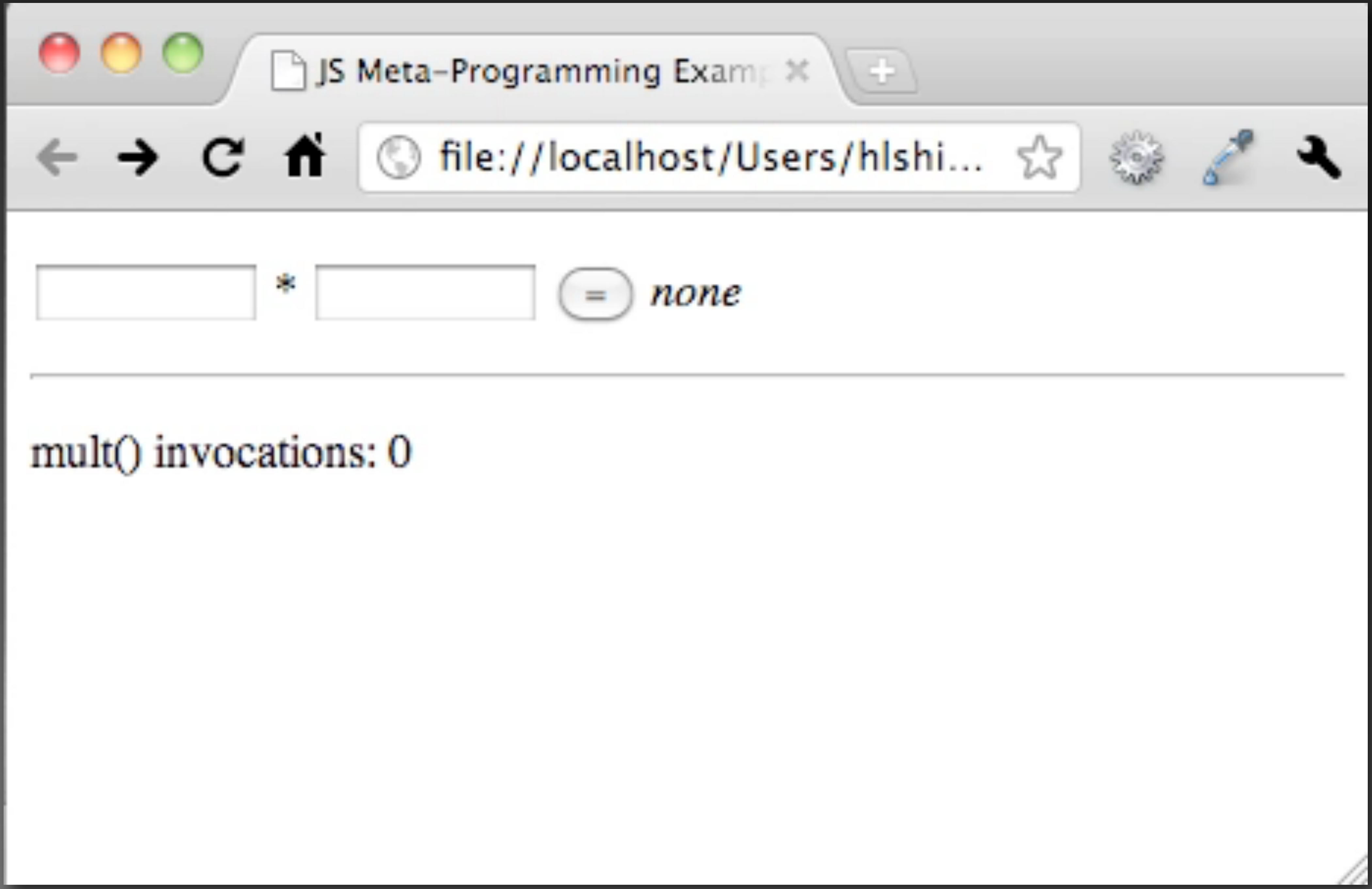


Meta-Programming Dynamic Languages

```
<p>
  <input type="text" size="10" id="v1"> *
  <input type="text" size="10" id="v2">
  <button id="calc">=</button>
  <span id="result"><em>none</em></span>
</p>
<hr>
<p>
  mult() invocations: <span id="count">0</span>
</p>
```

```
var count = 0;

function mult(v1, v2) {

  $("#count").text(++count);

  return v1 * v2;
}
```

```
$(function() {
  $("#calc").click(function() {
    var v1 = parseInt($("#v1").val());
    var v2 = parseInt($("#v2").val());
    $("#result").text(mult(v1, v2));
  });
});
```



mult() invocations: 0

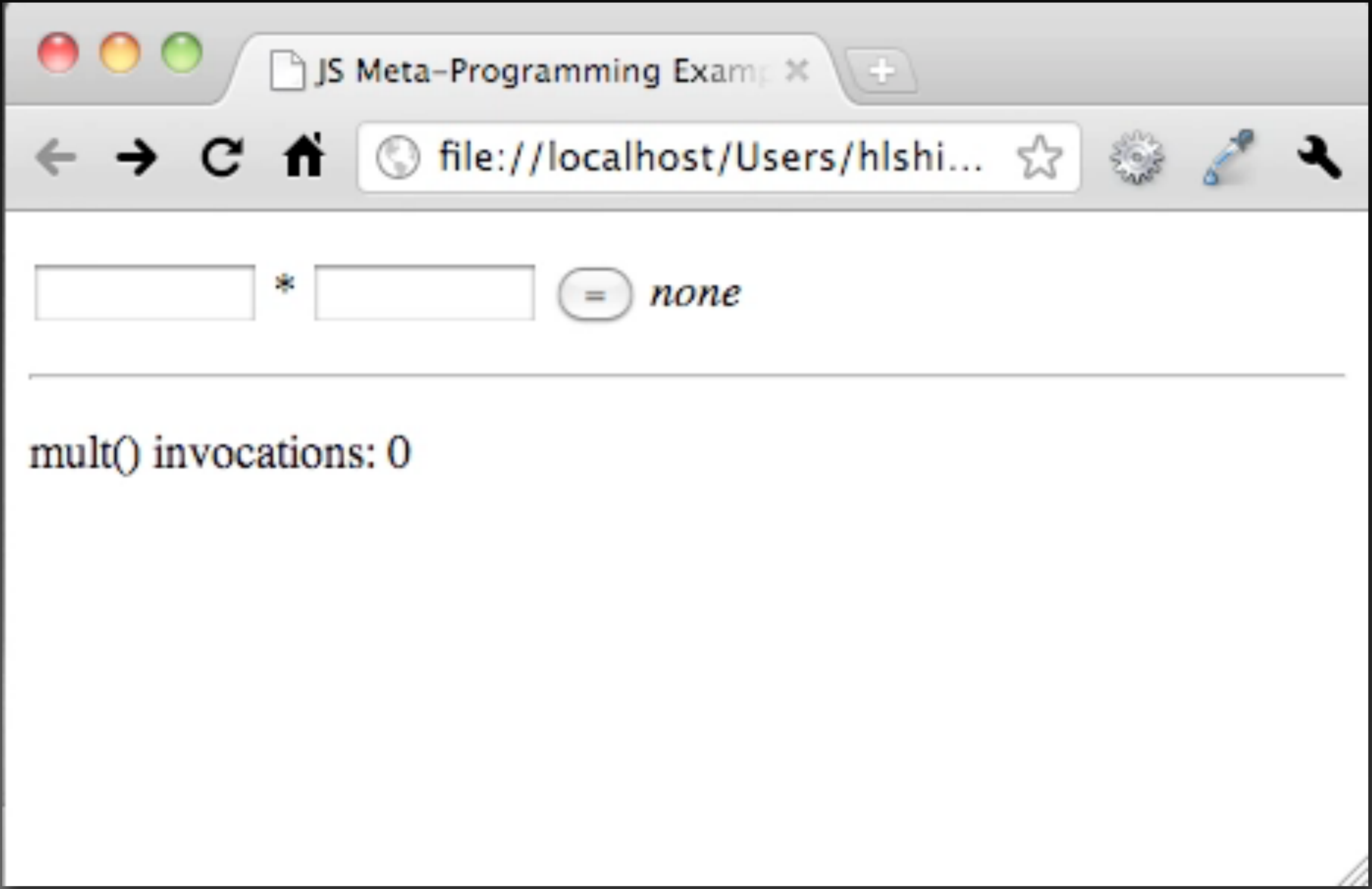
function |ˈfə ng k sh ən|

noun

A function, in a mathematical sense, expresses the idea that one quantity (the argument of the function, also known as the input) ***completely determines*** another quantity (the value, or the output).

```
function memoize(originalfn) {

  var invocationCache = {};

  return function() {
    var args = Array.prototype.slice.call(arguments);

    var priorResult = invocationCache[args];

    if (priorResult !== undefined) {
      return priorResult;
    }

    var result = originalfn.apply(null, arguments);

    invocationCache[args] = result;

    return result;
  };
}

mult = memoize(mult);
```

Danger!



none

mult() invocations: 0

Clojure

```
(defn memoize
  "Returns a memoized version of a referentially transparent function. The
  memoized version of the function keeps a cache of the mapping from arguments
  to results and, when calls with the same arguments are repeated often, has
  higher performance at the expense of higher memory use."
  {:added "1.0"
   :static true}
  [f]
  (let [mem (atom {})]
    (fn [& args]
      (if-let [e (find @mem args)]
        (val e)
        (let [ret (apply f args)]
          (swap! mem assoc args ret)
          ret)))))
```

Python

```python
def memoize(function):
    cache = {}
    def decorated_function(*args):
        if args in cache:
            return cache[args]
        else:
            val = function(*args)
            cache[args] = val
            return val
    return decorated_function
```

Ruby

```ruby
module Memoize
   def memoize(name)
      cache = {}

      (class<<self; self; end).send(:define_method, name) do |*args|
         unless cache.has_key?(args)
            cache[args] = super(*args)
         end
         cache[args]
      end
      cache
   end
end
```

http://programmingzen.com/2009/05/18/memoization-in-ruby-and-python/

Java?

Uh, maybe a Subclass?

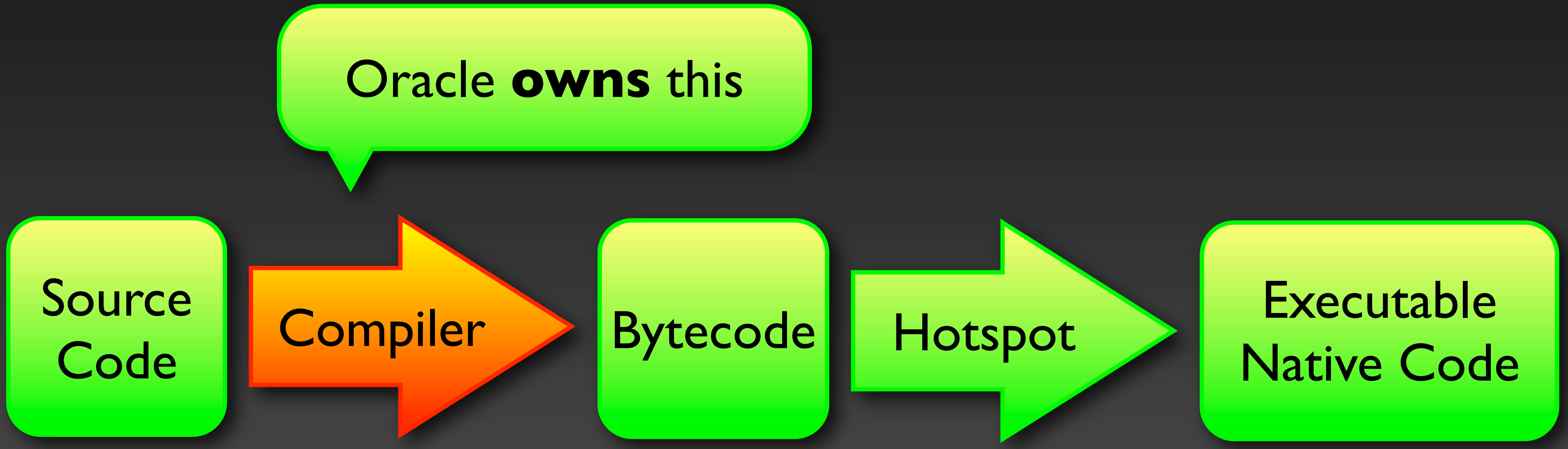
"Closed" Language Challenges

“Java is C++ without the guns, knives, and clubs”

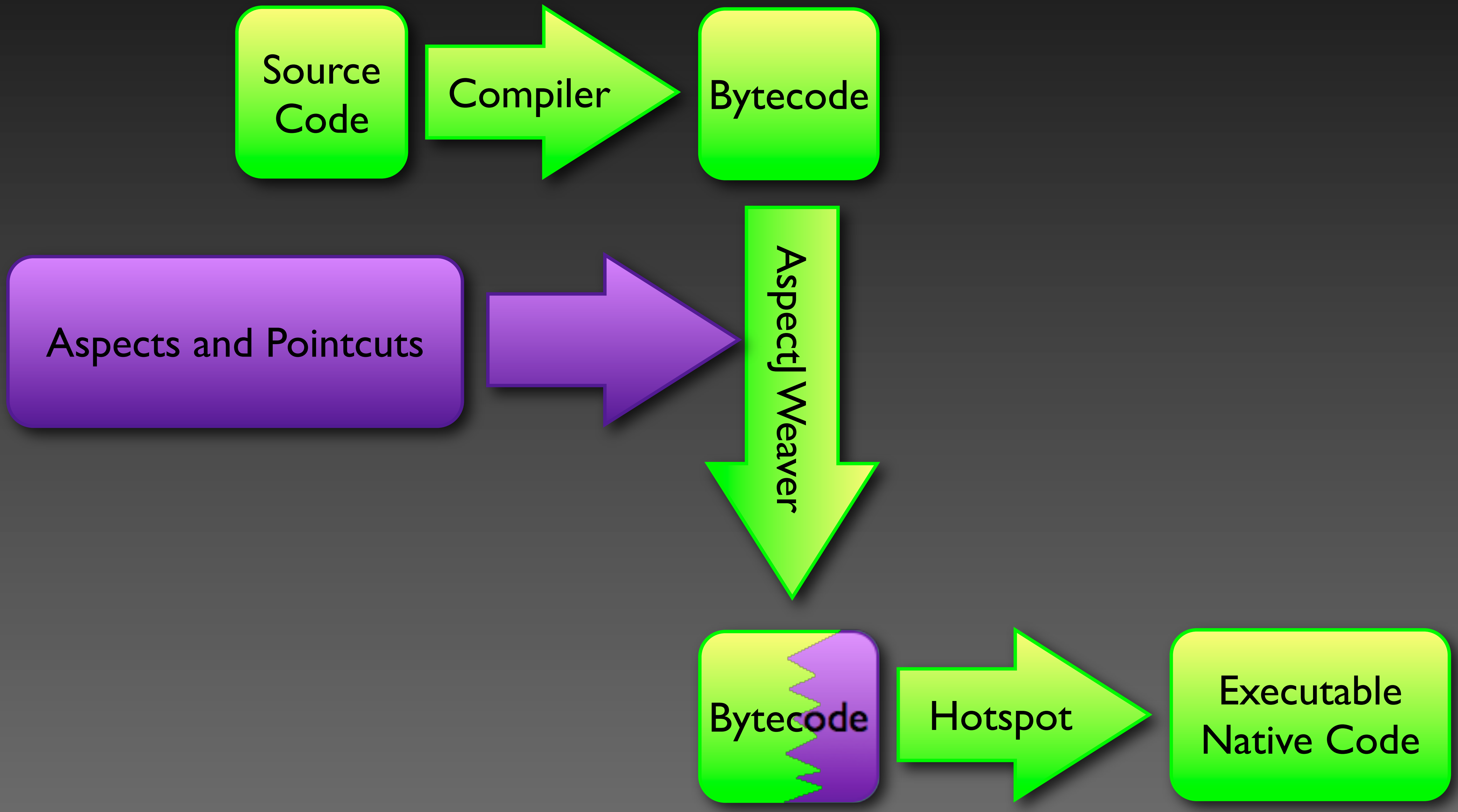
James Gosling

“JavaScript has more in common with functional languages like Lisp or Scheme than with C or Java”

Douglas Crockford


Oracle **owns** this
Source Code
Compiler
Bytecode
Hotspot
Executable Native Code

AspectJ



http://www.eclipse.org/aspectj/

Not *targeted* on any specific class or method

```
public abstract aspect Memoize pertarget(method()){
    HashMap cache = new HashMap();

    String makeKey(Object[] args) {
        String key = "";
        for (int i = 0; i < args.length; i++) {
            key += args[i].toString();
            if (i < (args.length - 1)) {

                key += ",";
            }
        }
        return key;
    }

    abstract pointcut method();

    Object around(): method() {
        String key = makeKey(thisJoinPoint.getArgs());
        if (cache.containsKey(key)) {
            return cache.get(key);
        } else {
            Object result = proceed();
            cache.put(key,result);
            return result;
        }
    }
}
```

The key is formed from the toString() values of the parameters. Ick.

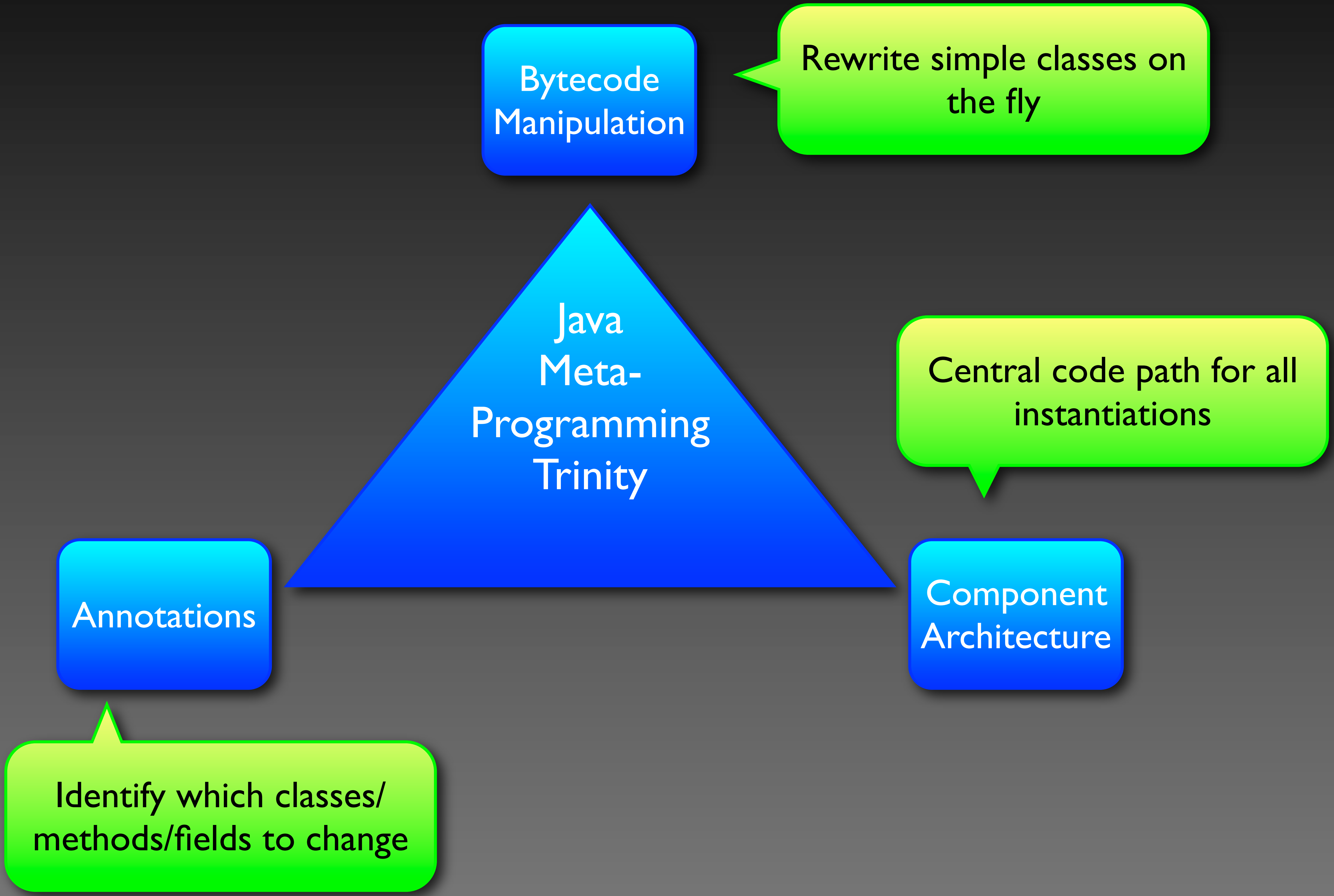
http://www.jroller.com/mschinc/entry/memoization_with_aop_and_aspectj

Extend Memoize to apply to specific classes & methods

```
public aspect MemoizeFib extends Memoize {
    pointcut method(): call(int Test.fib(int));
}
```

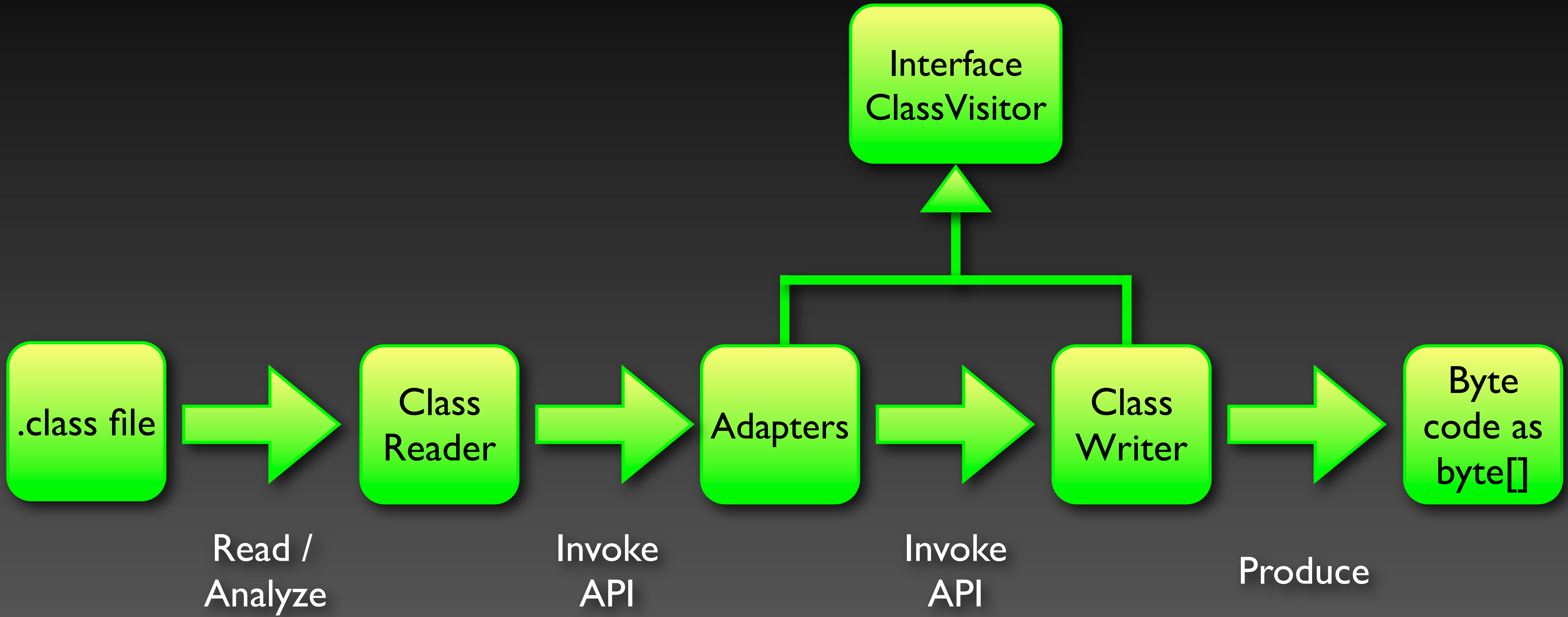
Identify method(s) to be targeted by Aspect

Tapestry Plastic


Bytecode Manipulation
Rewrite simple classes on the fly
Java Meta-Programming Trinity
Central code path for all instantiations
Component Architecture
Annotations
Identify which classes/ methods/fields to change

ASM 3.3.1

- Small & Fast
- Used in:
  - Clojure
  - Groovy
  - Hibernate
  - Spring
  - JDK


Interface ClassVisitor
.class file
Read / Analyze
Class Reader
Invoke API
Adapters
Invoke API
Class Writer
Produce
Byte code as byte[]

Reader: parse bytecode, invoke methods on ClassVisitor

```
ClassReader reader = new ClassReader("com.example.MyClass");
ClassWriter writer = new ClassWriter(reader, 0);

ClassVisitor visitor = new AddFieldAdapter(writer, ACC_PRIVATE,
  "invokeCount", "I");

reader.accept(visitor, 0);

byte[] bytecode = writer.toByteArray();
```

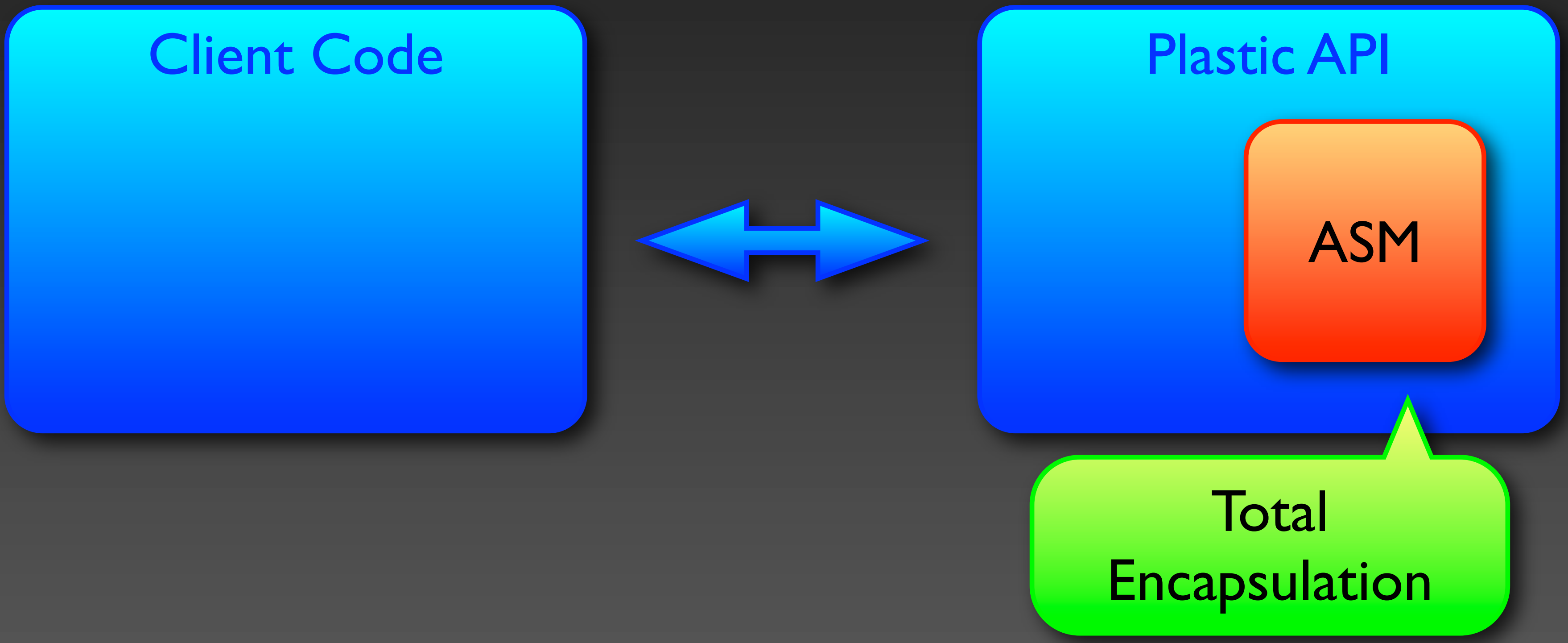
Adaptor: Sits between Reader & Writer

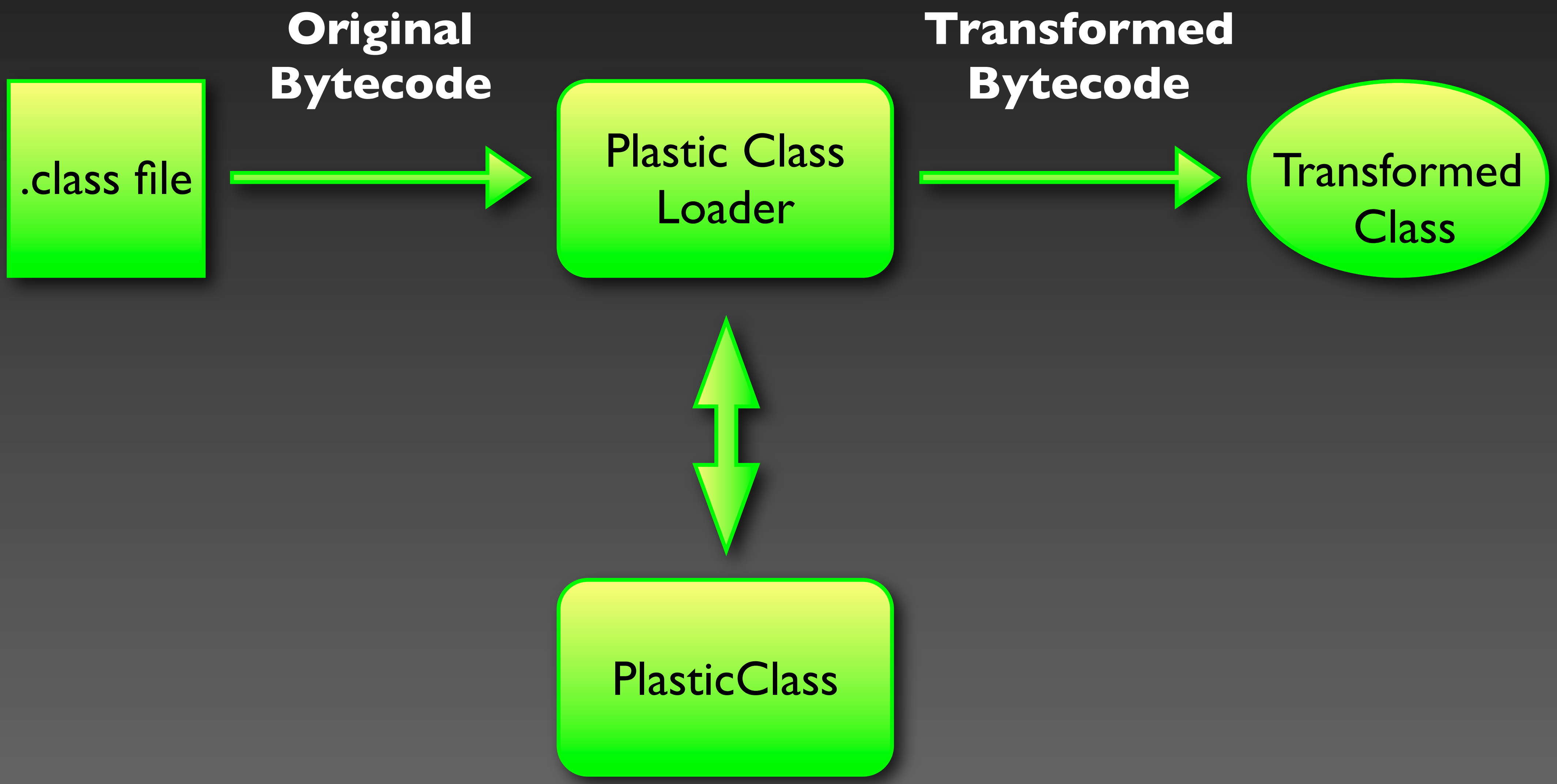
Writer: methods construct bytecode

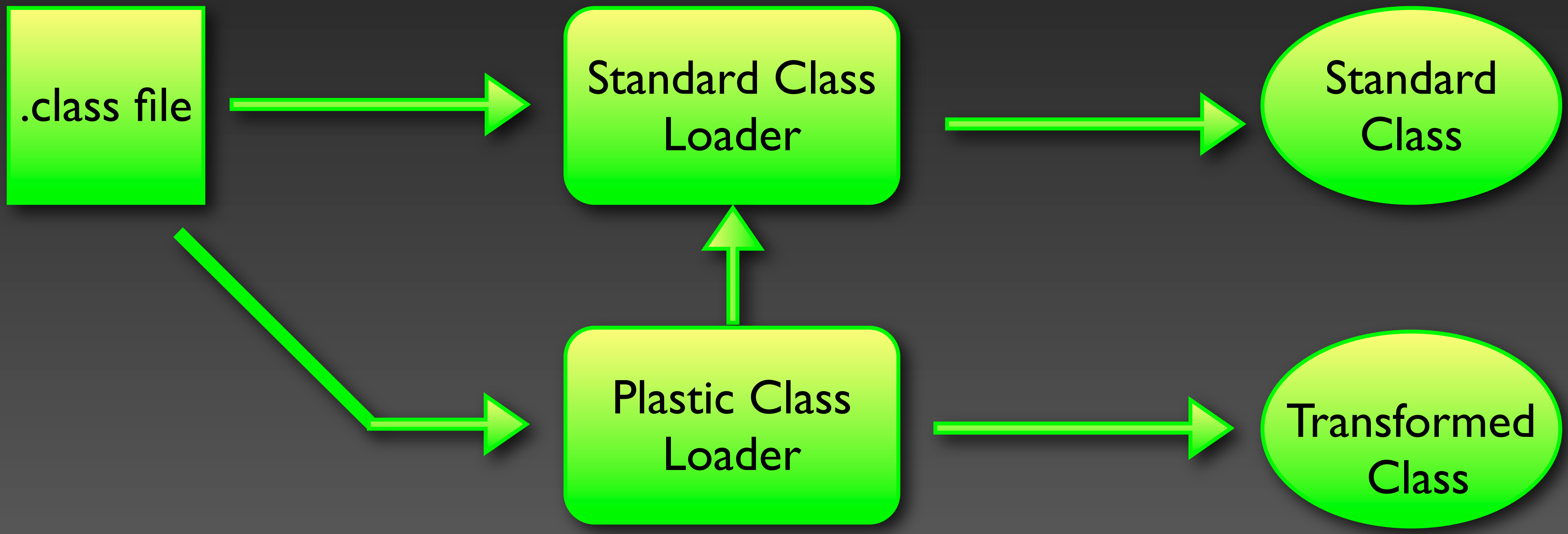
Most methods delegate to ClassVisitor

```
public class AddFieldAdapter extends ClassAdapter {
  private final int fAcc;
  private final String fName;
  private final String fDesc;

  public AddFieldAdapter(ClassVisitor cv, int fAcc, String fName, String fDesc) {
    super(cv);
    this.fAcc = fAcc;
    this.fName = fName;
    this.fDesc = fDesc;
  }

  @Override
  public void visitEnd() {
    FieldVisitor fv = cv.visitField(fAcc, fName, fDesc, null, null);
    if (fv != null) {
      fv.visitEnd();
    }
    cv.visitEnd();
  }
}
```

"*Simulate*" a field read from the input class after everything else

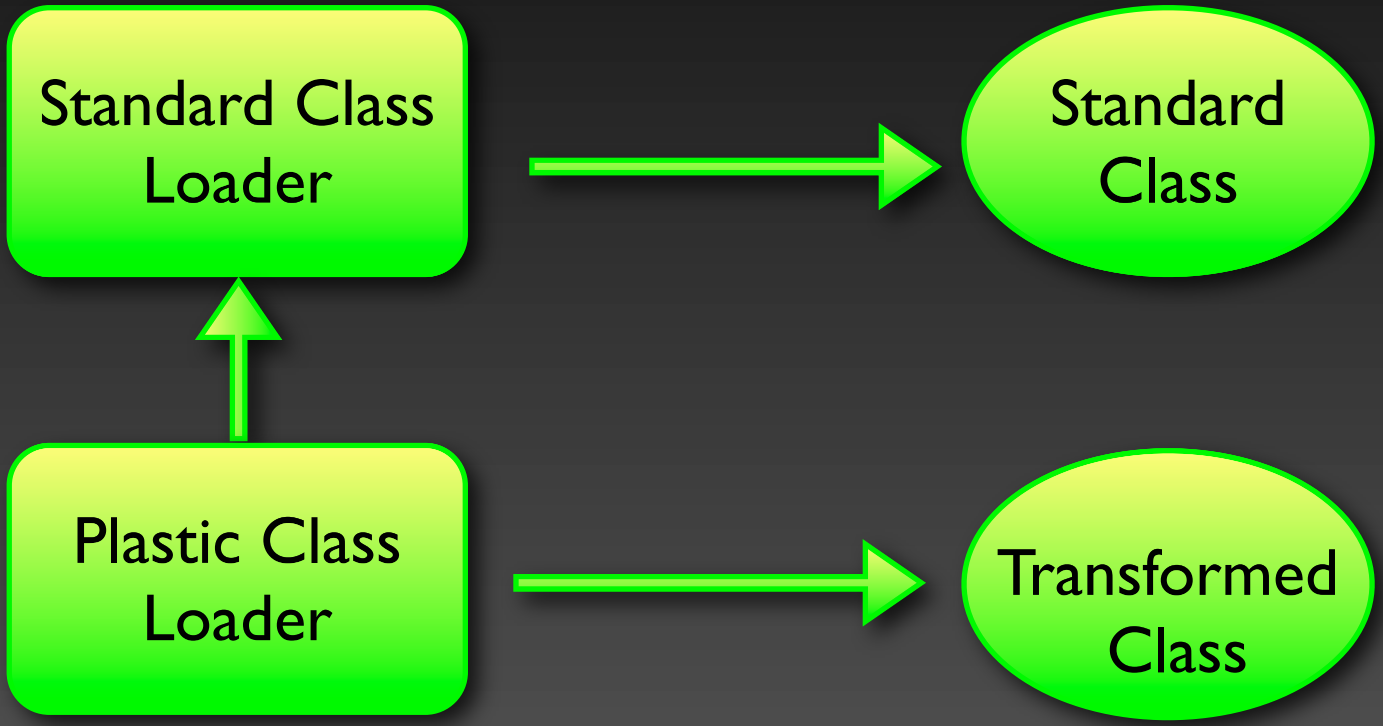

Client Code
Plastic API
ASM
Total Encapsulation


Original Bytecode
Transformed Bytecode
.class file
Plastic Class Loader
Transformed Class
PlasticClass


.class file
Standard Class Loader
Standard Class
Plastic Class Loader
Transformed Class

Leaky Abstractions

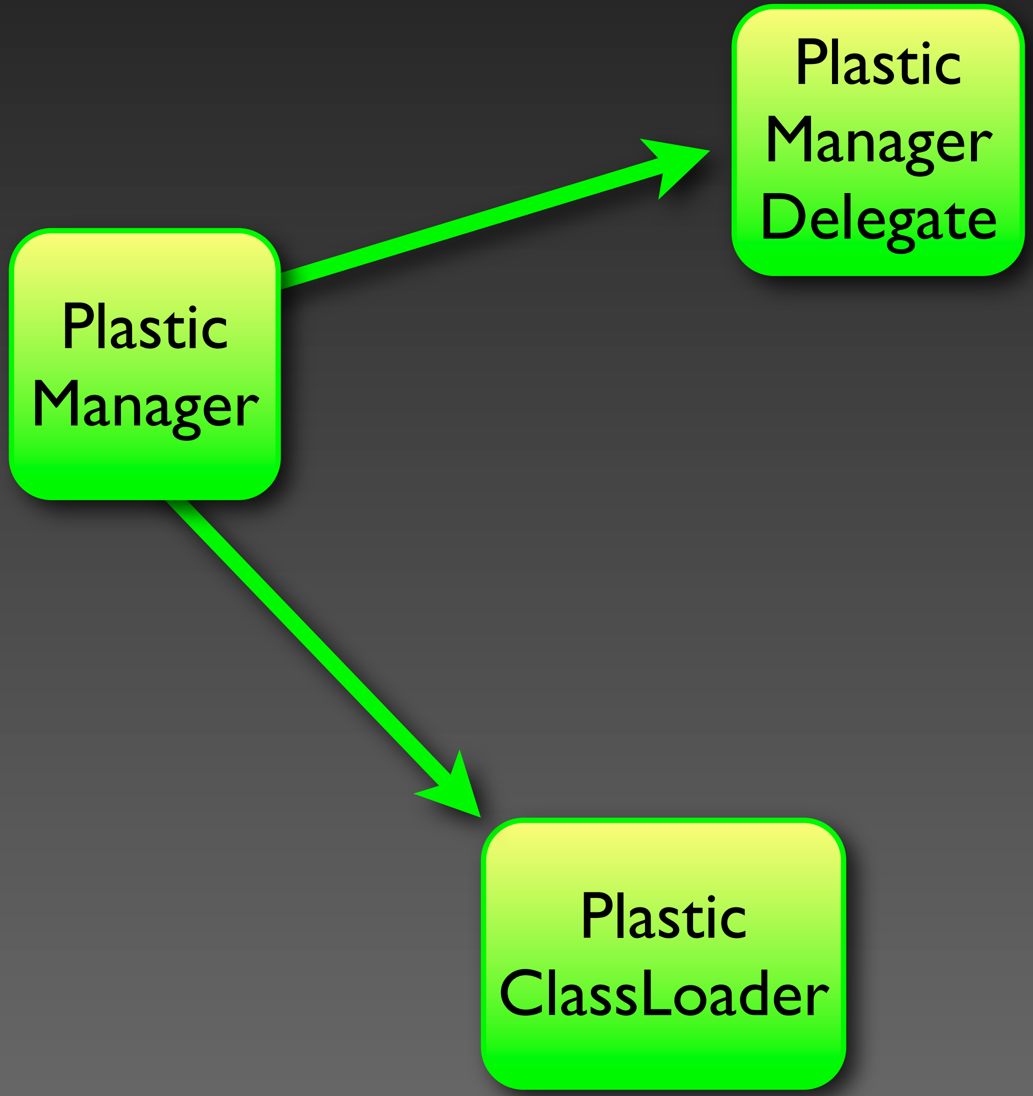
torsdag den 17. november 2011



```
ClassCastException: org.apache.tapestry5.corelib.components.Grid
can not be cast to org.apache.tapestry5.corelib.components.Grid
```

Performs transformations on PlasticClass

Which classes are transformed (by package)
Access to ClassInstantiator



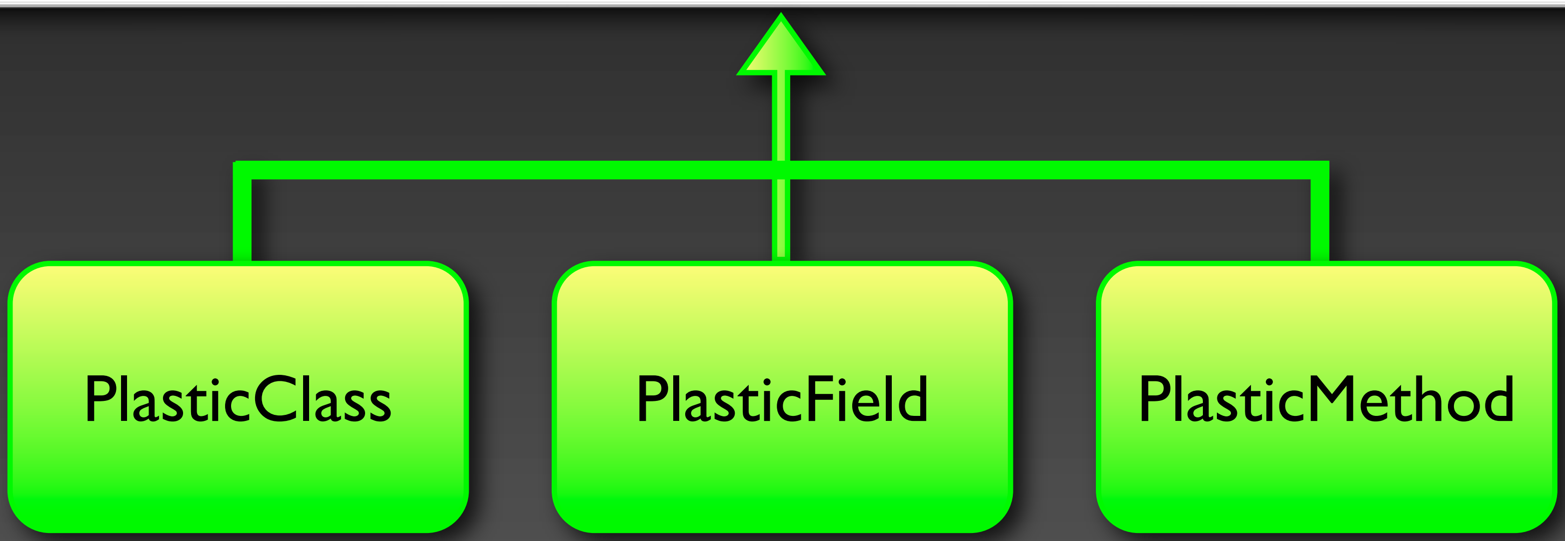
```
public class PlasticManager {

  public ClassLoader getClassLoader() { … }

  public <T> ClassInstantiator<T> getClassInstantiator(String className) { … }

  public <T> ClassInstantiator<T> createClass(Class<T> baseClass,
      PlasticClassTransformer callback) { … }

  public <T> ClassInstantiator<T> createProxy(Class<T> interfaceType,
      PlasticClassTransformer callback) { … }

  …
}
```

For instantiating existing components

For creating proxies and other objects

```
public interface PlasticClassTransformer
{
  void transform(PlasticClass plasticClass);
}
```

```
public interface AnnotationAccess
{
  <T extends Annotation> boolean hasAnnotation(Class<T> annotationType);

  <T extends Annotation> T getAnnotation(Class<T> annotationType);
}
```



Extending Methods with Advice

Method Advice

```
public class EditUser
{
  @Inject
  private Session session;

  @Propery
  private User user;

  @CommitAfter
  void onSuccessFromForm() {
    session.saveOrUpdate(user);
  }
}
```

Advise method to manage transaction commit

```
void onSuccessFromForm() {
  try {
    session.saveOrUpdate();
    commit-transaction();
  } catch (RuntimeException ex) {
    rollback-transaction();
    throw ex;
  }
}
```

Introduce Method

```
public interface PlasticClass {

  Set<PlasticMethod> introduceInterface(Class interfaceType);

  PlasticMethod introduceMethod(MethodDescription description);

  PlasticMethod introduceMethod(Method method);

  PlasticMethod introducePrivateMethod(String typeName,
    String suggestedName,
    String[] argumentTypes,
    String[] exceptionTypes);
```

```
public interface PlasticMethod {

  …

  PlasticMethod addAdvice(MethodAdvice advice);
```

Define an annotation

Create a *worker* for the annotation

Apply annotation to class

Test transformed class

MethodInvocation

```
public interface MethodAdvice {
  void advise(MethodInvocation invocation);
}
```

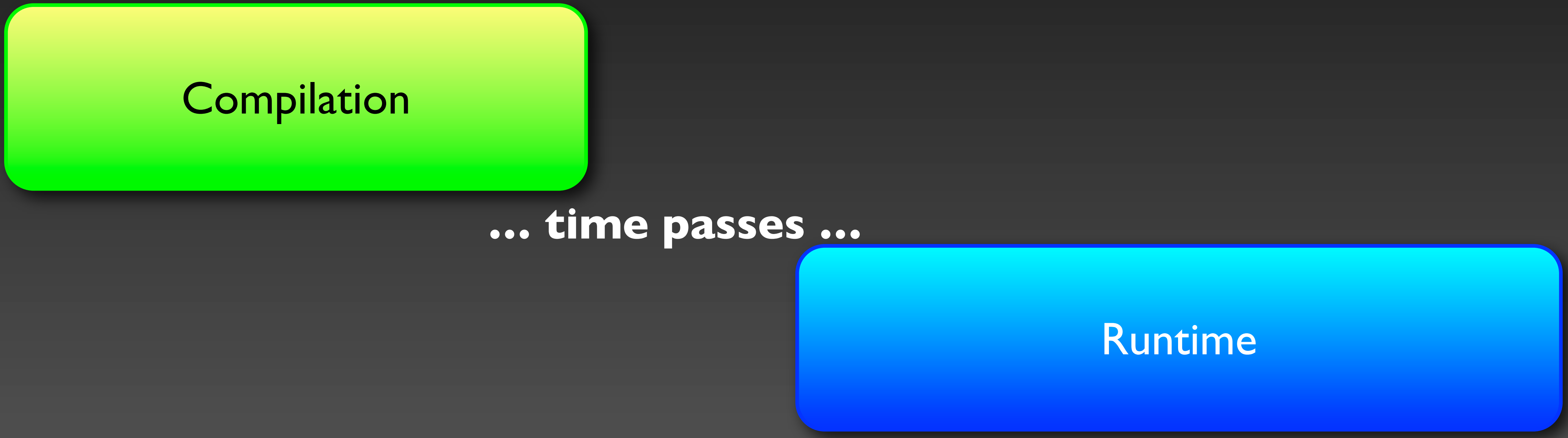
- Inspect method parameters
- Override method parameters
- Proceed
- Inspect / Override return value
- Inspect / Override thrown checked exception

```
public class CommitAfterWorker implements ComponentClassTransformWorker2 {
  private final HibernateSessionManager manager;

  private final MethodAdvice advice = new MethodAdvice() {
    …
  };

  public CommitAfterWorker(HibernateSessionManager manager) {
    this.manager = manager;
  }

  public void transform(PlasticClass plasticClass,
                        TransformationSupport support,
                        MutableComponentModel model) {
    for (PlasticMethod method :
           plasticClass.getMethodsWithAnnotation(CommitAfter.class)) {
      method.addAdvice(advice);
    }
  }
}
```

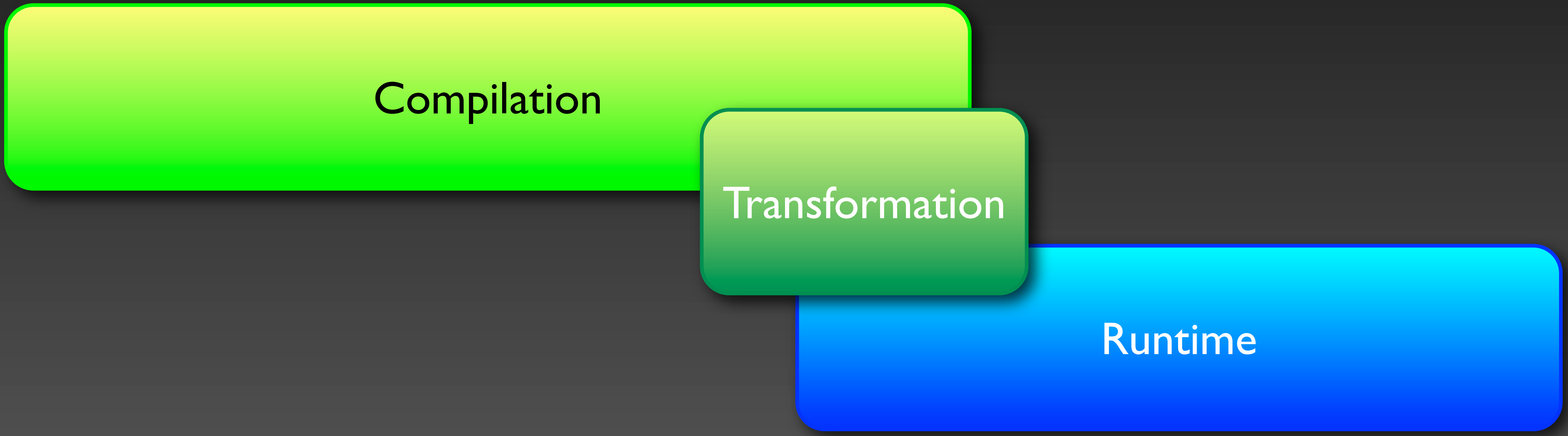
Shared Advice

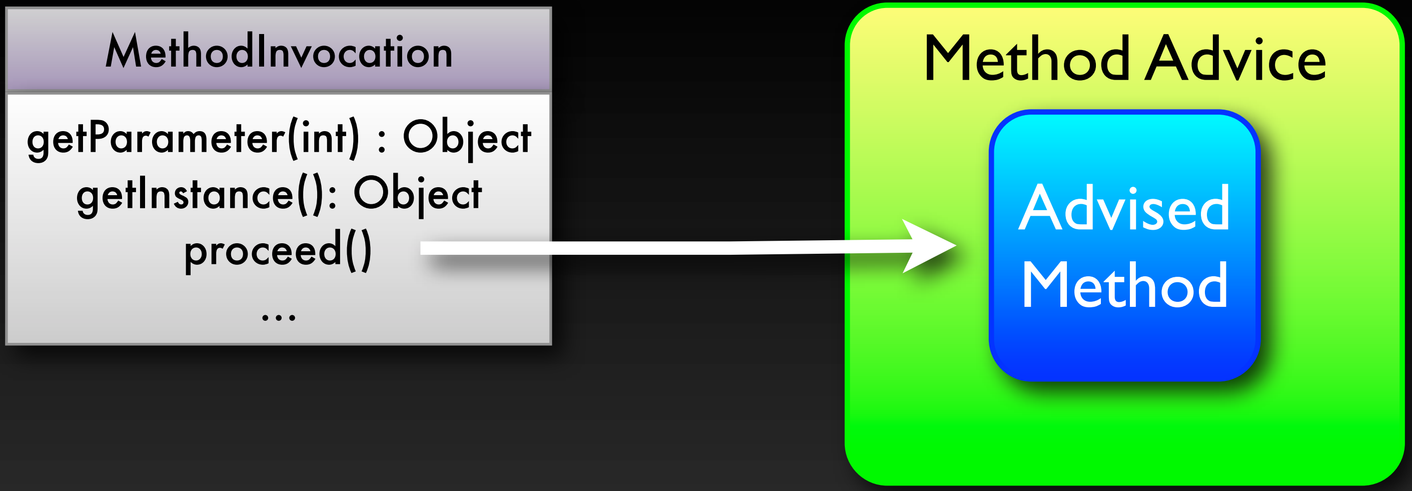
Advice object receives control when method invoked

Traditional Java



With Transfomations

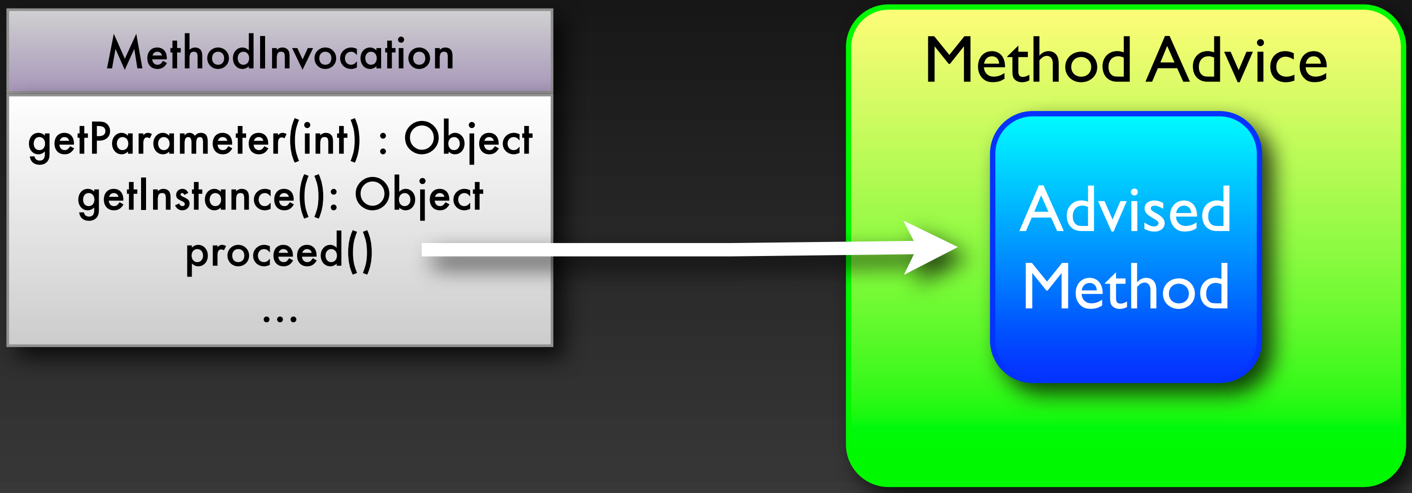


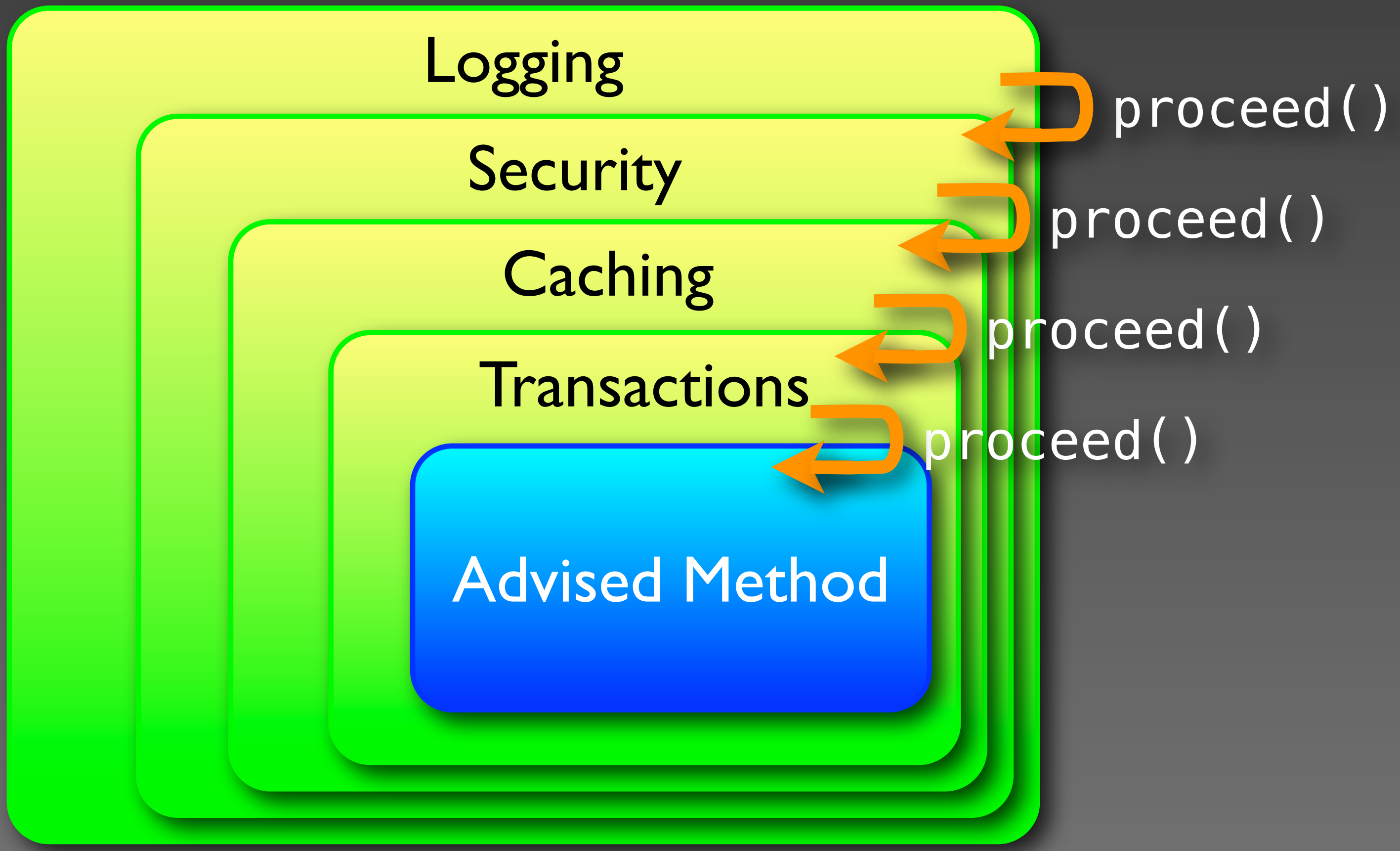


```
private final MethodAdvice advice = new MethodAdvice() {
  public void advise(MethodInvocation invocation) {
    try {
      invocation.proceed();

      // Success or checked exception:

      manager.commit();
    } catch (RuntimeException ex) {
      manager.abort();

      throw ex;
    }
  }
};
```

To the method **OR** next advice

Layering of Concerns





```
public class MemoizeAdvice implements MethodAdvice {
  private final Map<MultiKey, Object> cache = new HashMap<MultiKey, Object>();

  public void advise(MethodInvocation invocation) {
    MultiKey key = toKey(invocation);

    if (cache.containsKey(key)) {
      invocation.setReturnValue(cache.get(key));
      return;
    }

    invocation.proceed();
    invocation.rethrow();

    cache.put(key, invocation.getReturnValue());
  }

  private MultiKey toKey(MethodInvocation invocation) {
    Object[] params = new Object[invocation.getParameterCount()];

    for (int i = 0; i < invocation.getParameterCount(); i++) {
      params[i] = invocation.getParameter(i);
    }

    return new MultiKey(params);
  }
}
```

Not thread safe

Memory leak

Assumes parameters are immutable, implement equals() and hashCode()

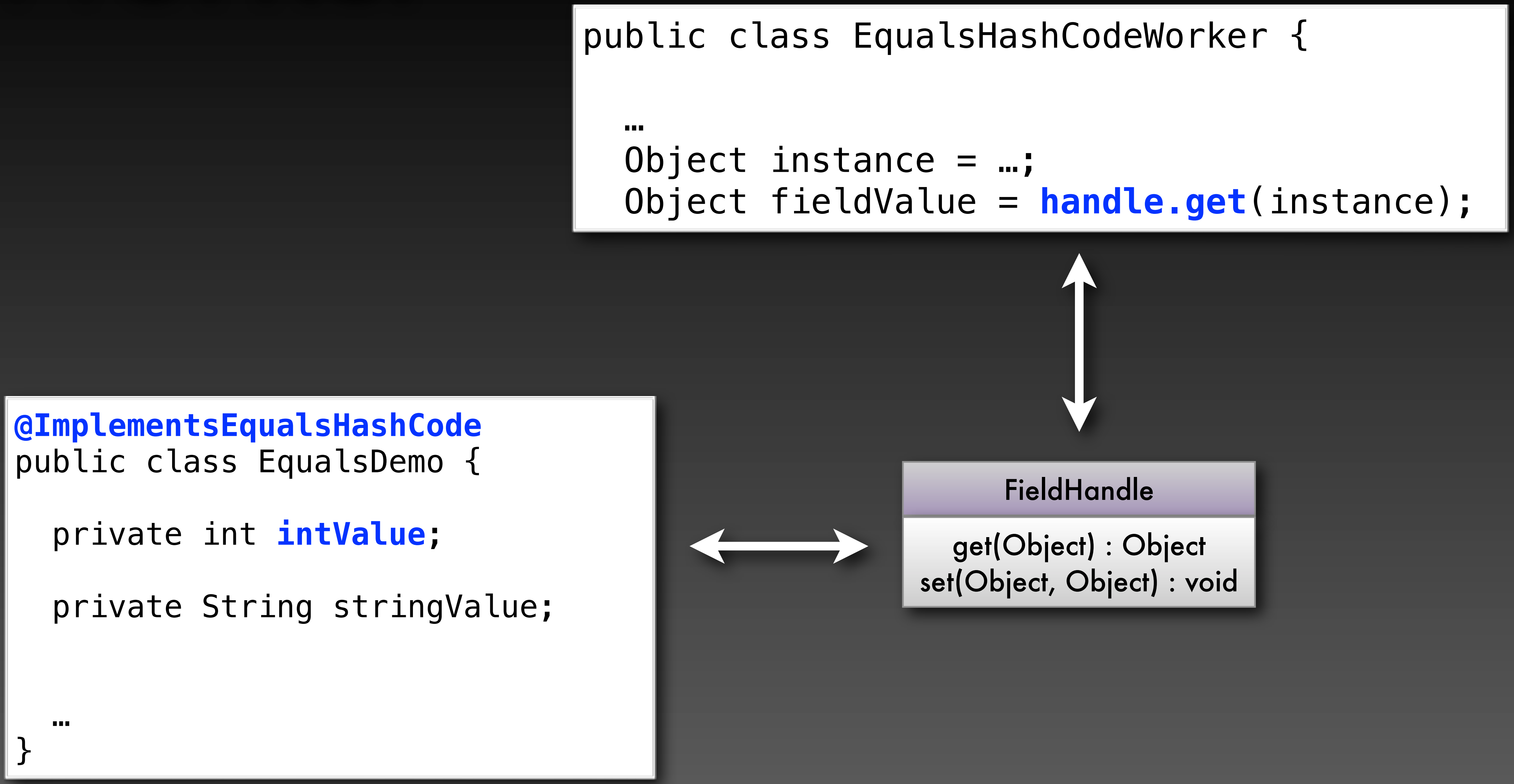
Implementing New Methods

```
@Target(ElementType.TYPE)
@Retention(RetentionPolicy.RUNTIME)
@Documented
public @interface ImplementsEqualsHashCode {

}
```

```
@ImplementsEqualsHashCode
public class EqualsDemo {

  private int intValue;

  private String stringValue;

  public int getIntValue() { return intValue; }

  public void setIntValue(int intValue) { this.intValue = intValue; }

  public String getStringValue() { return stringValue; }

  public void setStringValue(String stringValue) { this.stringValue = stringValue;   }
}
```

```java
public class EqualsDemo {

  private int intValue;

  private String stringValue;

  public int getIntValue() { return intValue; }

  public void setIntValue(int intValue) { this.intValue = intValue; }

  public String getStringValue() { return stringValue; }

  public void setStringValue(String stringValue) { this.stringValue = stringValue;   }

  public int hashCode() {
    int result = 1;

    result = 37 * result + new Integer(intValue).hashCode();
    result = 37 * result + stringValue.hashCode();

    return result;
  }
}
```

Worker

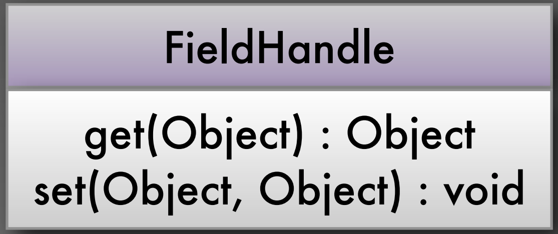


```
public class EqualsHashCodeWorker {

  …
  Object instance = …;
  Object fieldValue = handle.get(instance);
```

```
@ImplementsEqualsHashCode
public class EqualsDemo {

  private int intValue;

  private String stringValue;


  …
}
```

```
public class EqualsHashCodeWorker implements PlasticClassTransformer {

  …
  public void transform(PlasticClass plasticClass) {

    if (!plasticClass.hasAnnotation(ImplementsEqualsHashCode.class)) {
      return;
    }


  …
  }
}
```

```
List<PlasticField> fields = plasticClass.getAllFields();

final List<FieldHandle> handles = new ArrayList<FieldHandle>();

for (PlasticField field : fields) {
  handles.add(field.getHandle());
}
```



```java
private MethodDescription HASHCODE = new MethodDescription("int", "hashCode");

private static final int PRIME = 37;
```

Add a new method to the class with a default empty implementation

```java
plasticClass.introduceMethod(HASHCODE).addAdvice(new MethodAdvice() {

  public void advise(MethodInvocation invocation) {

    …
  }
});
```

```
public void advise(MethodInvocation invocation) {

  Object instance = invocation.getInstance();

  int result = 1;

  for (FieldHandle handle : handles) {

    Object fieldValue = handle.get(instance);

    if (fieldValue != null)
      result = (result * PRIME) + fieldValue.hashCode();
  }

  invocation.setReturnValue(result);

  // Don't proceed to the empty introduced method.
}
```

No Bytecode Required *

* For you

```
@ImplementsEqualsHashCode
public class EqualsDemo {

  private int intValue;

  private String stringValue;

  …

  public int hashCode() {

    return 0;
  }
}
```

Introduced method with default implementation.

```
public void advise(MethodInvocation invocation) {

  invocation.setReturnValue(…);
}
```

```
public interface ClassInstantiator {

    T newInstance();

    <V> ClassInstantiator<T> with(Class<V> valueType,
                                  V instanceContextValue);
}
```

Pass *per-instance* values to the new instance

```
Class.forName("com.example.components.MyComponent")
  .getConstructor(ComponentConfig.class)
  .newInstance(configValue);
```

```
manager.getClassInstantiator("com.example.components.MyComponent")
  .with(ComponentConfig.class, configValue)
  .newInstance();
```

```
public class EqualsDemo {

  private int intValue;

  private String stringValue;

  …

  public boolean equals(Object other) {

    if (other == null) return false;

    if (this == other) return true;

    if (this.getClass() != other.getClass()) return false;

     EqualsDemo o = (EqualsDemo)other;

     if (intValue != o.intValue) return false;

     if (! stringValue.equals(o.stringValue)) return false;

     return true;
   }
}
```

```
private MethodDescription EQUALS = new MethodDescription("boolean",
  "equals", "java.lang.Object");
```

```
plasticClass.introduceMethod(EQUALS).addAdvice(new MethodAdvice() {

  public void advise(MethodInvocation invocation) {

    Object thisInstance = invocation.getInstance();
    Object otherInstance = invocation.getParameter(0);

    invocation.setReturnValue(isEqual(thisInstance, otherInstance));

    // Don't proceed to the empty introduced method.
  }

  private boolean isEqual(…) { … }
}
```

```java
private boolean isEqual(Object thisInstance, Object otherInstance) {
  if (thisInstance == otherInstance) {
    return true;
  }

  if (otherInstance == null) {
    return false;
  }

  if (!(thisInstance.getClass() == otherInstance.getClass())) {
    return false;
  }

  for (FieldHandle handle : handles) {
    Object thisValue = handle.get(thisInstance);
    Object otherValue = handle.get(otherInstance);

    if (!(thisValue == otherValue || thisValue.equals(otherValue))) {
      return false;
    }
  }

  return true;
}
```

Testing with Spock

```
class EqualsHashCodeTests extends Specification {

  PlasticManagerDelegate delegate =
    new StandardDelegate(new EqualsHashCodeWorker())

  PlasticManager mgr = PlasticManager.withContextClassLoader()
    .packages(["examples.plastic.transformed"])
    .delegate(delegate)
    .create();


  ClassInstantiator instantiator = mgr.getClassInstantiator(EqualsDemo.class.name)
```

The delegate manages one or more workers

Only top-level classes in example.plastic.transformed are passed to the delegate

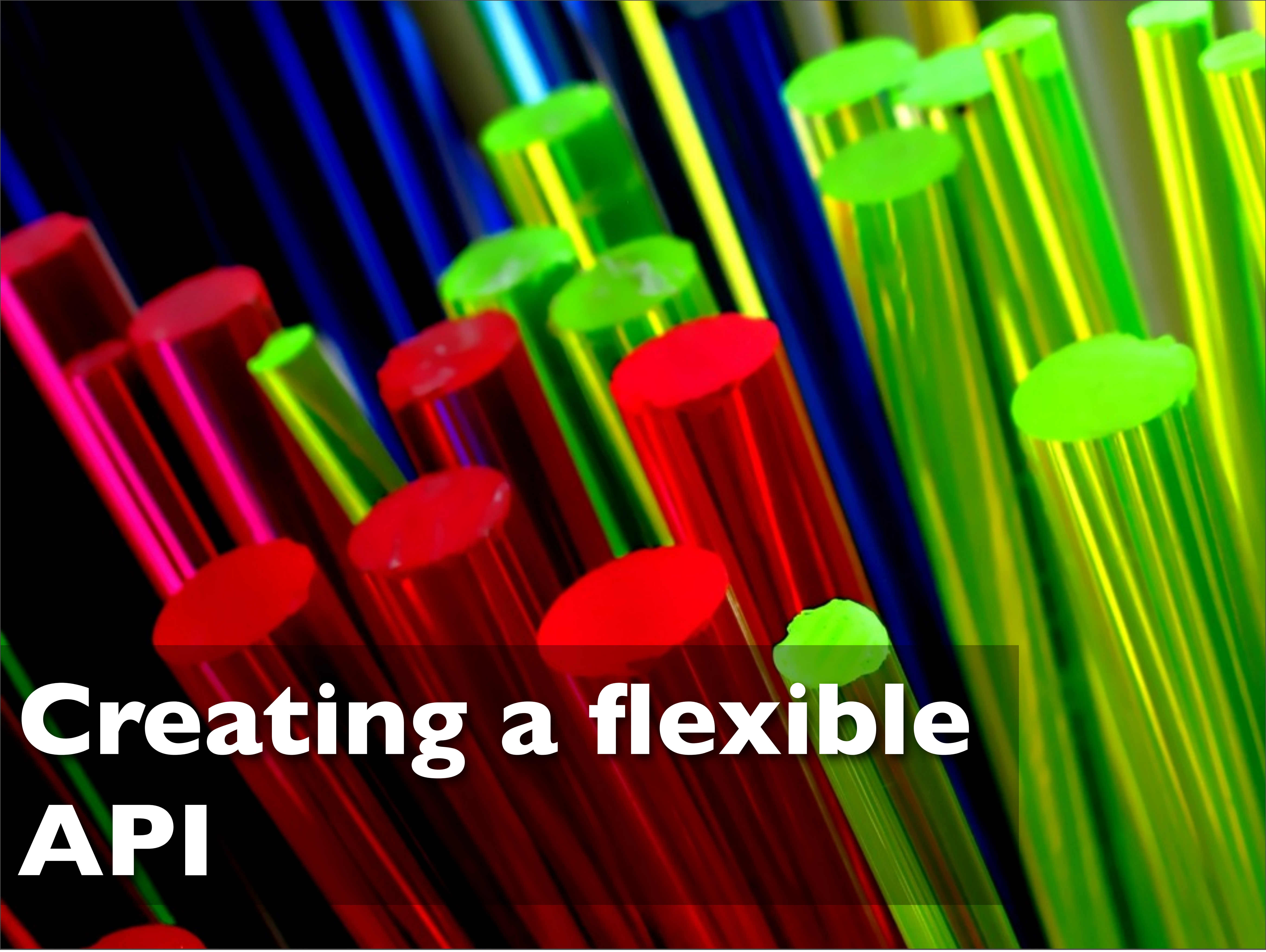
examples.plastic.transformed.EqualsDemo

http://code.google.com/p/spock/

```
def "simple comparison"() {
  def instance1 = instantiator.newInstance()
  def instance2 = instantiator.newInstance()
  def instance3 = instantiator.newInstance()
  def instance4 = instantiator.newInstance()

  instance1.intValue = 99
  instance1.stringValue = "Hello"

  instance2.intValue = 100
  instance2.stringValue = "Hello"

  instance3.intValue = 99
  instance3.stringValue = "Goodbye"

  instance4.intValue = 99
  instance4.stringValue = "Hello"

  expect:

  instance1 != instance2

  instance1 != instance3

  instance1 == instance4
}
```

Groovy invokes getters & setters

Groovy: == operator invokes equals()



Creating a flexible API

API != Interface

Layout.tml

```
<t:actionlink t:id="reset">reset session</t:actionlink>
```

```
public class Layout {

  @Inject
  private Request request;

  void onActionFromReset() {
    request.getSession(true).invalidate();
  }
}
```

Naming convention + expected behavior == API

Using Traditional API

```
public class Layout implements HypotheticalComponentAPI {

  @Inject
  private Request request;

  public void registerEventHandlers(ComponentEventRegistry registry) {

    registry.addEventHandler("action", "reset",
      new ComponentEventListener() {
        public boolean handle(ComponentEvent event) {
          request.getSession(true).invalidate();
          return true;
        }
      });
  }
}
```

Essence

```
public class Layout {

  @Inject
  private Request request;

  void onActionFromReset() {
    request.getSession(true).invalidate();
  }
}
```

Essence

```java
public class Layout implements Component {

  @Inject
  private Request request;

  void onActionFromReset() {
    request.getSession(true).invalidate();
  }

  public boolean dispatchComponentEvent(ComponentEvent event) {
    boolean result = false;

    if (event.matches("action", "reset", 0)) {
      onActionFromReset();
      result = true;
    }

    …

    return result;
  }

  …
}
```

0 is the parameter count

There's our rigid interface

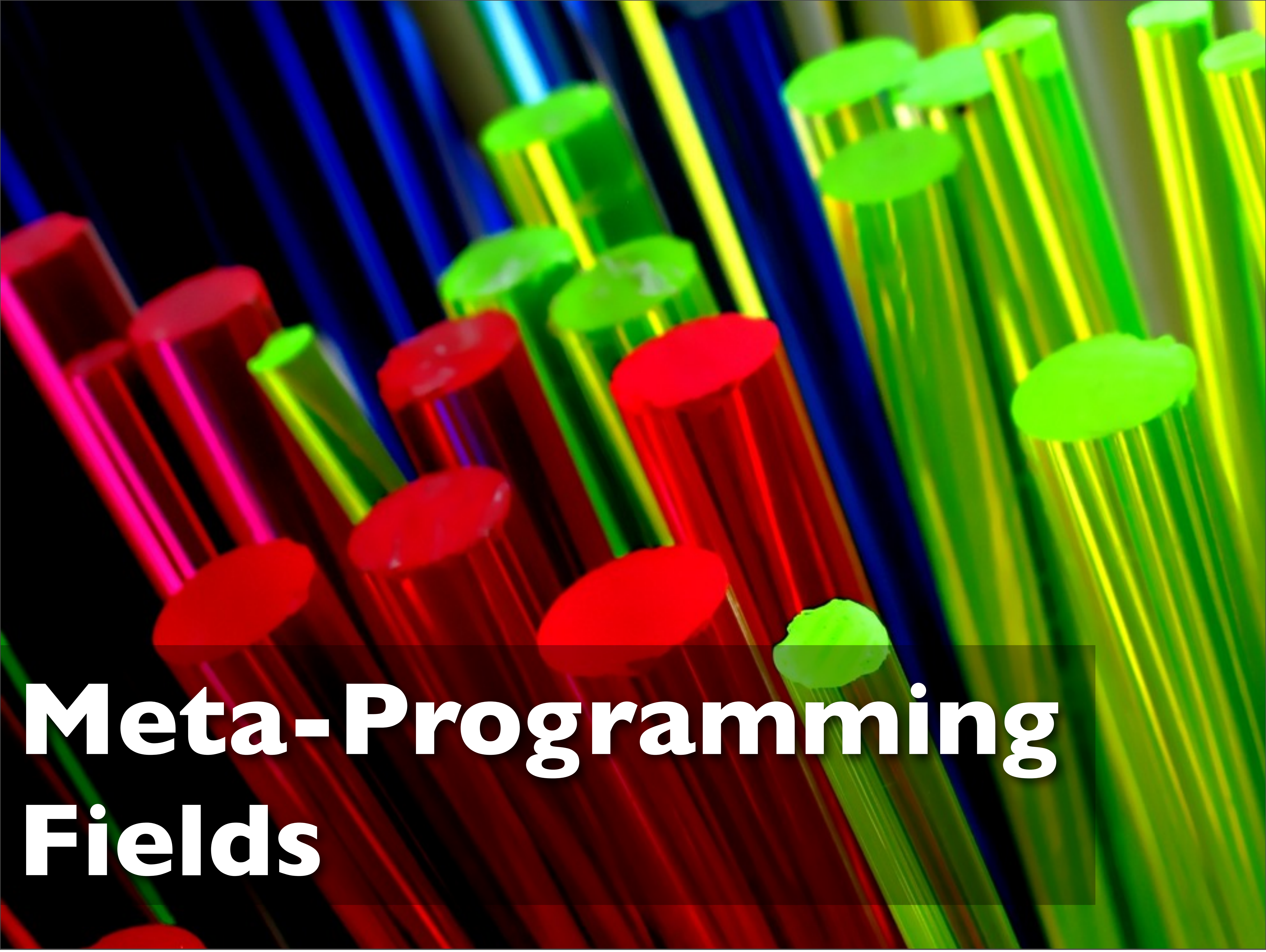
```java
public interface Component {

  boolean dispatchComponentEvent(ComponentEvent event);

  …
}
```

```
for (PlasticMethod method : matchEventMethods(plasticClass)) {

  String eventName = toEventName(method);
  String componentId = toComponentId(method);

  MethodAdvice advice = createAdvice(eventName, componentId, method);

  PlasticMethod dispatch =   plasticClass.introduceMethod(
    TransformConstants.DISPATCH_COMPONENT_EVENT_DESCRIPTION);

  dispatch.addAdvice(advice);
}
```

```
private MethodAdvice createAdvice(final String eventName,
    final String componentId,
    PlasticMethod method) {
  final MethodHandle handle = method.getHandle();

  return new MethodAdvice() {
    public void advise(MethodInvocation invocation) {

    invocation.proceed();

    ComponentEvent event = (ComponentEvent) invocation.getParameter(0);

    if (event.matches(eventName, componentId, 0)) {
      handle.invoke(invocation.getInstance());
      invocation.rethrow();

      invocation.setReturnValue(true);
    }
  };
}
```

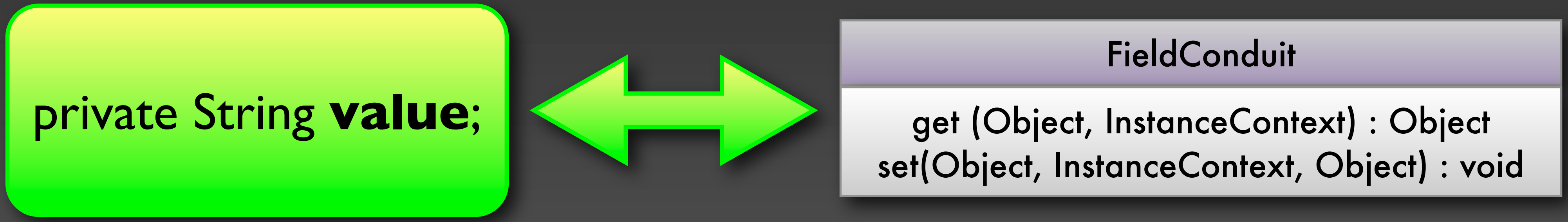
Invoke default or super-class implementation first

Simplification – real code handles methods with parameters



Meta-Programming Fields

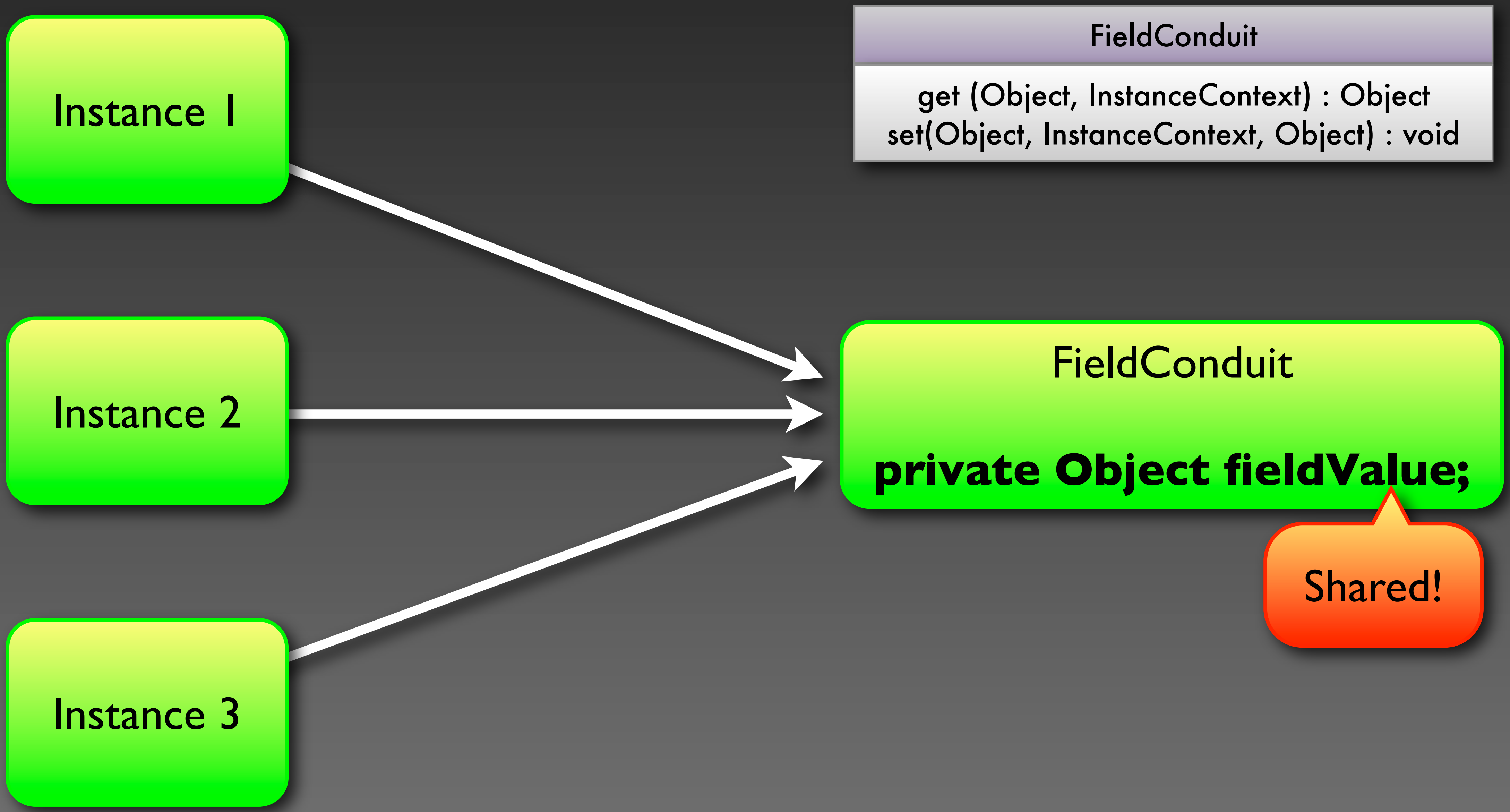
```
/**
 * Identifies a field that may not store the value null.
 *
 */
@Target(ElementType.FIELD)
@Retention(RetentionPolicy.RUNTIME)
@Documented
public @interface NotNull {

}
```


private String **value**;
FieldConduit
get (Object, InstanceContext) : Object
set(Object, InstanceContext, Object) : void

```
public class NullCheckingConduit implements FieldConduit<Object> {

  private final String className;

  private final String fieldName;

  private Object fieldValue;

  private NullCheckingConduit(String className, String fieldName) {
    this.className = className;
    this.fieldName = fieldName;
  }

  public Object get(Object instance, InstanceContext context) {
    return fieldValue;
  }

  public void set(Object instance, InstanceContext context, Object newValue) {

    if (newValue == null)
      throw new IllegalArgumentException(String.format(
          "Field %s of class %s may not be assigned null.",
          fieldName, className));

    fieldValue = newValue;
  }
```

Replaces actual field

```
field.setConduit(new NullCheckingConduit(className, fieldName));
```


FieldConduit
get (Object, InstanceContext) : Object
set(Object, InstanceContext, Object) : void
Instance 1
Instance 2
Instance 3
FieldConduit
private Object fieldValue;
Shared!

```
@SuppressWarnings({"unchecked"})
public void transform(PlasticClass plasticClass) {
  for (PlasticField field : plasticClass
          .getFieldsWithAnnotation(NotNull.class)) {

    final String className = plasticClass.getClassName();
    final String fieldName = field.getName();

    field.setComputedConduit(new ComputedValue() {

      public Object get(InstanceContext context) {
        return new NullCheckingConduit(className, fieldName);
      }
    });
  }
}
```

ComputedValue: A **Factory** that is executed inside the transformed class' **constructor**

```
class NotNullTests extends Specification {

  …

  def "store null is failure"() {

    def o = instantiator.newInstance()

    when:

    o.value = null

    then:

    def e = thrown(IllegalArgumentException)

    e.message == "Field value of class 
examples.plastic.transformed.NotNullDemo may not be assigned null."
  }
}
```

Private Fields Only

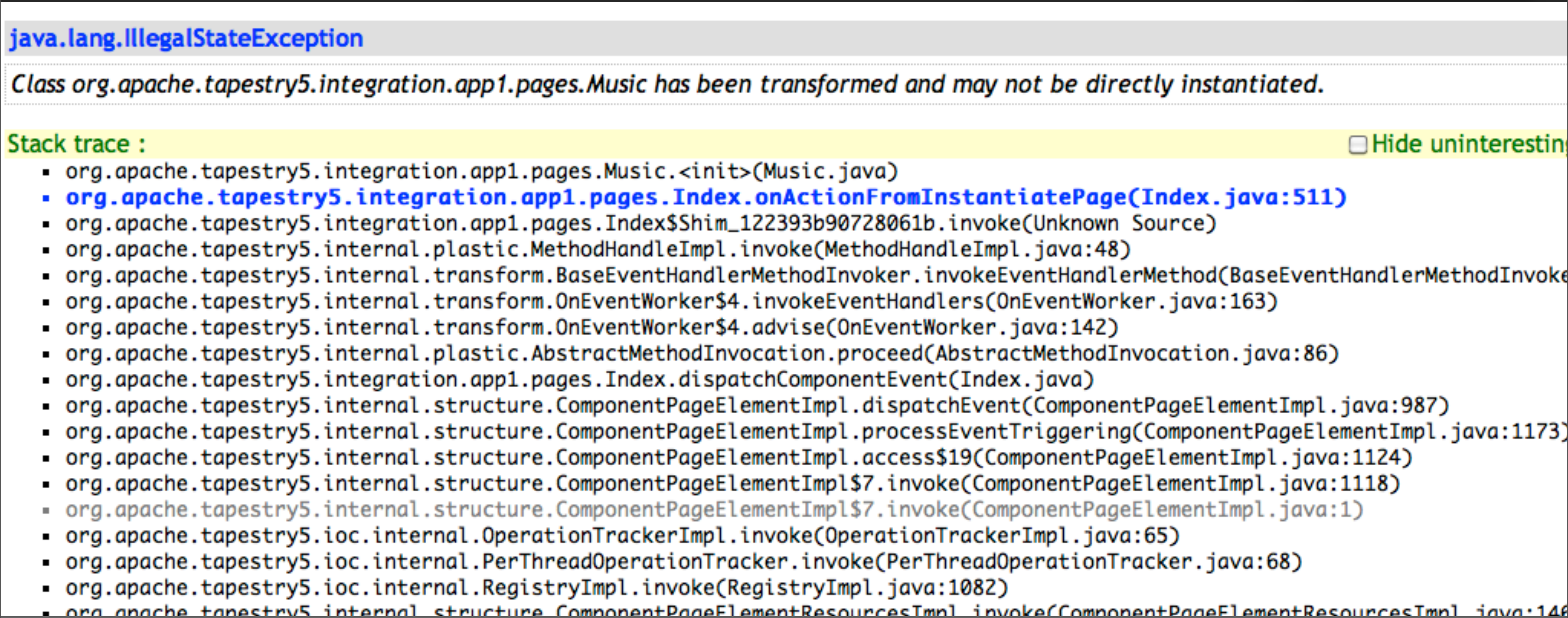
```
package examples.plastic.transformed;

import examples.plastic.annotations.NotNull;

public class NotNullDemo {

  @NotNull
  public String value;

}
```

```
java.lang.IllegalArgumentException: Field value of class examples.plastic.transformed.NotNullDemo is
not private. Class transformation requires that all instance fields be private.
  at org.apache.tapestry5.internal.plastic.PlasticClassImpl.<init>()
  at org.apache.tapestry5.internal.plastic.PlasticClassPool.createTransformation()
  at org.apache.tapestry5.internal.plastic.PlasticClassPool.getPlasticClassTransformation()
  at org.apache.tapestry5.internal.plastic.PlasticClassPool.loadAndTransformClass()
  at org.apache.tapestry5.internal.plastic.PlasticClassLoader.loadClass()
  at java.lang.ClassLoader.loadClass()
  at org.apache.tapestry5.internal.plastic.PlasticClassPool.getClassInstantiator()
  at org.apache.tapestry5.plastic.PlasticManager.getClassInstantiator()
  at examples.plastic.PlasticDemosSpecification.createInstantiator()
  at examples.plastic.NotNullTests.setup()
```

```java
package examples.plastic.transformed;

import examples.plastic.annotations.NotNull;

public class NotNullDemo {

  @NotNull
  private String value;

  public String getValue() {
    return get_value();  // was return value;
  }

  public void setValue(String value) {
    set_value(value); // was this.value = value;
  }


  …
}
```

```java
package examples.plastic.transformed;

import examples.plastic.annotations.NotNull;

public class NotNullDemo {
  …

  private final InstanceContext ic;

  private final FieldConduit valueConduit;

  public NotNullDemo(StaticContext sc, InstanceContext ic) {
    this.ic = ic;
    valueConduit = (FieldConduit) ((ComputedValue) sc.get(0)).get(ic);
  }

  String get_value() { return (String) valueConduit.get(this, ic); }

  void set_value(String newValue) {
    valueConduit.set(this, ic, newValue);
  }
}
```

No More **new**



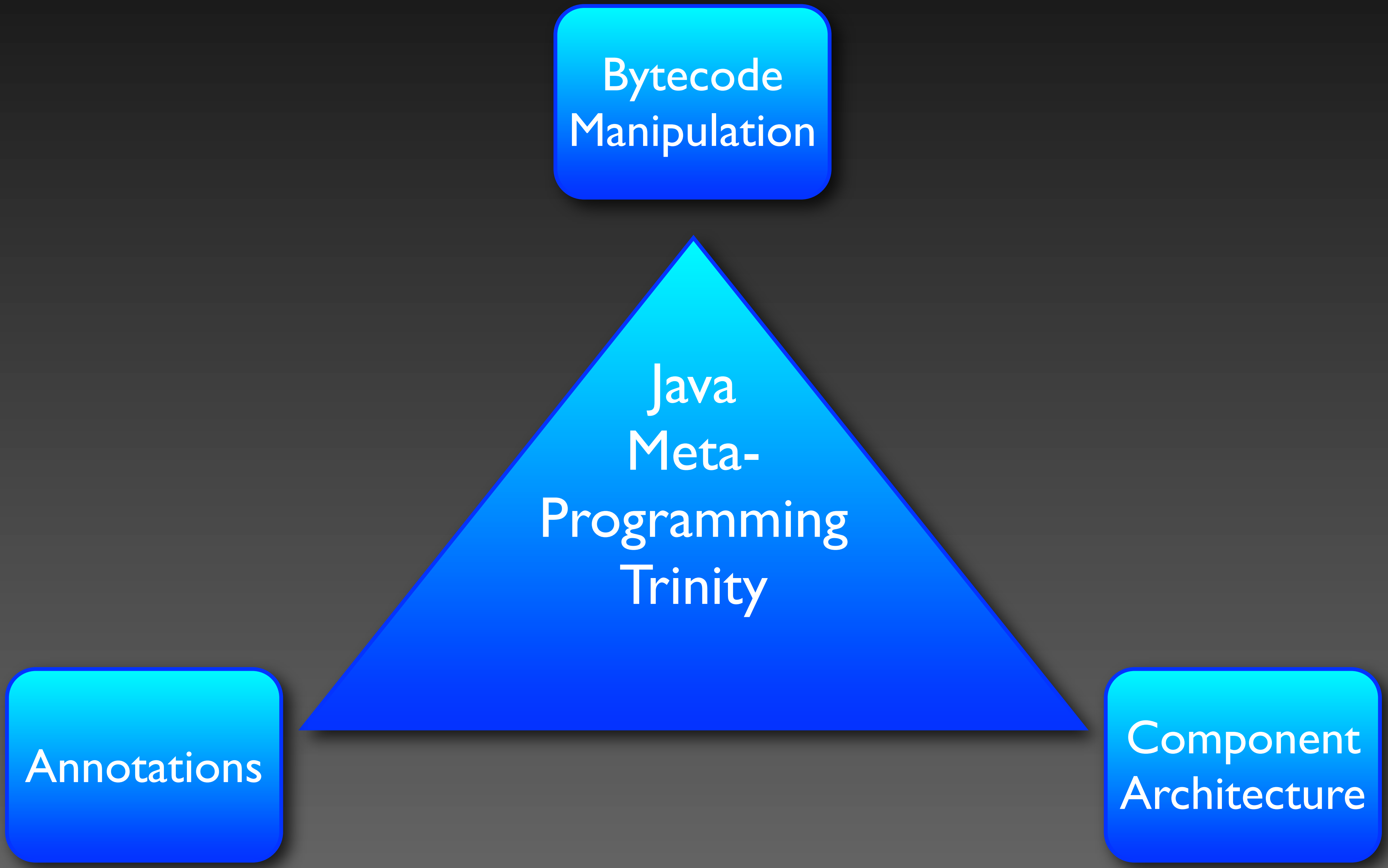
java.lang.IllegalStateException

Class org.apache.tapestry5.integration.app1.pages.Music has been transformed and may not be directly instantiated.

Stack trace : ☐ Hide uninterestin

```
▪ org.apache.tapestry5.integration.app1.pages.Music.<init>(Music.java)
▪ org.apache.tapestry5.integration.app1.pages.Index.onActionFromInstantiatePage(Index.java:511)
▪ org.apache.tapestry5.integration.app1.pages.Index$Shim_122393b90728061b.invoke(Unknown Source)
▪ org.apache.tapestry5.internal.plastic.MethodHandleImpl.invoke(MethodHandleImpl.java:48)
▪ org.apache.tapestry5.internal.transform.BaseEventHandlerMethodInvoker.invokeEventHandlerMethod(BaseEventHandlerMethodInvok
▪ org.apache.tapestry5.internal.transform.OnEventWorker$4.invokeEventHandlers(OnEventWorker.java:163)
▪ org.apache.tapestry5.internal.transform.OnEventWorker$4.advise(OnEventWorker.java:142)
▪ org.apache.tapestry5.internal.plastic.AbstractMethodInvocation.proceed(AbstractMethodInvocation.java:86)
▪ org.apache.tapestry5.integration.app1.pages.Index.dispatchComponentEvent(Index.java)
▪ org.apache.tapestry5.internal.structure.ComponentPageElementImpl.dispatchEvent(ComponentPageElementImpl.java:987)
▪ org.apache.tapestry5.internal.structure.ComponentPageElementImpl.processEventTriggering(ComponentPageElementImpl.java:1173
▪ org.apache.tapestry5.internal.structure.ComponentPageElementImpl.access$19(ComponentPageElementImpl.java:1124)
▪ org.apache.tapestry5.internal.structure.ComponentPageElementImpl$7.invoke(ComponentPageElementImpl.java:1118)
▪ org.apache.tapestry5.internal.structure.ComponentPageElementImpl$7.invoke(ComponentPageElementImpl.java:1)
▪ org.apache.tapestry5.ioc.internal.OperationTrackerImpl.invoke(OperationTrackerImpl.java:65)
▪ org.apache.tapestry5.ioc.internal.PerThreadOperationTracker.invoke(PerThreadOperationTracker.java:68)
▪ org.apache.tapestry5.ioc.internal.RegistryImpl.invoke(RegistryImpl.java:1082)
▪ org.apache.tapestry5.internal.structure.ComponentPageElementResourcesImpl.invoke(ComponentPageElementResourcesImpl.java:14
```

Conclusion


Bytecode Manipulation
Java Meta-Programming Trinity
Annotations
Component Architecture

Structure

- Best With Managed Lifecycle
  - **new** no longer allowed
  - Instantiation by class name
- Framework / Code Interactions via Interface(s)
  - Modify components to implement Interface(s)
- Blurs lines between compile & runtime

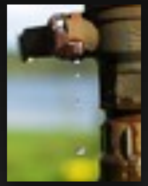
Not Covered

- Field injection
- Direct bytecode builder API

- Home Page
  http://howardlewisship.com
- Tapestry Central Blog
  http://tapestryjava.blogspot.com/
- Examples Source
  https://github.com/hlship/plastic-demos
- Apache Tapestry
  http://tapestry.apache.org
- ASM
  http://asm.ow2.org/

Q & A

Image Credits



© 2008 Andrei Niemimäki
http://www.flickr.com/photos/andrein/2502654648

© 2008 Don Solo
http://www.flickr.com/photos/donsolo/2234406328/

© 2006 Alexandre Duret-Lutz
http://www.flickr.com/photos/donsolo/2234406328/

© 2010 Trey Ratcliff
http://www.flickr.com/photos/stuckincustoms/4820290530

© 2007 Sandra Gonzalez
http://www.flickr.com/photos/la-ultima-en-saber/1209594912/

© 2009 Darwin Bell
http://www.flickr.com/photos/53611153@N00/3645850983/

© 2008 *Katch*
http://www.flickr.com/photos/13533187@N00/2371264501

© 2010 Christian Guthier
http://www.flickr.com/photos/60364452@N00/4445705276/

© 2011 Randall Muroe
http://xkcd.com/974/