



The Codeless Code

*Fables and Kōans for the
Software Developer*

**thecodelesscode.com github.
com/karianna/codelesscode**

README!

This is a storytelling presentation - whilst you may glean some snippets of enlightenment from reading these slides, the experience will be like the fish living in the Ocean...

It does not comprehend that there is a sky

Disclaimers

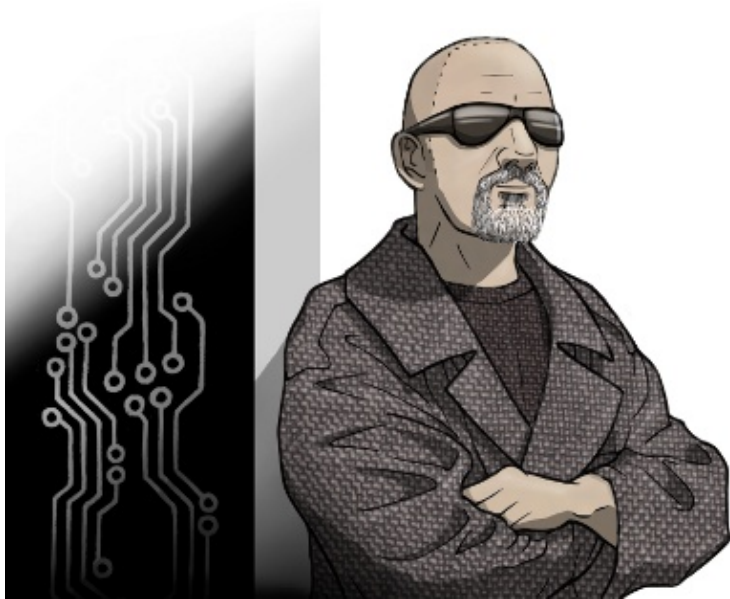
The stories in this collection are works of fiction, synthesized from ideas acquired over many years. Any resemblance to persons living or dead, or to corporations, situations, religions, philosophies, gross acts of injustice, or other works of fiction — past, present, or future imperfect — is purely coincidental.

No* *actual monks were harmed in the making of this presentation.*

*** Well, maybe a few...**

"Cultural appropriation" is a topic that seems to come up a lot these days when discussing art of all types. We humbly accept suggestions on how to run this presentation more sensitively - especially if you're a member of one of the groups that you feel is being misrepresented in a cringe-worthy way.

Qi - The Scribe



Creator & Illustrator
[Codelesscode.com](http://codelesscode.com)

Diabolical Developer - Story Teller



Java Champion &
CEO @ **www.jclarity.**
com

In a land far far away....





Section I - Code

- *Case I - The small stuff*
- *Case III - Encapsulation*
- *Case VI - Empty*
- *Case XIII - Evolution*
- *Case XIX - By any other name*
- *Case XXXIV - The naming of names*

Case 1 - The small stuff

*If you take care of the
little things, the big things
take care of themselves*



Case 1 - The small stuff (in Java)

```
public class TheSmallStuff {  
  
    private final static Logger LOGGER = LoggerFactory.getLogger(TheSmallStuff.class);  
  
    private int addend1 = 0;  
    private int addend2 = 0;  
  
    public static void main(String[] args) {  
        TheSmallStuff me = new TheSmallStuff();  
        me.generateSum();  
    }  
  
    private void generateSum() {  
        LOGGER.info("Sum of two addends is: " + addend1 + addend2 + "');  
    }  
}
```

Case III - Encapsulation

*Keep your innards to
yourself*



Case III – Encapsulation (in Java)

```
public class Monk {  
    public Stomach stomach = new Stomach();  
    private Mouth mouth = new Mouth();  
    public Mouth getMouth() { return mouth; }  
}
```

Case III - Encapsulation (in Java)

```
package com.codelesscode.presentation.encapsulation;

/** Case III - Encapsulation */
public class Encapsulation {

    private Monk monk = new Monk();

    public static void main(String[] args) {
        Encapsulation me = new Encapsulation();
        me.feedingTime();
    }

    private void feedingTime() {
        // Implementation goes here...
        monk.stomach.feed();
    }
}
```

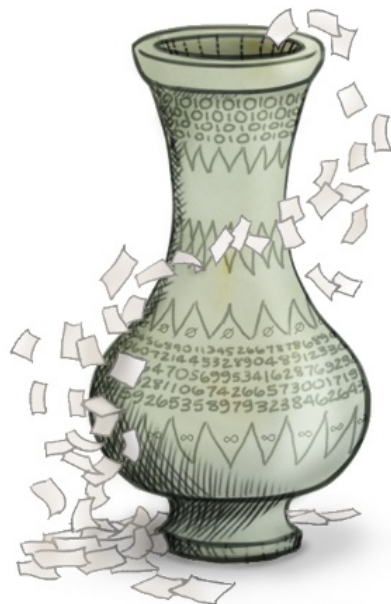
Case III - Result

ERROR c.c.p.encapsulation.Stomach

- Monk ends up with food mashed into robes.

Case VI - Empty

*Empty collections are
better than `null`*



Case VI - Empty (in Java)

```
public class Empty {  
  
    private final static Logger LOGGER = LoggerFactory.getLogger(Empty.class);  
  
    /** Is initialised to null */  
    List<?> dataCollection = null; // new ArrayList<>();  
  
    public static void main(String[] args) {  
        Empty me = new Empty();  
        me.printCollectionSize();  
    }  
  
    private void printCollectionSize() {  
        LOGGER.info("Collection Size is:" + dataCollection.size());  
    }  
  
}
```


Case XIII – Evolution

*Stand on the shoulders of Giants
but not follow others blindly*

Case XIII - Evolution (In Java)

```
/* What if this runs on a single core machine? */  
public static void main(String args[]) {  
    Collection<String> filteredNames = getNames()  
                                     .parallelStream()  
                                     .filter(s -> s.startsWith("Foo"))  
                                     .collect(Collectors.toList());  
  
    System.out.print(filteredNames);  
}
```

Case XIX - By any other name

*Magic numbers have
no semantic meaning*

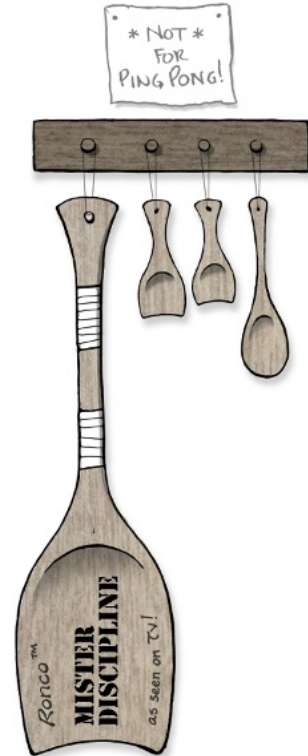


Case XIX - By any other name (in Java)

```
public class ByAnyOtherName {  
    public static int NUMBER_OF_DAYS_I_WILL_PERFORM_PENANCE = 7;  
}
```

Case XXXIV - The naming of names

"Verb your expletive nouns!"



Case XXXIV - The naming of names

```
// This is an action (verb), not a noun
public class Walking {

    private Human human;

    // This is a noun, not an action (verb)
    public void human() {
        human.walk();
    }
}
```

```
// A Noun!
public class Human {

    // A verb!
    public void walk() {
        // TODO Get that lazy Human walking
    }
}
```

Case XXXIV - The naming of names

```
public class TheNamingOfNames {  
    // Do NOT do this  
    int x;  
    int x1;  
    int x2;  
    int x3;  
    int x4;  
    int x5;  
    int x6;  
    int x7;  
    int x8;  
    int x9;  
    int x10;  
  
    public int performTaxCalculation() {  
        // Hmm was it x2 or x3 we needed to divide by?  
        return x1 / x2 * 100;  
    }  
}
```


Case XXXIV - The naming of names

```
public void lambdaNaming() {  
    Stream<String> nameStream = names.stream();  
    // n is not a good name  
    List<String> tNames = nameStream.filter(n -> n.startsWith("T")).  
                                   collect(Collectors.toList());  
}  
  
public void exceptionNaming() {  
    try {  
        Files.delete(Paths.get("Nonsense Path"));  
    }  
    // e is not a good name  
    catch (IOException e) {  
        e.printStackTrace();  
    }  
}
```



Section II - Testing

- *Case VII - The enemy of Good*
- *Case XXIX - Goldfish*
- *Case XLIV - The backwards monk*
- *Case LVIII - Enough to hang yourself by*
- *Case LXIII - Shackles*

Case VII – The enemy of good

*Perfection is the
enemy of good*



Case XXIX - Goldfish

*You are a programmer,
automate*

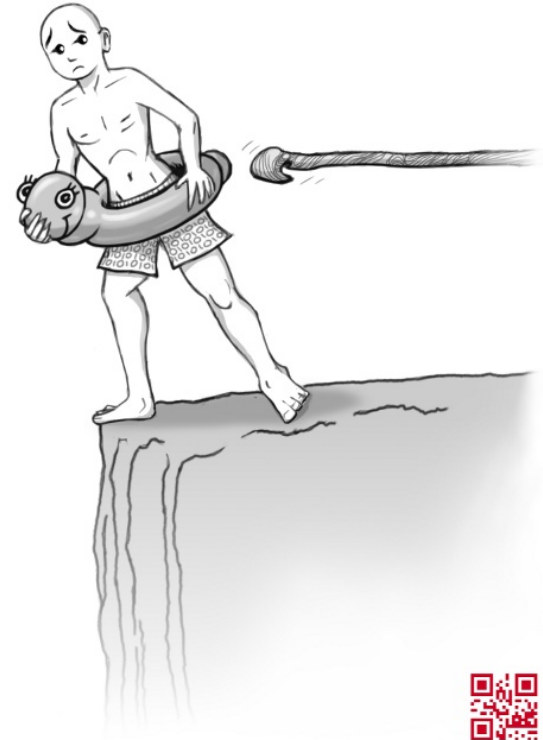


Case XXIX - Goldfish (In Chef)

```
# Upgrade if current version is out of date
ark jpdm_dir do
  url jpdm_url
  path node[:jpdm][:install_location]
  owner node[:jpdm][:user]
  group node[:jpdm][:user]
  action :put
  strip_components 0
  subscribes :create, "remote_file[#{tmp_version_file}]"
  only_if {InstallJpdm.getLocalVersion(version_file) != InstallJpdm.getLatestVersion()}
  not_if {force_install}
  notifies :restart, "service[jpdm]"
  ignore_failure ignore_errors
end
```

Case XLIV – The backwards monk

*To go forwards you
must go backwards*



Case XLIV – The backwards monk

```
@Test
public void returnFalseIfURIIsBad() {
    LicenseClient licenseClient = new LicenseClient("username", "", "http://<server>:<port>");
    assertFalse(licenseClient.requestFloatingLicense());
}
```


Case LVIII – Enough to hang yourself by

The whole system matters



Case LXIII - Shackles

Let it go



Case LXIII - Shackles (In Java)

rm -rf *



Section III - Performance

- *Case IX - Infinities*
- *Case ?? - Heroism*

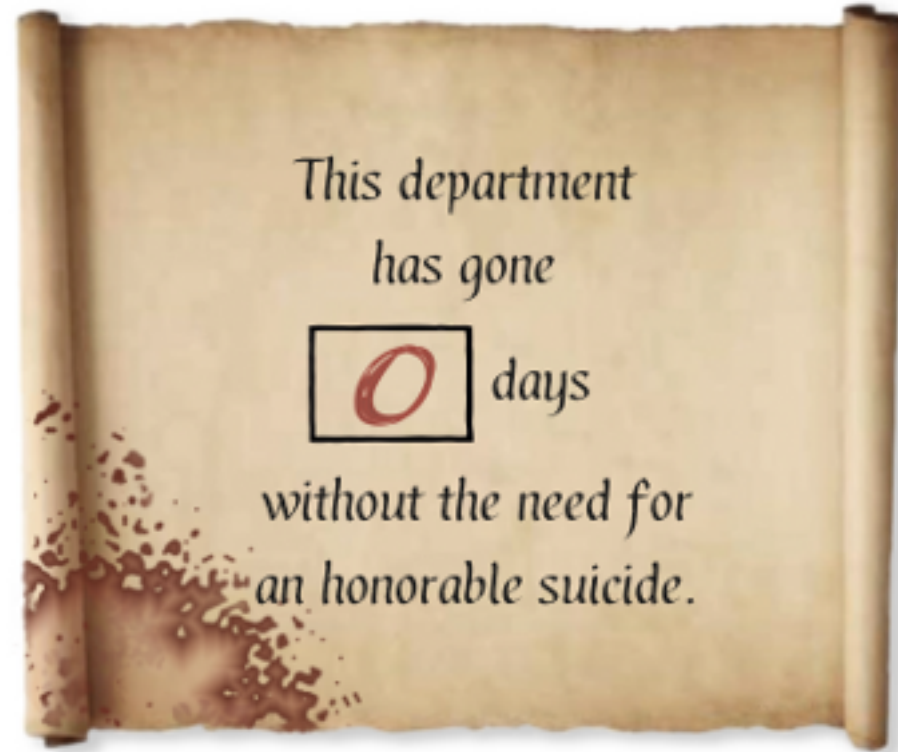
Case IX - Infinities

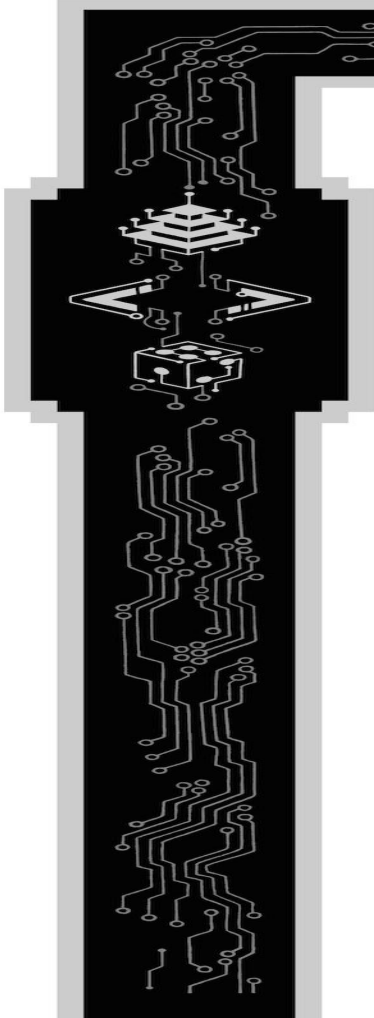
*All things must
come to an end*



Case ?? - Heroism

Heroes die young





Section IV - Non Code

- *Case XVI - Documentation*
- *Case XXVI - The uncorrected monk*
- *Case XXXII - Weeds*
- *Case LXIX - Up To Date*

Case XVI – Documentation

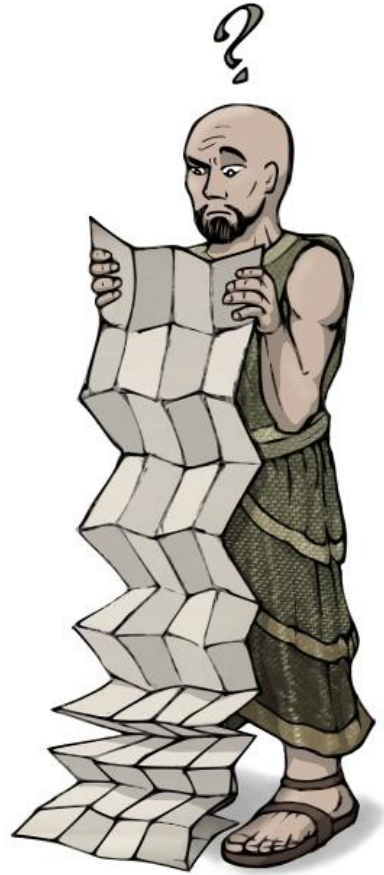
Code is not everything

Case XVI - Documentation

```
//113.325: [Full GC (System.gc()) 37M->32M(96M), 0.0943030 secs]
else if ((trace = FULL_GC.parse(line)) != null) {
    if (trace.gcCause(3, 0) == GCause.JAVA_LANG_SYSTEM)
        forwardReference = new G1SystemGC(trace.getDateTimeStamp(), trace.getPauseTime());
    else
        forwardReference = new G1FullGC(trace.getDateTimeStamp(), trace.gcCause(3, 0), trace.getPauseTime());
}
```

Case LXIX - Up to date

The source is long...



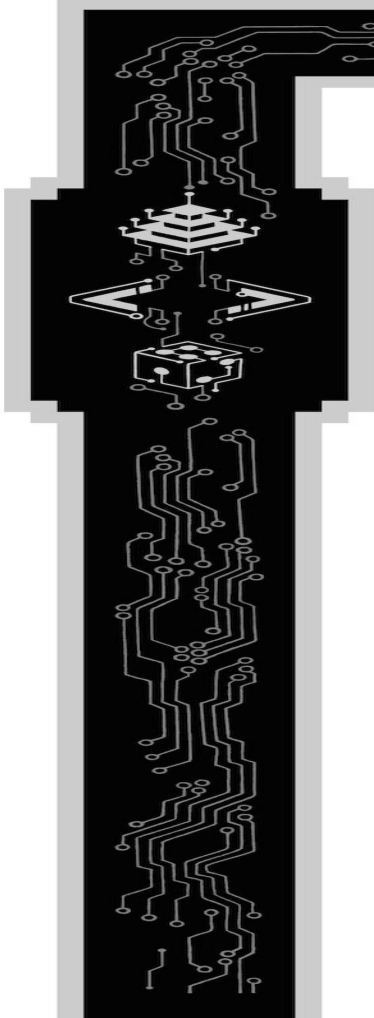
Case XXVI – The uncorrected monk

*Lead by example and
do not tolerate excuses*

Case XXXII - Weeds

*The path to the
answer is as important
as the answer...*





Section V – Third Parties

- *Case X – Pride*
- *Case XXII – Safety*

Case XXII - Safety

DefaultImpl extends ***Abstract***
implements ***Interface***

Case X - Pride

NotInventedHere.java

Case X - Pride (at jClarity)

```
package com.jclarity.jpdm.communications.logging;

/**
 * This is not meant to be a generic logging interface, simply something that
 * must be used within jpdm-comms and jpdm-stopwatch.
 */
public interface NIHLogger {

    ... public void error(String message, Throwable cause);

    ... public void error(String string);

    ... public void debug(String message);

    ... public boolean isEnabled();

    ... public void debug(String string, Throwable t);

    ... public void trace(String message);
}
```



Enlightened?

thecodelesscode.com
github.com/karianna/codelesscode



Case LXV - Two choices

*The method that takes
Object will cheerfully accept
LeftFoot.*



Case XXV – Compromise

Prototype

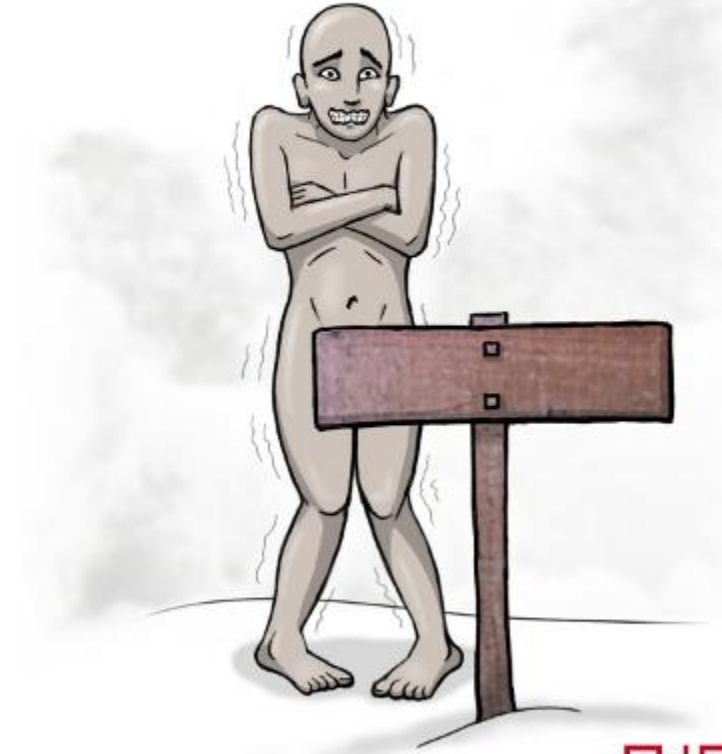


Section VI - Misc

- *Case LXXII - Too Eager*
- *Case LXXIII - The white pearl*
- *Case LXIV - Bathwater*

Case LXXII – Too eager

Handle laziness with grace



Case LXXIII - The white pearl

Logging causes haystacks



Case LXXIV - Bathwater

Levels matter



Case 67

Case 73

Case 41 - Garbage

Case 39 - The river of tea

Case 38 - Language

