Batman v Super<u>name</u>

Dawn of Legacy Code

Written, directed and presented by Björn Kimminich / @bkimminich

Björn Kimminich

- Senior Manager IT Architecture @ Kuehne + Nagel
- Performing 2-day Clean Code Training Workshops @ KN
- Lecturer for Software Development @ Nordakademie
- Read both books and watched >38 Clean Code episodes
- Speaker @ Clean Code Days, JavaLand & Agile.ee (et al.)

Intent

Can you figure this out?

```
// ...
player1.setmenOCom(1);
player2.setmenOCom(0);
// ...
```

The Batman Mode™ Metaphor

When there is a mystery or crime to be solved, Batman will utilize his brain and all kinds of fancy gadgets to get it done! He will analyze, investigate and deduce until he has the answer. For him as a *costumed Super Hero Detective* it's part of the job! *Software engineers* should **never** have to go into Batman Mode™ to investigate about names used in the code!

Clarifying(?) Declaration

```java
/**
 * setzt menOCom, 0 = Mensch, 1 = Computer
 *
 * @param int fuer menOCom
 */
```

```java
public void setmenOCom(int a) {
  this.menOCom = a;
}
```

Reveal your Intent!

```
import static PlayerType.*;
// ...
player1.setType(HUMAN);
player2.setType(COMPUTER);
// ...
```

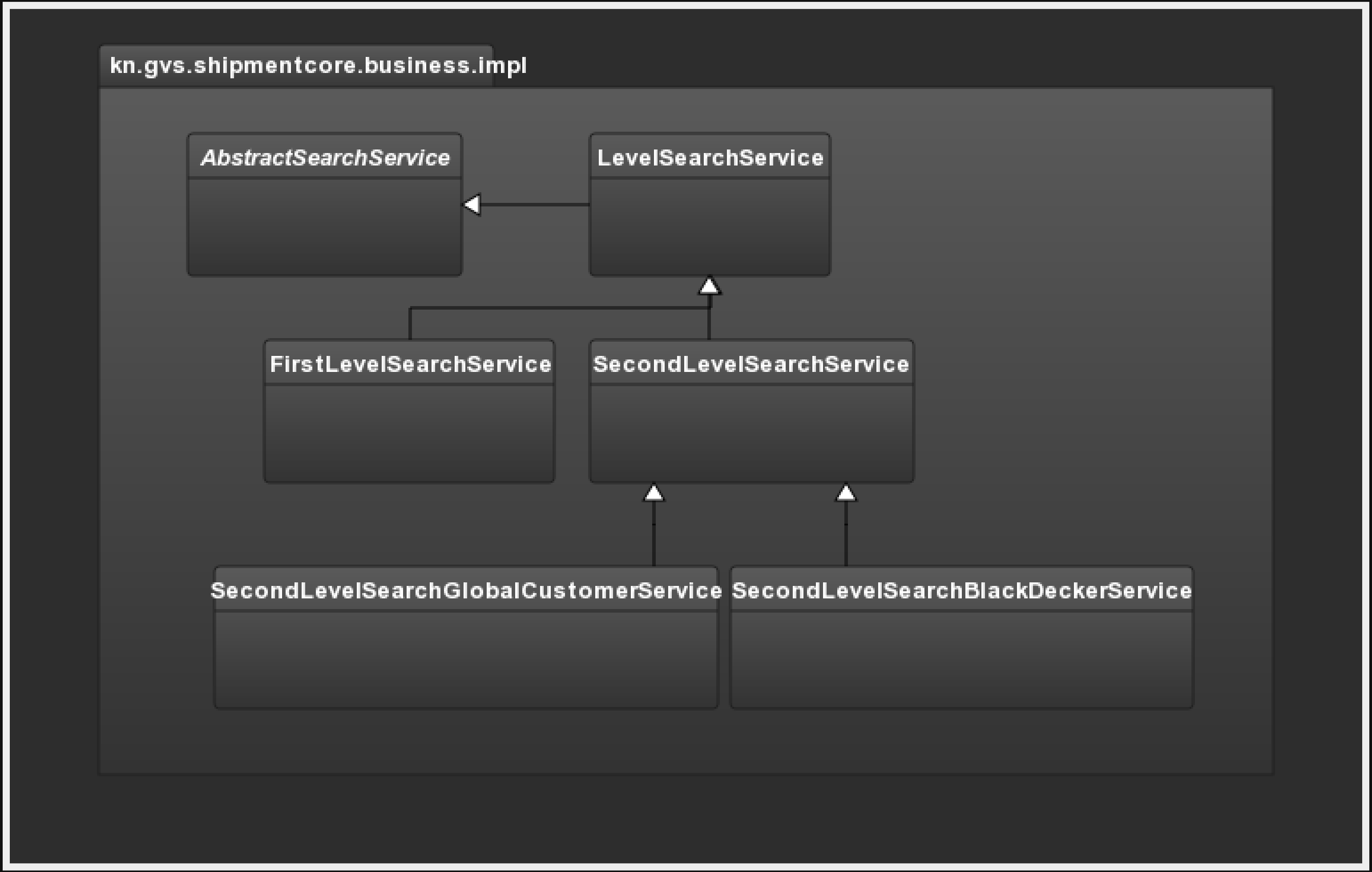
```
public void setType(PlayerType type) {
  this.type = type;
}

public enum PlayerType {
  HUMAN, COMPUTER
}
```

Any ideas what these are?

```java
/** The Indirect to Direct Map */

protected Map<K, Object> itod;

/** The Direct to Indirect Map */

protected Map<Object, K> dtoi;
```

Meaning

What might these services do?


kn.gvs.shipmentcore.business.impl
AbstractSearchService
LevelSearchService
FirstLevelSearchService
SecondLevelSearchService
SecondLevelSearchGlobalCustomerService
SecondLevelSearchBlackDeckerService

Javadoc to the rescue!

```
/**
 * level search service
 *
 * @author [censored]
/*
public class LevelSearchService {...}
```

Sidenote: This is a perfect White Belt Skill Level example for the art of UnCamelCasing!

Maybe in the subclasses?

```
/**
 * This service provides methods for the Level-1 Shipment search.
 *
 * @author [censored]
/*
public class FirstLevelSearchService extends LevelSearchService {...}
```

```
/**
 * This service provides methods for the Level-2 Shipment search.
 *
 * @author [censored]
/*
public class SecondLevelSearchService extends LevelSearchService {...}
```

Sidenote: These could be accidentally denounced as Green Belt Skill Level examples for UnCamelCasing, but at least there is a tiny piece of extra information: `Shipment`! Still a very weak comment as the package name `kn.gvs.shipmentcore` gave that information away earlier already!

Let me explain...

At Kuehne + Nagel the search for ***publicly visible tracking information*** *of shipments is often called* ***First Level*** *Search.*

The search for ***more detailed (and sensitive) information*** *(which requires authentication) we often call* ***Second Level*** *Search.*

These are by no means official logistics terms! They are not even used corporate-wide in Kuehne + Nagel!

Say what you mean!

```
public class PublicTrackingShipmentSearchService extends ShipmentSearchService {...}

public class FullVisibilityShipmentSearchService extends ShipmentSearchService {...}
```

Disinformation

Entrenched v Intended Meaning

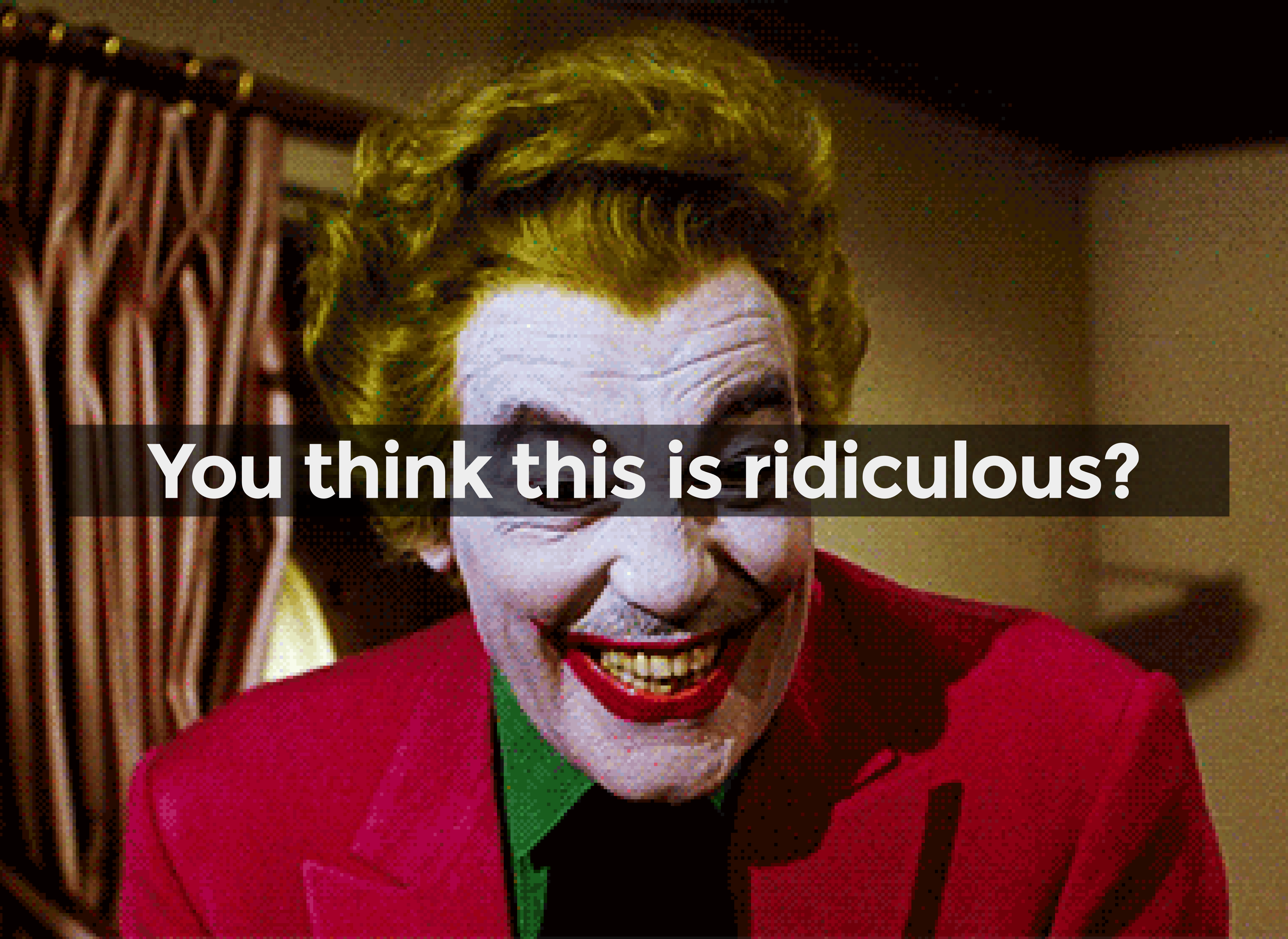
```
private final ?????????????????? ssd;
private final ???????????? sd;
private final ????????????? cd;
```

Entrenched v Intended Meaning

```
private final IShipmentSearchDao ssd;
private final IShipmentDao sd;
private final IContainerDao cd;
```

Abbreviations <u>always</u> fail at some point

```java
private final IShipmentSearchDao ssd;
private final IShipmentDao sd;
private final IContainerSearchDao csd;
private final IContainerDao cd;
private final IShipmentStatusSearchDao sstsd;
private final IShipmentStatusDao sstd;
private final ISpecialShipmentSearchDao spssd;
private final ISpecialShipmentDao spsd;
private final IStatusSearchDao stsd;
private final IStatusDao std;
```



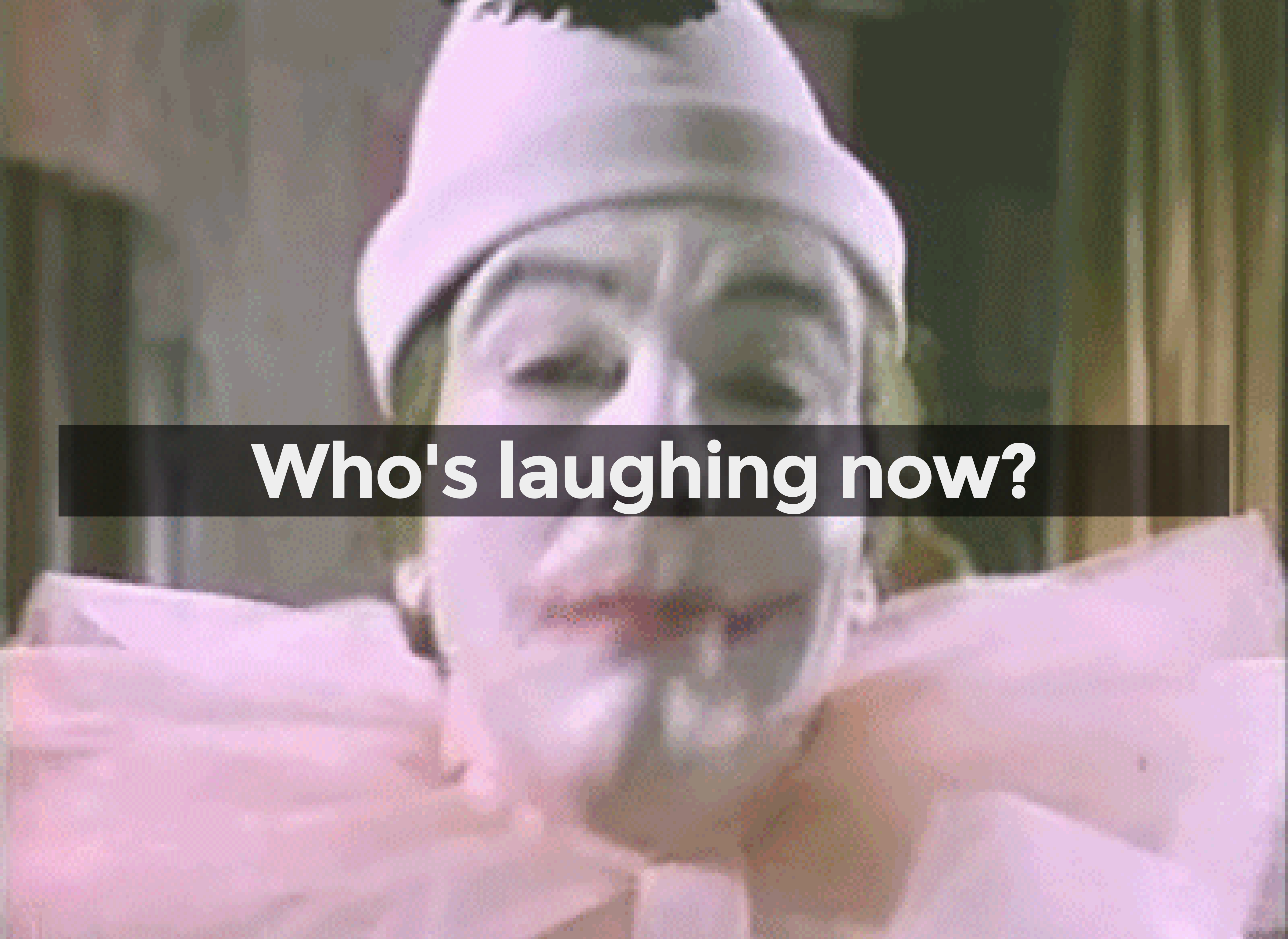
You think this is ridiculous?

Then you shouldn't mind a serious question!

`sstd` **is an instance of ______?**

Please **silently** raise your hand if you remember!

`IShipmentStatusDao`



Who's laughing now?

Language

No Language Mashups

```java
private void maxFourEqualValues(int[] werte) {
  int testValue = werte[0];
  int equalValues = 1;

  for (int i = 1; i < 7; i++) {
    if (testValue == werte[i]) {
      equalValues++;
    } else {
      equalValues = 1;
      testValue = werte[i];
    }
    if (equalValues == 5) {
      throw new IllegalArgumentException(
            "Ein Wert wurde häufiger als 4x übergeben.
            + Betroffener Wert: " + testValue);
    }
  }
}
```

10101

l0IO1 are horrible to distinguish

```
int a = l;
if (O == l)
  a = 0l;
else
  I = 0l;
```

Encodings

Hungarian Notation...

...should be extinct by now. But just to be sure:

```
boolean bBusyFightingCrime;
boolean fBusyFightingCrime; // flag
int cBatGadgets; // count of items
float fpBatMobileMaxSpeed; // floating point
SuperVillain[] rgVillains; // range
```

IUgly & IUnnecessary

Never put an "I" in front of your interface names!

Rationale against the "`I`"

It visually mutilates the type you probably **use most** in your code!

If it can't be avoided, prefer to mutilate the implementation type with a "`Default`"-prefix or "`Impl`"-suffix instead. because you should **only use it once** during instantiation! If you only have one implementation for an interface, you might also ask yourself why you created an interface in the first place.

Context

Redundant Context Noise

```
de.nordakademie.pla.actionblocks.ActionAction
de.nordakademie.pla.actionblocks.ActionBlock
de.nordakademie.pla.actionblocks.ActionCard
de.nordakademie.pla.actionblocks.ActionDamage
de.nordakademie.pla.actionblocks.ActionDiscard
de.nordakademie.pla.actionblocks.ActionEvaluate
de.nordakademie.pla.actionblocks.ActionLevel
de.nordakademie.pla.actionblocks.ActionResource
de.nordakademie.pla.actionblocks.ActionSpecial_Blood
de.nordakademie.pla.actionblocks.ActionSpecial_Flood
de.nordakademie.pla.actionblocks.ActionSpecial_LuckyFind
de.nordakademie.pla.actionblocks.ActionSpecial_Parity
de.nordakademie.pla.actionblocks.ActionSpecial_PureMagic
de.nordakademie.pla.actionblocks.ActionSpecial_Raise
de.nordakademie.pla.actionblocks.ActionSpecial_Santa
de.nordakademie.pla.actionblocks.ActionSpecial_Shift
de.nordakademie.pla.actionblocks.ActionSpecial_Smith
de.nordakademie.pla.actionblocks.ActionSpecial_Spy
de.nordakademie.pla.actionblocks.ActionSpecial_Thief
```

After Renaming/Relocation

```
de.nordakademie.pla.actions.Action
de.nordakademie.pla.actions.Block
de.nordakademie.pla.actions.Card
de.nordakademie.pla.actions.Damage
de.nordakademie.pla.actions.Discard
de.nordakademie.pla.actions.Evaluate
de.nordakademie.pla.actions.Level
de.nordakademie.pla.actions.Resource
de.nordakademie.pla.actions.special.Blood
de.nordakademie.pla.actions.special.Flood
de.nordakademie.pla.actions.special.LuckyFind
de.nordakademie.pla.actions.special.Parity
de.nordakademie.pla.actions.special.PureMagic
de.nordakademie.pla.actions.special.Raise
de.nordakademie.pla.actions.special.Santa
de.nordakademie.pla.actions.special.Shift
de.nordakademie.pla.actions.special.Smith
de.nordakademie.pla.actions.special.Spy
de.nordakademie.pla.actions.special.Thief
```

Note: The sub-package `pla` is not very clear either, but if you have no idea what it means, just leave it as is. Never let an unanswered question like this stop you from doing small-step improvements in the code you understood!

Consistency

Many Words for one Concept

```
interface BatComputer {
  BatReport<Chemical> analyseChemical(Chemical chemical);
  BatReport<Explosive> analyzeExplosive(Explosive explosive);
  BatReport<Tissue> dissectTissue(Tissue tissue);
  BatReport<Fingerprint> parseFingerprint(Fingerprint fingerprint);
}
```

Pick one Word per Concept

```
interface BatComputer {
  BatReport<Chemical> analyzeChemical(Chemical chemical);
  BatReport<Explosive> analyzeExplosive(Explosive explosive);
  BatReport<Tissue> analyzeTissue(Tissue tissue);
  BatReport<Fingerprint> analyzeFingerprint(Fingerprint fingerprint);
}
```

One Word for two Purposes

```
interface BatComputer {
  BatReport<Chemical> analyzeChemical(Chemical chemical);
  BatReport<Explosive> analyzeExplosive(Explosive explosive);
  BatReport<Tissue> analyzeTissue(Tissue tissue);
  BatReport<Fingerprint> analyzeFingerprint(Fingerprint fingerprint);

  boolean analyzeBatPhoneOnHook(BatPhone phone);
  boolean analyzeBatCapeIroned(BatCape cape);
  double analyzeBatMobileGasoline(BatMobile car);
  int analyzeBatCopterKerosene(BatCopter helicopter);
}
```

Two Words for two Purposes

```
interface BatComputer {
  BatReport<Chemical> analyzeChemical(Chemical chemical);
  BatReport<Explosive> analyzeExplosive(Explosive explosive);
  BatReport<Tissue> analyzeTissue(Tissue tissue);
  BatReport<Fingerprint> analyzeFingerprint(Fingerprint fingerprint);

  boolean monitorBatPhoneOnHook(BatPhone phone);
  boolean monitorBatCapeIroned(BatCape cape);
  double monitorBatMobileGasoline(BatMobile car);
  int monitorBatCopterKerosene(BatCopter helicopter);
}
```

Even better: Split by Responsibility

```
interface BatComputer {
  BatReport<Chemical> analyzeChemical(Chemical chemical);
  BatReport<Explosive> analyzeExplosive(Explosive explosive);
  BatReport<Tissue> analyzeTissue(Tissue tissue);
  BatReport<Fingerprint> analyzeFingerprint(Fingerprint fingerprint);
}

interface BatMonitor {
  boolean monitorBatPhoneOnHook(BatPhone phone);
  boolean monitorBatCapeIroned(BatCape cape);
  double monitorBatMobileGasoline(BatMobile car);
  int monitorBatCopterKerosene(BatCopter helicopter);
}
```

Scope

Scope Rule for Methods

- Long Scope = Short and evocative Names
- Short Scope = Long and precise Names

You should not have to read the body of a method to know what it does - its name should tell you.

Too verbose for Long Method Scope

```
/**
 * This service handles the storage and retrieval of the data used by the netsurvey dia
 */
public interface NetsurveyService extends Service {
  Long saveNewNetsurveyRun(NetsurveyRunDts netsurveyRun);
  void updateContentEncoding4NetsurveyRun(String contentEncoding, Long id);
  // ...
}
```

Assumption: This interface is part of a public service API.

Evocative Names for Long Method Scope

```
/**
 * This service handles the storage and retrieval of the data used by the netsurvey dia
 */
public interface NetsurveyService extends Service {
  Long create(NetsurveyRunDts netsurveyRun);
  void updateEncoding(String contentEncoding, Long netSurveyRunId);
  // ...
}
```

Sidenote: Sometimes you can move some verbosity from a method name into a parameter name.

Tests should have Short Scopes...

```
@Test
void testDamage() {
  bot = aBot().withIntegrity(100).withArmor(10).build();

  bot.takeDamage(20);
  assertEquals(90, bot.getIntegrity());

  bot.takeDamage(60);
  assertEquals(40, bot.getIntegrity());

  bot.takeDamage(5);
  assertEquals(40, bot.getIntegrity(), "Armor will not reduce damage below zero");

  bot.takeDamage(9999);
  assertEquals(0, bot.getIntegrity(), "Integrity cannot drop below zero");
}
```

...and therefore Long (and Precise) Names

```java
@Test
void damageTakenIsReducedByArmorAndIntegrityCannotDropBelowZero() {
  bot = aBot().withIntegrity(100).withArmor(10).build();

  bot.takeDamage(20);
  assertEquals(90, bot.getIntegrity());

  bot.takeDamage(60);
  assertEquals(40, bot.getIntegrity());

  bot.takeDamage(5);
  assertEquals(40, bot.getIntegrity(), "Armor will not reduce damage below zero");

  bot.takeDamage(9999);
  assertEquals(0, bot.getIntegrity(), "Integrity cannot drop below zero");
}
```

"Seriously damageTakenIsReducedByArmorAndIntegrityCannotDropBelowZero()...?!?"

Name is too long = Scope is too long

```java
@Test
void damageTakenIsReducedByArmor() {
  bot = aBot().withIntegrity(100).withArmor(10).build();

  bot.takeDamage(20);
  assertEquals(90, bot.getIntegrity());

  bot.takeDamage(60);
  assertEquals(40, bot.getIntegrity());

  bot.takeDamage(5);
  assertEquals(40, bot.getIntegrity(), "Armor will not reduce damage below zero");

@Test
void integrityCannotDropBelowZero() {
  bot = aBot().withIntegrity(100).build();

  bot.takeDamage(9999);
  assertEquals(0, bot.getIntegrity());
}
```

"Extract till you drop" - You should try to further split up a method to get *more* **but** *smaller* scopes which are easier to find names for! For Tests this typically correlates well with the *Single Assert* rule!

Scope Rule for Variables

- Long/Global Scope = Long and self-explaining Names
- Short/Local Scope = Short and reasonable Names

Which Variables need renaming?

```java
public class JAsteroids extends AbstractGame {
  static int FRAMERATE = 60;
  static int HEIGHT = 600;
  static int WIDTH = 800;
  static Dimension SIZE = new Dimension(WIDTH, HEIGHT);
  static boolean SHOW_FPS = true;

  private static final double ACC = 0.05;
  private static final double MAX_ACC = 5.0;
  private static final int BULLET_SPEED = 4;
  private static final double ROTATION = 5.0;
  private static final long DEFAULT_INV_TIME = 3000;

  private static final int NORMAL = 0;
  private static final int INV = 1;
  private static final int MEGA_SHIELD = 2;

  // ... ~1700 lines of game code
}
```

Long Variable Scope = Long Names

```java
public class JAsteroids extends AbstractGame {
  static int FRAMERATE = 60;
  static int SCREEN_HEIGHT = 600;
  static int SCREEN_WIDTH = 800;
  static Dimension SCREEN_SIZE = new Dimension(SCREEN_WIDTH, SCREEN_HEIGHT);
  static boolean SHOW_FRAMERATE = true;

  private static final double ACCELERATION = 0.05;
  private static final double MAX_ACCELERATION = 5.0;
  private static final int BULLET_SPEED = 4;
  private static final double SHIP_ROTATION = 5.0;
  private static final long DEFAULT_INVULNERABLE_TIME = 3000;

  private static final int STATE_NORMAL = 0;
  private static final int STATE_INVULNERABLE = 1;
  private static final int STATE_MEGA_SHIELD = 2;

  // ... ~1700 lines of game code
}
```

Sidenote: Over 1700 LOC is a *very large* scope. `JAsteroids` might need more refactorings than just renaming...

Short Variable Scope

- Short names are allowed in a Short Scope...
- ...but they must still be reasonably easy to understand...
- ...so don't go overboard with crazy abbreviations!

1-or-2-Letter Names...

...can be tolerated as Counter Variables or Exception Instances:

```
for (int i = 0; i < villains.length; i++) {
  try {
    gordon.arrest(villains[i]);
  } catch (TooSmartToBeCaughtByPoliceException ex) {
    gordon.callForHelp(ex.getMessage(), batman, robin);
  }
}
```

...but **never** elsewhere!

```
for (int i = 0; i < v.length; i++) {
  try {
    g.arrest(v[i]);
  } catch (TooSmartToBeCaughtByPoliceException ex) {
    g.callForHelp(ex.getMessage(), b, r);
  }
}
```

We're climbing straight to the climax of interactivity...

Pronunciation

Reading Exercise

Please read the following class names out loud!

SwotService

MINIATURE GAME
COMMISSIONER GORDON
& SWAT TEAM
SWAT
TM & © DC Comics (s13)
THE DARK KNIGHT
TRILOGY

KnlobiLocation

SegmentG041Data

Dx2FltrShipmentCustPartyXDto

BaseDxoProcessMilestone7600LstBo

GyqfaChBppResDao

Reading those class names out loud *among like-minded people* should already feel embarassing.

Now consider talking about these classes to your peers *in the cantina* with *attractive colleagues from HR or Sales* sitting across the table.

And *this* is how they'd look at you, the *"crazy computer weirdos"*!

If this makes you feel bad, then I accomplished my mission!

Thanks for your attention...

- ...and for **thinking about good names** for everything in your code!
- ...and for **renaming badly named things** when you deciphered their meaning!
- ...and for **talking about unweird topics** while having lunch when normal people are around!

Most code examples shown in this presentation are from real-life project code. The `BatComputer` example definitely isn't. All `@author` tags or other personal data have been anonymized in order to protect the authors from prosecution by Clean Code zealots or other Quality Assurance authorities. No software engineers were harmed during or after the creation of this presentation! Please consider the environment before printing this slide deck! It contains many animated GIFs which do not work so well on paper anyway...