Clean Code

What is Clean Code?

- Can be read and enhanced by a developer other than its original author
- Has unit and acceptance tests
- Does one thing well
- Looks like it was written by someone who cares
- Never obscures the designer's intent
- Each routine you read turns out to be pretty much what you expected
- Reads like well written prose
- Provides a clear and minimal API
- Elegant & efficient

Martin: The Robert C. Martin Clean Code Collection, Pos. 815-922

Names

What might these functions do? (🦇)

```
// ...
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

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

Comment (🇩🇪) to the rescue

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

Reveal your intent (💡)

- Any Variable/Method/Class name should tell
  - Why it exists
  - What it does
  - How it is used
- If you need a comment to explain the name, the name is probably ill chosen

Disinformation

What might `ssd`, `sd` and `cd` mean?

```
private final XXXXXXXXXXXXXXXXX ssd;
private final XXXXXXXXXXXX sd;
private final XXXXXXXXXXXXX cd;
```

Entrenched vs. Intended Meaning

```
private final XXXXXXXXXXXXXXXXX ssd;
private final XXXXXXXXXXXX sd;
private final XXXXXXXXXXXX cd;
```

Image from https://edv-webdesign-seo.de/datenrettung.html

Entrenched vs. *Intended* Meaning

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

Abbreviations easily get out of control

```
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;
```

Avoid indistinguishable characters

```
int a = l;
if (O == l)
  a = 0l;
else
  I = 01;
```

even if they might be more distinguishable in certain fonts and highlighting!

```
int a = l;
if (O == l)
  a = 0l;
else
  I = 01;
```

No Disinformation (💡)

- Do not leave false clues
- Do not obscure the meaning of code
- Consider entrenched vs. intended meaning
- Avoid inconsistent spelling

No language mashups

```
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);
    }
  }
}
```

No unpronounceable names (🔇)

```
class BaseDxoProcessMilestone7600LstBo {}
class Dx2FltrShipmentCustPartyXDto {}
class GyqfaChBppResDao {}
class SegmentG041Data {}
class KnlobiLocation {}
class SwotService {}
```

Avoid prefix `I` for interfaces

- Preceding `I` is a distraction at best...
- ...and too much information at worst

🎯 **Leave interfaces unadorned!**

Martin: The Robert C. Martin Clean Code Collection, Pos. 1185

Redundant context noise

```
ty.universi.cardgame.ActionAction
ty.universi.cardgame.ActionBlock
ty.universi.cardgame.ActionCard
ty.universi.cardgame.ActionDamage
ty.universi.cardgame.ActionDiscard
ty.universi.cardgame.ActionEvaluate
ty.universi.cardgame.ActionLevel
ty.universi.cardgame.ActionResource
ty.universi.cardgame.ActionSpecial_Blood
ty.universi.cardgame.ActionSpecial_LuckyFind
ty.universi.cardgame.ActionSpecial_Parity
ty.universi.cardgame.ActionSpecial_PureMagic
ty.universi.cardgame.ActionSpecial_Raise
ty.universi.cardgame.ActionSpecial_Shift
ty.universi.cardgame.ActionSpecial_Smith
ty.universi.cardgame.ActionSpecial_Spy
ty.universi.cardgame.ActionSpecial_Thief
```

Proper packaging over repetitive naming

```
ty.universi.cardgame.Card
ty.universi.cardgame.Level
ty.universi.cardgame.Resource
ty.universi.cardgame.actions.AbstractAction
ty.universi.cardgame.actions.Block
ty.universi.cardgame.actions.Damage
ty.universi.cardgame.actions.Discard
ty.universi.cardgame.actions.Evaluate
ty.universi.cardgame.actions.special.Blood
ty.universi.cardgame.actions.special.LuckyFind
ty.universi.cardgame.actions.special.Parity
ty.universi.cardgame.actions.special.PureMagic
ty.universi.cardgame.actions.special.Raise
ty.universi.cardgame.actions.special.Shift
ty.universi.cardgame.actions.special.Smith
ty.universi.cardgame.actions.special.Spy
ty.universi.cardgame.actions.special.Thief
```

Encoding & Context (💡)

- Stick to problem or solution domain names
- Follow commonly accepted naming conventions...
- ...and change any home-grown bad ones

🎯 *Don't force the reader to translate your names into ones they use and understand.*

Martin: The Robert C. Martin Clean Code Collection, Pos. 1177

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);
}
```

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);
}
```

Split interfaces by responsibility (💯)

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

interface BatStatusMonitor {
  boolean isBatPhoneOnHook(BatPhone phone);
  boolean isBatCapeIroned(BatCape cape);
  double getRemainingBatMobileGasoline(BatMobile car);
  int getRemainingBatCopterKerosene(BatCopter helicopter);
}
```

Consistent Lexicon (💡)

- Pick one word per **concept**...
- ...while avoiding to use the same word for two **purposes**

Martin: The Robert C. Martin Clean Code Collection, Pos. 1211-1240

Key takeaways (🎯)

- Think about **good names** for *everything* in your code
- **Rename** all *badly named things* once you've deciphered their meaning
- *Always* avoid sending readers of your code into **Batman Mode** (🦇)

Exercise 2.1 (🏠)

🔧 TODO

Functions

Comments

Formatting