Clean Code

Hönnun og smíði hugbúnaðar

Haust 2022

The Cost of Owning a Mess

- Engin breyting verður augljós
- Hver breyting brýtur kerfið
- Ekki hægt að bæta við fleiri forriturum

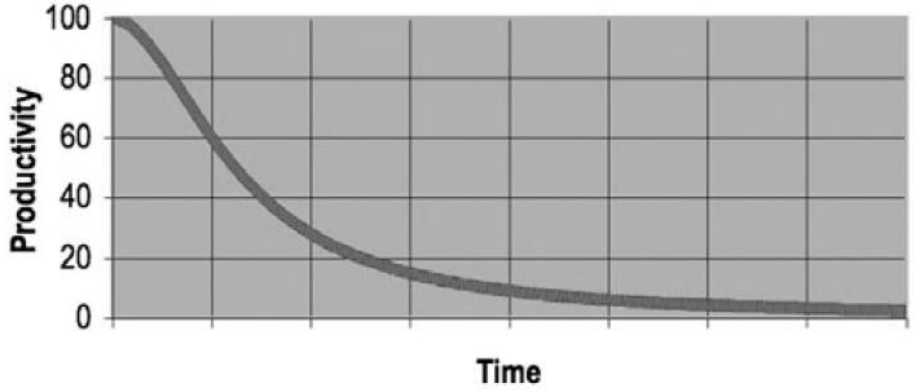


Figure 1-1
Productivity vs. time

Defend your Code

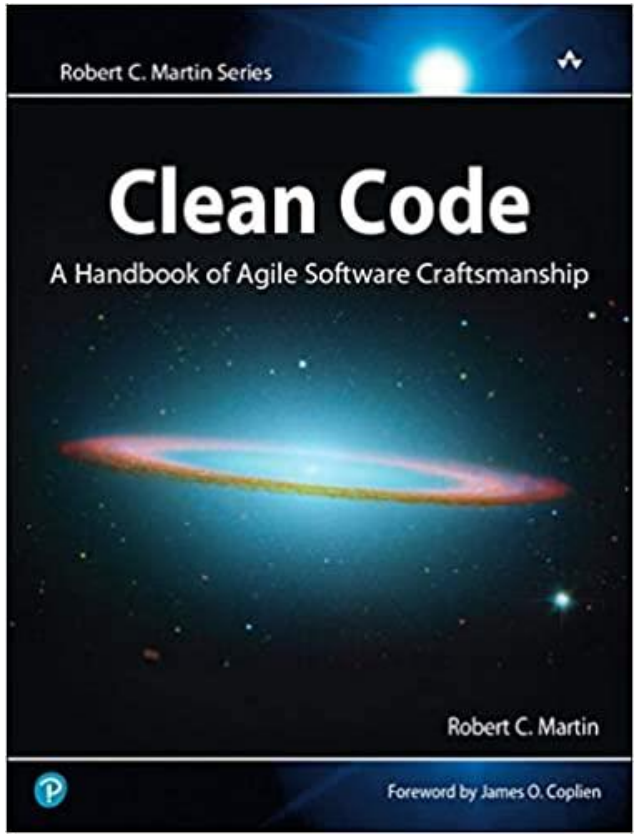
- Okkar að verja kóðann
- Okkar að ýta til baka á móti verkefnastjórum
  - Flestir vilja heyra sannleikann
- Okkar að refactor-a
- Okkar að gera ekki hakk fix til að mæta deadlines
  - Heldur okkur frá því að mæta deadlines í framtíðinni
- Okkar að tryggja að kóðinn helst *clean*
- *“One broken window starts the process towards decay”*

Hvað er *Clean Code*?

- Clean Code er læsilegur
- Clean Code er auðskiljanlegur
- Clean Code er einfaldur
- Clean Code er sveigjanlegur
- Eins og að mála mynd
  - Flest vita hvenær málverk er gott eða slæmt
  - Færri vita hvernig á að mála góða mynd
- *“Clean Code always looks like it was written by someone who cares”*

Bókin

- Full af tillögum ekki reglum
- Sumir hlutir umdeilanlegir
- Þurfið ekki að vera sammála öllu
- Flest er þó 👌 þótt ég segi sjálfur frá



The Boy Scout Rule

- “Leave the campground cleaner than you found it”
- Skilið kóðann eftir hreinni og betri en þegar þið komuð að honum
- Refactor þegar þið sjáið eitthvað sem þarfnast refactoringu
- Þarf ekki mikið
  - Betra breytunafn
  - Minni endurtekt
  - Betri test
  - Etc.

Meaningful Names

Use Intention-Revealing Names

- Nöfn falla, breyta, klasa, file-a… ætti að svara stóru spurningunum
- Tilgangur
- Hvað gerir hluturinn?
- Forðist *Mental Mapping*
- Löng góð nöfn > Stutt óskýr nöfn
- Ættu ekki að þurfa comment, ættu að vera comment-in!

```
public List<int[]> getThem() {
  List<int[]> list1 = new ArrayList<int[]>();
  for (int[] x : theList)
    if (x[0] == 4)
      list1.add(x);
  return list1;
}
```

vs

```
public List<int[]> getFlaggedCells() {
  List<int[]> flaggedCells = new ArrayList<int[]>();
  for (int[] cell : gameBoard)
    if (cell[STATUS_VALUE] == FLAGGED)
      flaggedCells.add(cell);
  return flaggedCells;
}
```

Avoid Disinformation

- Forðist nöfn með afvegaleiðandi upplýsingum
- `accountList` vs `accounts`
  - Listi þýðir eitthvað sérstakt
  - Best að innihalda ekki týpu breyturnar í nafninu

Make Meaningful Distinctions

- `ProductInfo vs ProductData`
- `NameString vs Name`
- `MoneyAmount vs Money`
- `theMessage vs message`

Allt dæmi um slæmar aðgreiningar breyta

```
public static void copyChars(char a1[], char a2[]) {
  for (int i = 0; i < a1.length; i++) {
    a2[i] = a1[i];
  }
}
```

Betra: `source`

Betra: `dest`

Use Pronounceable Names

- Við erum góð í orðum
- Erum þróuð til að hugsa í töluðum orðum

(generation date, year, month, day, hour, minute, and second)

```
class DtaRcrd102 {
  private Date genymdhms;
  private Date modymdhms;
  private final String pszqint = "102";
  /* ... */
};
```

vs

```
class Customer {
  private Date generationTimestamp;
  private Date modificationTimestamp;;
  private final String recordId = "102";
  /* ... */
};
```

Use Searchable Names

- Nöfn ættu að vera augjlós
- Nöfn ættu að vera aðgreinanleg

```
for (int j=0; j<34; j++) {
  s += (t[j]*4)/5;
}
```

vs

```
int realDaysPerIdealDay = 4;
const int WORK_DAYS_PER_WEEK = 5;
int sum = 0;
for (int j=0; j < NUMBER_OF_TASKS; j++) {
  int realTaskDays = taskEstimate[j] * realDaysPerIdealDay;
  int realTaskWeeks = (realdays / WORK_DAYS_PER_WEEK);
  sum += realTaskWeeks;
}
```

Avoid Encodings

```
PhoneNumber phoneString;
// name not changed when type changed!
```

- Interfaces og Útfærslur
  - `IShapeFactory` vs `ShapeFactory`
  - `ShapeFactory` vs `ShapeFactoryImpl`

Bókin vill meina að ef þú þarft að encode-a eitthvað þá ætti það að vera útfærslan **en þetta er umdeilt**

Class Names

- Ættu að vera nafnorð
- T.d. `Customer`, `WikiPage`, `Account`, `AddressParser`.

Method Names

- Ættu að vera sagnorð / sagnorðs *frasi*
  - T.d. `postPayment`, `deletePage`, eða `save`
- Breytur í föllum ættu að vera nafnorð
  - T.d. `write(name)`
- *Accessor* ætti að hafa `get` forskeyti
- *Mutator* ætti að hafa `set` forskeyti
- *Predicate* ætti að hafa `is`,`are`,`should` forskeyti

```
string name = employee.getName();
customer.setName("mike");
if (paycheck.isPosted())...
```

Umdeilt

- Overloaded constructor ætti að vera lýsandi

```
Complex fulcrumPoint = Complex.FromRealNumber(23.0);
```
VS
```
Complex fulcrumPoint = new Complex(23.0);
```

Don’t Be Cute

- `HolyHandGrenade` vs `DeleteItems`
- “Choose clarity over entertainment value”

Pick One Word Per Concept

- Consistency is Key
- `Fetch` vs `retrieve` vs `get`
- `Controller` vs `Manager` vs `Driver`

Use Solution and Domain Names

- Solution Domain Names
  - Þeir sem lesa kóðann eru forritarar
  - Notið computer science hugtök þegar við á
    - JobQueue, AccountVisitor, …
- Problem Domain Names
  - Nota problem domain nöfn þegar við á
  - Hægt að leita til domain sérfræðinga með spurningar

Add Meaningful Context

- Nöfn merkja lítið ein og sér
- Þurfa samhengi
  - T.d. Klasi, fall, scope etc.
- `firstName`, `lastName`, `street`, `houseNumber`, `city`, `state`, og `zipcode`.
  - Merkja eitthvað saman (Address)
  - `State` eitt og sér hefur litla merkingu
  - Hægt að vefja saman í Address klasa

Add Meaningful Context frh.

Listing 2-1
Variables with unclear context.

```
private void printGuessStatistics(char candidate, int count) {
    String number;
    String verb;
    String pluralModifier;
    if (count == 0) {
      number = "no";
      verb = "are";
      pluralModifier = "s";
    } else if (count == 1) {
      number = "1";
      verb = "is";
      pluralModifier = "";
    } else {
      number = Integer.toString(count);
      verb = "are";
      pluralModifier = "s";
    }
    String guessMessage = String.format(
      "There %s %s %s%s", verb, number, candidate, pluralModifier
    );
    print(guessMessage);
  }
```

Listing 2-2
Variables have a context.

```
public class GuessStatisticsMessage {
  private String number;
  private String verb;
  private String pluralModifier;

  public String make(char candidate, int count) {
    createPluralDependentMessageParts(count);
    return String.format(
      "There %s %s %s%s",
       verb, number, candidate, pluralModifier );
  }

  private void createPluralDependentMessageParts(int count) {
    if (count == 0) {
      thereAreNoLetters();
    } else if (count == 1) {
      thereIsOneLetter();
    } else {
      thereAreManyLetters(count);
    }
  }

  private void thereAreManyLetters(int count) {
    number = Integer.toString(count);
    verb = "are";
    pluralModifier = "s";
  }

  private void thereIsOneLetter() {
    number = "1";
    verb = "is";
    pluralModifier = "";
  }

  private void thereAreNoLetters() {
    number = "no";
    verb = "are";
    pluralModifier = "s";
  }
}
```

Functions

Small

- Föll ættu að vera lítil
- 20 línur max viðmið
- Kóða línur ættu að vera stuttar
  - Ef löng lína þá breyta í fall -> eykur læsileika út af nafnafalli

Listing 3-1
HtmlUtil.java (FitNesse 20070619)

```
public static String testableHtml(
  PageData pageData,
  boolean includeSuiteSetup
) throws Exception {
  WikiPage wikiPage = pageData.getWikiPage();
  StringBuffer buffer = new StringBuffer();
  if (pageData.hasAttribute("Test")) {
    if (includeSuiteSetup) {
      WikiPage suiteSetup =
        PageCrawlerImpl.getInheritedPage(
                SuiteResponder.SUITE_SETUP_NAME, wikiPage
        );
      if (suiteSetup != null) {
        WikiPagePath pagePath =
          suiteSetup.getPageCrawler().getFullPath(suiteSetup);
        String pagePathName = PathParser.render(pagePath);
        buffer.append("!include -setup .")
              .append(pagePathName)
              .append("\n");
      }
    }
    WikiPage setup =
      PageCrawlerImpl.getInheritedPage("SetUp", wikiPage);
    if (setup != null) {
      WikiPagePath setupPath =
        wikiPage.getPageCrawler().getFullPath(setup);
      String setupPathName = PathParser.render(setupPath);
      buffer.append("!include -setup .")
            .append(setupPathName)
            .append("\n");
    }
  }
  buffer.append(pageData.getContent());
  if (pageData.hasAttribute("Test")) {
    WikiPage teardown =
      PageCrawlerImpl.getInheritedPage("TearDown", wikiPage);
    if (teardown != null) {
      WikiPagePath tearDownPath =
        wikiPage.getPageCrawler().getFullPath(teardown);
      String tearDownPathName = PathParser.render(tearDownPath);
      buffer.append("\n")
            .append("!include -teardown .")
            .append(tearDownPathName)
            .append("\n");
    }
    if (includeSuiteSetup) {
      WikiPage suiteTeardown =
        PageCrawlerImpl.getInheritedPage(
                SuiteResponder.SUITE_TEARDOWN_NAME,
                wikiPage
        );
      if (suiteTeardown != null) {
        WikiPagePath pagePath =
          suiteTeardown.getPageCrawler().getFullPath (suiteTeardown);
        String pagePathName = PathParser.render(pagePath);
        buffer.append("!include -teardown .")
              .append(pagePathName)
              .append("\n");
      }
    }
  }
  pageData.setContent(buffer.toString());
  return pageData.getHtml();
}
```

VS

Listing 3-2
HtmlUtil.java (refactored)

```
public static String renderPageWithSetupsAndTeardowns(
  PageData pageData, boolean isSuite
) throws Exception {
  boolean isTestPage = pageData.hasAttribute("Test");
  if (isTestPage) {
    WikiPage testPage = pageData.getWikiPage();
    StringBuffer newPageContent = new StringBuffer();
    includeSetupPages(testPage, newPageContent, isSuite);
    newPageContent.append(pageData.getContent());
    includeTeardownPages(testPage, newPageContent, isSuite);
    pageData.setContent(newPageContent.toString());
  }

  return pageData.getHtml();
}
```

VS

Listing 3-3
HtmlUtil.java (re-refactored)

```
public static String renderPageWithSetupsAndTeardowns(
  PageData pageData, boolean isSuite) throws Exception {
  if (isTestPage(pageData))
    includeSetupAndTeardownPages(pageData, isSuite);
  return pageData.getHtml();
}
```

Do One Thing

- Fall ætti að gera **einn hlut** og **gera það vel**
- Erfitt að vita hvað *einn hlutur* er
  - Afstætt
  - Fer eftir samhengi
  - Fer eftir *abstraction*

One Level of Abstraction per Function

- Öll skref í falli ættu að vera á sama *abstract level*
- Föll innihalda fallaköl í næsta (lægra) abstraction level

```
public List<WeatherDataDto> GetWeatherReport()
{
    var weatherDtoList = new List<WeatherDataDto>();

    var weatherReport = JsonConvert.DeserializeObject<WeatherDataModel[]>(File.ReadAllText("weather.json"));

    if (weatherReport != null && weatherReport.Length > 0)
    {
        foreach (var report in weatherReport)
        {
            var weatherDto = new WeatherDataDto()
            {
                City = report.city,
                Temparature = report.temp,
                ReportedBy = report.reportedBy
            };
            weatherDtoList.Add(weatherDto);
        }
    }

    return weatherDtoList;
}
```

Abstr 2

Abstr 1

Abstr 3

vs

```
1 reference
public List<WeatherDataDto> GetWeatherReport()
{
    var weatherDtoList = new List<WeatherDataDto>();

    var weatherReport :WeatherDataModel[] = ReadWeatherReport();

    if (IsValidWeatherReport(weatherReport))
    {
        MapToReportDto(weatherDtoList, weatherReport);
    }


    return weatherDtoList;
}

1 reference
static WeatherDataModel[] ReadWeatherReport()
{
    return JsonConvert.DeserializeObject<WeatherDataModel[]>(File.ReadAllText( path: "weather.json"))!;
}

1 reference
static bool IsValidWeatherReport(WeatherDataModel[] weatherReport)
{
    return weatherReport != null && weatherReport.Length > 0;
}

1 reference
static void MapToReportDto(List<WeatherDataDto> weatherDtoList, WeatherDataModel[] weatherReport)
{
    foreach (var report :WeatherDataModel in weatherReport)
    {
        var weatherDto = new WeatherDataDto()
        {
            City = report.city,
            Temparature = report.temp,
            ReportedBy = report.reportedBy
        };
        weatherDtoList.Add(weatherDto);
    }
}
```

Function Arguments

- Því færri því betri
- Því fleiri breytur því ólæislegra er fallakallið
  - `writeField(name) vs writeField(output-Stream, name)`
- Þrjár eða fleiri breytur ætti að forðast
- Frekar hópa inntaks breytur í sér klasa
  - `makeCircle(double x, double y, double radius)` vs
    `makeCircle(Point center, double radius);`
- Undantekning
  - Argument Lists
  - `public String format(String format, Object... args)`
  - `format("%s worked %.2f hours.", name, hours);`
- Output breytur ætti að forðast
  - Input sem breytur output sem return

Auðvelt að lesa fyrst breyturnar eru *meðhöndlaðar* eins. Format er þannig séð *dyadic* fall

Have No Side Effects

- Föll ættu ekki að hafa hliðarverkun
- Viljum ekki óvæntar breytingar/virkni
- Föll ættu að gera einn hlut og ekkert *auka*

Listing 3-6
UserValidator.java

```
public class UserValidator {
  private Cryptographer cryptographer;

  public boolean checkPassword(String userName, String password) {
    User user = UserGateway.findByName(userName);
    if (user != User.NULL) {
      String codedPhrase = user.getPhraseEncodedByPassword();
      String phrase = cryptographer.decrypt(codedPhrase, password);
      if ("Valid Password".equals(phrase)) {
        Session.initialize();
        return true;
      }
    }
    return false;
  }
}
```

Fallið ***lofar*** að eingöngu check-a password, ekki að frumstilla *session*

Command Query Separation

- Föll ættu annað hvort að *gera eitthvað* eða *skila einhverju* ekki bæði
  - Gera eitthvað -> Command
  - Skila einhverju -> Query
- Fall verður óskýrt ef bæði er gert
  - `public boolean set(String attribute, String value);`
  - `if (set("username", "unclebob"))...`
    - Er verið að tjékka hvort username er *set* sem “unclebob” ?
    - Er verið að tjékka hvort *set*-ið tókst?

```
if (attributeExists("username")) {
  setAttribute("username", "unclebob");
  ...
}
```

Betra, aðskilja *command*-ið frá *query*-inu

Prefer Exceptions to Returning Error Codes

- Að skila Error Codes er brot á Command Query Separation
  - `if (deletePage(page) == E_OK)`

```
if (deletePage(page) == E_OK) {
  if (registry.deleteReference(page.name) == E_OK) {
    if (configKeys.deleteKey(page.name.makeKey()) == E_OK){
      logger.log("page deleted");
    } else {
      logger.log("configKey not deleted");
    }
  } else {
    logger.log("deleteReference from registry failed");
  }
} else {
  logger.log("delete failed");
  return E_ERROR;
}
```

vs

```
try {
  deletePage(page);
  registry.deleteReference(page.name);
  configKeys.deleteKey(page.name.makeKey());
}
catch (Exception e) {
  logger.log(e.getMessage());
}
```

Error Handling Is **One Thing**

- Villu meðhöndlun er *einn hlutur*
- Fall sem sér um villur ætti eingöngu að gera það

```
public void delete(Page page) {
  try {
    deletePageAndAllReferences(page);
  }
  catch (Exception e) {
    logError(e);
  }
}

private void deletePageAndAllReferences(Page page) throws Exception {
  deletePage(page);
  registry.deleteReference(page.name);
  configKeys.deleteKey(page.name.makeKey());
}

private void logError(Exception e) {
  logger.log(e.getMessage());
}
```

Þetta fall sér eingöngu um villur

Prefer Exceptions to Returning Error Codes Frh.

- Umdeild regla
- Kostir
  - Læsileiki
  - Decoupling
  - Hægt að grípa villur nokkrum stigum fyrir ofan
- Gallar
  - Hætta á ómeðhöndluðum villum
  - Óljóst flæði forrits
- Alternatives?
  - Hægt að fara ákveðinn milliveg
  - https://youtu.be/a1ye9eGTB98

Comments

"Comments are, at best, a necessary evil"

Comments

- Vandamál?
  - Kóðinn er source of truth
    - Kóði breytist, comment fylgja oft ekki
    - Comments ljúga oft
    - Engin comment eru betri en röng comment
  - Comments brjóta flæðið við lestur

- Hvenær?
  - Comment ættu að vera notuð til að bæta upp fyrir okkar misheppnuðu tilraun til tjá okkur auðskiljanlega í kóða
  - Okkur tekst ekki alltaf að skrifa læsilegan kóða.
  - Stundum þarf til að lýsa ástæðu ákvörðunar.
  - Sem viðvörun á afleiðingum keyrslu
  - Stundum þarf fyrir legal reasons (höfundaréttur / hugverkaréttur)
  - Til að magna mikilvægi kóða
  - TODO comment

- Ef ekki comment, hvað þá?
  - Kóði ætti að vera *self-documenting*
  - Reyna að gera kóða læsilegri áður en þú skrifar comment
    - T.d. Betri nöfn / fleiri breytur með nöfnum (mega vera löng nöfn)

Comments dæmi

```
// Check to see if the employee is eligible for full benefits
if ((employee.flags & HOURLY_FLAG) &&
    (employee.age > 65))
```

vs

```
if (employee.isEligibleForFullBenefits())
```

```
// Returns an instance of the Responder being tested.
protected abstract Responder responderInstance();
```

vs

```
protected abstract Responder responderBeingTested();
```

```
// does the module from the global list <mod> depend on the
// subsystem we are part of?
if (smodule.getDependSubsystems().contains(subSysMod.getSubSystem()))
```

vs

```
ArrayList moduleDependees = smodule.getDependSubsystems();
String ourSubSystem = subSysMod.getSubSystem();
if (moduleDependees.contains(ourSubSystem))
```

Comments dæmi frh.

```
    InputStreamResponse response = new InputStreamResponse();
    response.setBody(formatter.getResultStream(), formatter.getByteCount());
//  InputStream resultsStream = formatter.getResultStream();
//  StreamReader reader = new StreamReader(resultsStream);
//  response.setContent(reader.read(formatter.getByteCount()));
```

Aldrei comment-a út kóða, eyddu honum, ef þú þarft hann seinna þá ertu með git

```
    // Don't run unless you
    // have some time to kill.
    public void _testWithReallyBigFile()
    {
      writeLinesToFile(10000000);

      response.setBody(testFile);
      response.readyToSend(this);
      String responseString = output.toString();
      assertSubString("Content-Length: 1000000000", responseString);
      assertTrue(bytesSent > 1000000000);
    }
```

Gilt Comment sem viðvörun

```
String listItemContent = match.group(3).trim();
// the trim is real important.  It removes the starting
// spaces that could cause the item to be recognized
// as another list.
new ListItemWidget(this, listItemContent, this.level + 1);
return buildList(text.substring(match.end()));
```

Gilt Comment sem mögnun á mikilvægi

Formatting

Formatting Rules

- Consistency is key
- Viljum hafa formatting reglur
- T.d. Eslint , Prettier (Javascript)
- Hjálpar okkur að vera samkvæm okkur sjálfum og öðrum
- Reglur dæmi:
  - Bool breytur byrja á is, should, are …
  - Breytur ættu að vera snake case (hello_world)
  - Lengd file-a
  - Lengd lína
  - Bil á milli falla
  - Space á milli -> “=“ , “,” …
  - Indentation
  - Etc.

Vertical Distance

- Skildir hlutir ættu að vera *vertically* nálægt hvort öðru
- Breytir ættu að vera skilgreind nálægt notkun
- Ef eitt fall kallar á annað ættu þau að vera vertically nálægt, caller fyrir ofan calle

Robert C. Martin Formatting Rules

```
public class CodeAnalyzer implements JavaFileAnalysis {
  private int lineCount;
  private int maxLineWidth;
  private int widestLineNumber;
  private LineWidthHistogram lineWidthHistogram;
  private int totalChars;

  public CodeAnalyzer() {
    lineWidthHistogram = new LineWidthHistogram();
  }

  public static List<File> findJavaFiles(File parentDirectory) {
    List<File> files = new ArrayList<File>();
    findJavaFiles(parentDirectory, files);
    return files;
  }

  private static void findJavaFiles(File parentDirectory, List<File> files) {
    for (File file : parentDirectory.listFiles()) {
      if (file.getName().endsWith(".java"))
        files.add(file);
      else if (file.isDirectory())
        findJavaFiles(file, files);
    }
  }
```

Klasabreytur saman með ekkert bil á milli

Ekkert bil á milli *færibreytur* falla

Bil á milli færibreyta og skilgreiningu

Alltaf *function body* aldrei one-liner

Bil á milli breyta

Alltaf indentation

High-level fyrir ofan low-level og skildleiki saman

Robert C. Martin Formatting Rules frh.

```
public void analyzeFile(File javaFile) throws Exception {
  BufferedReader br = new BufferedReader(new FileReader(javaFile));
  String line;
  while ((line = br.readLine()) != null)
    measureLine(line);
}

private void measureLine(String line) {
  lineCount++;
  int lineSize = line.length();
  totalChars += lineSize;
  lineWidthHistogram.addLine(lineSize, lineCount);
  recordWidestLine(lineSize);
}

private void recordWidestLine(int lineSize) {
  if (lineSize > maxLineWidth) {
    maxLineWidth = lineSize;
    widestLineNumber = lineCount;
  }
}

public int getLineCount() {
  return lineCount;
}

public int getMaxLineWidth() {
  return maxLineWidth;
}
```

Línur max 120 chars

Bil á milli operators

Breytur skilgreindar nálægt notkun

Objects and Data Structures

Objects and Data Structures

- Objects
  - Fela gögn bakvið abstractions
  - Expose-a föllum til að sækja / vinna á gögnum
- Data Structures
  - Expose-a gögnum
  - Hafa engin ***meaningful*** föll
  - Entites, DTOs, Models
- Hybrids
  - Expose-a gögnum og hafa meaningful föll
  - Tilgangur verður óskýr -> ætti að forðast

Objects and Data Structures Dæmi

```
public class Square {
  public Point topLeft;
  public double side;
}

public class Rectangle {
  public Point topLeft;
  public double height;
  public double width;
}

public class Circle {
  public Point center;
  public double radius;
}

public class Geometry {
  public final double PI = 3.141592653589793;

  public double area(Object shape) throws NoSuchShapeException
  {
    if (shape instanceof Square) {
      Square s = (Square)shape;
      return s.side * s.side;
    }
    else if (shape instanceof Rectangle) {
      Rectangle r = (Rectangle)shape;
      return r.height * r.width;
    }
    else if (shape instanceof Circle) {
      Circle c = (Circle)shape;
      return PI * c.radius * c.radius;
    }
    throw new NoSuchShapeException();
  }
}
```

Data Structures

Object

Null

Don’t Return Null

- NullPointerException er algengasta ógripna villan
- Null check gerir kóðann ólæsinn

```
List<Employee> employees = getEmployees();
if (employees != null) {
  for(Employee e : employees) {
    totalPay += e.getPay();
  }
}
```

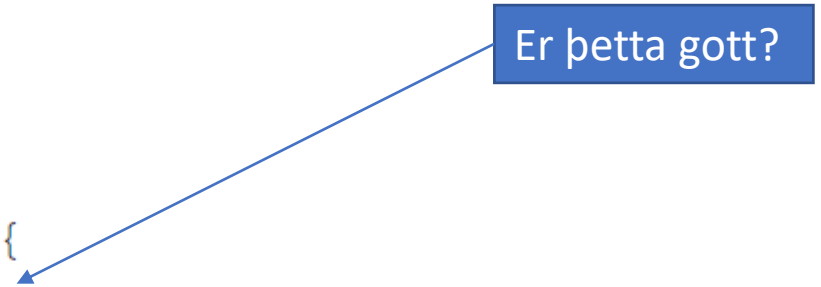
vs

```
List<Employee> employees = getEmployees();
for(Employee e : employees) {
  totalPay += e.getPay();
}
```

Tómur listi í stað null

Don’t Pass Null

- Forðist að senda null inn í föll
- Ólæsilegri kóði
- Óvíst að fallið meðhöndli null




```
public double xProjection(Point p1, Point p2) {
  if (p1 == null || p2 == null) {
    throw InvalidArgumentException(
      "Invalid argument for MetricsCalculator.xProjection");
  }
  return (p2.x - p1.x) * 1.5;
}
```