Clean Code 2

A Handbook of Agile Software Craftsmanship

Robert C. Martin

Note by waegaein@github.com

CH2 Meaningful Names

Variable Names

Function Names

Class Names

CH2 Meaningful Names

Names should

- tell you **why** it exists, **what** it does, and **how** it is used.

def sendtostore_packagestock0
→ def calculate_stock_with_notifying_store

CH2 Meaningful Names

Names should

- tell you why it exists, what it does, and how it is used.
- allow readers to understand difference **without comment** nor deduction.

def sendtostore_packagestock0
→ def calculate_stock_with_notifying_store

CH2 Meaningful Names

Names should

- tell you why it exists, what it does, and how it is used.
- allow readers to understand difference without comment nor deduction.
- include **semantic** words.

```
def sendtostore_packagestock0
def sendtostore_packagestock1
```

CH2 Meaningful Names

Names should

- tell you why it exists, what it does, and how it is used.
- allow readers to understand difference without comment nor deduction.
- include semantic words.
- include words that are technical or related to the **problem domain**.

def calculate_rate
→ def calculate_surcharge

CH2 Meaningful Names

Names should

- tell you why it exists, what it does, and how it is used.
- allow readers to understand difference without comment nor deduction.
- include semantic words.
- include words that are technical or related to the problem domain.
- have **context** from its scope.

Package_stock.package_stock_id
→ Package_stock.id

CH2 Meaningful Names

Names should

- tell you why it exists, what it does, and how it is used.
- allow readers to understand difference without comment nor deduction.
- include semantic words.
- include words that are technical or related to the problem domain.
- have context from its scope.
- consists of **words**.

daydelay
→ days_delayed

CH2 Meaningful Names

Names should **not**

- include **obscure clues** that may lead to false conclusions.

def create_invoice_with_assurance
→ def create_invoice_with_retry

CH2 Meaningful Names

Names should **not**

- include obscure clues that may lead to false conclusions.
- use **inconsistent synonyms** for a single concept.

('calculate_stock_with_notifying_store' exists)
def send_mail_to_branch
→ def send_mail_to_store

CH2 Meaningful Names

Names should **not**

- include obscure clues that may lead to false conclusions.
- use inconsistent synonyms for a single concept.
- include **syntactic** encodings.

v_count
→ count

CH2 Meaningful Names

Names should **not**

- include obscure clues that may lead to false conclusions.
- use inconsistent synonyms for a single concept.
- include syntactic encodings.
- consists of **over-abbreviated** letters.

def get_psscharge
→ def get_package_stock_surchage

CH2 Meaningful Names

- Prefer **clarity** to smartness and entertainment.

```
_, _ = select.select([], [])
→ discarded, discarded = select.select(list(), list())
```

CH2 Meaningful Names

- Prefer clarity to smartness and entertainment.
- Class names are nouns or **noun phrases**.

class SlackNotify
→ class SlackNotification

CH2 Meaningful Names

- Prefer clarity to smartness and entertainment.
- Class names are nouns or noun phrases.
- Method names are verbs or **verb phrases**.

def surcharge_calculation
→ def calculate_surcharge

CH3 Functions

Functions should

- be as **small** as possible.

CH3 Functions

Functions should

- be as small as possible.
- have the **indent level of one or two**.

CH3 Functions

Functions should

- be as small as possible.
- have the indent level of one or two.
- do only **one thing**.

CH3 Functions

Functions should

- be as small as possible.
- have the indent level of one or two.
- do only one thing.
- include steps within **one level of abstraction**.

CH3 Functions

Functions should

- be as small as possible.
- have the indent level of one or two.
- do only one thing.
- include steps within one level of abstraction.
- have **two arguments at maximum**, taking advantage of argument objects/lists.

def send_mail_with_file(email_from, email_to, subject, html_content, attach_file_path, attach_file_name, mime_type)
→ def send_mail_with_file(base_email, attachment)

CH3 Functions

Functions should **not**

- **expose switch statements**, which do multiple things inherently, to the high level.

CH3 Functions

Functions should **not**

- expose switch statements, which do multiple things inherently, to the high level.
- pass Boolean **flag argument** as it forces more than one behavior.

calculate_surcharge(is_promotion=False, …)
→ calculate_surcharge, calcuate_surcharge_for_promotion

CH3 Functions

Functions should **not**

- expose switch statements, which do multiple things inherently, to the high level.
- pass Boolean flag argument as it forces more than one behavior.
- **transform input argument** instead of using return value.

```
...
charge_list.sort()
...
return
```

→

```
...
charge_list_sorted = sorted(charge_list)
...
return charge_list_sorted
```

CH3 Functions

Functions should **not**

- expose switch statements, which do multiple things inherently, to the high level.
- pass Boolean flag argument as it forces more than one behavior.
- transform input argument instead of using return value.
- have side effects that are **not implied by names**.

CH3 Functions

Functions should **not**

- expose switch statements, which do multiple things inherently, to the high level.
- pass Boolean flag argument as it forces more than one behavior.
- transform input argument instead of using return value.
- have side effects that are not implied by names.
- perform command and query **at the same time**.
- perform action and error handling **at the same time**.

CH3 Functions

Functions should **not**

- expose switch statements, which do multiple things inherently, to the high level.
- pass Boolean flag argument as it forces more than one behavior.
- transform input argument instead of using return value.
- have side effects that are not implied by names.
- perform command and query at the same time.
- perform action and error handling at the same time.
- have **multiple entries or exits**.

CH3 Functions

```
import sys


f = sys.stdin
pairs = tuple(tuple(line.strip().split()) for line in f.readlines()[1:])

for word1, word2 in pairs:
    occurrences1 = dict()
    for character in word1:
        if character in occurrences.keys():
            occurrences1[character] += 1
        else:
            occurrences1[character] = 1
    occurrences2 = dict()
    for character in word2:
        if character in occurrences.keys():
            occurrences2[character] += 1
        else:
            occurrences2[character] = 1

    if occurrences1 == occurrences2:
        print("{0} & {1} are anagrams.".format(word1, word2))
    else:
        print("{0} & {1} are NOT anagrams.".format(word1, word2))
```

```
1  import sys
2
3  def parse_input():
4      f = sys.stdin
5      return tuple(tuple(line.strip().split()) for line in f.readlines()[1:])
6
7  def get_occurrences(word):
8      occurrences = dict()
9      for character in word:
10         if character in occurrences.keys():
11             occurrences[character] += 1
12         else:
13             occurrences[character] = 1
14
15     return occurrences
16
17 def are_anagrams(word1, word2):
18     return get_occurrences(word1) == get_occurrences(word2)
19
20 def run():
21     pairs = parse_input()
22     for word1, word2 in pairs:
23         if are_anagrams(word1, word2):
24             print("{0} & {1} are anagrams.".format(word1, word2))
25         else:
26             print("{0} & {1} are NOT anagrams.".format(word1, word2))
27
28     return
29
30 run()
31
```