Clean Code Fundamentals

Form

Pre-work

- Video: https://cleancoders.com/episode/clean-code-episode-5
- Exam: https://cleancoders.com/episode/clean-code-episode-5/exam

Chapters

Timetable

Activity	Time
Warmup	5 min
Exercise 1	20 min
Exercise 2	20 min
Exercise 3	20 min
Wrap up	5 min

Warmup

- What are your formatting preferences? (e.g., tabs vs spaces, 2 spaces vs 4 spaces, `snake_case` vs `camelCase`, etc.)
  - Type in the meeting chat

Exercise 1

- Prompt
  - Compare classes and data structures
- Time limit: 10 minutes

Possible answer

- Classes
  - Cohesive data and functions that operate on that data
  - Private data, public functions
  - Tell, don't ask
  - Polymorphism
  - Protects from new types, exposes to new methods
- Data Structures
  - Cohesive data, simple functions like getters and setters
  - Public data
  - Ask, don't tell
  - Switch statements
  - Protects from new methods, exposes to new types

Exercise 2

- Code-review practice (part 1)
- Use the worksheet
- Go through scenarios 1-3
- Suggest code improvements, and explain why
  - Use the worksheet to record your suggestions
  - Use active voice (e.g., “introduce parameter object” instead of “parameter object should be introduced”)
- Time limit: 15 minutes
- Scenarios adopted from Eder Diaz blog

Scenario 1

```
1  def can_buy_beer(age, money):
2      if age >= 21 and money >= 20:
3          return True
4
5      return False
```

Scenario 1 solution

- Introduce constants to clarify the *magic numbers*
- Collapse if-statement when evaluating boolean expressions

```
1  LEGAL_DRINKING_AGE = 21
2  BEER_PRICE = 20
3
4
5  def can_buy_beer(age, money):
6      return age >= LEGAL_DRINKING_AGE and money >= BEER_PRICE
```

Scenario 2

```python
def should_show_image(item_index, article, show_all_images):
    return (
        bool(article.image_url)
        if item_index in (0, 1, 2)
        else bool(article.image_url)
        if show_all_images
        else False
    )
```

Ternary operators note:

- C++: `condition ? a : b`
- Python: `a if condition else b`

Scenario 2 solution

- Use explanatory variables to clarify sub-expressions
- Remove code duplication

```
1  def should_show_image(item_index, article, show_all_images):
2      is_first_three_items = item_index in (0, 1, 2)
3      has_image = article.image_url is not None
4
5      return (is_first_three_items or show_all_images) and has_image
```

Scenario 3

```python
import math


def get_area(shape, width, height, radius):
    if shape == "circle":
        return math.pi * radius * radius
    elif shape == "square":
        return width * width
    elif shape == "rectangle":
        return width * height
```

Scenario 3 solution

- Introduce polymorphism to protect from new shapes
- Replace shape names with classes to make it less error-prone

```
1  import abc
2  import math
3
4
5  class Shape(abc.ABC):
6      @abc.abstractmethod
7      def get_area(self):
8          raise NotImplementedError
9
10
11 class Circle(Shape):
12     def __init__(self, radius):
13         self.radius = radius
14
15     def get_area(self):
16         return math.pi * self.radius * self.radius
```

Exercise 3

- Code-review practice (part 2)
- Use the worksheet
- Go through scenarios 4-6
- Suggest code improvements, and explain why
  - Use the worksheet to record your suggestions
  - Use active voice (e.g. “introduce parameter object” instead of “parameter object should be introduced”)
- Time limit: 15 minutes

Scenario 4

```
chemical_symbols = {
    "Sodium": "Na",
    "Hydrogen": "H",
    "Helium": "He",
    "Oxygen": "O",
}


def get_symbol(name):
    symbol = chemical_symbols.get(name)
    if symbol:
        return symbol

    print("symbol not found")
    return "not found"
```

Scenario 4 solution

- Introduce error handling
- Use PEP8 naming conventions

```
class SymbolNotFoundError(Exception):
    pass


CHEMICAL_SYMBOLS = {
    "Sodium": "Na",
    "Hydrogen": "H",
    "Helium": "He",
    "Oxygen": "O",
}


def get_symbol(name):
    try:
        return CHEMICAL_SYMBOLS[name]
    except KeyError:
        raise SymbolNotFoundError("symbol not found")
```

Scenario 5

```
def get_display_image(article, watermark=None):
    image = None

    if article["image"] and article["displayImage"]:
        if watermark:
            image = apply_watermark(article["image"], watermark)
        else:
            image = article["image"]

    return image
```

Scenario 5 solution 1

- Use early return to separate main logic from corner cases
- Use separate function for `watermark=None` argument

```python
def has_image_and_display_image(article):
    return article["image"] and article["displayImage"]


def get_display_image_with_watermark(article, watermark):
    if not has_image_and_display_image(article):
        return None
    return apply_watermark(article["image"], watermark)


def get_display_image(article):
    if not has_image_and_display_image(article):
        return None
    return article["image"]
```

Scenario 5 solution 2

- Introduce a class

```
class Article:
    def __init__(self, image, display_image):
        self.image = image
        self.display_image = display_image

    def has_image_to_display(self):
        return self.image and self.display_image

    def get_display_image(self):
        if not self.has_image_to_display():
            return None
        return self.image

    def get_display_image_with_watermark(self, watermark):
        if not self.has_image_to_display():
            return None
        return apply_watermark(self.image, watermark)
```

Scenario 6

```
# The code checks if auth is enabled, if it's enforced,
# and if user already registered (has email field).
if (
    type(AUTH_ENABLED) == str
    and AUTH_ENABLED == "true"  # enabled!
    and not skip_authentication  # enforced!!
    and user.email  # already registered!!!
):
    print(f"Welcome back, {user.name}")
```

Scenario 6 solution

- Replace comments with explanatory variables

```
is_auth_enabled = AUTH_ENABLED == "true"
is_auth_enforced = is_auth_enabled and not skip_authentication
is_registered_user = user.email is not None

if is_auth_enforced and is_registered_user:
    print(f"Welcome back, {user.name}")
```

Summary

- Comments
- Formatting
- Classes vs data structures
- Abstraction boundaries

What is next?

- Expect an e-mail with instructions for upcoming coding dojo

Final words

Always leave the code better than you found it.
– The Software Craftsmanship Rule