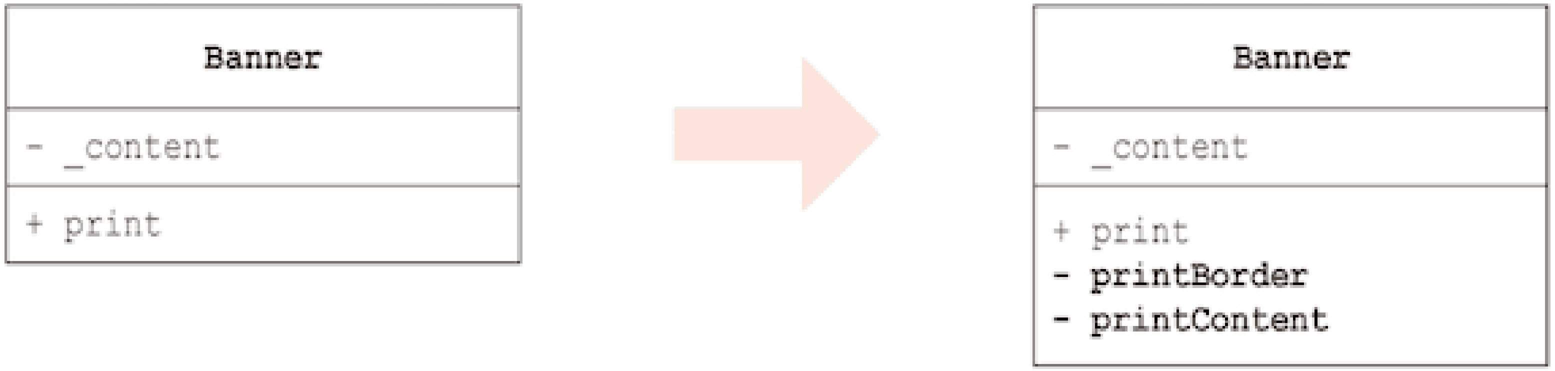
Extract Method

Extract Method refactoring involves taking a code fragment and turning it into a separate method

Code Smell

We have lengthy print code.

```
def print():
  for (i in content):
    ... # lengthy print code

=>

def print():
  printBorder()

def printBorder():
  for (i in content):
    ... # lengthy print code
```


Banner
- _content
+ print
Banner
- _content
+ print
- printBorder
- printContent

When method becomes longer

- We add code after:
  - fixing bugs
  - adding features
- When we have to scroll to read the method, it's time to use the extract method refactoring.

Example: Banner

Before:

```python
class Banner:
    def __init__(self, content):
        self.content = content

    def print_banner(self, times):
        # Print top border
        print("+", end="")
        for i in range(len(self.content)): print("-", end="")
        print("+")

        # Print content
        for i in range(times): print(f"|{self.content}|")

        # Print bottom border (duplicate code)
        print("+", end="")
        for i in range(len(self.content)): print("-", end="")
        print("+")
```

After: apply Extract method

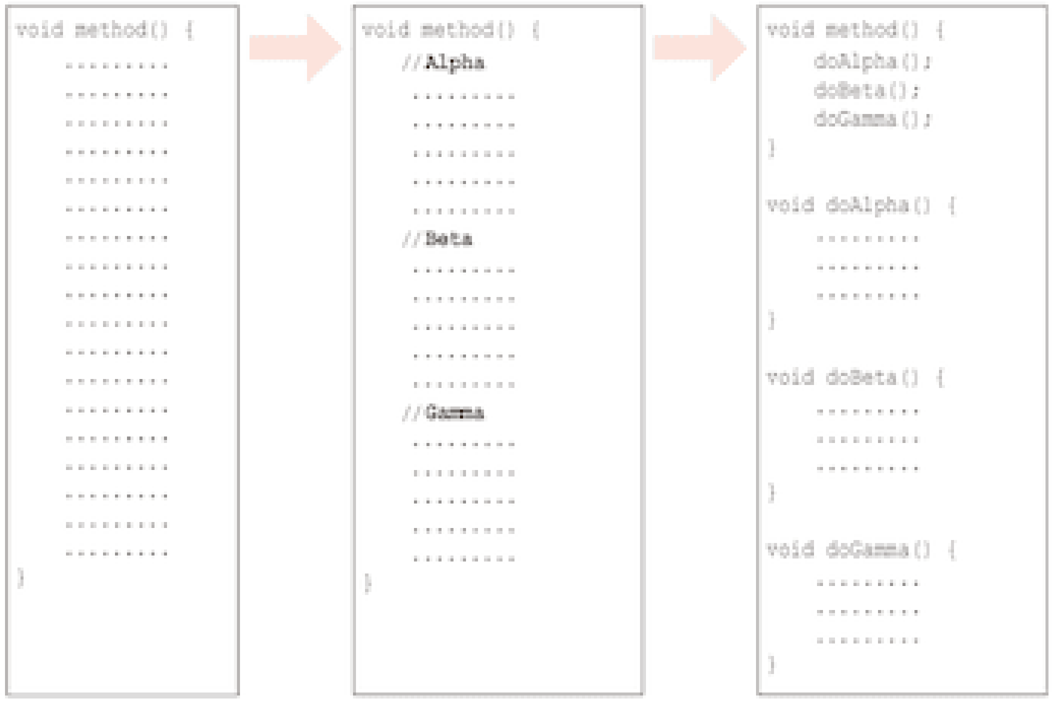
```python
class Banner:
    """Banner class with extracted methods (after refactoring)"""

    def __init__(self, content):
        self.content = content

    def print_banner(self, times):
        self._print_border()
        self._print_content(times)
        self._print_border()

    def _print_border(self):
        """Extracted method for printing border"""
        print("+", end="")
        for i in range(len(self.content)):
            print("-", end="")
        print("+")

    def _print_content(self, times):
        """Extracted method for printing content"""
        for i in range(times):
            print(f"|{self.content}|")
```

Tips

- Be careful with the name.
    - Be specific to express what it does (not how)
- Make the method private

```
def _print_border(self):
    ...
def _print_content(self, times):
    ...
```

- Making a lot of comments is the code smell for the refactoring



```
void method() {
    .........
    .........
    .........
    .........
    .........
    .........
    .........
    .........
    .........
    .........
    .........
    .........
    .........
    .........
    .........
    .........
    .........
    .........
}
```

```
void method() {
  //Alpha
    .........
    .........
    .........
    .........
    .........
  //Beta
    .........
    .........
    .........
    .........
    .........
  //Gamma
    .........
    .........
    .........
    .........
    .........
}
```

```
void method() {
    doAlpha();
    doBeta();
    doGamma();
}

void doAlpha() {
    .........
    .........
    .........
}

void doBeta() {
    .........
    .........
    .........
}

void doGamma() {
    .........
    .........
    .........
}
```

Inline Method: Reverse of Extracting Method

- When the method is too short, we put the method in the method that calls it.
  - Reduce the number of methods.

Discussion

Benefits of Extract Method

1. **Improves readability** - shorter, more focused methods
2. **Reduces duplication** - extracted code can be reused
3. **Better naming** - meaningful method names document intent
4. **Easier testing** - smaller methods are easier to test
5. **Single Responsibility** - each method has one clear purpose

Code Smell for Extract Method

- **Long methods** - methods that are hard to understand at a glance
- **Comments explaining sections** - comments like "// calculate discount" indicate extractable code
- **Nested loops and conditions** - complex nested structures
- **Duplicate code** - similar code fragments that can be extracted and reused