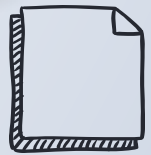
Identificar erros se o projeto está no padrão Template Method

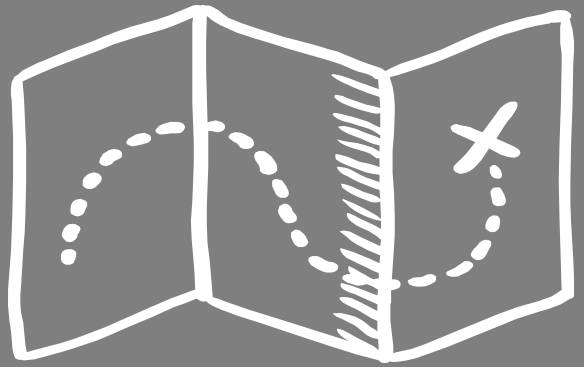
Prof. Ricardo Terra

Nome: Andrew Takeshi
Gabriel Amorim



O que será apresentado ?

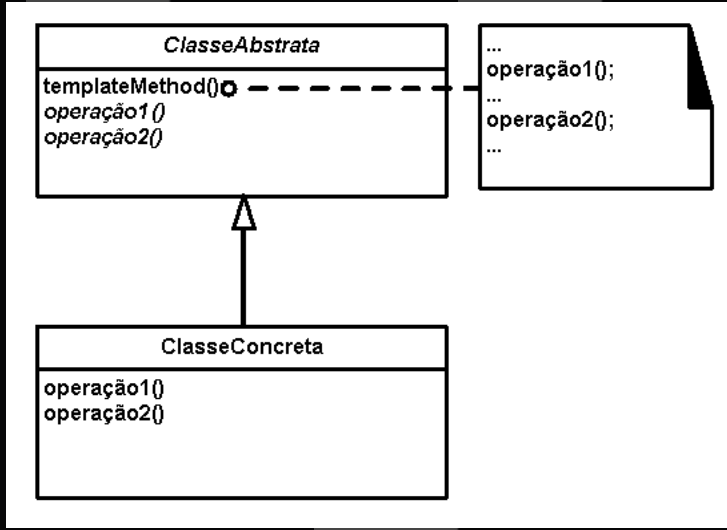
- Contextualização
- Execução / Realização
- Limitações
- Difculdades



Contextualização

```python
import sys
from abc import ABC, abstractmethod


class Algorithm(ABC):

    def template_method(self):
        """Skeleton of operations to perform. DON'T override me.

        The Template Method defines a skeleton of an algorithm in an operati
        and defers some steps to subclasses.
        """
        self.__do_absolutely_this()
        self.do_step_1()
        self.do_step_2()
        self.do_something()

    def __do_absolutely_this(self):
        """Protected operation. DON'T override me."""
        this_method_name = sys._getframe().f_code.co_name
        print('{}.{}'.format(self.__class__.__name__, this_method_name))
```

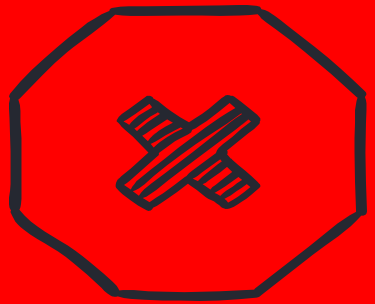
```python
@abstractmethod
def do_step_1(self):
    """Primitive operation. You HAVE TO override me, I'm a placeholder."""
    pass

@abstractmethod
def do_step_2(self):
    """Primitive operation. You HAVE TO override me, I'm a placeholder."""
    pass

def do_something(self):
    """Hook. You CAN override me, I'm NOT a placeholder."""
    print('do something')
```



```
class AlgorithmA(Algorithm):

    def do_step_1(self):
        print('do step 1 for Algorithm A')

    def do_step_2(self):
        print('do step 2 for Algorithm A')


class AlgorithmB(Algorithm):

    def do_step_1(self):
        print('do step 1 for Algorithm B')

    def do_step_2(self):
        print('do step 2 for Algorithm B')

    def do_something(self):
        print('do something else')
```

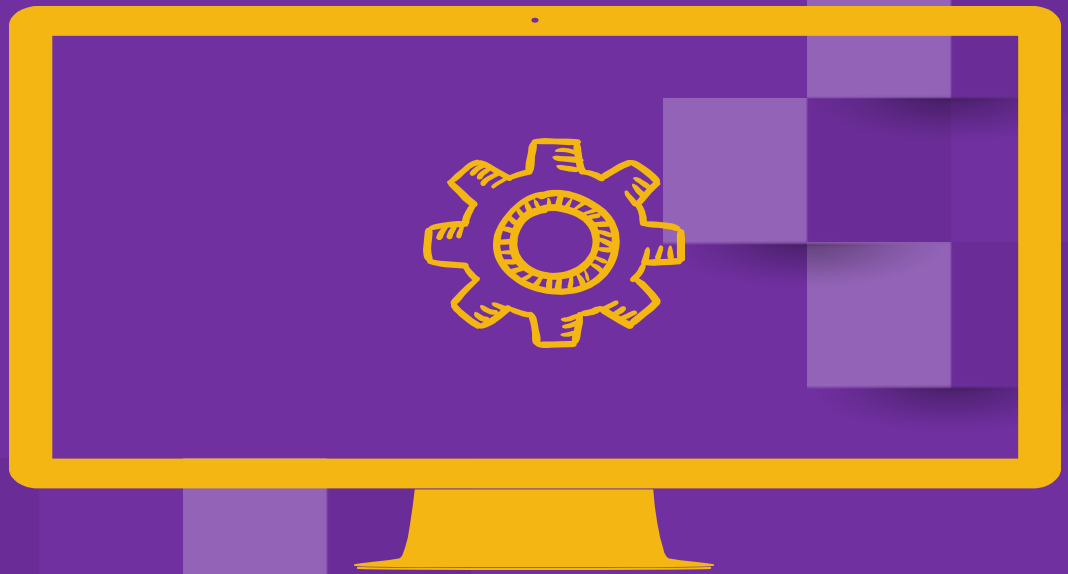
```
"""
Define the skeleton of an algorithm in an operation, deferring some
steps to subclasses. Template Method lets subclasses redefine certain
steps of an algorithm without changing the algorithm's structure.
"""

import abc


class AbstractClass(metaclass=abc.ABCMeta):
    """
    Define abstract primitive operations that concrete subclasses define
    to implement steps of an algorithm.
    Implement a template method defining the skeleton of an algorithm.
    The template method calls primitive operations as well as operations
    defined in AbstractClass or those of other objects.
    """

    def template_method(self):
        self._primitive_operation_1()
        self._primitive_operation_2()

    @abc.abstractmethod
    def _primitive_operation_1(self):
        pass

    @abc.abstractmethod
    def _primitive_operation_2(self):
        pass


class ConcreteClass(AbstractClass):
    """
    Implement the primitive operations to carry out
    subclass-specificsteps of the algorithm.
    """

    def _primitive_operation_1(self):
        pass

    def _primitive_operation_2(self):
        pass


def main():
    concrete_class = ConcreteClass()
    concrete_class.template_method()


if __name__ == "__main__":
    main()
```

- Não existe classe abstrata com o TM;
- Template Methods (TM) duplicados na classe abstrata;
- Existe mais de um TM;
- Casos de não existir TM na classe abstrata;
- Se as classes filhas implementam os métodos abstrados.



- Não tem Hooks (classes filhas), para o Design Pattern.

Execução / Realização



Limitações [$\lim_{x \to +\infty}$]

Dificuldades


OBRIGADO !