

METODA SZABLONOWA.

M. G. K. S. T. P.

Wydział Fizyki, Matematyki i Informatyki
Politechnika Krakowska im. Tadeusza Kościuszki

17.05.2019

Plan.

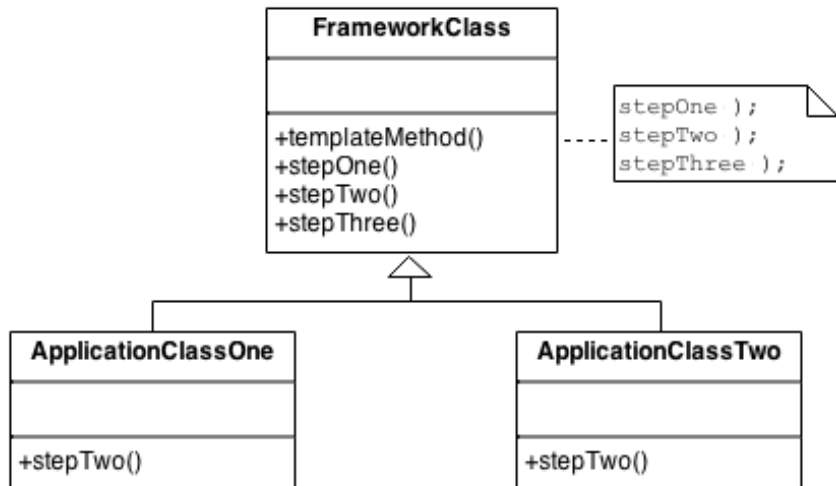
- 1 Wstęp.
- 2 Struktura wzorca.
- 3 Diagram UML.
- 4 Cel.
- 5 Przykład zastosowania.
- 6 Źródła.

Metoda szablonowa to wzorec projektowy, którego zadaniem jest zdefiniowanie metody, będącej szkieletem algorytmu. Algorytm ten może być następnie dokładnie definiowany w klasach pochodnych. Niezmienna część algorytmu zostaje opisana w metodzie szablonowej, której klient nie może nadpisać. W metodzie szablonowej wywoływane są inne metody, reprezentujące zmienne kroki algorytmu. Mogą one być abstrakcyjne lub definiować domyślne zachowania. Klient, który chce skorzystać z algorytmu, może wykorzystać domyślną implementację bądź może utworzyć klasę pochodną i nadpisać metody opisujące zmienne fragmenty algorytmu. Najczęściej metoda szablonowa ma widoczność publiczną, natomiast metody do przesłonięcia mają widoczność chronioną lub prywatną, tak, aby klient nie mógł ich użyć bezpośrednio.

Struktura wzorca.

Wzorzec składa się z co najmniej dwóch klas. Klasą podstawową jest `FrameworkClass` definiuje ona szkielet algorytmu i jest bazą, z której korzysta klient. Składa się z metody szablonowej `templateMethod` która jest publiczna i której klient nie ma możliwości rozszerzyć, oraz z metod prywatnych (lub chronionych) `stepOne` , `stepTwo` , `stepThree` , które są wykorzystywane w metodzie `templateMethod` i zawierają domyślną implementację algorytmu. Klient chcący zmienić któryś z kroków algorytmu, musi zdefiniować dodatkowo klasy pochodne `ApplicationClassOne` , `ApplicationClassTwo` w których przedefiniuje jedną bądź wszystkie prywatne metody, implementujące kolejne kroki algorytmu.

Diagram UML.



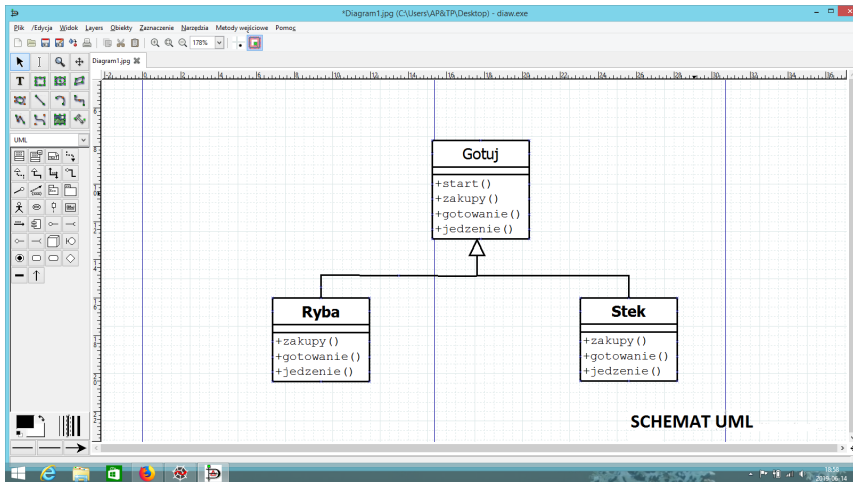
Celem stosowania wzorca **Metoda szablonowa** jest:

- zdefiniowanie szkieletu algorytmu,
- oddzielenie części logiki do klas dziedziczących po klasie **AbstractClass** bez zmieniania podstawowej struktury algorytmu,
- zaimplementowanie niezmienniej części algorytmu.

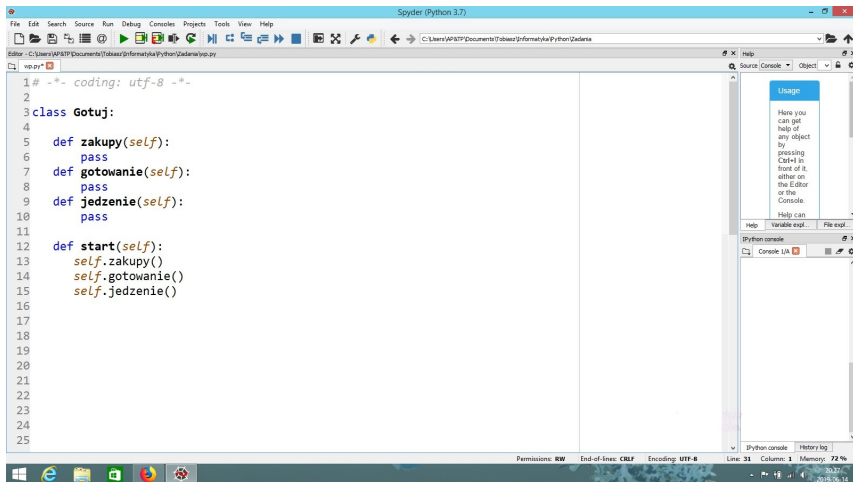
Przykład zastosowania.

Tworzymy klasę podstawową **Gotuj** , która definiuje szkielet algorytmu. Klasa ta składa się z metody **start** oraz z metod **zakupy**, **gotowanie**, **jedzenie** , które są używane w metodzie **start** i zawierają domyślną implementację algorytmu. Tworzymy dodatkowo dwie klasy pochodne **Ryba** i **Stek** .

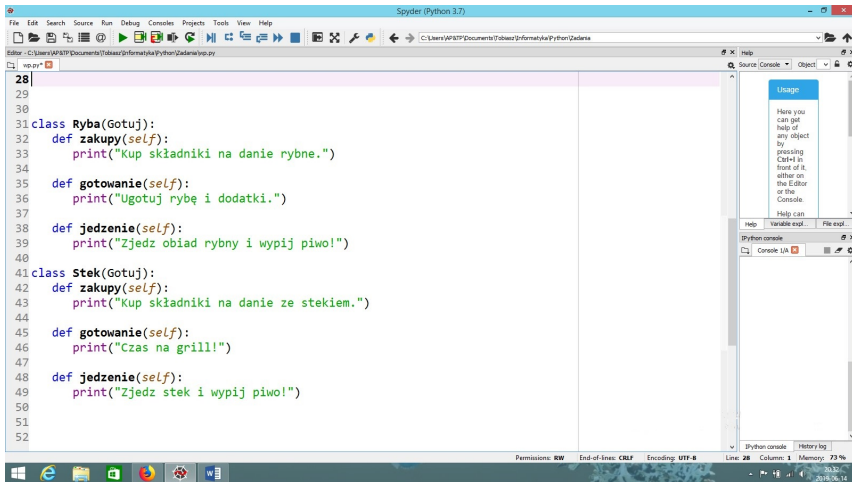
Przykład zastosowania.



Przykład zastosowania.



Przykład zastosowania.

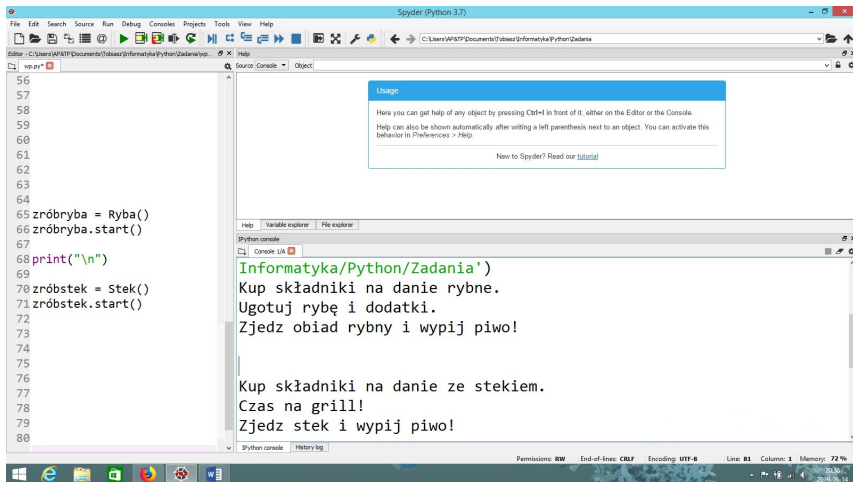


The screenshot shows the Spyder Python IDE interface. The main editor window displays a Python script with two classes, `Ryba` and `Stek`, both inheriting from `Gotuj`. Each class has three methods: `zakupy`, `gotowanie`, and `jedzenie`. The `zakupy` method prints a message about buying ingredients, `gotowanie` prints a message about cooking, and `jedzenie` prints a message about eating. The `Stek` class's `jedzenie` method includes the instruction to 'eat the steak and drink beer'.

```
28|
29|
30|
31| class Ryba(Gotuj):
32|     def zakupy(self):
33|         print("Kup składniki na danie rybne.")
34|
35|     def gotowanie(self):
36|         print("Ugotuj rybę i dodatki.")
37|
38|     def jedzenie(self):
39|         print("Zjedz obiad rybny i wypij piwo!")
40|
41| class Stek(Gotuj):
42|     def zakupy(self):
43|         print("Kup składniki na danie ze stekiem.")
44|
45|     def gotowanie(self):
46|         print("Czas na grill!")
47|
48|     def jedzenie(self):
49|         print("Zjedz stek i wypij piwo!")
50|
51|
52|
```

The right sidebar contains a 'Usage' panel with instructions on how to get help for an object by pressing `Ctrl+I` in the editor or the console. Below it is the 'Python console' panel, which is currently empty. The bottom status bar shows the file permissions, end-of-line characters, encoding, and the current line and column numbers.

Przykład zastosowania.



Źródła.



<https://sourcemaking.com/designpatterns/templatemethod>



<https://www.youtube.com/watch?v=MfAvs0n9uMs>