Разработка в ML

В смысле разработка?!

В смысле разработка?!

1. Проектирование ML систем
2. Использование алгоритмов и подходов к написанию кода

В смысле разработка?!

1. Проектирование ML систем
2. Использование алгоритмов и подходов к написанию кода
3. Знание Python за пределами Jupyter

В смысле разработка?!

1. Проектирование ML систем
2. Использование алгоритмов и подходов к написанию кода
3. Знание Python за пределами jupyter
4. Написание production кода
5. Представление как происходит deploy сервисов

Зачем это нужно

Зачем это нужно

1. Учёт особенностей внедрения в DS

Зачем это нужно

1. Учёт особенностей внедрения в DS
2. Скорость внедрения фичей

Зачем это нужно

1. Учёт особенностей внедрения в DS
2. Скорость внедрения фичей
3. Контроль над ситуацией

Зачем это нужно

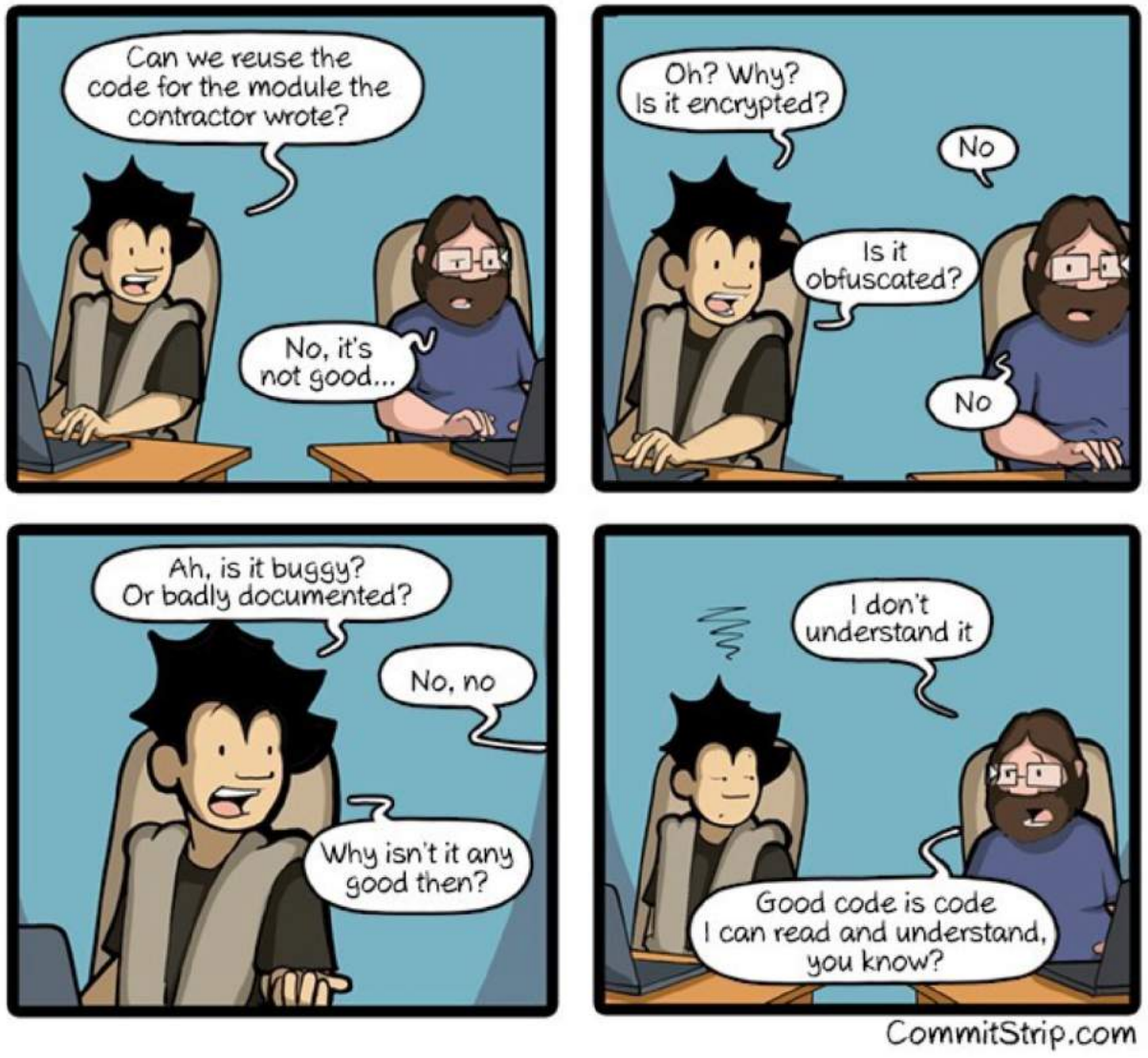
1. Учёт особенностей внедрения в DS
2. Скорость внедрения фичей
3. Контроль над ситуацией
4. Решение трудных задач

Зачем это нужно

1. Учёт особенностей внедрения в DS
2. Скорость внедрения фичей
3. Контроль над ситуацией
4. Решение трудных задач
5. Поддержка крупных проектов

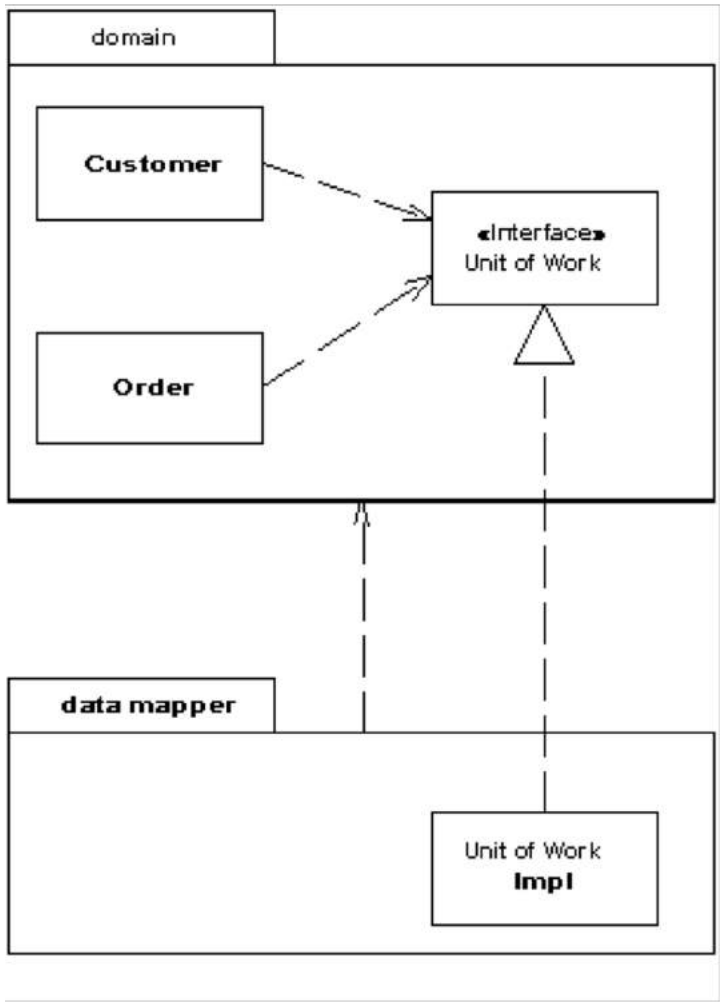
1. Паттерны программирования

Мотивация

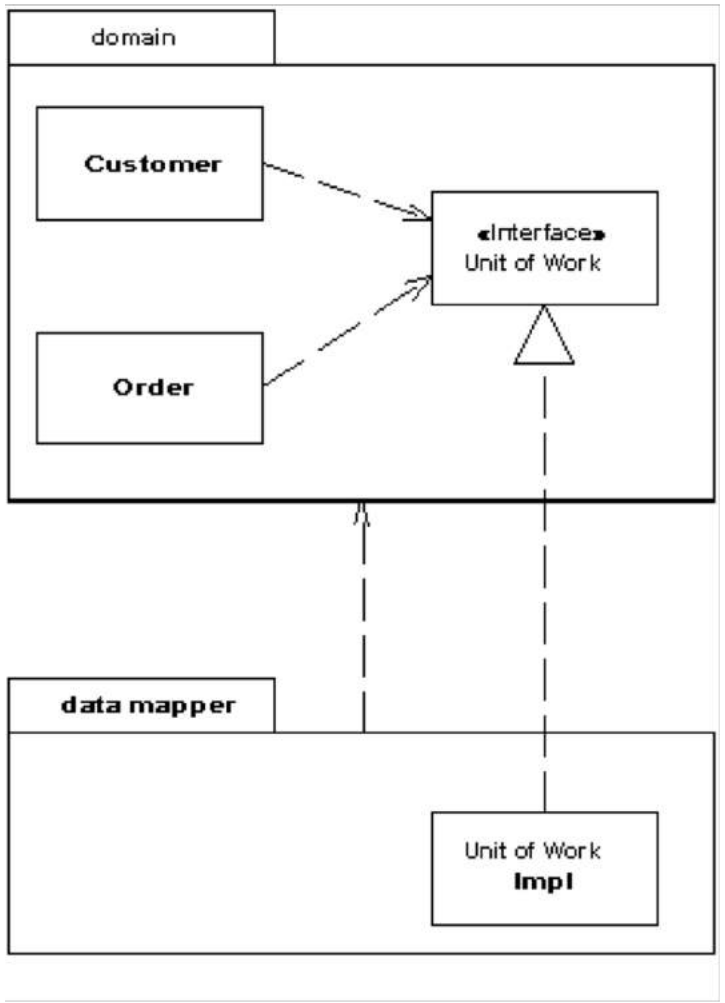

Can we reuse the code for the module the contractor wrote?
No, it's not good...
Oh? Why? Is it encrypted?
No
Is it obfuscated?
No
Ah, is it buggy? Or badly documented?
No, no
Why isn't it any good then?
I don't understand it
Good code is code I can read and understand, you know?
CommitStrip.com


Иными словами, Фабрика делегирует создание объектов наследникам родительского класса
Мужик, я тебя вообще не понимаю, я же, чёрт подери, просто лев!

Интерфейс

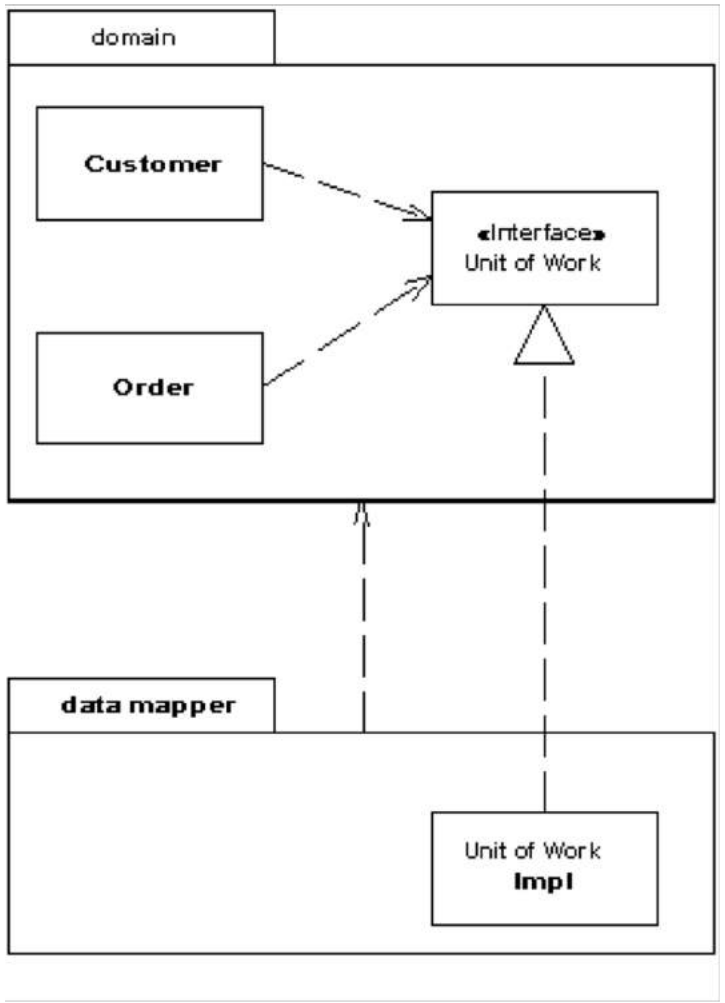

domain
Customer
«Interface»
Unit of Work
Order
data mapper
Unit of Work
Impl

Интерфейс



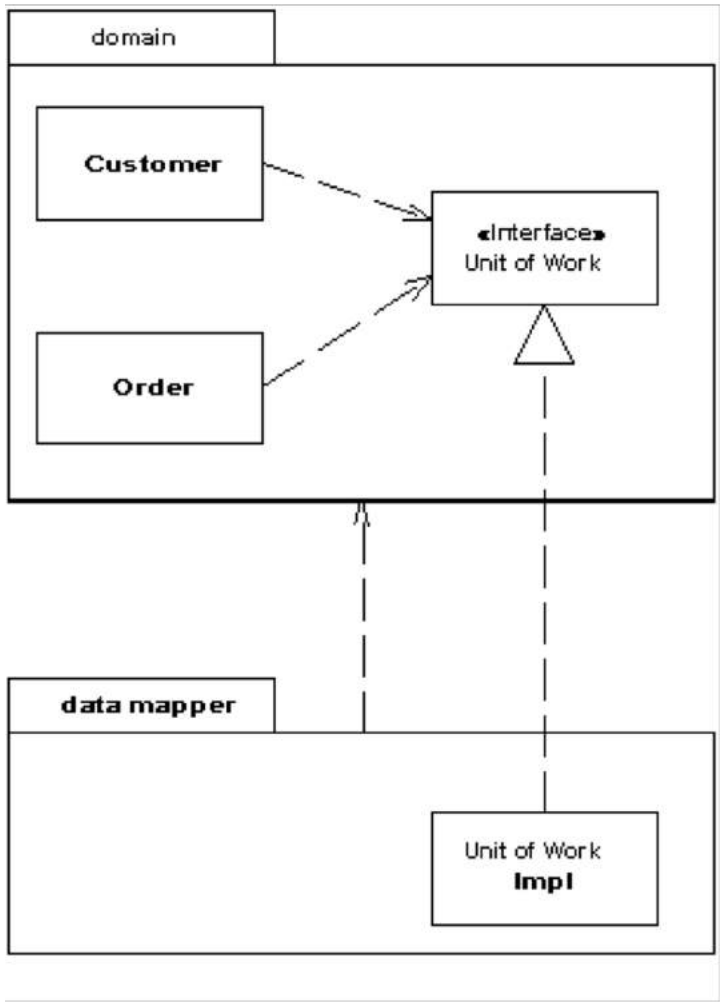
1. Отделяем «важную» часть логики

Интерфейс



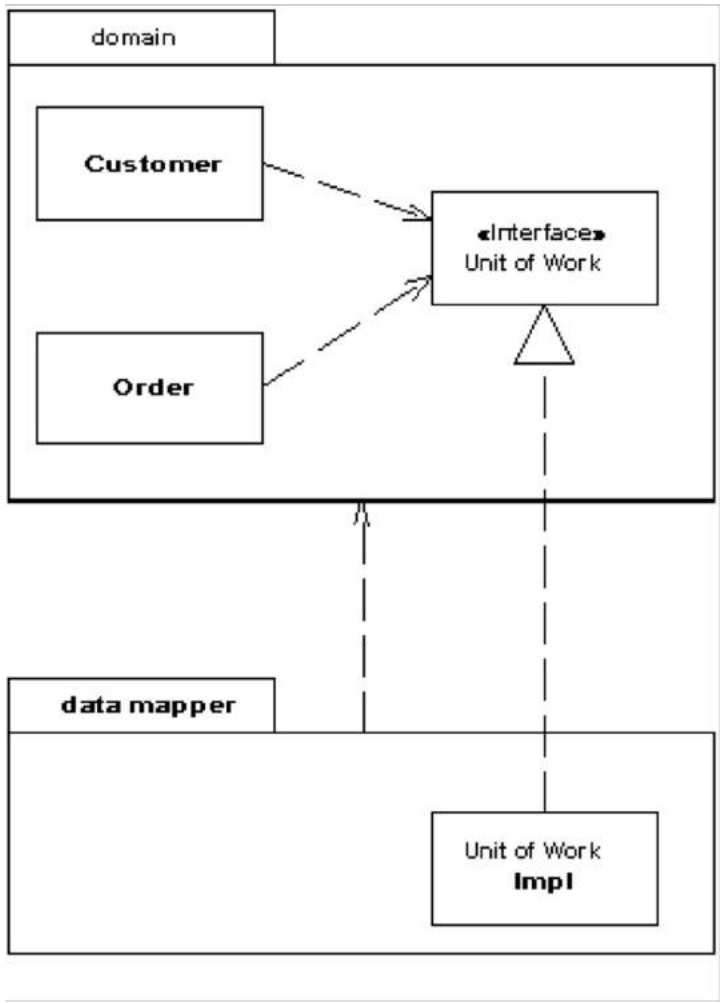
1. Отделяем «важную» часть логики
2. У каждого интерфейса свои «контракты»

Интерфейс



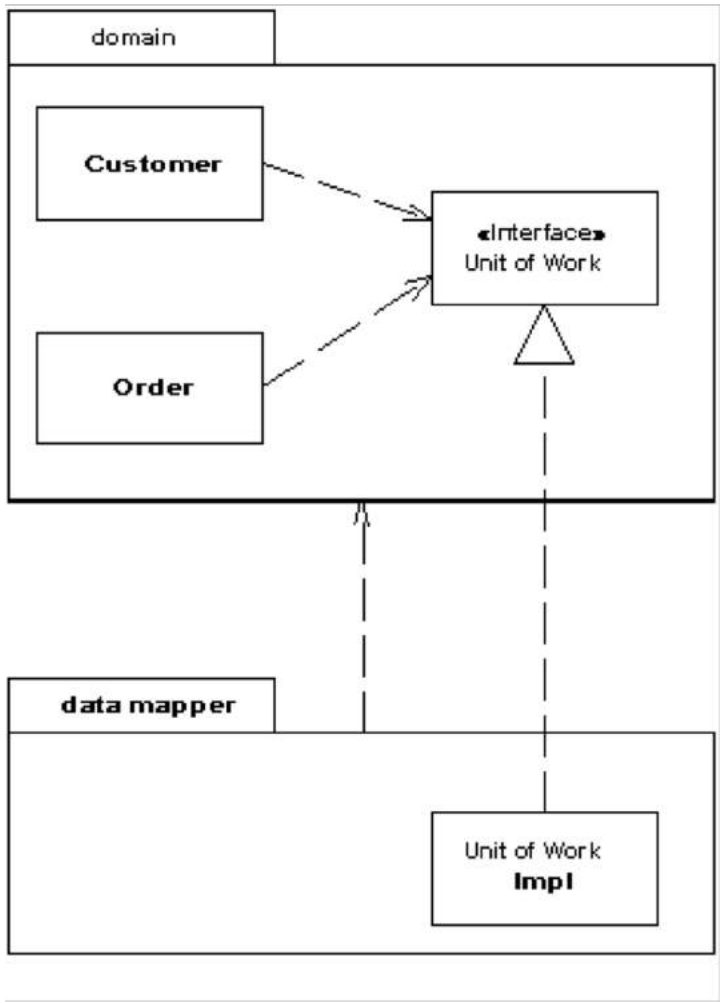
1. Отделяем «важную» часть логики
2. У каждого интерфейса свои «контракты»
3. На этих «контрактах» основывается взаимодействие

Интерфейс



1. Отделяем «важную» часть логики
2. У каждого интерфейса свои «контракты»
3. На этих «контрактах» основывается взаимодействие
4. Есть реализации интерфейсов и они взаимозаменяемы

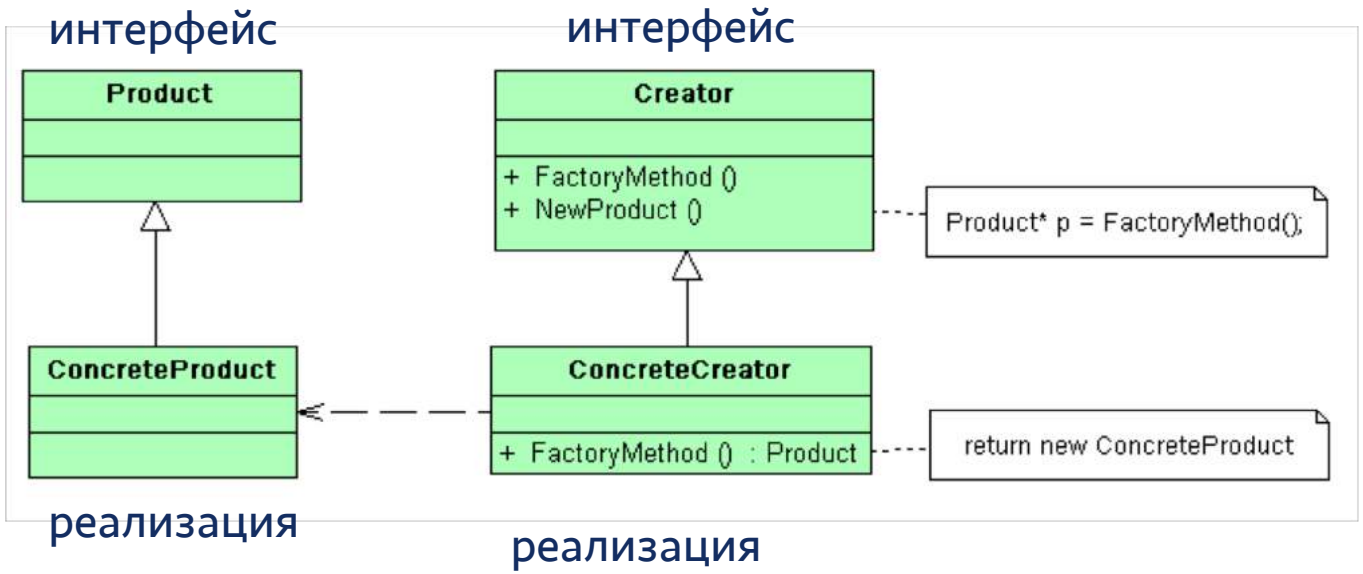
Интерфейс



1. Отделяем «важную» часть логики
2. У каждого интерфейса свои «контракты»
3. На этих «контрактах» основывается взаимодействие
4. Есть реализации интерфейсов и они взаимозаменяемы

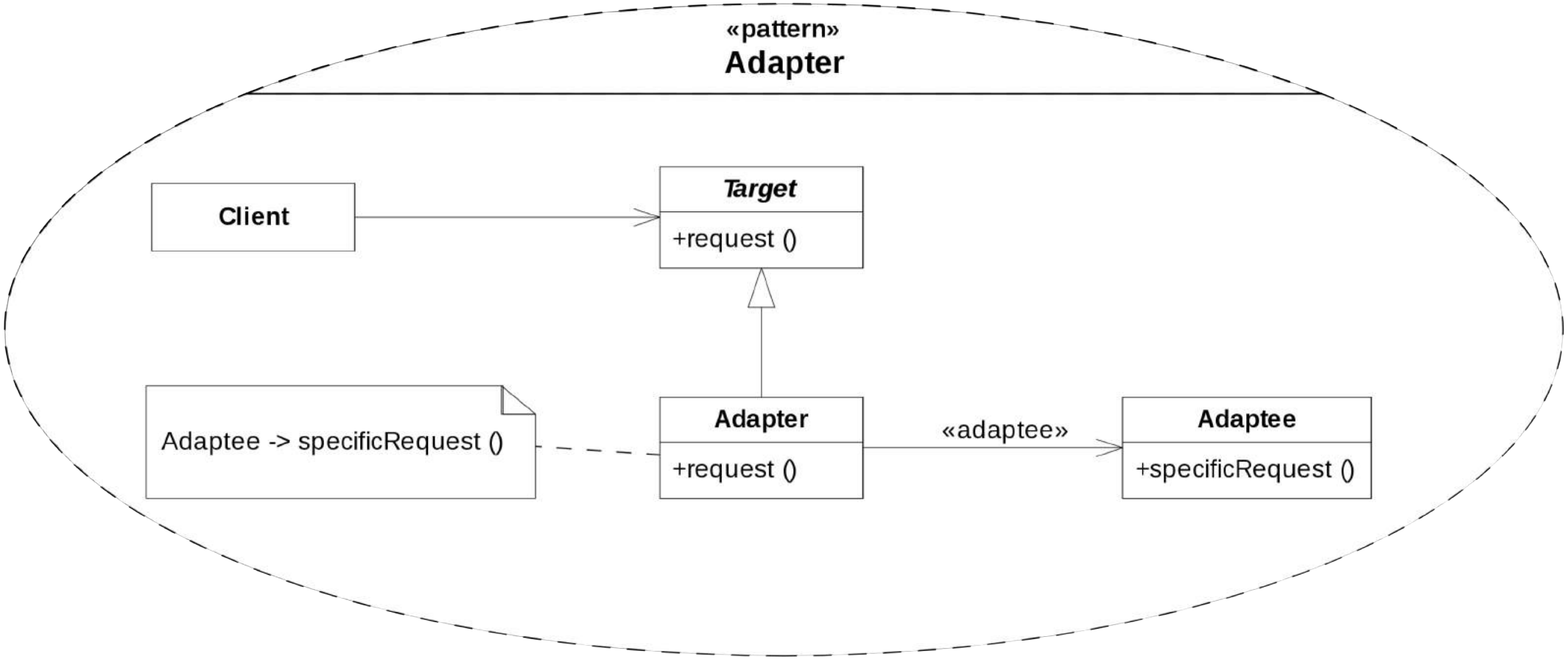
sklearn.base

Фабричный метод

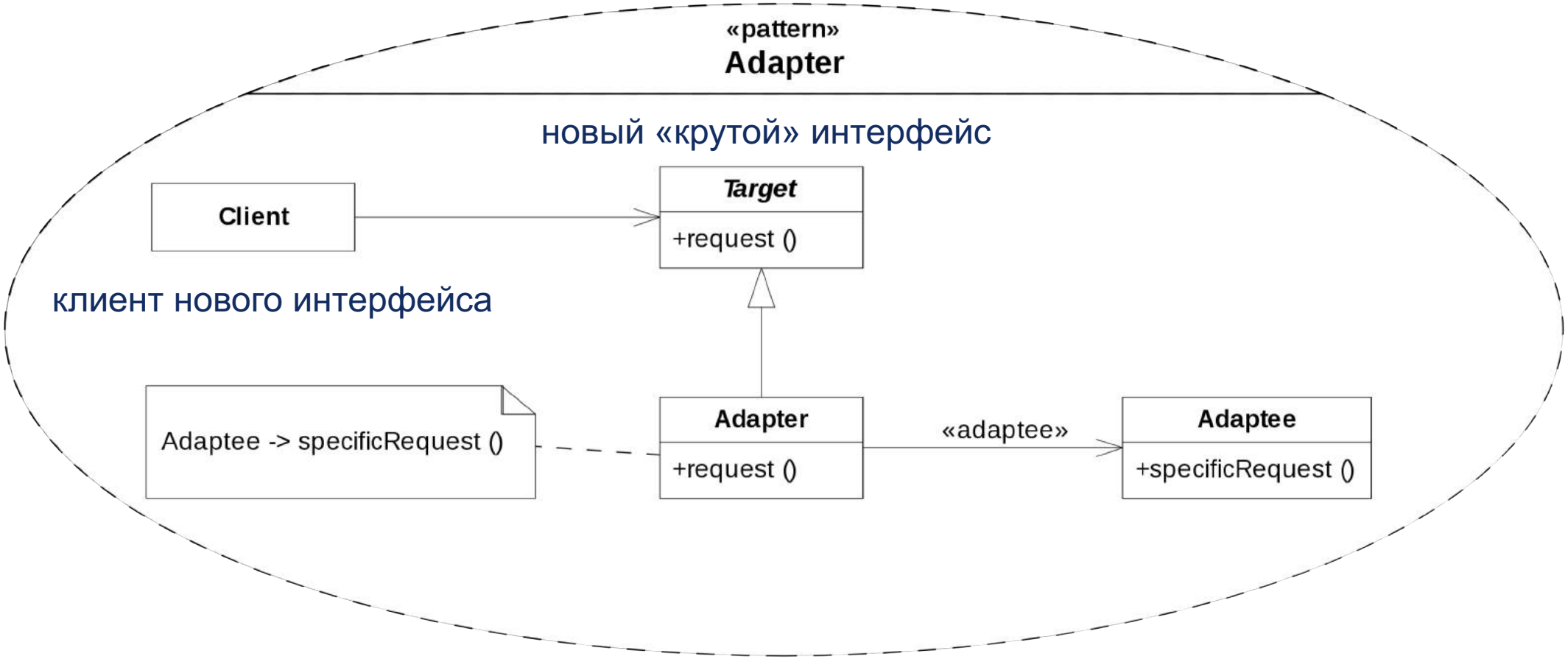


1. Есть интерфейс создания объектов (опционально)
2. Способы создания объекта отделяются от самого объекта
3. Позволяет сделать объекты более переносимыми

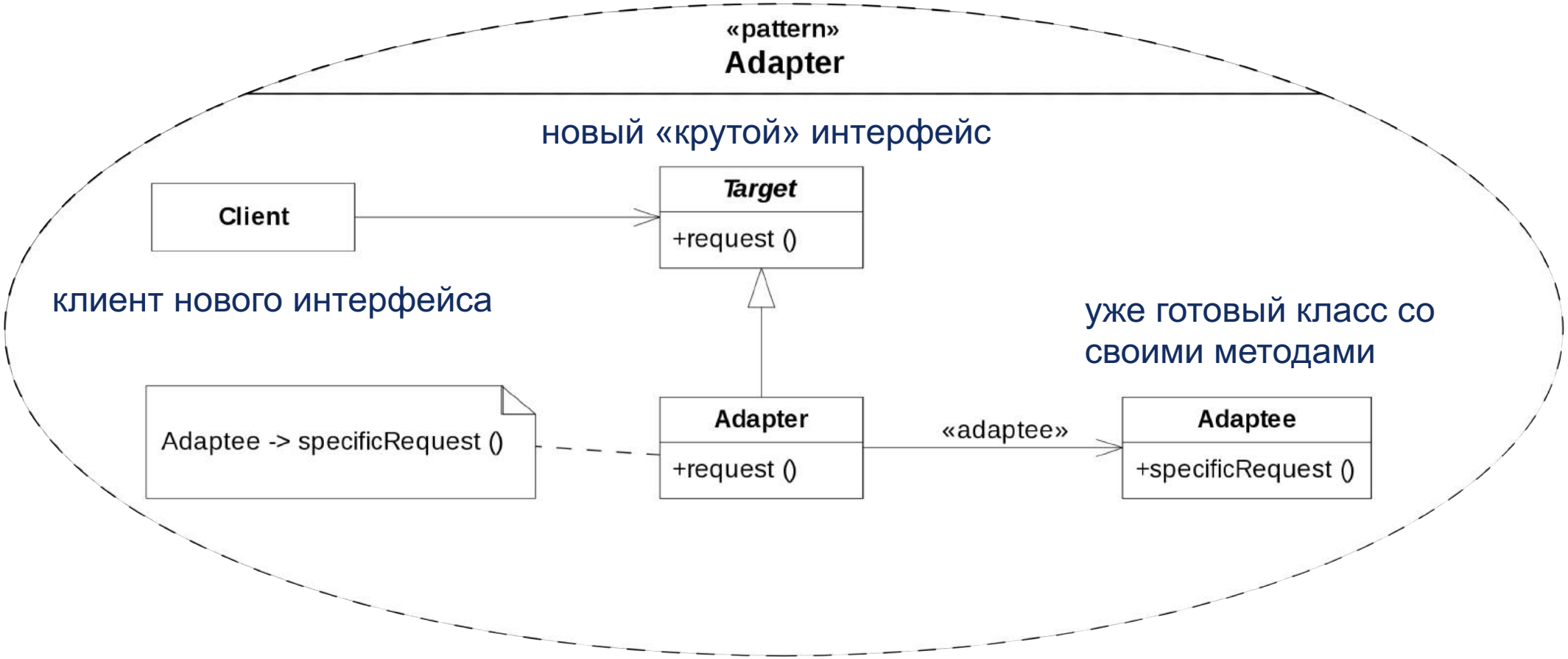
Адаптер


«pattern»
Adapter
Client
Target
+request ()
Adaptee -> specificRequest ()
Adapter
+request ()
«adaptee»
Adaptee
+specificRequest ()

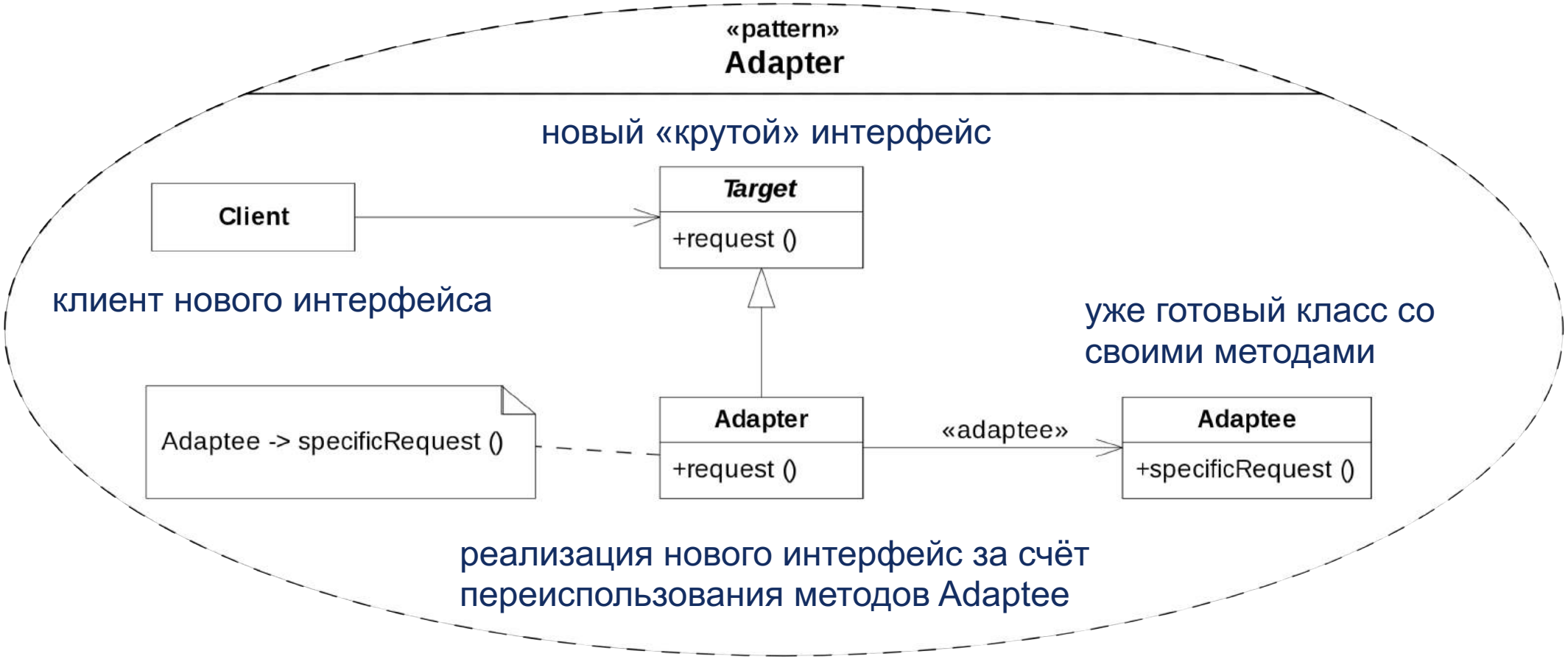
Адаптер


«pattern»
Adapter
новый «крутой» интерфейс
Client
Target
+request ()
клиент нового интерфейса
Adaptee -> specificRequest ()
Adapter
+request ()
«adaptee»
Adaptee
+specificRequest ()

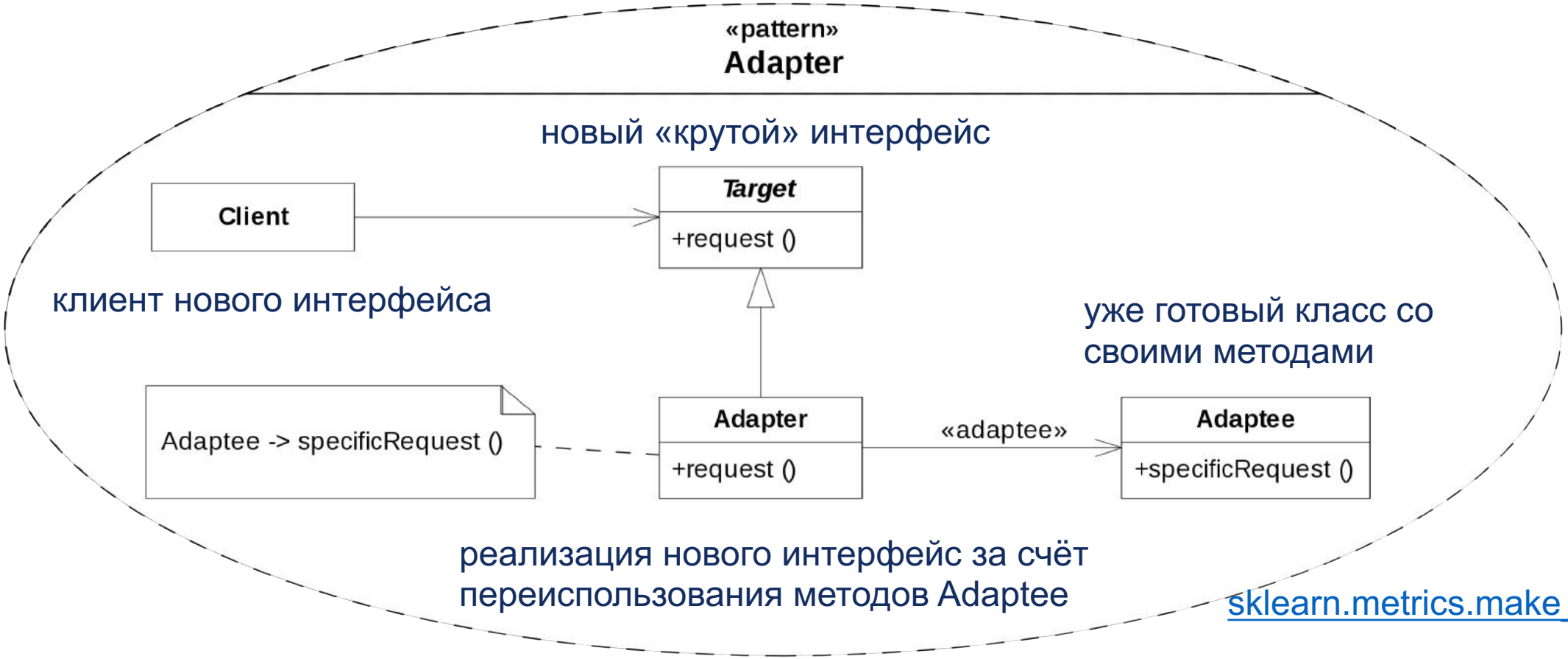
Адаптер


«pattern»
Adapter
новый «крутой» интерфейс
Client
Target
+request ()
клиент нового интерфейса
уже готовый класс со своими методами
Adaptee -> specificRequest ()
Adapter
+request ()
«adaptee»
Adaptee
+specificRequest ()

Адаптер


«pattern»
Adapter
новый «крутой» интерфейс
Client
Target
+request ()
клиент нового интерфейса
уже готовый класс со своими методами
Adaptee -> specificRequest ()
Adapter
+request ()
«adaptee»
Adaptee
+specificRequest ()
реализация нового интерфейс за счёт переиспользования методов Adaptee

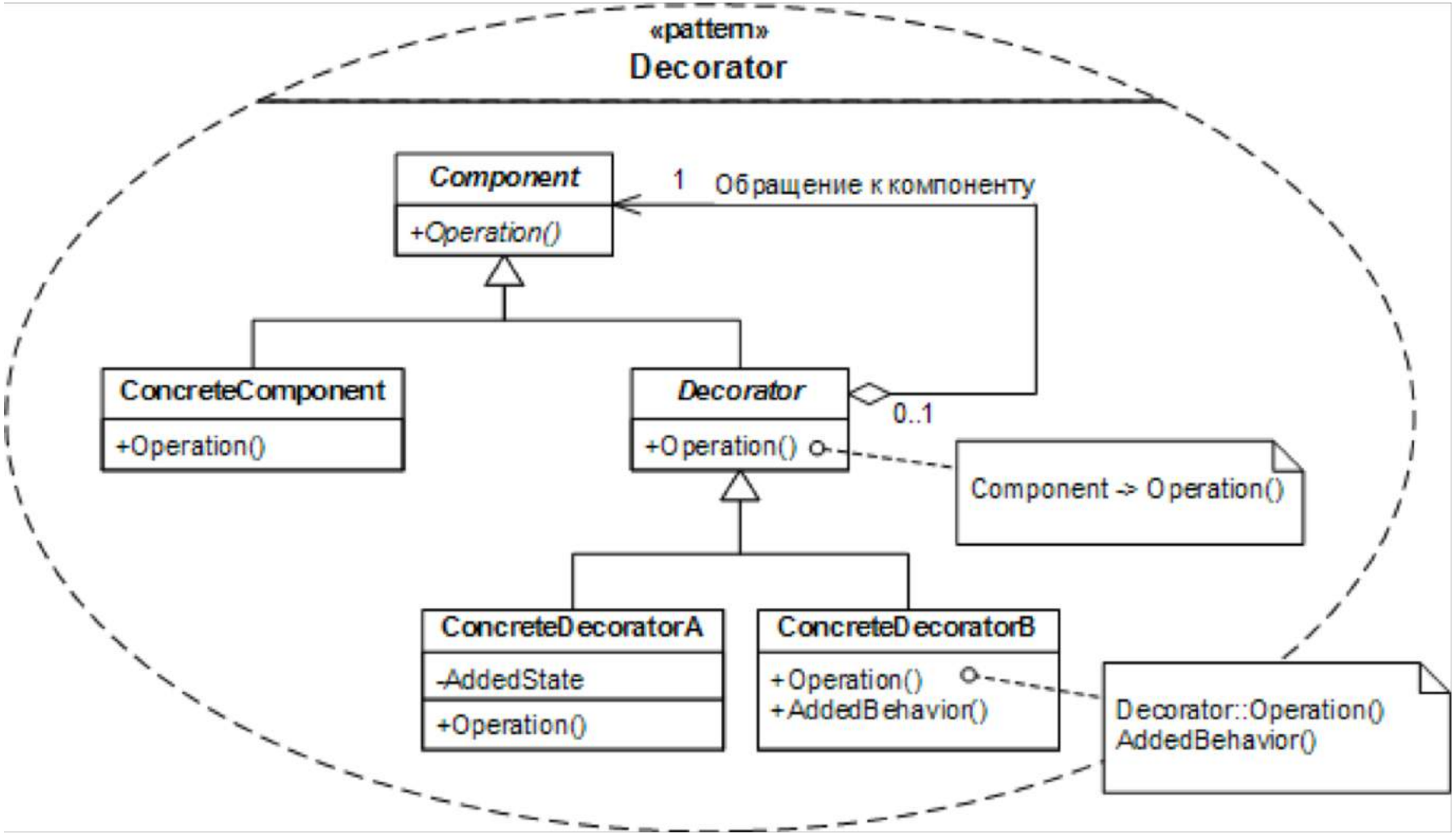
Адаптер



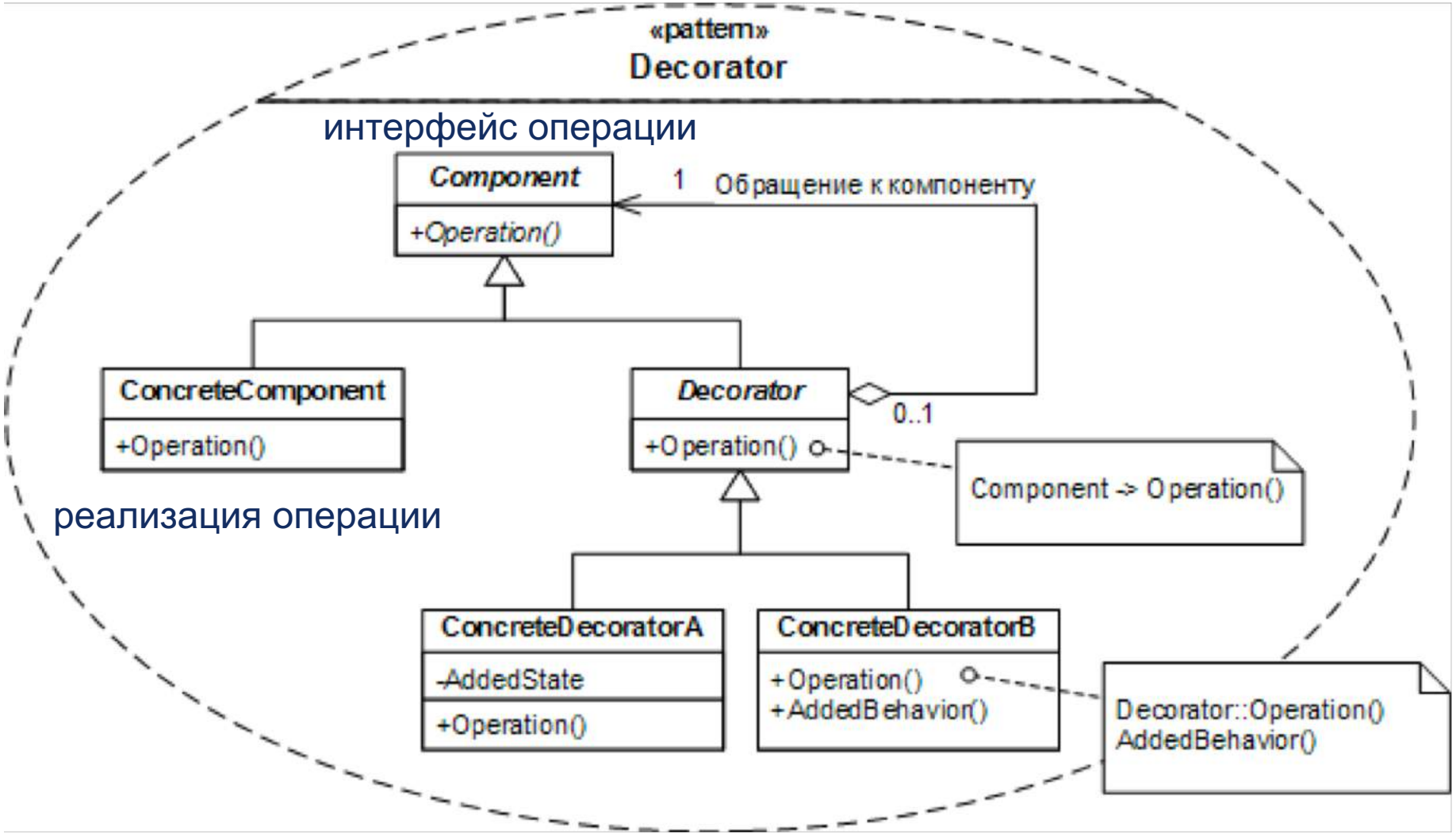
[sklearn.metrics.make_scorer]

[XGBoostRegressor]

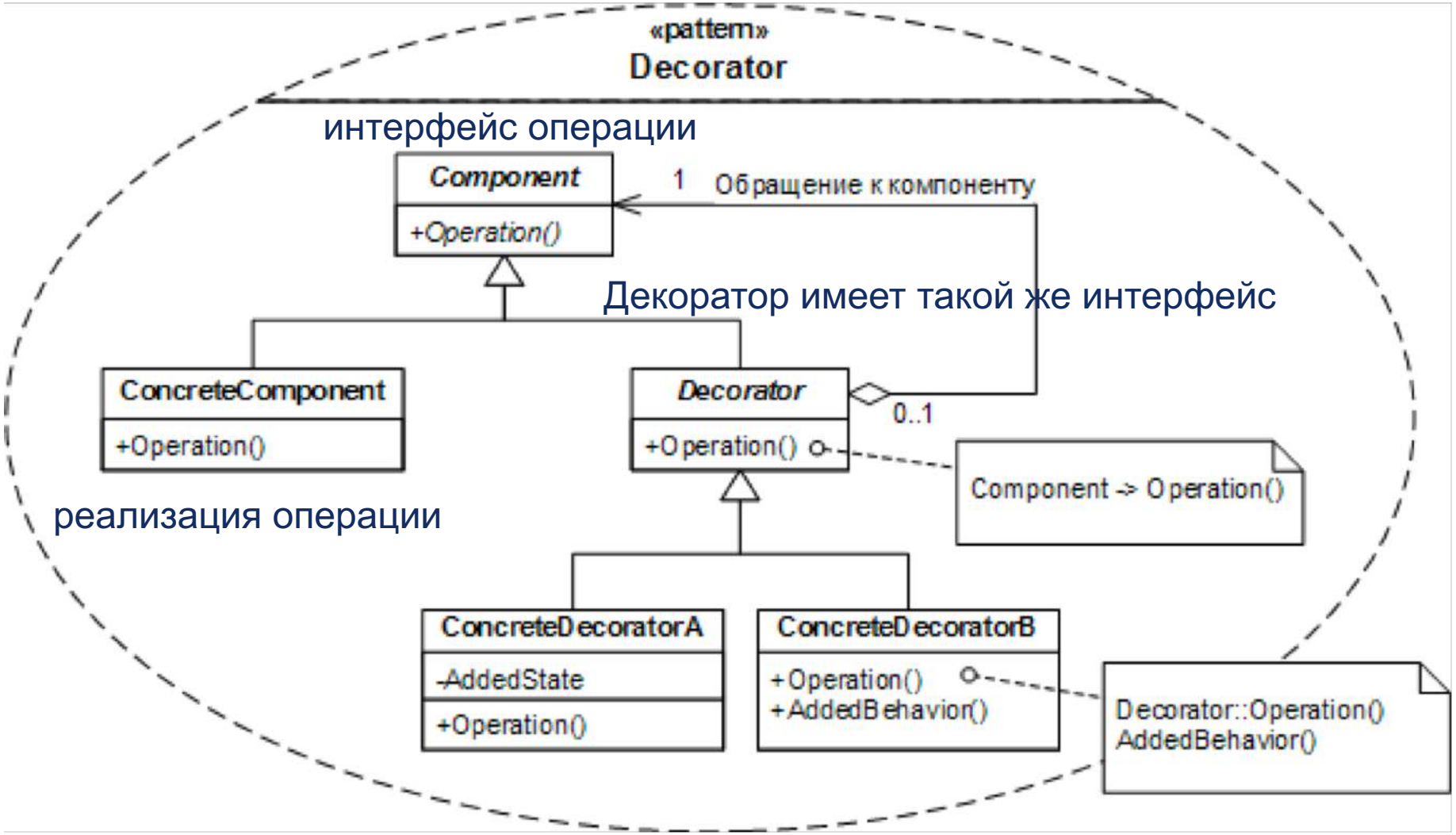
Декоратор


«pattern»
Decorator
Component
+Operation()
1
Обращение к компоненту
ConcreteComponent
+Operation()
Decorator
+Operation()
0..1
Component -> Operation()
ConcreteDecoratorA
-AddedState
+Operation()
ConcreteDecoratorB
+Operation()
+AddedBehavior()
Decorator::Operation()
AddedBehavior()

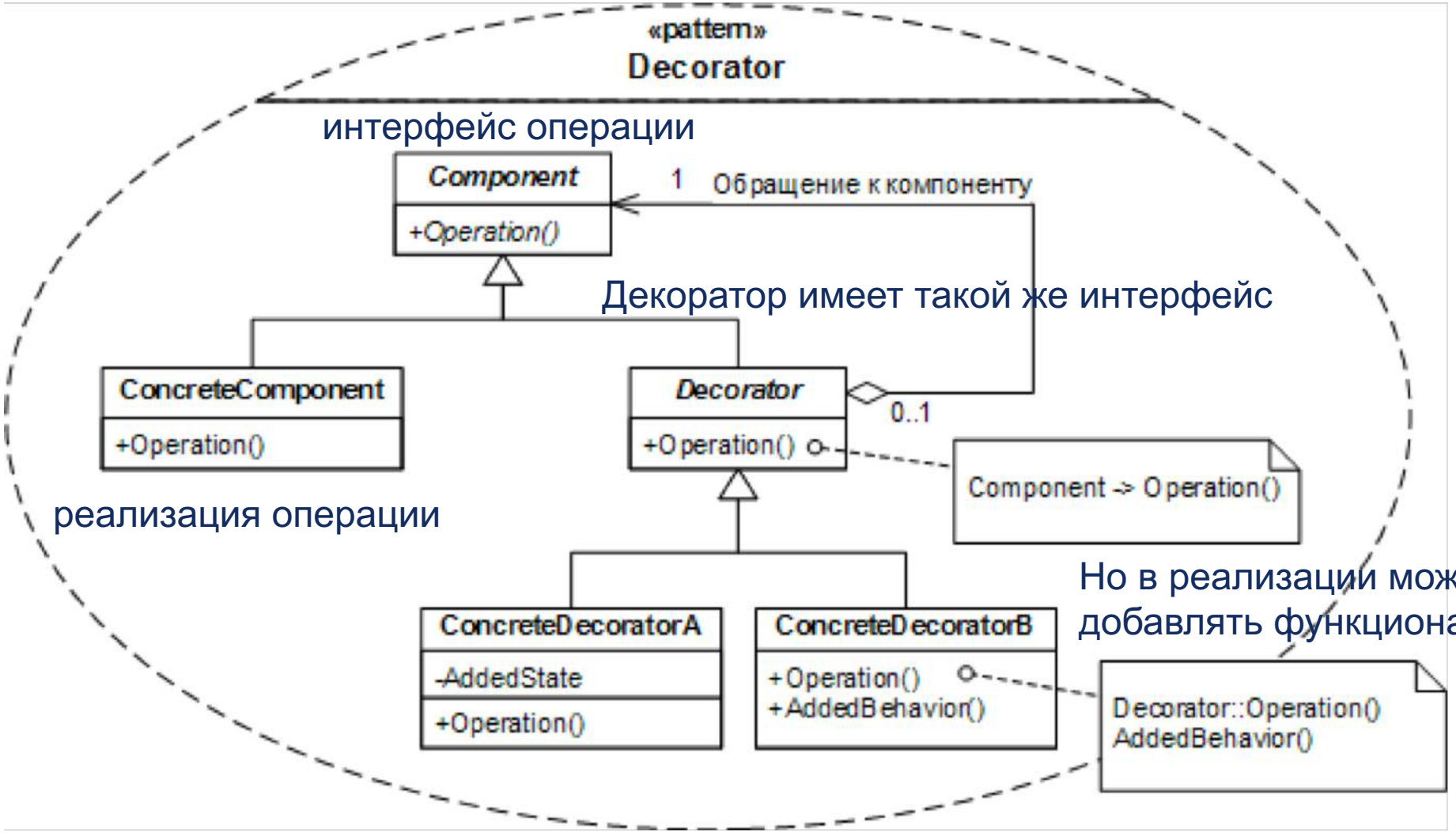
Декоратор


«pattern»
Decorator
интерфейс операции
Component
+Operation()
1
Обращение к компоненту
ConcreteComponent
+Operation()
Decorator
+Operation()
0..1
реализация операции
Component -> Operation()
ConcreteDecoratorA
-AddedState
+Operation()
ConcreteDecoratorB
+Operation()
+AddedBehavior()
Decorator::Operation()
AddedBehavior()

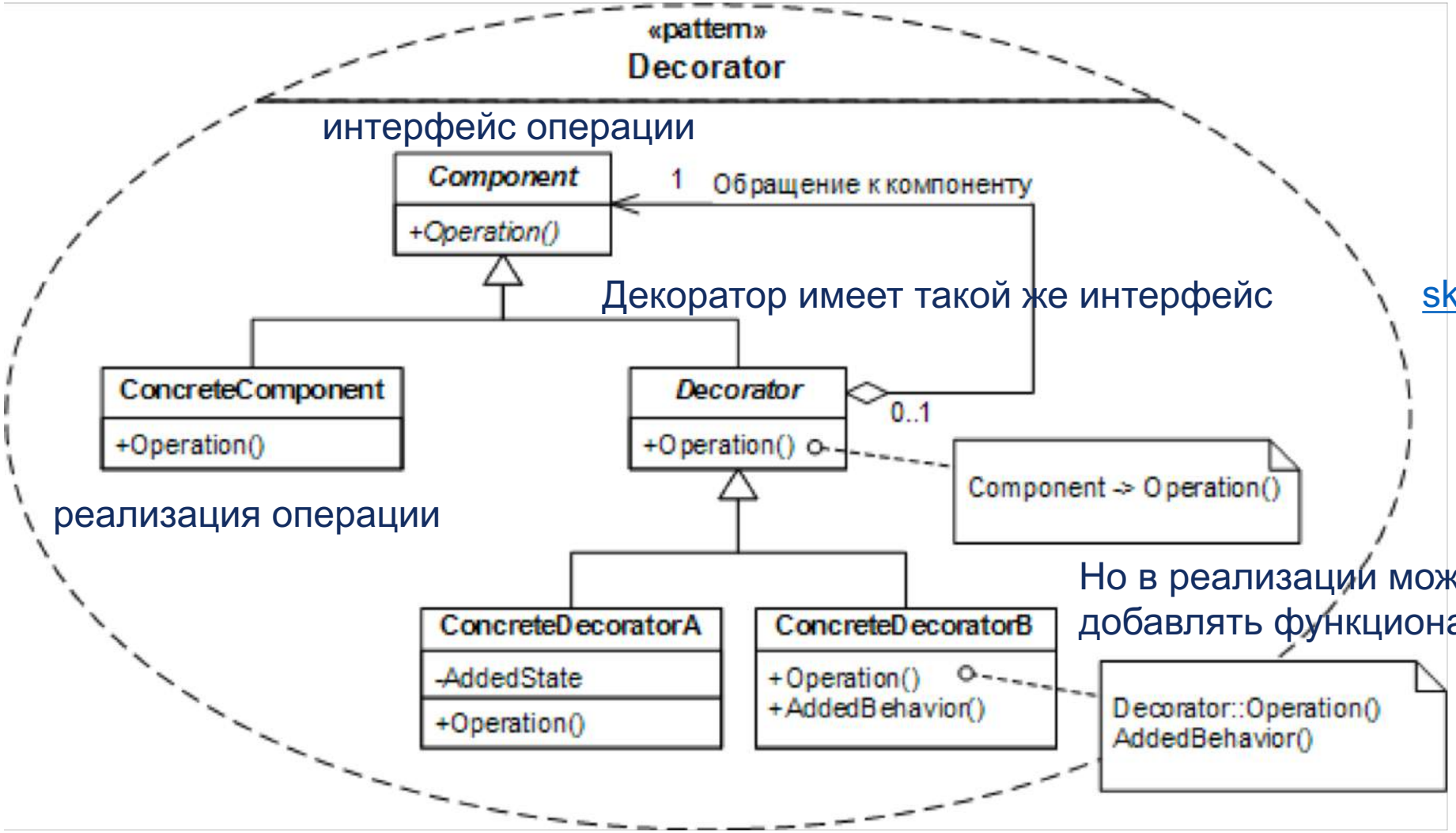
Декоратор


«pattern»
Decorator
интерфейс операции
Component
+Operation()
1
Обращение к компоненту
Декоратор имеет такой же интерфейс
ConcreteComponent
+Operation()
Decorator
+Operation()
0..1
Component -> Operation()
реализация операции
ConcreteDecoratorA
-AddedState
+Operation()
ConcreteDecoratorB
+Operation()
+AddedBehavior()
Decorator::Operation()
AddedBehavior()

Декоратор

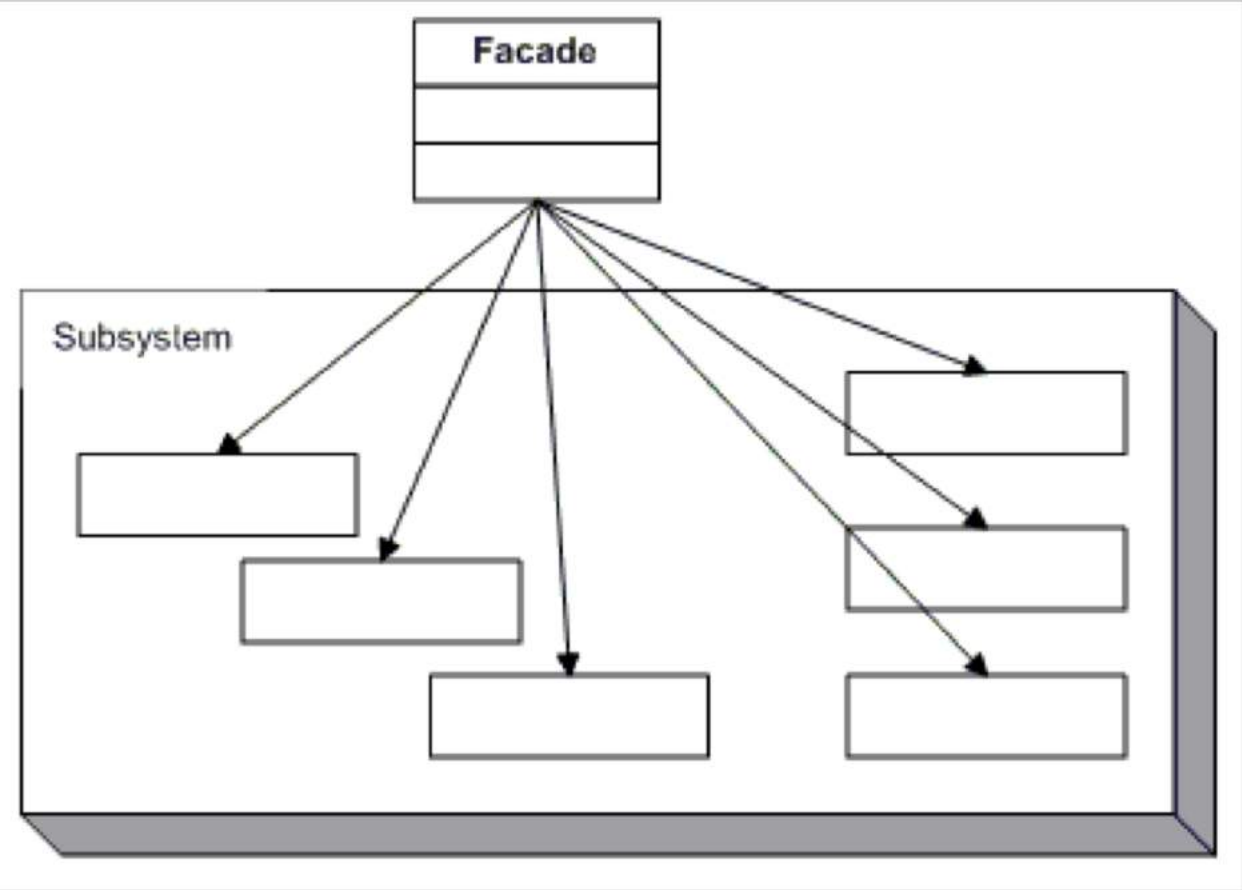

«pattern»
Decorator
интерфейс операции
Component
+Operation()
1
Обращение к компоненту
Декоратор имеет такой же интерфейс
ConcreteComponent
+Operation()
Decorator
+Operation()
0..1
Component -> Operation()
реализация операции
ConcreteDecoratorA
-AddedState
+Operation()
ConcreteDecoratorB
+Operation()
+AddedBehavior()
Но в реализации может
добавлять функциональность
Decorator::Operation()
AddedBehavior()

Декоратор

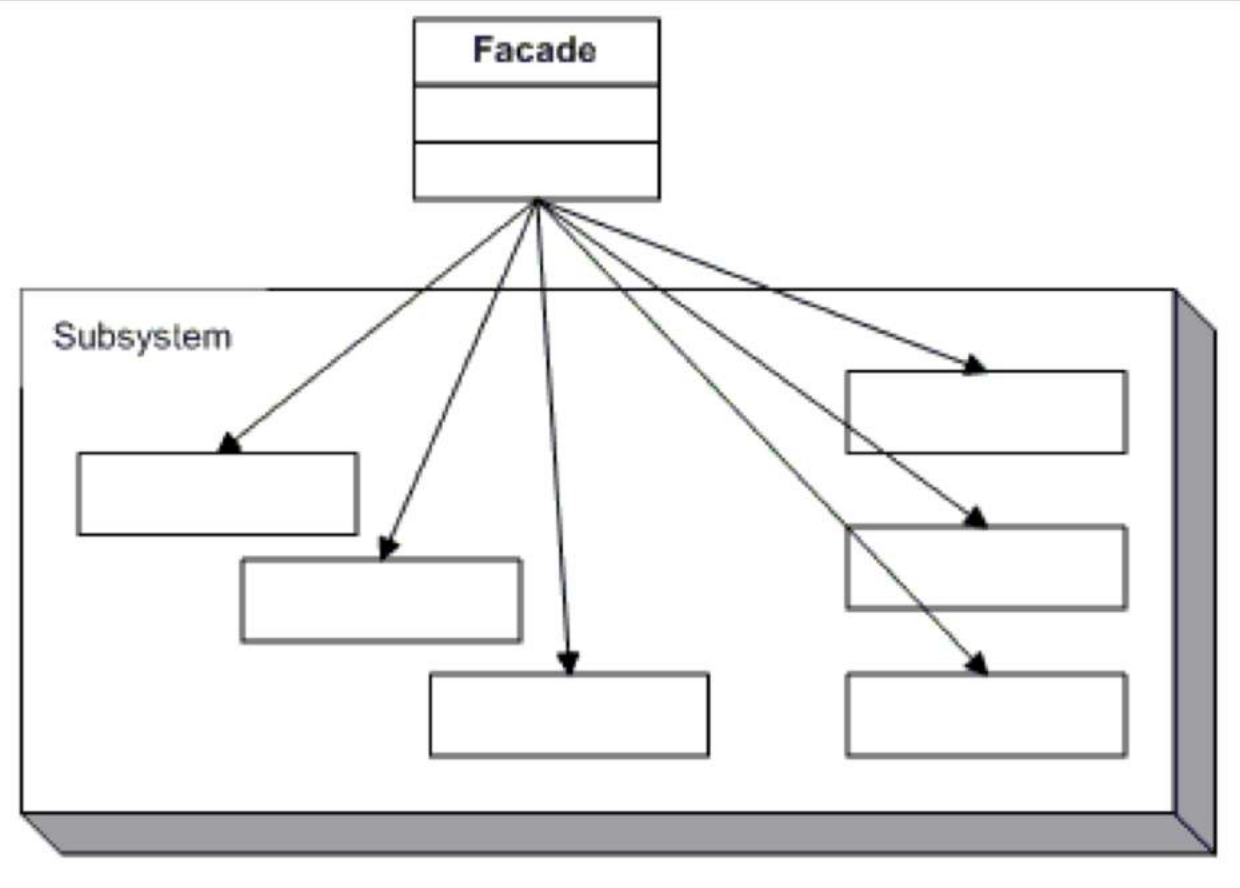


[sklearn.utils.deprecated]

Фасад

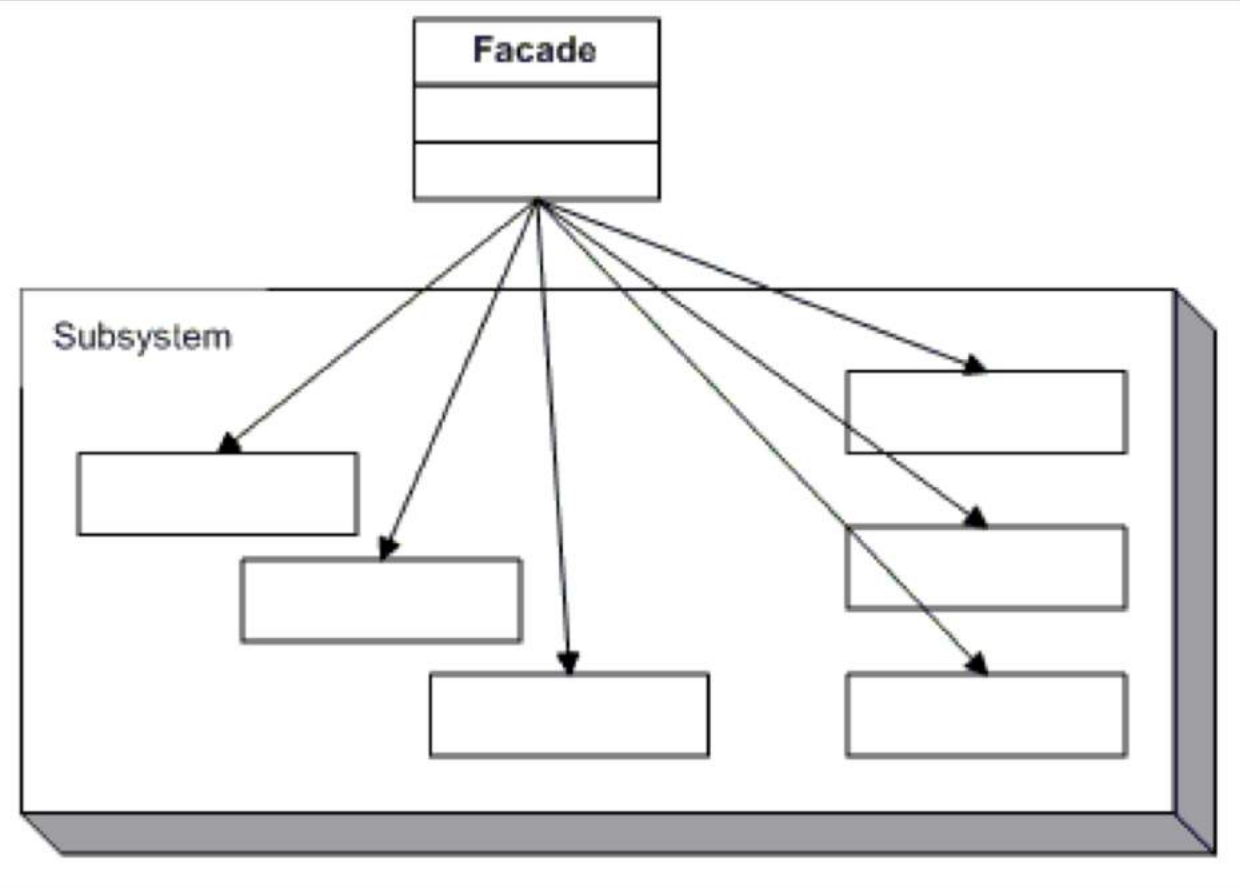


Фасад



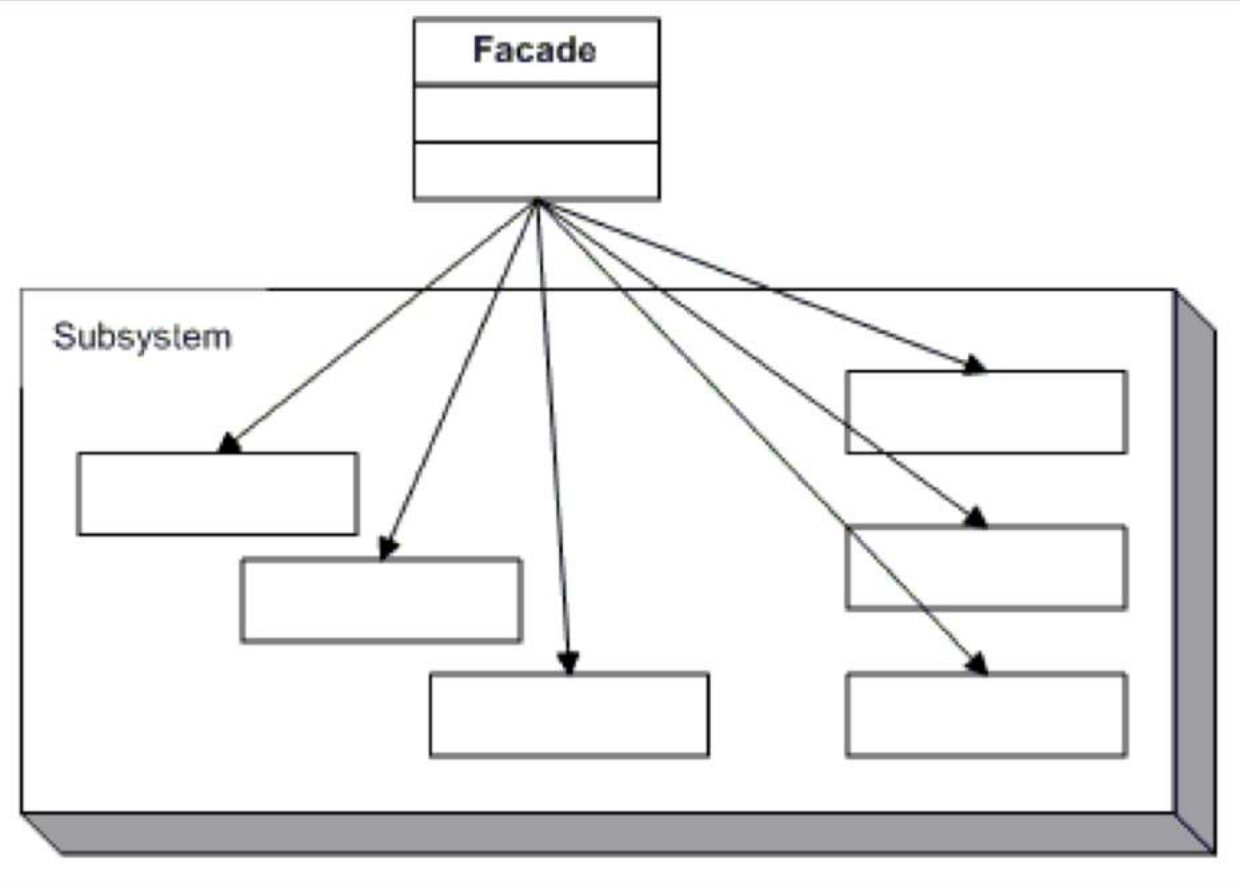
1. Есть уже рабочая система, например, кастомная ml либа вашей команды

Фасад



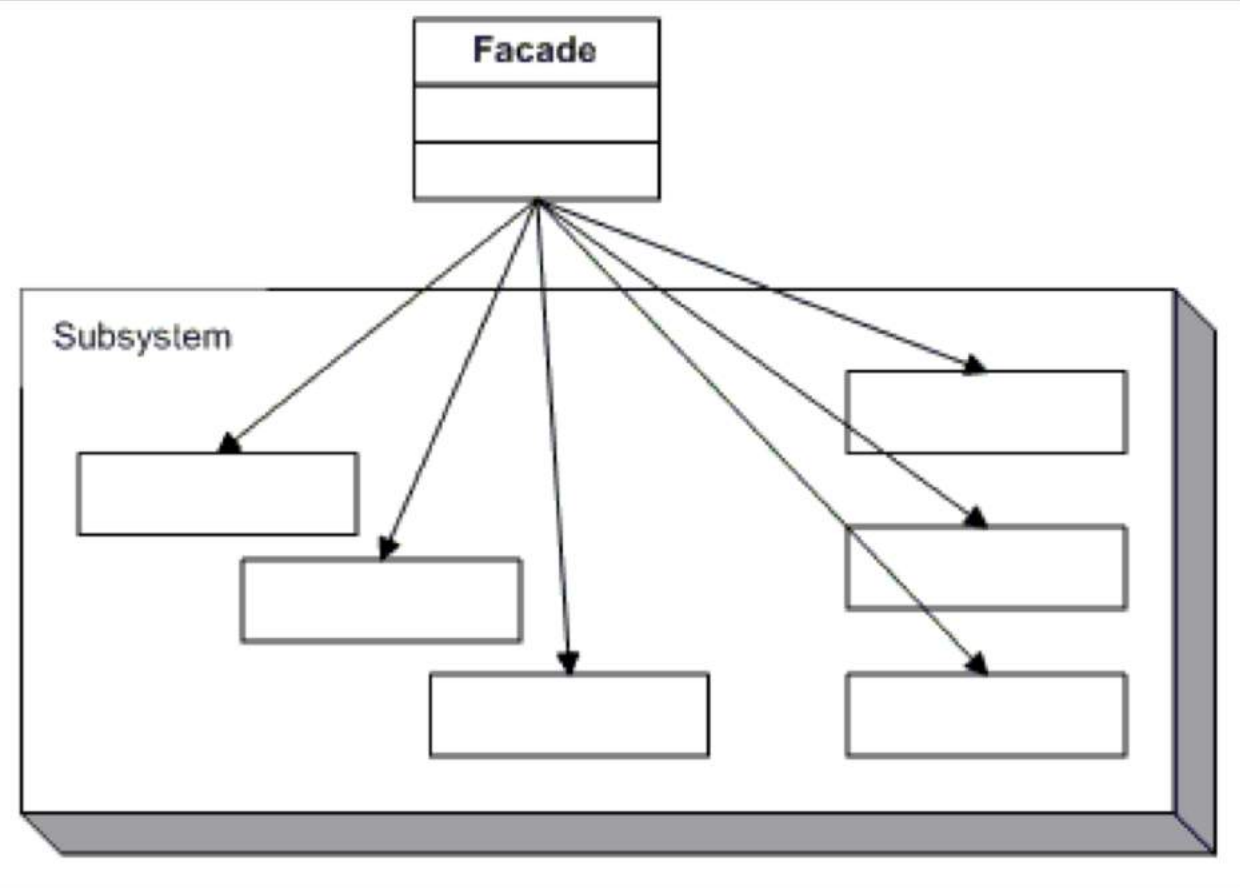
1. Есть уже рабочая система, например, кастомная ml либа вашей команды
2. Интерфейсы были подобраны под вас, чтобы вам было удобно

Фасад



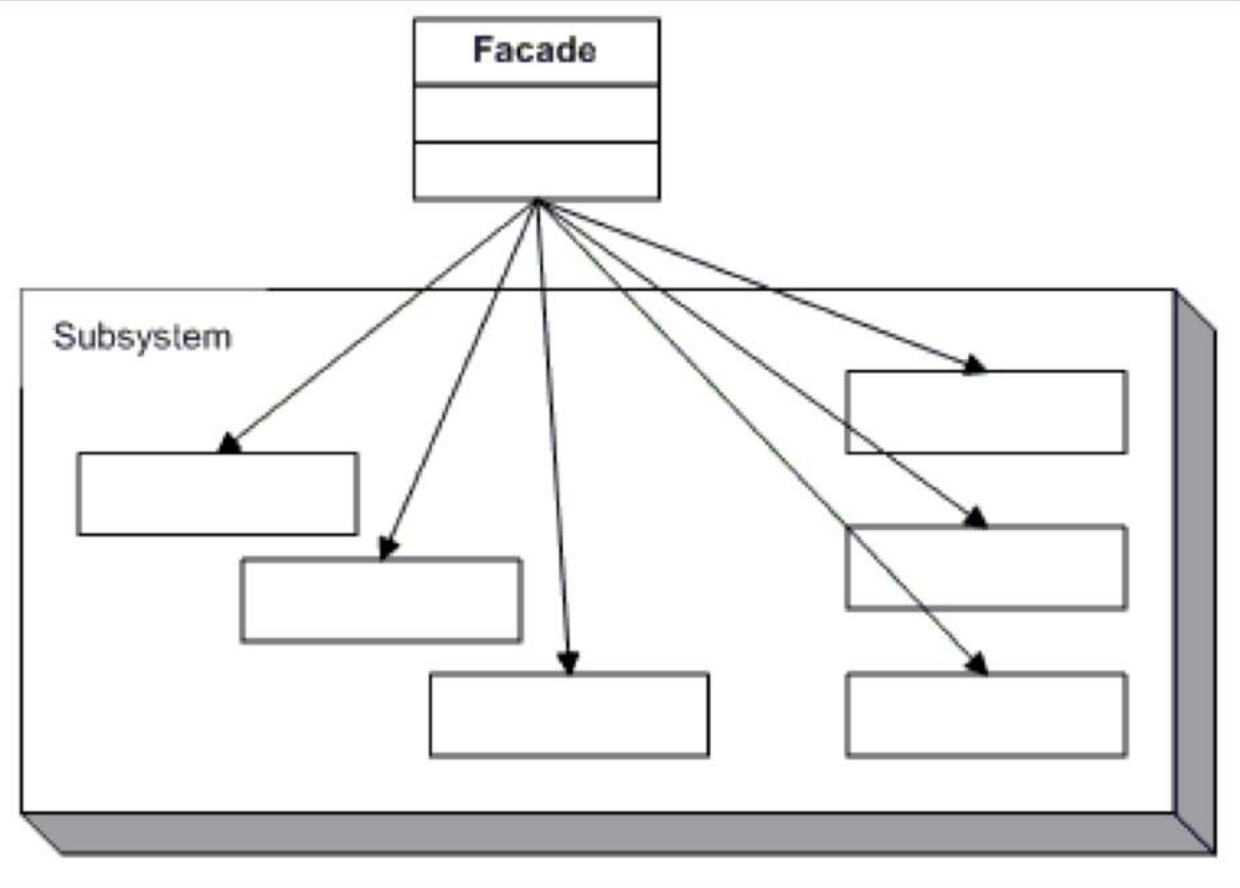
1. Есть уже рабочая система, например, кастомная ml либа вашей команды
2. Интерфейсы были подобраны под вас, чтобы вам было удобно
3. Другая команда пилит решение поверх вашей библиотеки

Фасад



1. Есть уже рабочая система, например, кастомная ml либа вашей команды
2. Интерфейсы были подобраны под вас, чтобы вам было удобно
3. Другая команда пилит решение поверх вашей библиотеки
4. Им не нужна вся функциональность

Фасад

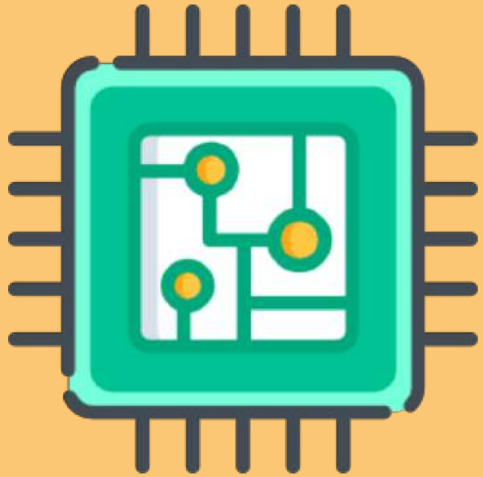


1. Есть уже рабочая система, например, кастомная ml либа вашей команды
2. Интерфейсы были подобраны под вас, чтобы вам было удобно
3. Другая команда пилит решение поверх вашей библиотеки
4. Им не нужна вся функциональность
5. Вы предоставляете удобный для их нужд интерфейс доступа

И многие другие

1. Паттернов программирования очень много!
2. Их знание поможет лучше понимать как работают системы
3. Почитайте Design Patterns это поможет

2. Примеры дизайна из ML



DataContext

Есть много сырых таблиц с данными на MR

Их расположение может меняться

Есть сложные способы подсчёта разных статистик по данным

Они могут друг друга переиспользовать и иметь общие части

DataContext

Есть много сырых таблиц с данными на MR

Их расположение может меняться

Есть сложные способы подсчёта разных статистик по данным

Они могут друг друга переиспользовать и иметь общие части

```python
class DataContext():
    def __init__(self, *args, **kwargs):
        # save parametrs
        self.cache = dict()

    def get_orders(self):
        if 'order' not in self.cache:
            # evaluation orders by data
            self.cache['orders'] = None
        return self.cache['orders']

    def get_carts(self):
        if 'carts' not in self.cache:
            # evaluation carts by data
            self.cache['carts'] = None
        return self.cache['carts']

    def get_orders_with_carts(self):
        if 'orders_with_carts' not in self.cache:
            # join orders and carts
            self.cache['orders_with_carts'] = join(
                self.get_orders(),
                self.get_carts()
            )
        return self.cache['orders_with_carts']
```

TimePoints

1. Решаем задачу, где важен момент предсказания
2. Как задать датасет в такой системе?
3. Как гарантировать незаглядывание в будущее?
4. Чем отличаются target-ы и признаки?

TimePoints

```
class TimePoint:
    timestamp: int,
    key: str,
    timepoint_id: str

class StatisticsEvaluator:
    def __init__(self, data_context, **kwargs):
        self.data_context = data_context
        # save other parameters

    def __call__(self, time_points):
        # 1. return <timepoint_id, statistic>
        # 2. return time_points with statistics
```

ExperimentManager + Factory

1. Хотим конфигурировать эксперимент, не меняя код
2. Имея уже созданные объекты, логика их использования может быть сложной

ExperimentManager + Factory

1. Хотим конфигурировать эксперимент, не меняя код
2. Имея уже созданные объекты, логика их использования может быть сложной
3. Нужно разделить логику создания объектов и их использования

ExperimentManager + Factory

```
class Factory:
    def __init__(self, config):
        self.config = config
    # creating objects methods depending on self.config

class Experiment:
    def __init__(factory, other_params):
        self.factory = factory
        self.other_params = other_params

    def step1():
        step1_object = self.factory.create_step1_object()
        # do step1 with step1_object

    def step2():
        step2_object = self.factory.create_step2_object()
        # do step1 with step2_object

    # ...
```

3. Интерактив

Задача

- Нужно задизайнить pipeline обучения для рекомендательной системы
- Давайте проясним важные моменты

Задача

- Нужно задизайнить pipeline обучения для рекомендательной системы
- Давайте проясним важные моменты:
  - Как представляются данные?
  - Какие части есть в обучении?
  - Какие из частей будут использоваться при применении?
  - Как они взаимодействуют?
  - Какие есть гарантии на взаимодействия?

Как представляются данные?

Как представляются данные?

Обычно данных много и они хранятся на MR

Поэтому будет считать, что у нас поток запросов вида:
<тело запроса с данными; целевой объект>

Какие части есть в обучении?

Какие части есть в обучении?

Извлечение из запроса множество кандидатов

Определение для кандидата его релевантности

Определение для кандидатов его признаков

Сэмплирование негативных примеров

Формирование датасета и определение весов объектов

Разделение на тест и трейн

Форматирование датасета под алгоритм обучения

Обучение

Применение модели к запросу – получение рекомендаций

Оценка качества

Что используется в проде?

Извлечение из запроса множество кандидатов

Определение для кандидата его релевантности

Определение для кандидатов его признаков

Сэмплирование негативных примеров

Формирование датасета и определение весов объектов

Разделение на тест и трейн

Форматирование датасета под алгоритм обучения

Обучение

Применение модели к запросу – получение рекомендаций

Оценка качества

Что используется в проде?

Извлечение из запроса множество кандидатов

Определение для кандидатов его признаков

Применение модели к запросу – получение рекомендаций

Как они взаимодействуют и с какими гарантиями?

CandidatesExtractor(***) -> ***

Как они взаимодействуют и с какими гарантиями?

CandidatesExtractor(request_data) -> list of objects

Как они взаимодействуют и с какими гарантиями?

CandidatesExtractor(request_data) -> list of objects

Relenacer(***) -> ***

Как они взаимодействуют и с какими гарантиями?

CandidatesExtractor(request_data) -> list of objects

Relenacer(candidates, target) -> list of relevances

FeaturesExtractor(***) -> ***

Как они взаимодействуют и с какими гарантиями?

CandidatesExtractor(request_data) -> list of objects

Relenacer(candidates, target) -> list of relevances

FeaturesExtractor(request_data, candidates) -> matrix of numbers

Sampler(***) -> ***

Как они взаимодействуют и с какими гарантиями?

CandidatesExtractor(request_data) -> list of objects

Relenacer(candidates, target) -> list of relevances

FeaturesExtractor(request_data, candidates) -> matrix of numbers

Sampler(request_data , candidates, target, feature_matrix) -> list of pairs of index and its weight

Splitter(***) -> ***

Как они взаимодействуют и с какими гарантиями?

CandidatesExtractor(request_data) -> list of objects

Relenacer(candidates, target) -> list of relevances

FeaturesExtractor(request_data, candidates) -> matrix of numbers

Sampler(request_data , candidates, target, feature_matrix) -> list of pairs of index and its weight

Splitter(request_data) -> bool

RecModel(***) -> ***

Как они взаимодействуют и с какими гарантиями?

CandidatesExtractor(request_data) -> list of objects

Relenacer(candidates, target) -> list of relevances

FeaturesExtractor(request_data, candidates) -> matrix of numbers

Sampler(request_data , candidates, target, feature_matrix) -> list of pairs of index and its weight

Splitter(request_data) -> bool

RecModel(request_data) -> list of objects

4. Алгоритмистика

Пример ML задачи

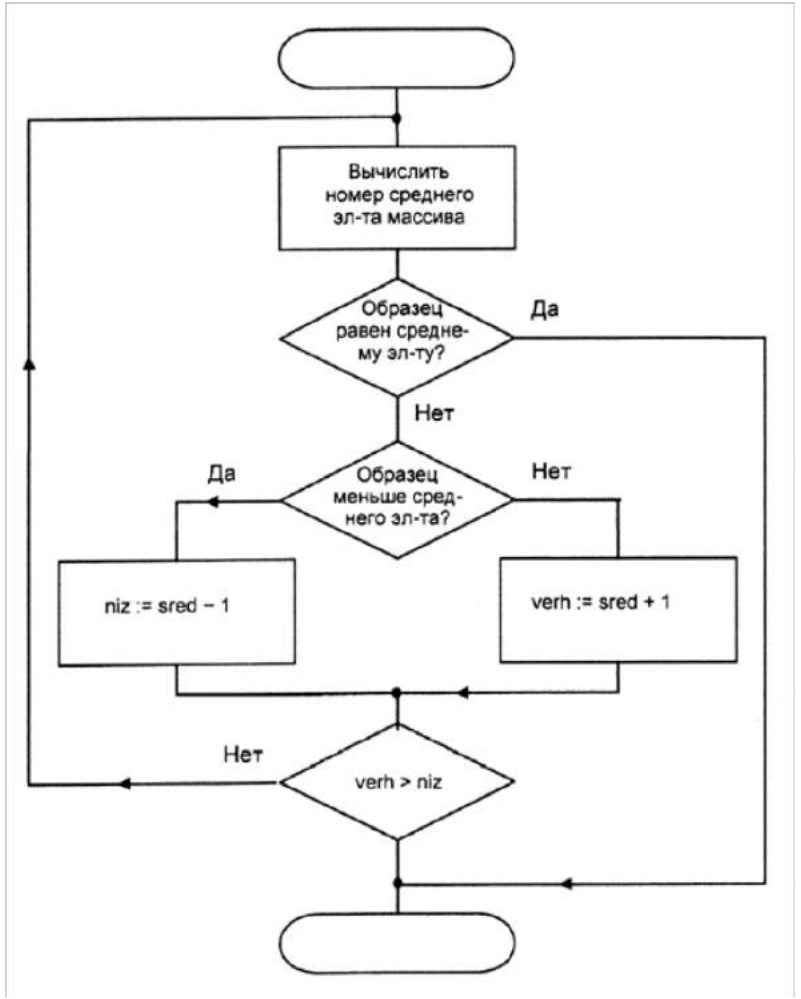
Есть поминутное число заказов, чтобы считать фичи, нужно научиться для произвольной минуты t и числу k отвечать через сколько минут с момента t произойдёт k заказов

Пример задачи с собеседования

Напишите бинарный поиск

Как вы думаете, сколько человек решают эту задачу?

А с первого раза?

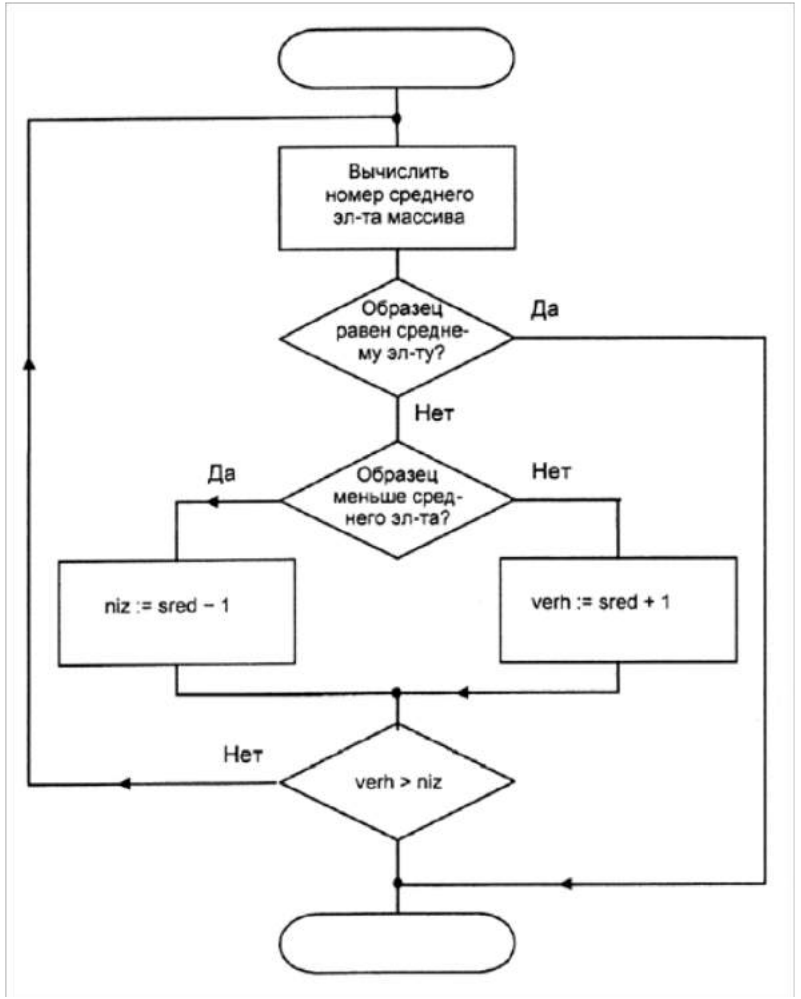


Пример задачи с собеседования

Напишите бинарный поиск

Как вы думаете, сколько человек решают эту задачу?

А с первого раза?



Пример ML задачи 2

Есть поток временных событий. Нужно для заданных моментов времени составить список предшествующих событий (но не более 1000).

Пример ML задачи 2

Есть поток временных событий. Нужно для заданных моментов времени составить список предшествующих событий за (но не более 1000 и не ранее чем за месяц).

Нужно создать очередь;

Добавлять элементы справа, пока выполняются ограничения;

Дсли они нарушаются, то удалять слева

Пример не совсем ML задачи 3

Вы хотите получить случайную перестановку чисел от 1 до n

Как это сделать?

Пример не совсем ML задачи 3

Вы хотите получить случайную перестановку чисел от 1 до n

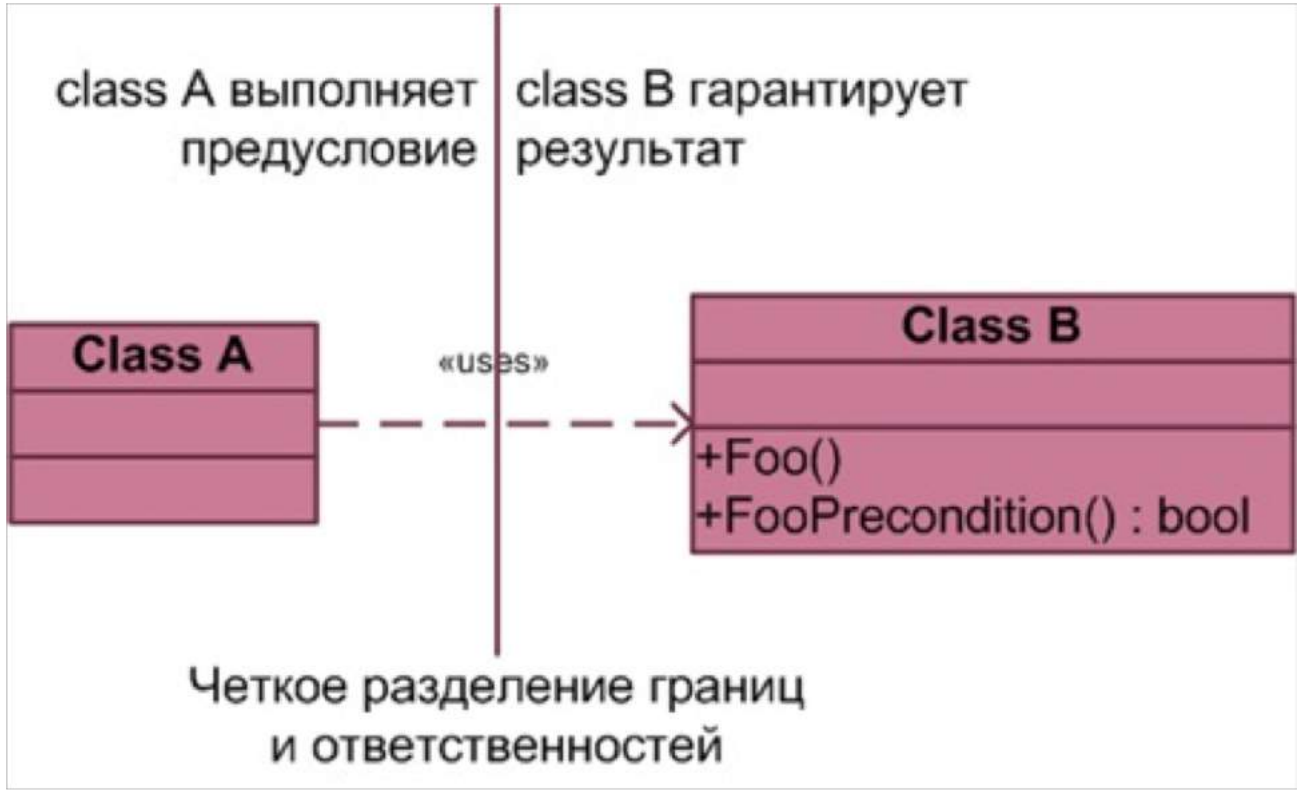
Как это сделать?

Пусть получена случайная перестановка чисел от 1 до n-1, нужно добавить n в конец и переставить со случайным элементом. Такой подход можно использовать в бутстрепе.

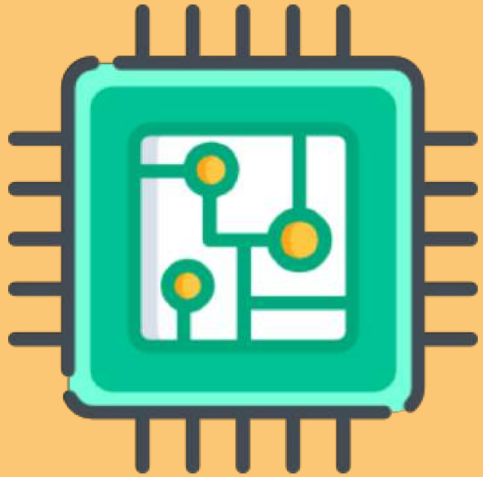
Общий совет

Нужно мыслить контрактами:

- **Типы данных**
- **Предусловие**
- **Инвариант**
- **Постусловие**



5. Про Python



Если коротко

1. Нужно знать особенности в поведении языка

Что выведет этот код?
А что ожидает нормальный человек?

```
list_of_functions = list()
fst_value = 0
for snd_value in range(10):
    list_of_functions.append(
        lambda: (fst_value, snd_value)
    )
    fst_value += 1

for function in list_of_functions:
    print(function())
```

Что выведет этот код?
А что ожидает нормальный человек?

```
list_of_functions = list()
fst_value = 0
for snd_value in range(10):
    list_of_functions.append(
        lambda: (fst_value, snd_value)
    )
    fst_value += 1

for function in list_of_functions:
    print(function())
```

```
(10, 9)
(10, 9)
(10, 9)
(10, 9)
(10, 9)
(10, 9)
(10, 9)
(10, 9)
(10, 9)
(10, 9)
```

Это код скрипта a.py

```
for _ in range(3):
    print(hash('10'), hash(10))
~
~
~
```

Что будет происходить при вызове команды python a.py?

Это код скрипта a.py

```
for _ in range(3):
    print(hash('10'), hash(10))
~
~
~
```

Что будет происходить при вызове команды python a.py?

А в разных версиях питона?

Если коротко

1. Нужно знать особенности в поведении языка
2. Нужно знать ограничения языка

Типы полей в классах

1. Public – видны всем
2. Protected – видны только внутри класса и в его наследниках
3. Private – видны только внутри класса

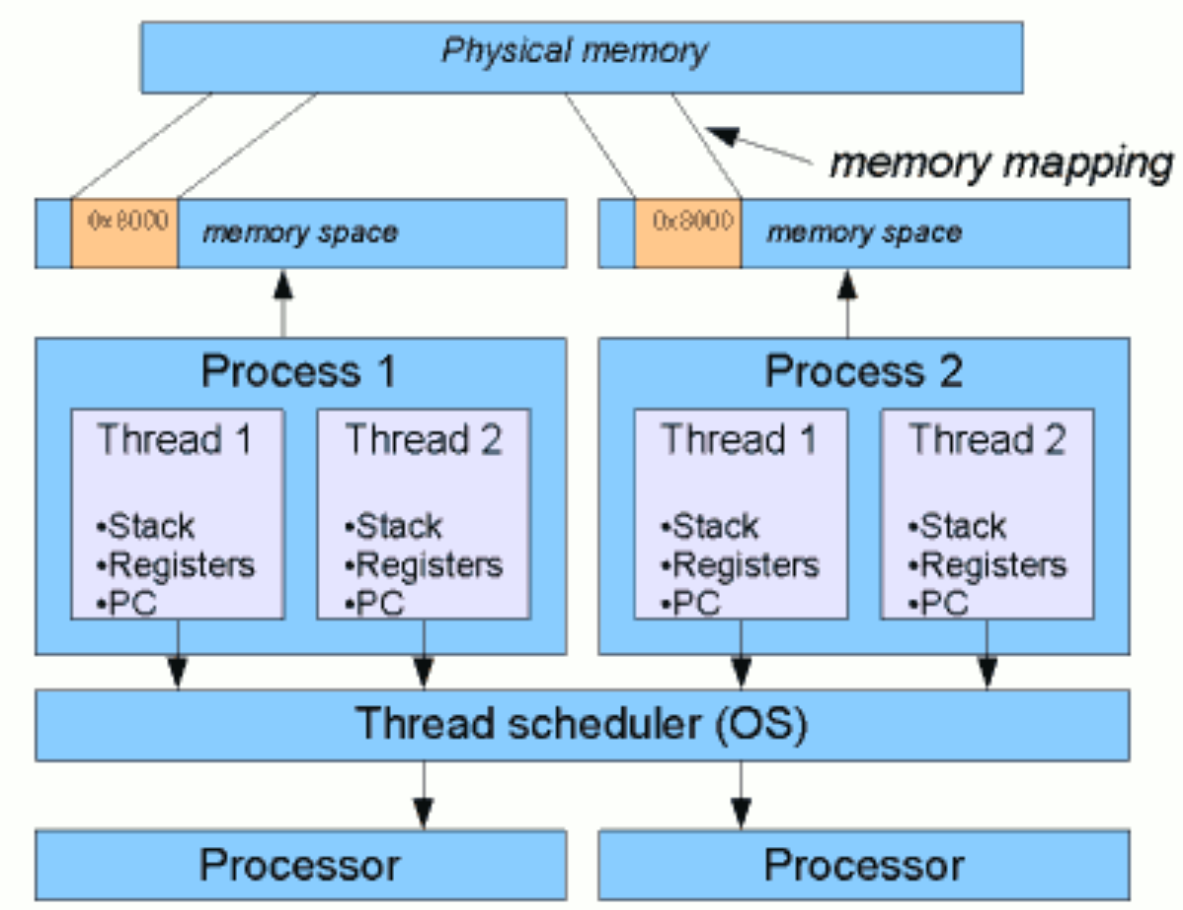
Ограничения

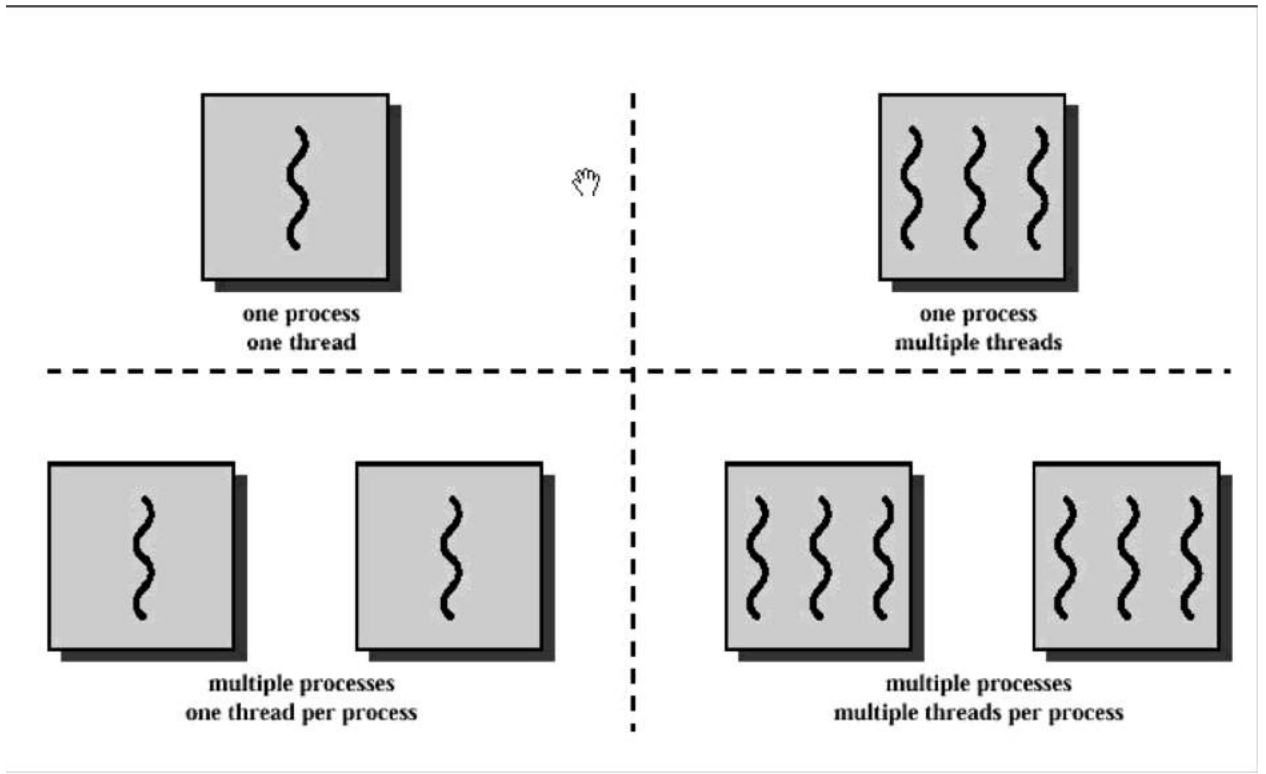
1. Как сделать private или protected поля в питоне?

Ограничения

1. Как сделать private или protected поля в питоне?
2. А как всё равно к ним обратиться?

Процессы и потоки





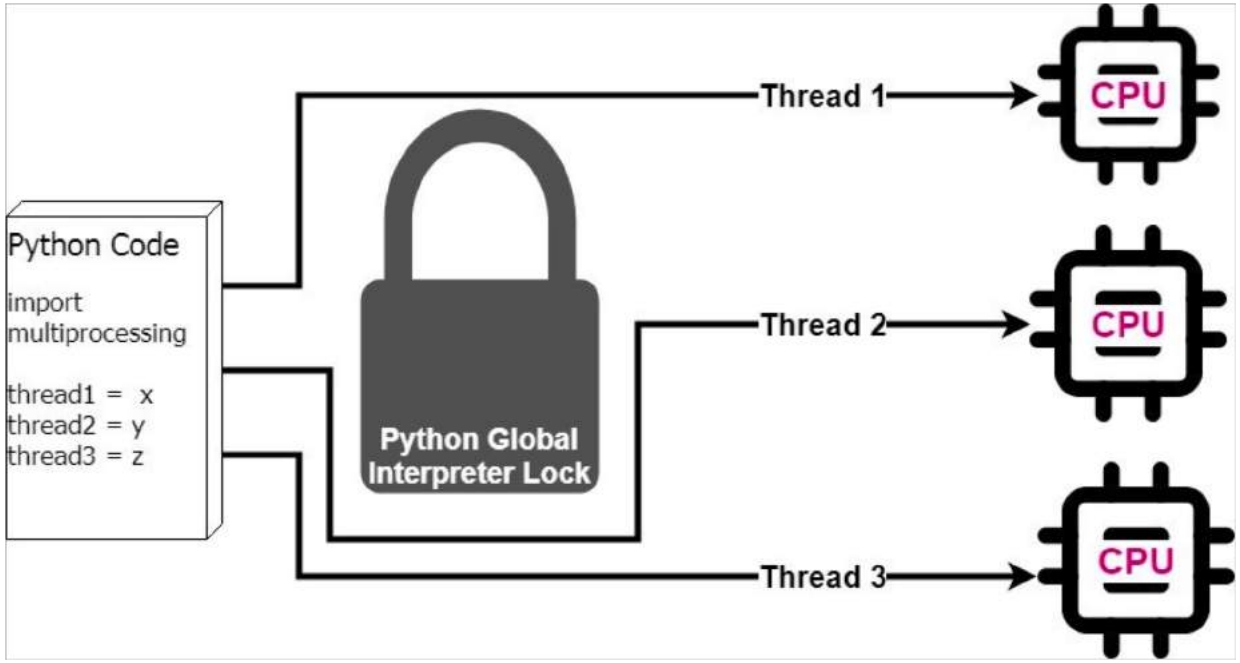
Процессы и потоки

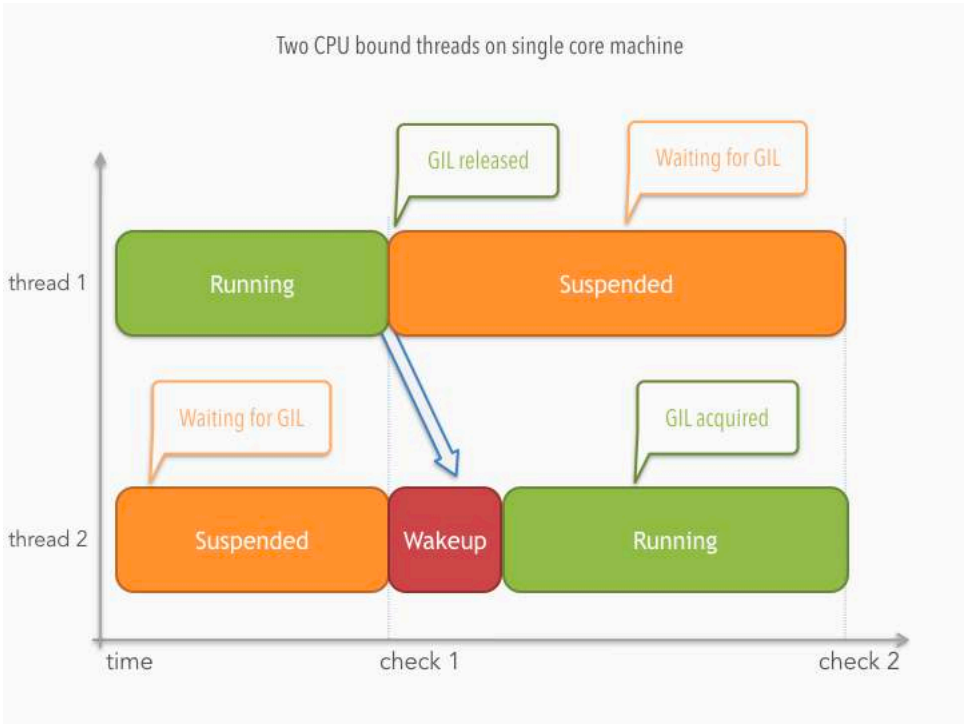
- Хорошо ли работают multiprocessing в python?
- Хорошо ли работают multithreading в python?

Процессы и потоки

- Хорошо ли работают multiprocessing в python?
- Хорошо ли работают multithreading в python?
- Почему?

Процессы и потоки: GIL





Процессы и потоки

- Хорошо ли работают multiprocessing в python?
- Хорошо ли работают multithreading в python?
- Почему?
- А когда работает?

Процессы и потоки

- Хорошо ли работают multiprocessing в python?
- Хорошо ли работают multithreading в python?
- Почему?
- А когда работает?
- А что делать?

Если коротко

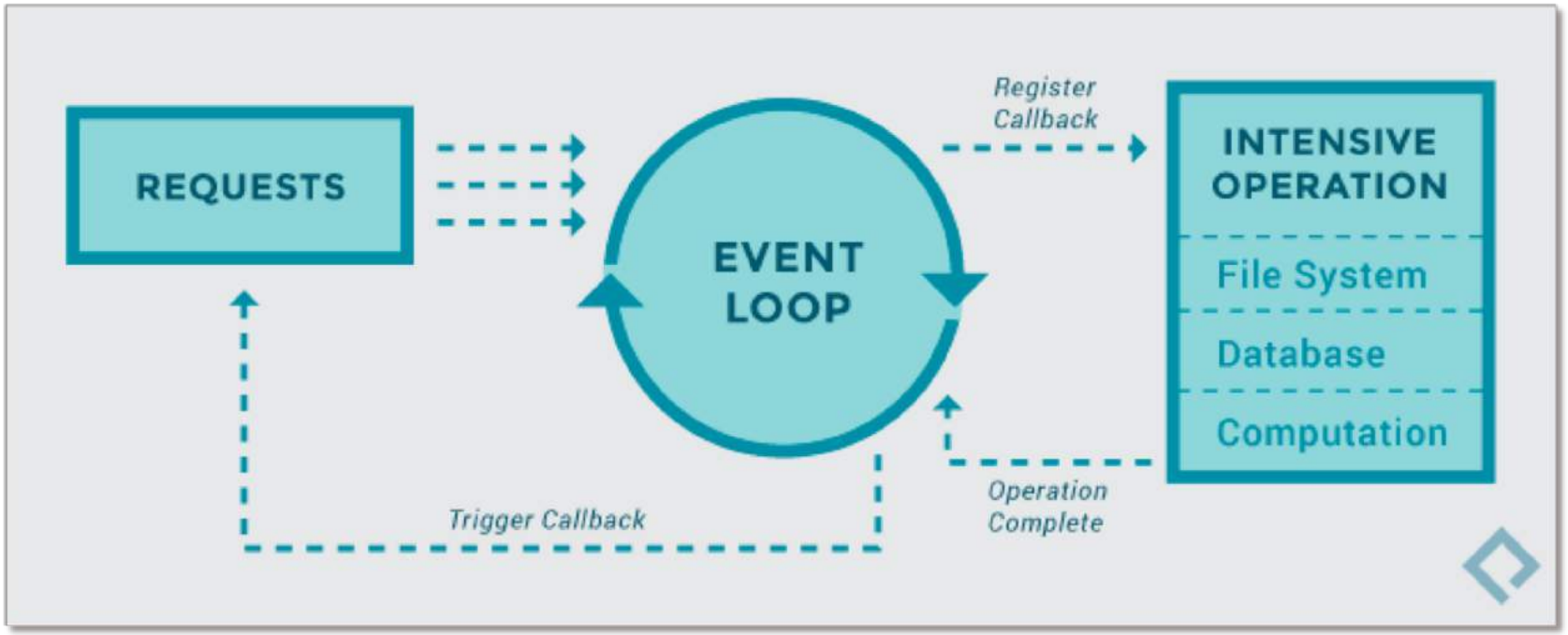
1. Нужно знать особенности в поведении языка
2. Нужно знать ограничения языка
3. Нужно знать новые фишки языка

Аннотации

```
def init(self, int = 3) -> None:
        """Construct semantic analyzer.
        """
        self.pyversion = pyversion

def visit_file(self, file_node: MypyFile, fnam: str = 3) -> int:
        self.errors.set_file(fnam)
        self.errors.set_ignored_lines(file_node.ignored_lines)
        self.cur_mod_node = file_node
        yield from expr
```

Asyncio


REQUESTS
EVENT LOOP
Register Callback
INTENSIVE OPERATION
File System
Database
Computation
Operation Complete
Trigger Callback

Asyncio

```
async def myCoroutine(future):
    magic_number = 150
    future.set_result(magic_number)

# get default event loop
loop = asyncio.get_event_loop()

future = asyncio.Future()
asyncio.ensure_future(myCoroutine(future))
loop.run_until_complete(future)
# print result of myCoroutine
print(future.result()) # output- 150
```

6. Про тестирование

Разные виды тестирования

Разные виды тестирования

- Юнит тесты
- Функциональные тесты
- Интеграционные
- Нагрузочные

Разные виды тестирования

- Юнит тесты
- Функциональные тесты
- Интеграционные
- Нагрузочные

- В ML: проверка качества и совместимости