Microsoft Partner
Silver Learning

Python Essential

Інкапсуляція та поліморфізм

ITVDN
IT VIDEO DEVELOPERS NETWORK

Після уроку обов'язково

Повторіть цей урок у відео форматі на ITVDN.com

Перевірте, як Ви засвоїли даний матеріал на TestProvider.com

Тема

Інкапсуляція та поліморфізм

План уроку:

1. Що таке інкапсуляція?
2. Модифікатори доступу
3. Різниця між модифікаторами доступу
4. Реалізація інкапсуляції в Python
5. Сеттери та геттери
6. Що таке поліморфізм?
7. Реалізація поліморфізму у Python

Інкапсуляція

- **Інкапсуляція** - це механізм мови, який дозволяє об'єднати всі методи та атрибути всередині одного класу. Ми використовували цей механізм постійно, коли всередині класу створювали багато методів та атрибутів.
- Ще одна властивість інкапсуляції: можливість обмеження доступу до методів та змінних. Інкапсуляція робить деякі методи або атрибути доступними тільки всередині самого об'єкту, але недоступними поза об'єктом.
- **Інкапсуляція в Python** працює на рівні угоди між програмістами про те, які атрибути
- загальнодоступні, а які — внутрішні.

Інкапсуляція

Інкапсуляція – це властивість системи, що дозволяє *об'єднати* дані та методи, що працюють з ними, у класі, і *приховати* деталі реалізації.

Інкапсуляція забезпечується наступними засобами:

- контроль доступу
- методи доступу
- властивості об'єкту

Інкапсуляція в Python

- Усі атрибути за замовчуванням є **публічними**.
- Атрибути, імена яких починаються з *одного символу підкреслення* (_) говорять програмісту про те, що вони відносяться до внутрішньої реалізації класу і не повинні використовуватися ззовні, але ніяк *не захищені*.
- Атрибути, імена яких починаються, але не закінчуються *двома символами підкреслення*, вважаються **приватними**. До них застосовується механізм "**name mangling**". Він не передбачає захисту даних від зміни ззовні, тому що до них все одно можна звернутися, знаючи ім'я класу і те, як Python змінює імена приватних атрибутів, проте *дозволяє захистити їх від випадкового перевизначення у класах-нащадках.*

Використання модифікаторів доступу до Python

Існують 3 типи модифікаторів доступів в ООП Python, які визначають видимість членів класу:

публічний — public (за замовчуванням);

приватний — private з використанням __name_attr;

захищений — protected з використанням _name_attr.

Ніколи не слід робити поля відкритими — це поганий стиль. Рекомендується використовувати методи доступу для звернення до поля.

Використання модифікаторів доступу до Python

```
class Person:
    def _get_secret(self):
        print("It is my big secret!")


me = Person()
me._get_secret()  # It is my big secret!
```

```
class Person:
    def _get_secret(self):
        print("It is my big secret!")

    def __get_the_biggest_secret(self):
        print("Quieter! It is my biggets secret!")


me = Person()
me._get_secret()  # It is my big secret!
me.__get_the_biggest_secret()  # AttributeError:
```

Ніколи не слід робити поля відкритими — це поганий стиль. Рекомендується використовувати методи доступу для звернення до поля.

Доступ до приватного метода у Python

```
class Person:
    def _get_secret(self):
        print("It is my big secret!")

    def __get_the_biggest_secret(self):
        print("Quieter! It is my biggets secret!")


me = Person()
me._Person__get_the_biggest_secret()
```

Ніколи не слід робити поля відкритими — це поганий стиль. Рекомендується використовувати методи доступу для звернення до поля.

Доступ до приватного метода у Python

```
class Test:
    def __init__(self, a):
        self.a = a

    def __test(self):
        self.a = 5
        print(self.a)

    def setTest(self, c):
        self.__test(c)

    def getTest(self):
        return self.__test()
```

```
t = Test(1)
# t._Test__test()
t.getTest()
```

Ніколи не слід робити поля відкритими — це поганий стиль. Рекомендується використовувати методи доступу для звернення до поля.

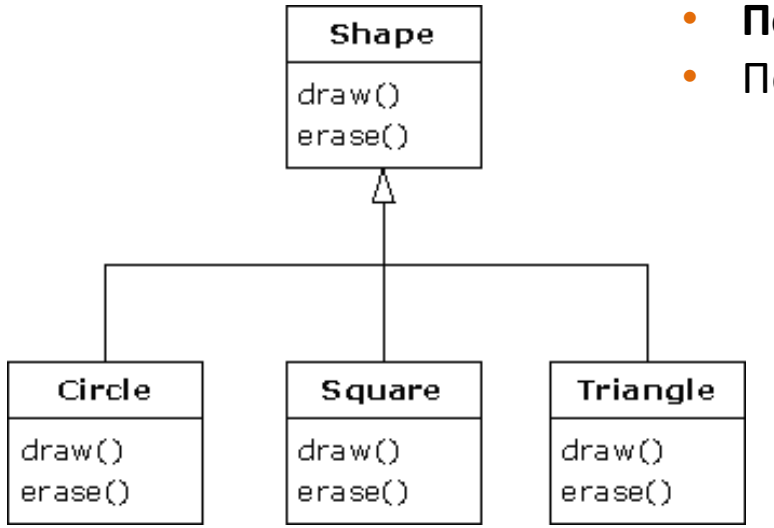
Доступ до приватного метода у Python

```
class Test:
    def __init__(self, test_value):
        self.__private_attr = test_value

    def get_private_attr(self):
        return self.__private_attr

    @staticmethod
    def __private_function():
        print("I'm private")

    def call_private(self):
        self.__private_function()
```

```
test = Test(test_value=15)
print(test.get_private_attr())
test.call_private()
```

Ніколи не слід робити поля відкритими — це поганий стиль. Рекомендується використовувати методи доступу для звернення до поля.

Поліморфізм

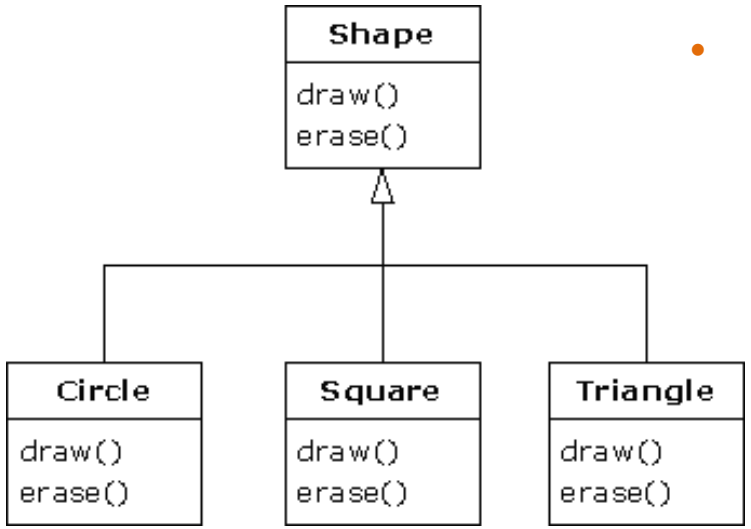


- **Поліморфізм** – це здатність однаковим чином обробляти дані різних типів.
- Поліморфізм є фундаментальною властивістю системи типів

- статична неполіморфна типізація
- статична поліморфна типізація
- *динамічна типізація*

- спеціальний поліморфізм
- параметричний поліморфізм
- *поліморфізм підтипів*
  (поліморфізм включень)

Поліморфізм



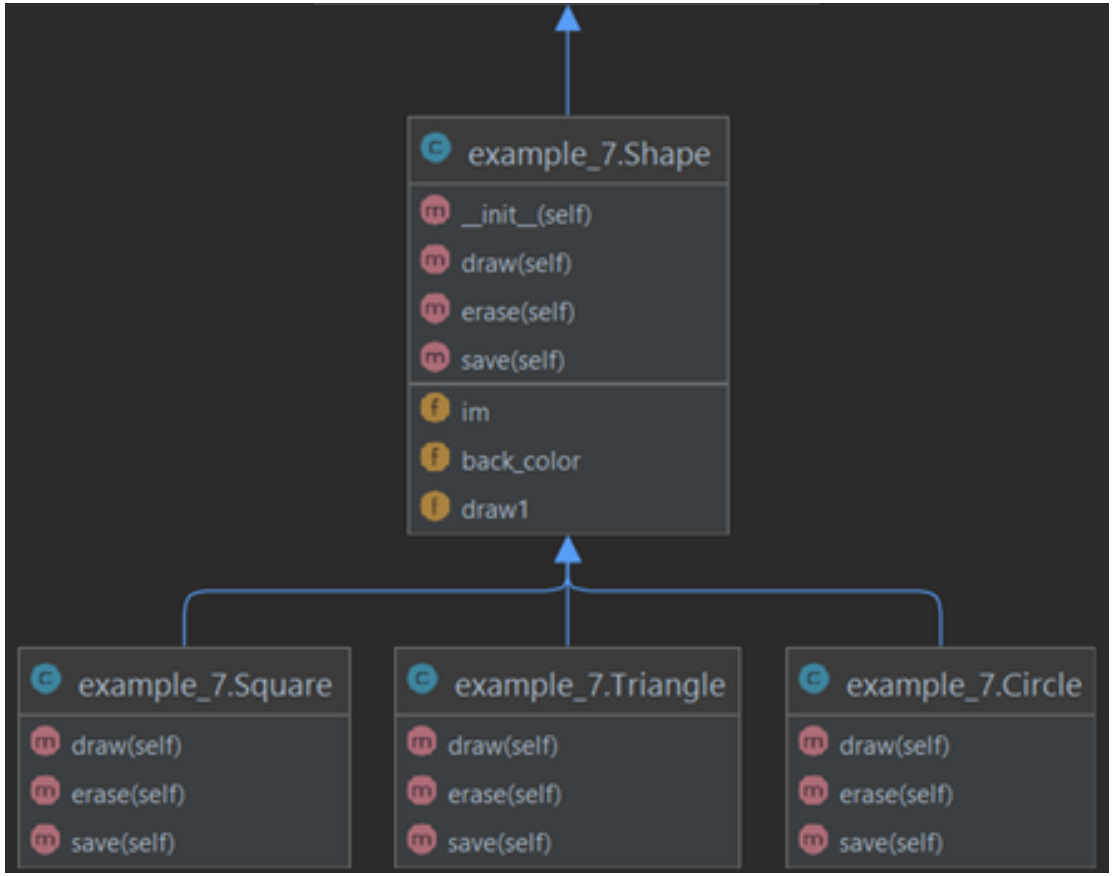
- **Поліморфізм** – це здатність однаковим чином обробляти дані різних типів.
- Поліморфізм є фундаментальною властивістю системи типів

- статична неполіморфна типізація
- статична поліморфна типізація
- *динамічна типізація*

- спеціальний поліморфізм
- параметричний поліморфізм
- *поліморфізм підтипів*
  (поліморфізм включень)

Поліморфізм

Діаграма класів проекту:



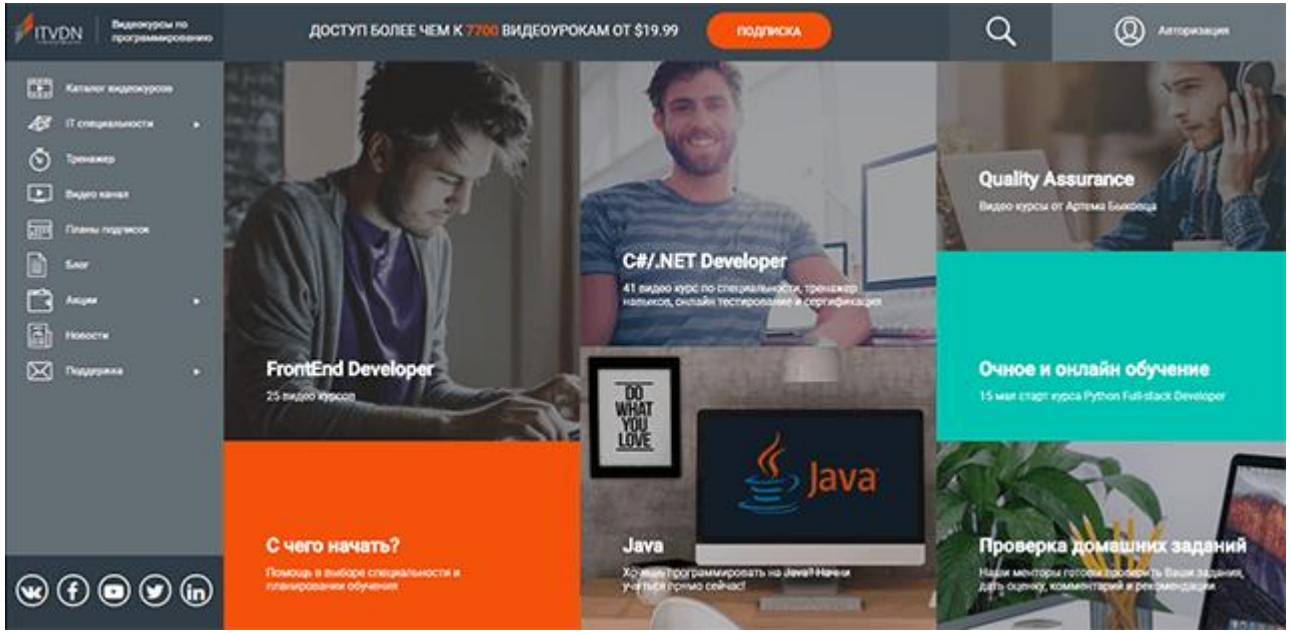
Качина типізація

- **Неявна типізація, латентна типізація** або **качина типізація** (англ. Duck typing) - вид динамічної типізації, при якій межі використання об'єкта визначаються його поточним набором методів і властивостей, на відміну від наслідування від певного класу. Тобто вважається, що об'єкт реалізує **інтерфейс**, якщо він містить усі методи цього інтерфейсу, незалежно від зв'язків в ієрархії наслідування та приналежності до якогось конкретного класу.
- Назва терміну походить від англійської «duck test» («качиний тест»), яка в оригіналі звучить як:

«If it looks like a duck, swims like a duck and quacks like a duck, then it probably is a duck».
(«Якщо це виглядає як качка, плаває як качка і крякає як качка, то, ймовірно, це і є качка».).

Дивіться наші уроки у відео форматі

ITVDN.com



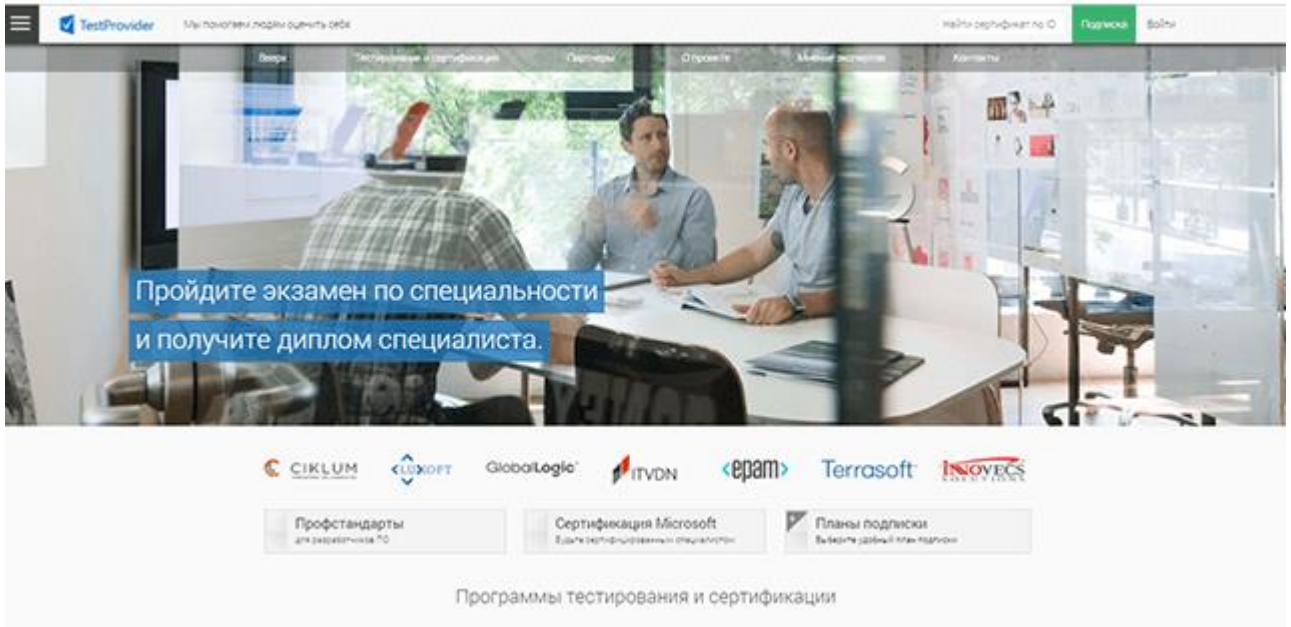
Перегляньте цей урок у відео форматі на освітньому порталі ITVDN.com
для закріплення пройденого матеріалу.

Курси записані сертифікованими тренерами, які працюють у навчальному центрі CyberBionic Systematics, та іншими висококваліфікованими розробниками.

ITVDN
IT VIDEO DEVELOPERS NETWORK

Перевірка знань

TestProvider.com



TestProvider – це online сервіс перевірки знань з інформаційних технологій. За його допомогою Ви можете оцінити Ваш рівень та виявити слабкі місця. Він буде корисним як у процесі вивчення технології, так і для загальної оцінки знань IT-спеціаліста.

Після кожного уроку проходьте тестування для перевірки знань на TestProvider.com

Успішне проходження фінального тестування дозволить Вам отримати відповідний Сертифікат.

TestProvider

Q&A

Інформаційний відеосервіс для розробників програмного забезпечення