Об'єктно-орієнтовне моделювання та проектування складних систем

Лекція 1 - ООП як спосіб мислення

Ніколаєнко Дмитро Володимирович

Зміст

- Об’єктно-орієнтовний підхід
- Основні властивості ооп
- Інші фундаментальні поняття

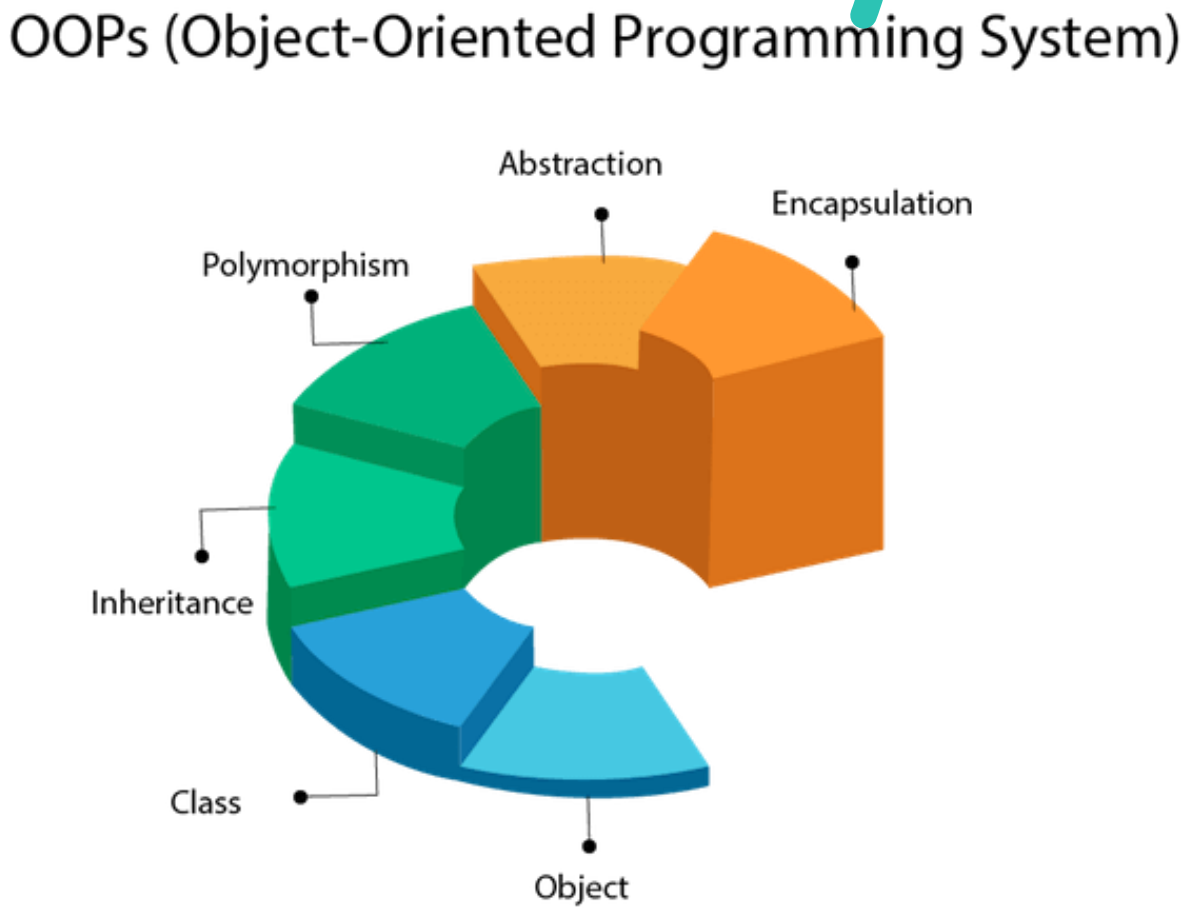
Вступ

Основна причина виникнення графічних мов моделювання полягає в тому, що мови програмування не забезпечують належний рівень абстракції, здатний полегшити процес моделювання.

Об’єктно-орієнтовний підхід

Об’єктно-орієнтовний підхід

- Все є об'єктами.
- Всі дії та розрахунки виконуються шляхом взаємодії (обміну даними) між об'єктами, під час якої один об'єкт потребує, щоб інший об'єкт виконав деяку дію.
- Об'єкти взаємодіють, надсилаючи й отримуючи повідомлення. Повідомлення – це запит на виконання дії, доповнений набором аргументів, які можуть знадобитися під час виконання дії.
- Кожен об'єкт має незалежну пам'ять, яка складається з інших об'єктів.
- Кожен об'єкт є представником (екземпляром, примірником) класу, який виражає загальні властивості об'єктів.
- У класі задається поведінка (функціональність) об'єкта. Таким чином усі об'єкти, які є екземплярами одного класу, можуть виконувати одні й ті ж самі дії.
- Класи організовані у єдину деревоподібну структуру з загальним корінням, яка називається ієрархією успадкування. Пам'ять та поведінка, зв'язані з екземплярами деякого класу, автоматично доступні будь-якому класу, розташованому нижче в ієрархічному дереві.



Основні властивості ооп

Основні властивості ооп

1. Абстрагування
2. Інкапсуляція
3. Поліморфізм
4. Спадкування

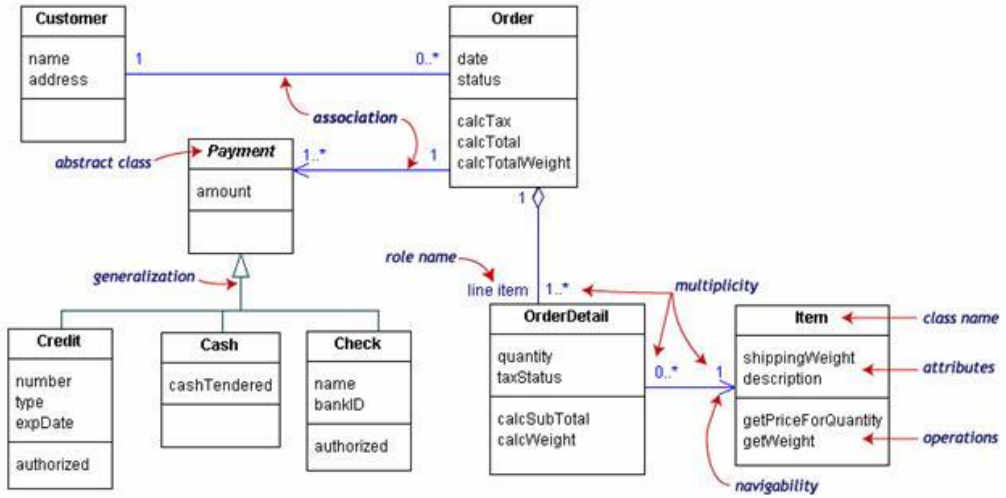
Абстрагування

Абстрагування – це спосіб бачення всього навкруги себе. Це спосіб мислення.

Це спрощення складної дійсності шляхом моделювання класів, що відповідають проблемі, та використання найприйнятнішого рівня деталізації окремих аспектів проблеми.

Абстрагування

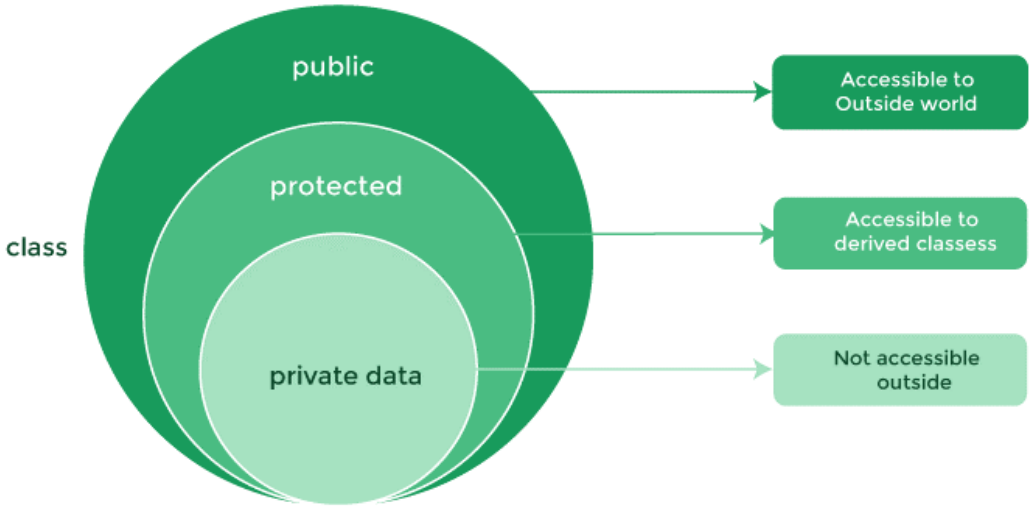
- Собака – це об’єкт
- Машина – це об’єкт
- …
- Користувач – це об’єкт
- Заказ – це об’єкт
- Елемент заказу – це об’єкт
- Продукт – це об’єкт
- …
- Касовий чек – це об’єкт
- …
- Всесвіт – це об’єкт



Інкапсуляція

- **Інкапсуляція** – можливість приховати внутрішні деталі під час опису загального інтерфейсу.

- Ключові слова **private, public, protected** визначають рівень доступу для забезпечення інкапсуляції (на рівні об’єктів).



Інкапсуляція

Реальний приклад:

Як водій ви знаєте як запустити машину повертаючи ключ в замку запалювання, проте процес запуску авто прихований від вас і ви не розумієте і не повинні розуміти як саме це відбувається.

Як у водія у вас є можливість викликати метод запуску двигуна передаючи йому правильний ключ.

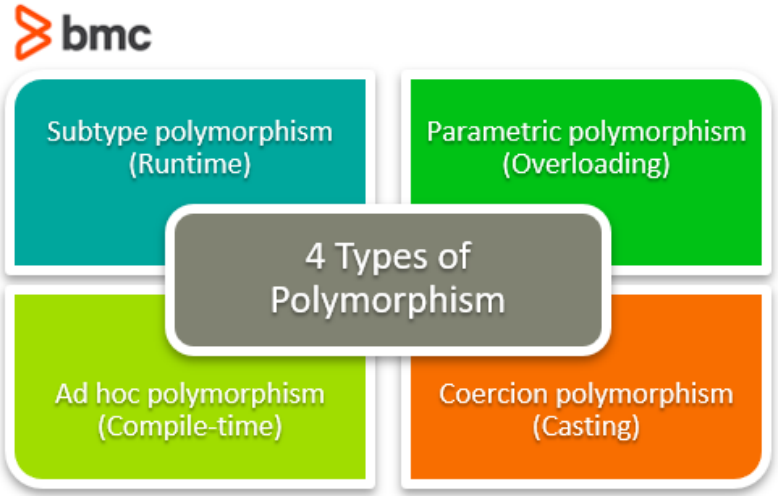
Все!

```
public class Account
{
  private decimal _accountBalance = 500.00m;

  public decimal CheckBalance()
  {
    return _accountBalance;
  }
}

static void Main()
{
    Account myAccount = new Account();
    decimal myBalance = myAccount.CheckBalance();
* This Main method can check the balance via the public
* "CheckBalance" method provided by the "Account" class
* but it cannot manipulate the value of "accountBalance"
}
```

Поліморфізм

- **Поліморфізм** в програмуванні - реалізація ряду підпрограм (функцій, процедур, методів тощо) з однаковими іменами, але з різними параметрами. Залежно від того, що передається, і вибирається відповідна підпрограма.

- **Поліморфізм в ООП** - можливість використання об'єктів (екземплярів) підкласів взамін батьківського класу.



Поліморфізм

Реалізація ряду підпрограм (функцій, процедур, методів тощо) з однаковими іменами, але з різними параметрами. Залежно від того, що передається, і вибирається відповідна підпрограма.

```
function Add(x, y : Integer) : Integer;

begin
  Add := x + y
end;

function Add(s, t : String) : String;
begin
  Add := Concat(s, t)
end;

begin
  (Add(1, 2)); (* Prints "3" *)
  (Add('Hello, ', 'Mammals!')); (* Prints "Hello, Mammals!" *)
end.
```

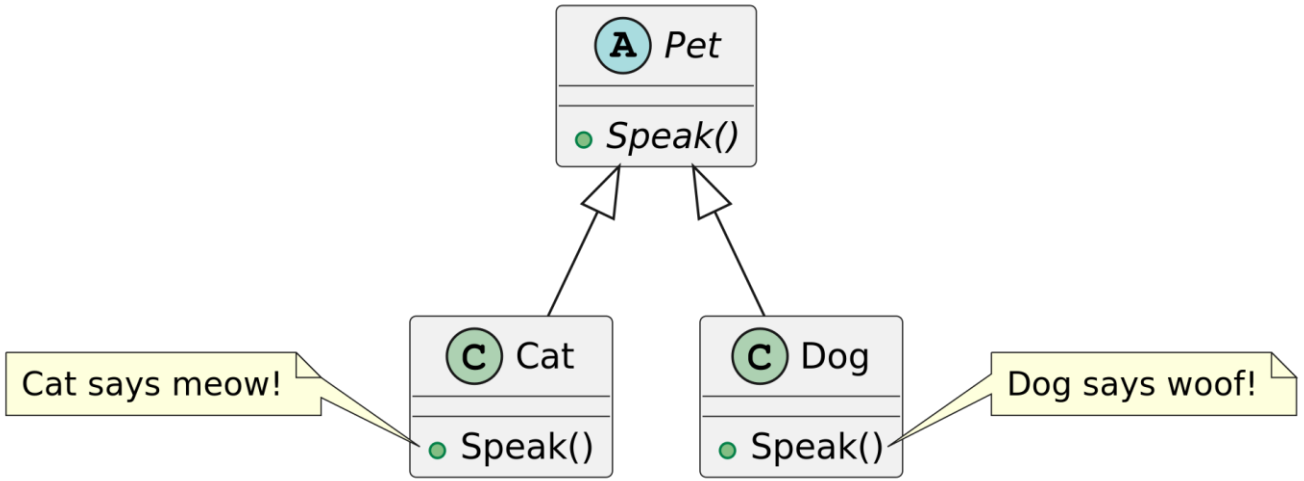
Поліморфізм

Реалізація ряду підпрограм (функцій, процедур, методів тощо) з однаковими іменами, але з різними параметрами. Залежно від того, що передається, і вибирається відповідна підпрограма.

```
function Add(x, y, z : Integer) : Integer;

begin
  Add := x + y + z
end;

function Add(x, y : Integer) : Integer;
begin
  Add := x + y
end;

begin
  (Add(1, 2)); (* Prints “3” *)
  (Add(1, 2, 3)); (* Prints “6” *)
end.
```

Поліморфізм

Можливість використання об'єктів (екземплярів) субкласів взамін суперкласів класу.

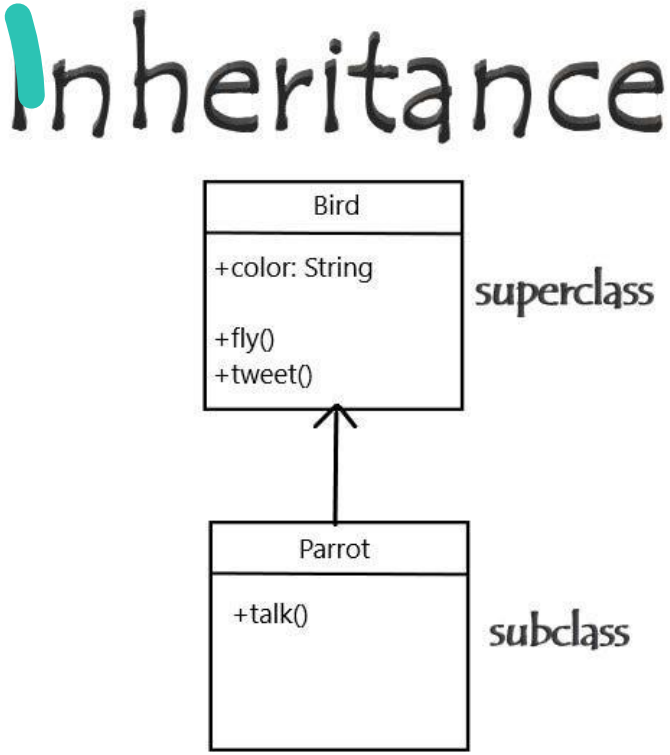



```
abstract class Pet {
  abstract String speak();
}

class Cat extends Pet {
  String speak() {
    return "Meow!";
  }
}

class Dog extends Pet {
  String speak() {
    return "Woof!";
  }
}

static void letsHear(final Pet pet) {
  println(pet.speak());
}

static void main(String[] args) {
  letsHear(new Cat());
  letsHear(new Dog());
}
```

Спадкування

Одна з найголовніших особливостей об’єктно-орієнтовного програмування.

Замість визначення функціоналу класу він спадкується від суперкласу, розширюється, змінюється та доповнюється як того потребує субклас клас.

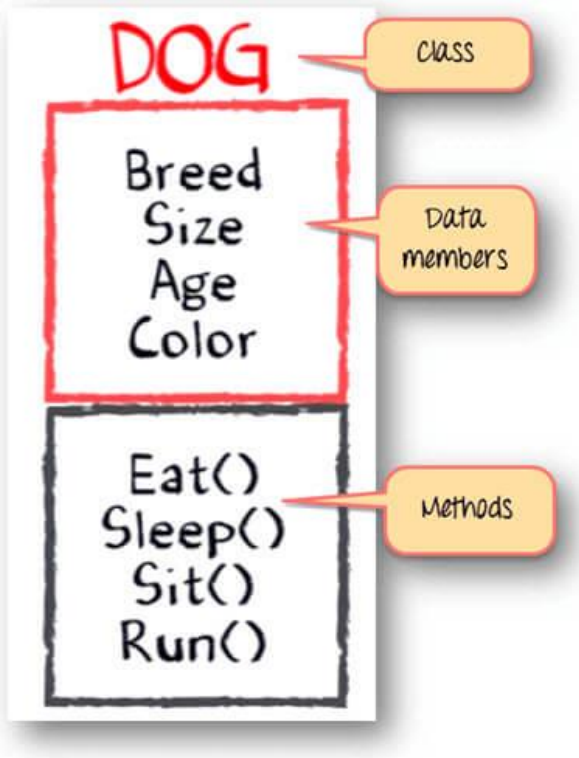


Інші фундаментальні поняття

Клас

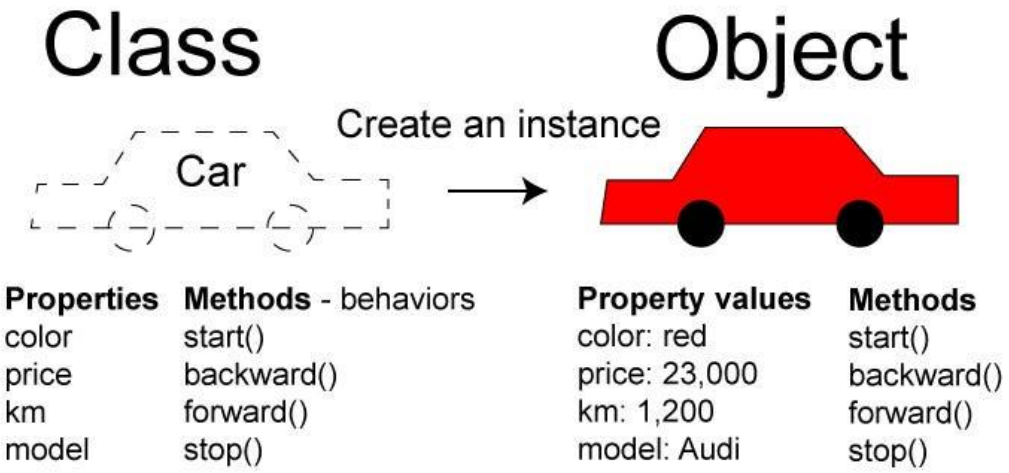
- Клас – це абстрактна сутність.
- Клас – не існує.
- Клас – це лише уява розробника.

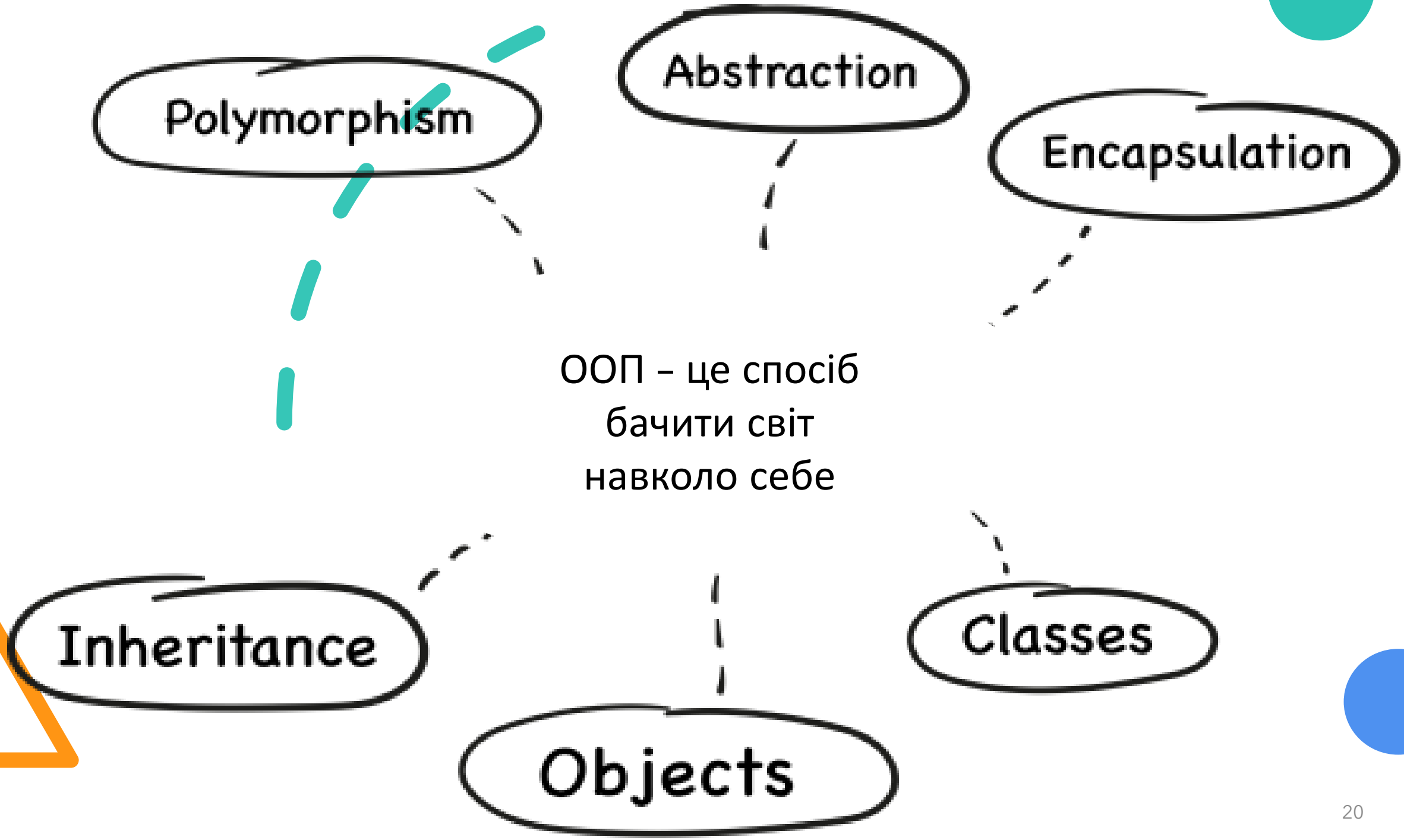
- Клас визначає абстрактні характеристики деякої сутності:
- Атрибути (властивості)
- Методи (поведінка)
- Події (повідомлення)



Об'єкт

Об’єкт – це
окремий *екземпляр* класу який створюється після запуску програми та ініціалізації полів класу.




Polymorphism
Abstraction
Encapsulation
ООП – це спосіб
бачити світ
навколо себе
Inheritance
Classes
Objects

Дякую

Ніколаєнко Дмитро
Володимирович