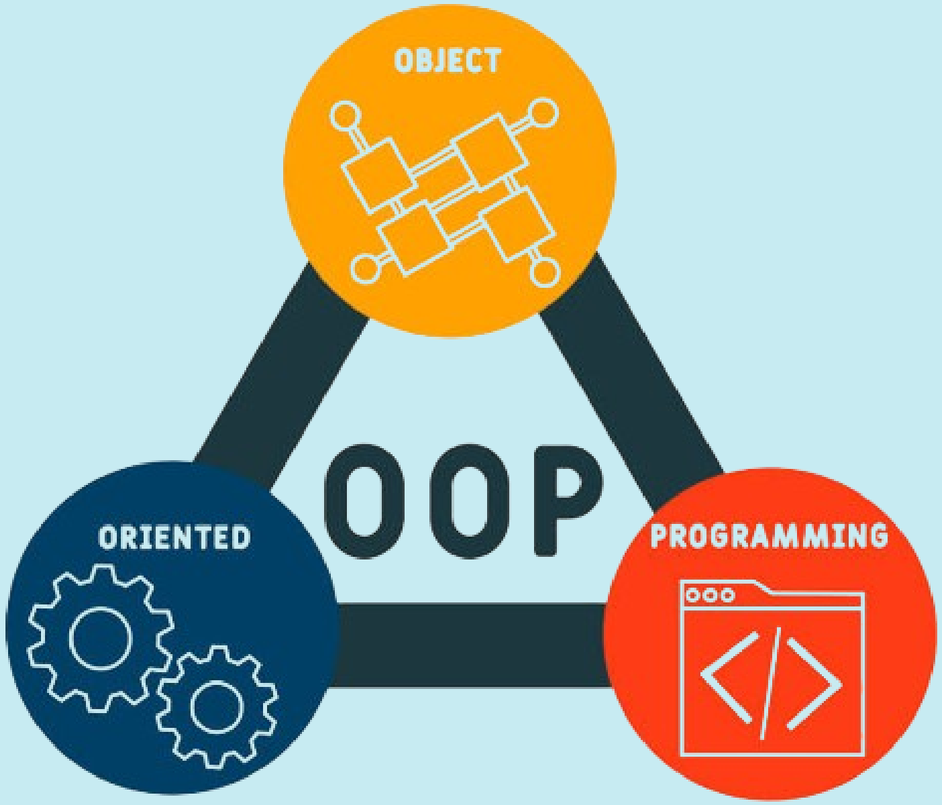
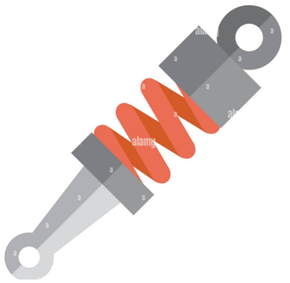
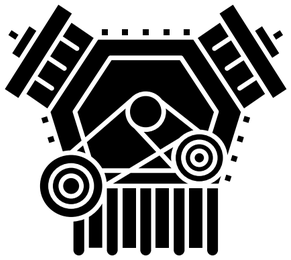
KELAS OTOMESYEN



Object Oriented Programming

KELAS
OTOMESYEN





Project

Component -> Object

KELAS OTOMESYEN

```python
# Tanpa OOP

# Buah pertama
apple_name = "Apel"
apple_color = "Merah"
apple_taste = "Manisss"

# Buah kedua
orange_name = "Jeruk"
orange_color = "Kuning"
orange_taste = "Asemm kek muka"

# Fungsi untuk menampilkan informasi buah
def display_fruit_info(name, color, taste):
    print(f"{name}: Color - {color}, Taste - {taste}")

# Memanggil fungsi untuk menampilkan informasi buah
display_fruit_info(apple_name, apple_color, apple_taste)
display_fruit_info(orange_name, orange_color, orange_taste)
```

```python
# Dengan OOP

class Fruit:
    def __init__(self, name, color, taste):
        self.name = name
        self.color = color
        self.taste = taste

    def display_info(self):
        print(f"{self.name}: Color - {self.color}, Taste - {self.taste}")

# Buah pertama
apple = Fruit("Apel", "Merah", "manisss")

# Buah kedua
orange = Fruit("Jeruk", "Kuning", "Asyeemmm")

# Memanggil metode untuk menampilkan informasi buah
apple.display_info()
orange.display_info()
```

KELAS OTOMESYEN

Class

Attribute

Method

Object

```python
class Tire:
    def __init__(self, brand, model):
        self.brand = brand
        self.model = model

    def display_info(self):
        print(f"{self.brand} {self.model}")

# Membuat objek dari class Tire
ban_depan = Tire("Bridgestone", "Techno 185/65 R15")
ban_depan.display_info()
```

KELAS OTOMESYEN

__init__

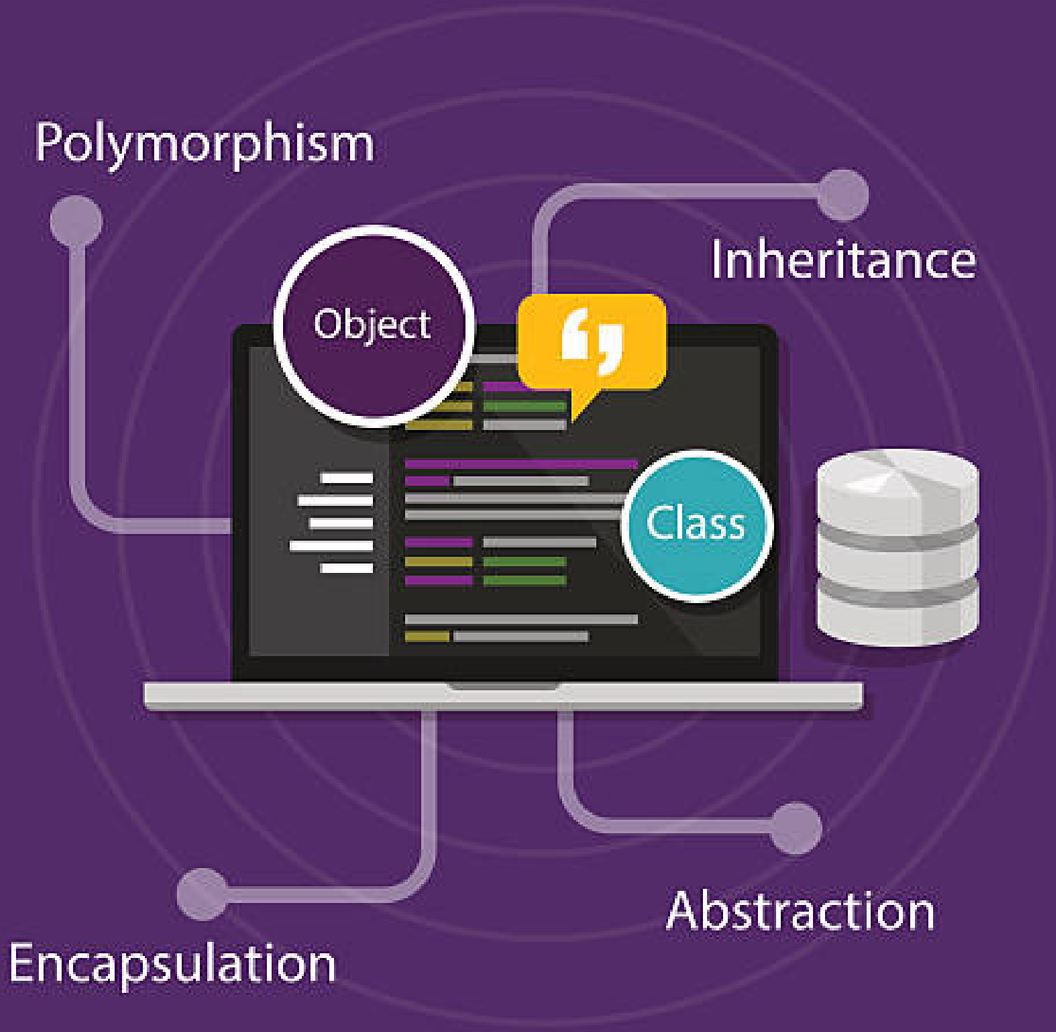
Constructor

Attribute

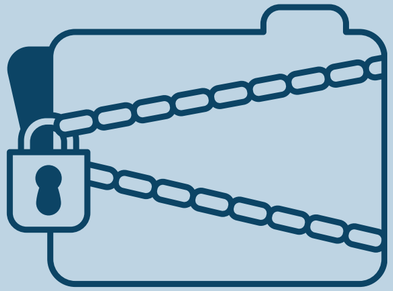
```python
class Tire:
    def __init__(self, brand, model):
        self.brand = brand
        self.model = model

    def display_info(self):
        print(f"{self.brand} {self.model}")

# Membuat objek dari class Tire
ban_depan = Tire("Bridgestone", "Techno 185/65 R15")
ban_depan.display_info()
```

KELAS OTOMESYEN

Prinsip OOP



KELAS OTOMESYEN



Encapsulation

Menjaga Rahasia di dalam Class

```python
class Mobil:
    def __init__(self, model, warna):
        self.__model = model  # Variabel privat dengan awalan __ (double underscore)
        self.__warna = warna

    def get_model(self):
        return self.__model

    def set_model(self, model):
        self.__model = model

    def get_warna(self):
        return self.__warna

    def set_warna(self, warna):
        self.__warna = warna

    def display_info(self):
        print(f"Model: {self.__model}, Warna: {self.__warna}")


# Membuat objek Mobil
mobil1 = Mobil(model="Sedan", warna="Merah")

# Mengakses dan menampilkan informasi menggunakan metode getter
print(f"Model: {mobil1.get_model()}")
print(f"Warna: {mobil1.get_warna()}")

# Mengubah informasi menggunakan metode setter
mobil1.set_model("SUV")
mobil1.set_warna("Biru")

# Menampilkan informasi setelah perubahan
mobil1.display_info()
```

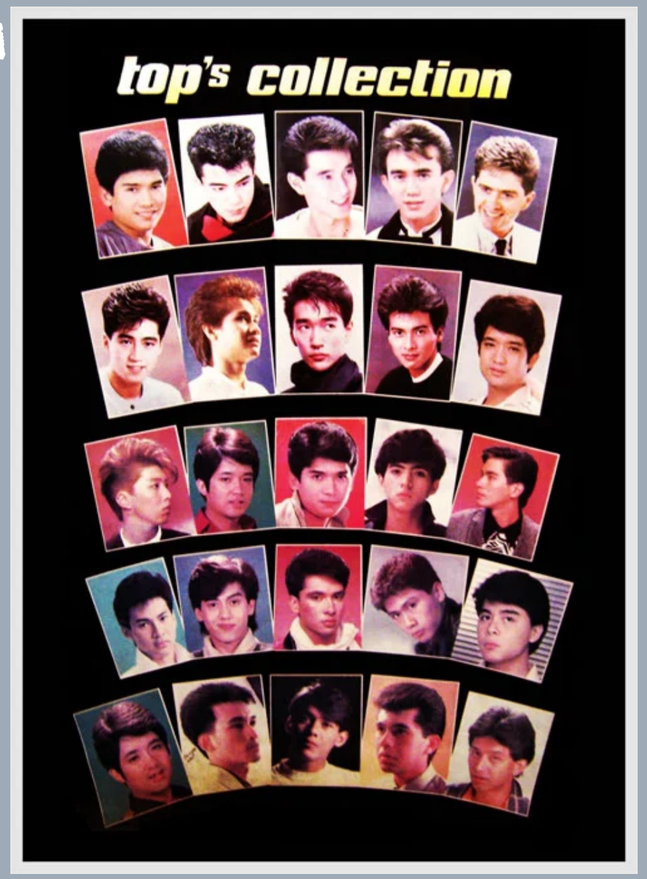
KELAS OTOMESYEN

Inheritance

Pewarisan antar Class

```python
class Animal:
    def __init__(self, name):
        self.name = name

    def speak(self):
        pass  # Metode ini akan diimplementasikan di class anakan

class Dog(Animal):
    def speak(self):
        return f"{self.name} says Woof!"

class Cat(Animal):
    def speak(self):
        return f"{self.name} says Meow!"

# Contoh penggunaan
dog = Dog("Blacky")
cat = Cat("Snowy")
print(dog.speak())
print(cat.speak())
```

KELAS OTOMESYEN



Polymorphism

Banyak Bentuk, Satu Nama

```python
class Bird:
    def speak(self):
        return "Tweet tweet!"

class Dog:
    def speak(self):
        return "Woof!"

class Cat:
    def speak(self):
        return "Meow!"

# Fungsi yang menerima objek apapun dengan function speak
def animal_sound(animal):
    print(animal.speak())

# Contoh penggunaan
bird = Bird()
dog = Dog()
cat = Cat()

animal_sound(bird)
animal_sound(dog)
animal_sound(cat)
```

KELAS OTOMESYEN

Abstraction

Menyembunyikan detail dan menampilkan fungsionalitas yang penting.