KU LEUVEN

Object Gerichte Software Ontwikkelling

Dr. Ing. Greet Baldewijns

Overerving

KU LEUVEN

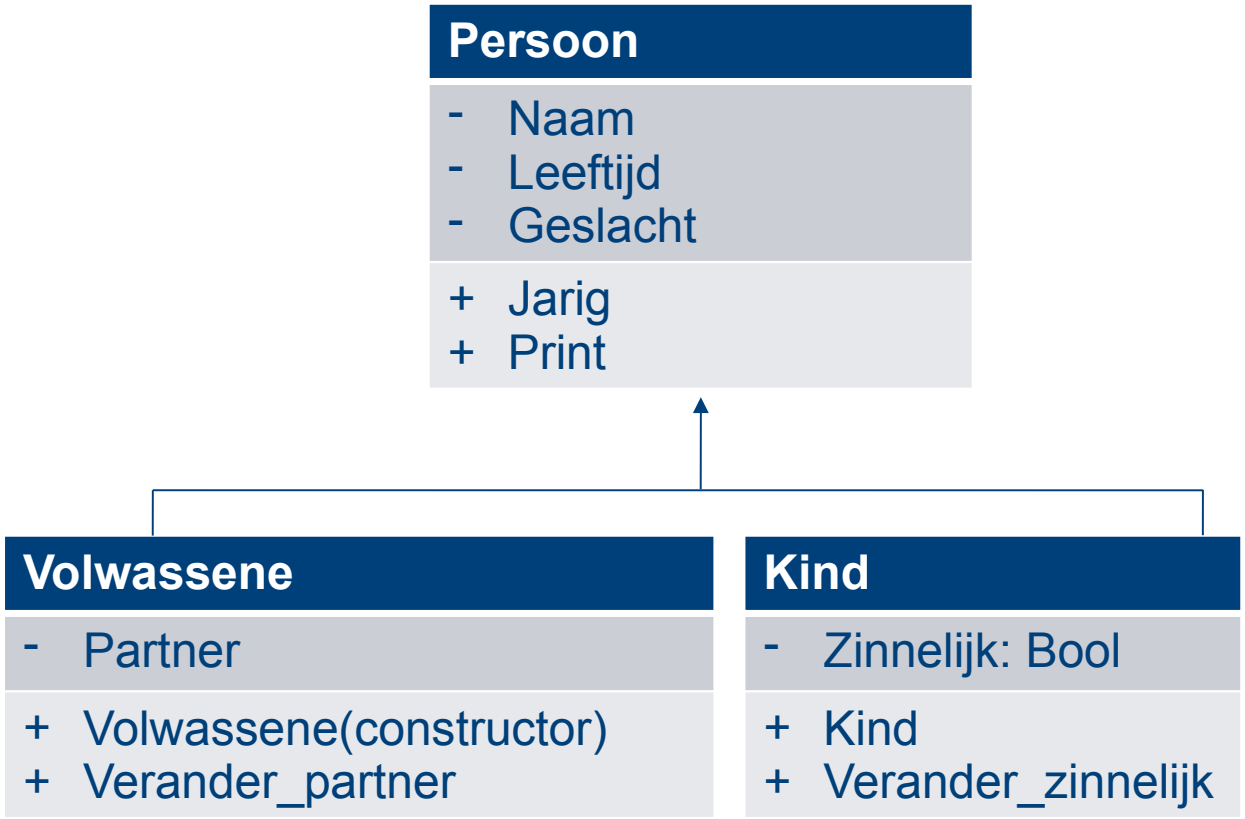
Generalisatie en specialisatie

Volwassene
- Naam - Leeftijd - Geslacht - Partner
+ Volwassene (constructor) + Jarig + Print + Verander_partner

Kind
- Naam - Leeftijd - Geslacht: String - Zinnelijk: Bool
+ Kind (constructor) + Jarig + Print + Verander_zinnelijk

Generalisatie en specialisatie

Generalisatie
Superklasse



Specialisatie
Subklasse

Super- en subklasses in python

```
class Automobile:

    def __init__(self, make, model, mileage, price):
        self.__make = make
        self.__model = model
        self.__mileage = mileage
        self.__price = price

    def set_make(self, make):
        self.__make = make

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

    def set_mileage(self, mileage):
        self.__mileage = mileage

    def set_price(self, price):
        self.__price = price

    def get_make(self):
        return self.__make

    def get_model(self):
        return self.__model

    def get_mileage(self):
        return self.__mileage

    def get_price(self):
        return self.__price
```

De klasse ‘Car’ erft de attributen en methodes van de klasse ‘Automobile’

```
class Car(Automobile):

    def __init__(self, make, model, mileage, price, doors):

        Automobile.__init__(self, make, model, mileage, price)

        self.__doors = doors

    def set_doors(self, doors):
        self.__doors = doors

    def get_doors(self):
        return self.__doors
```

Super- en subklasses in python

Naam van de module

Naam van de subklasse

```
import vehicles

def main():
    # Create an object from the Car class.
    # The car is a 2007 Audi with 12,500 miles, priced
    # at $21,500.00, and has 4 doors.
    used_car = vehicles.Car('Audi', 2007, 12500, 21500.00, 4)

    # Display the car's data.
    print('Make:', used_car.get_make())
    print('Model:', used_car.get_model())
    print('Mileage:', used_car.get_mileage())
    print('Price:', used_car.get_price())
    print('Number of doors:', used_car.get_doors())

# Call the main function.
main()
```

Methodes van super- en subklasse kunnen gebruikt worden

KU LEUVEN

Sub- en superklasses in python

```
class Truck(Automobile):

    def __init__(self, make, model, mileage, price, drive_type):
        Automobile.__init__(self, make, model, mileage, price)

        self.__drive_type = drive_type

    def set_drive_type(self, drive_type):
        self.__drive_type = drive_type

    def get_drive_type(self):
        return self.__drive_type
```

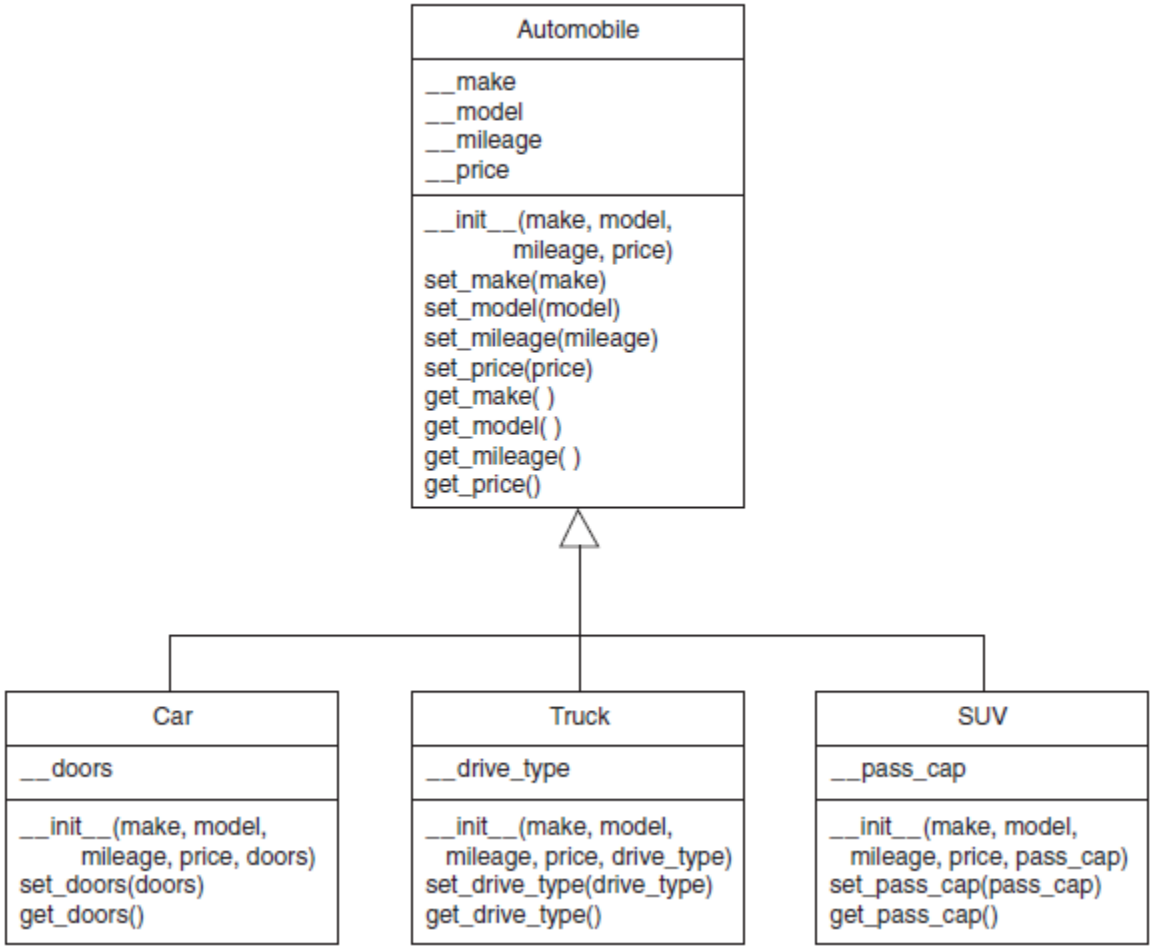
```
class SUV(Automobile):

    def __init__(self, make, model, mileage, price, pass_cap):
        Automobile.__init__(self, make, model, mileage, price)

        self.__pass_cap = pass_cap


    def set_pass_cap(self, pass_cap):
        self.__pass_cap = pass_cap


    def get_pass_cap(self):
        return self.__pass_cap
```

Sub- en superklassen in python

```
import vehicles

def main():
    car = vehicles.Car('BMW', 2001, 70000, 15000.0, 4)
    truck = vehicles.Truck('Toyota', 2002, 40000, 12000.0, '4WD')
    suv = vehicles.SUV('Volvo', 2000, 30000, 18500.0, 5)

    print('The following car is in inventory:')
    print('Make:', car.get_make())
    print('Model:', car.get_model())
    print('Mileage:', car.get_mileage())
    print('Price:', car.get_price())
    print('Number of doors:', car.get_doors())
    print()

    print('The following pickup truck is in inventory.')
    print('Make:', truck.get_make())
    print('Model:', truck.get_model())
    print('Mileage:', truck.get_mileage())
    print('Price:', truck.get_price())
    print('Drive type:', truck.get_drive_type())
    print()

    print('The following SUV is in inventory.')
    print('Make:', suv.get_make())
    print('Model:', suv.get_model())
    print('Mileage:', suv.get_mileage())
    print('Price:', suv.get_price())
    print('Passenger Capacity:', suv.get_pass_cap())

main()
```

UML voorstelling overerving

Figure 11-2 UML diagram showing inheritance



KU LEUVEN

Polymorfisme

KU LEUVEN

Polymorfisme

- Subklassen en superklassen hebben methodes met dezelfde naam.
- Het hoofdprogramma weet welke methode te gebruiken a.d.h.v. het object dat deze methode oproept.

Polymorfisme

```
class Mammal:

    def __init__(self, species):
        self.__species = species

    def show_species(self):
        print('I am a', self.__species)

    def make_sound(self):
        print('Grrrrr')
```

```
class Dog(Mammal):

    def __init__(self):
        Mammal.__init__(self, 'Dog')


    def make_sound(self):
        print('Woof! Woof!')
```

```
class Cat(Mammal):

    def __init__(self):
        Mammal.__init__(self, 'Cat')

    def make_sound(self):
        print('Meow')
```

```
import animals

def main():
    mammal = animals.Mammal('regular animal')
    dog = animals.Dog()
    cat = animals.Cat()

    # Display information about each one.
    print('Here are some animals and')
    print('the sounds they make.')
    print('--------------------------')
    mammal.show_species()
    mammal.make_sound()
    print()
    dog.show_species()
    dog.make_sound()
    print()
    cat.show_species()
    cat.make_sound()

main()
```

Als er geen methode in de subklasse is met dezelfde naam wordt de methode uit de hoofdklasse uitgevoerd

De isinstance functie

```
import animals

def main():
    mammal = animals.Mammal('regular animal')
    dog = animals.Dog()
    cat = animals.Cat()

    print('Here are some animals and')
    print('the sounds they make.')
    print('--------------------------')
    show_mammal_info(mammal)
    print()
    show_mammal_info(dog)
    print()
    show_mammal_info(cat)
    print()
    show_mammal_info('I am a string')

def show_mammal_info(creature):
    if isinstance(creature, animals.Mammal):
        creature.show_species()
        creature.make_sound()
    else:
        print('That is not a Mammal!')

main()
```

Controle of een object een instantie is van een bepaalde klasse of een subklasse van die klasse.

KU LEUVEN

Oefeningen

KU LEUVEN

Oefeningen

- Zie Toledo