Hogeschool van Arnhem en Nijmegen

OWE 3a: Python II



Geavanceerde concepten in Python en programmeren voor bio-informatica toepassingen

Opzet OWE 3a

Les	Onderwerp	Theorie	Opgaven
1	• Review Python I • Pseudocode • Flowcharts • Documenteren en Testen	H1..8 SowP*	Afvinkopdracht 1
2	• Graphs • Strings	MatPlotLib tutorial H9 More about Strings	Afvinkopdracht 2
3	• Datastructuren: • Dictionaries • Sets	H10 Dictionaries and Sets	Afvinkopdracht 3
4	• Text and Language Processing • Regular Expressions	H7 DiP**	Afvinkopdracht 4
5	• Object-Oriented Programming	H11 Classes and Object-Oriented Programming H12 Inheritance	Afvinkopdracht 5
6	• Recursion	H13 Recursion	Afvinkopdracht 6
7	• GUI Programming	H14 GUI Programming	Voorbeeld thematoets

HAN

**SowP: Starting out with Python*
***DiP Dive into Python*

Agenda

1. **Object oriëntatie**
2. **Classes**
3. **Instances**
4. **Designing**
5. **Inheritance**
6. **Polymorfisme**

Object Orientatie

- Object oriëntatie is een andere manier van programmeren
- Het benadert meer de werkelijkheid

Object Oriëntatie

- Met object oriëntatie creëer je objecten die afgeleid zijn van werkelijke objecten
- Zo bestaat een auto in een object georiënteerde programmeertaal uit een object stuur, objecten wielen enzovoort

Object oriëntatie en gaming

- Vrijwel alle games zijn object georiënteerd in opzet
- Alle grotere programma's zijn object georiënteerd

Voordelen aan object orientatie

- Is meer als de werkelijkheid
- Beter uitbreidbaar
- Beter hergebruik van code
- Gestructureerder

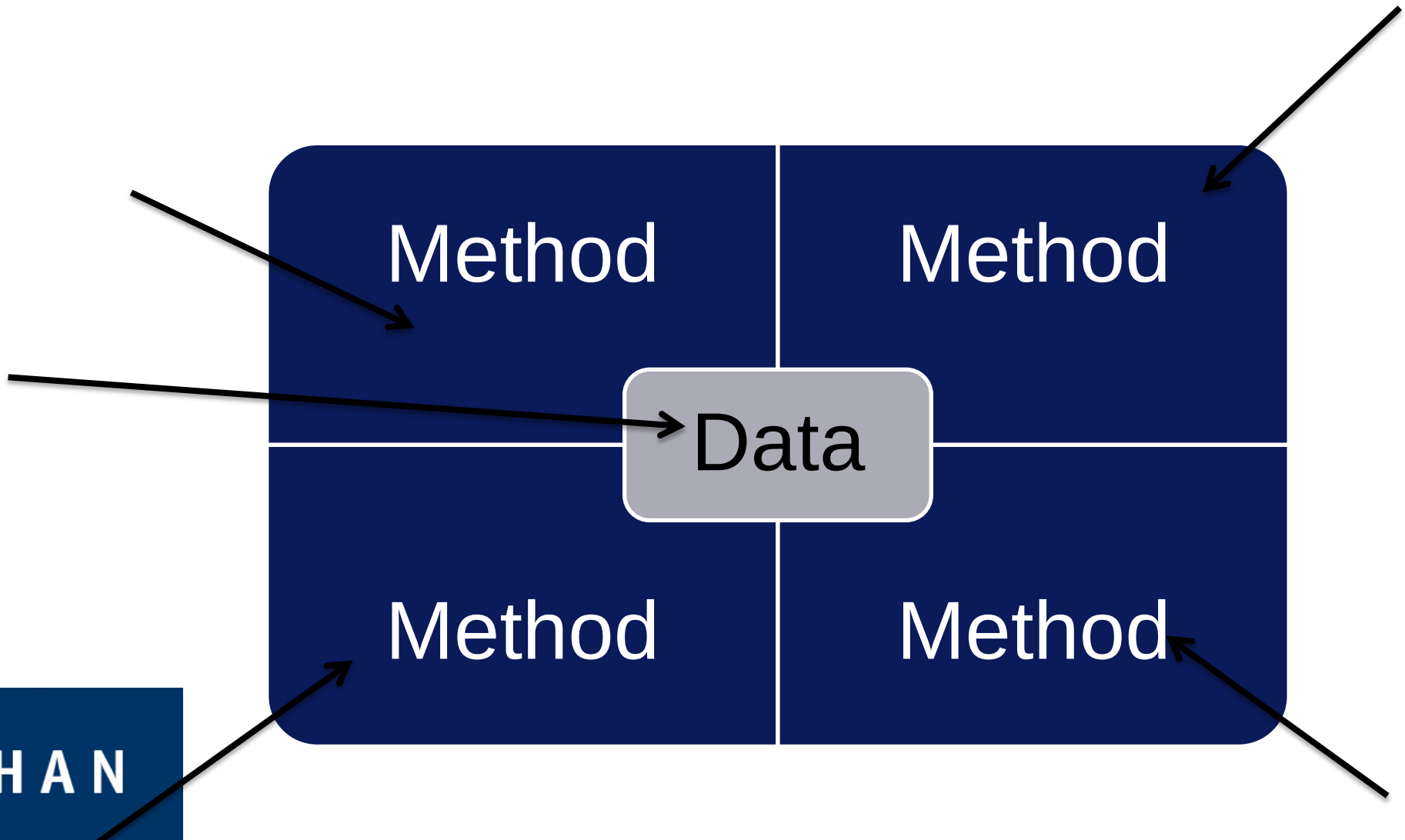
Procedureel vs. Object Georiënteerd

Procedureel	Object Georiënteerd
• Data en functies liggen los van elkaar • Functies dragen waardes over	• Data en functies horen bij elkaar • Objecten roepen elkaar aan

Kenmerken

- **De drie kenmerken van object oriëntatie**
  1. Encapsulatie
  2. Inheritance
  3. Polymorfisme

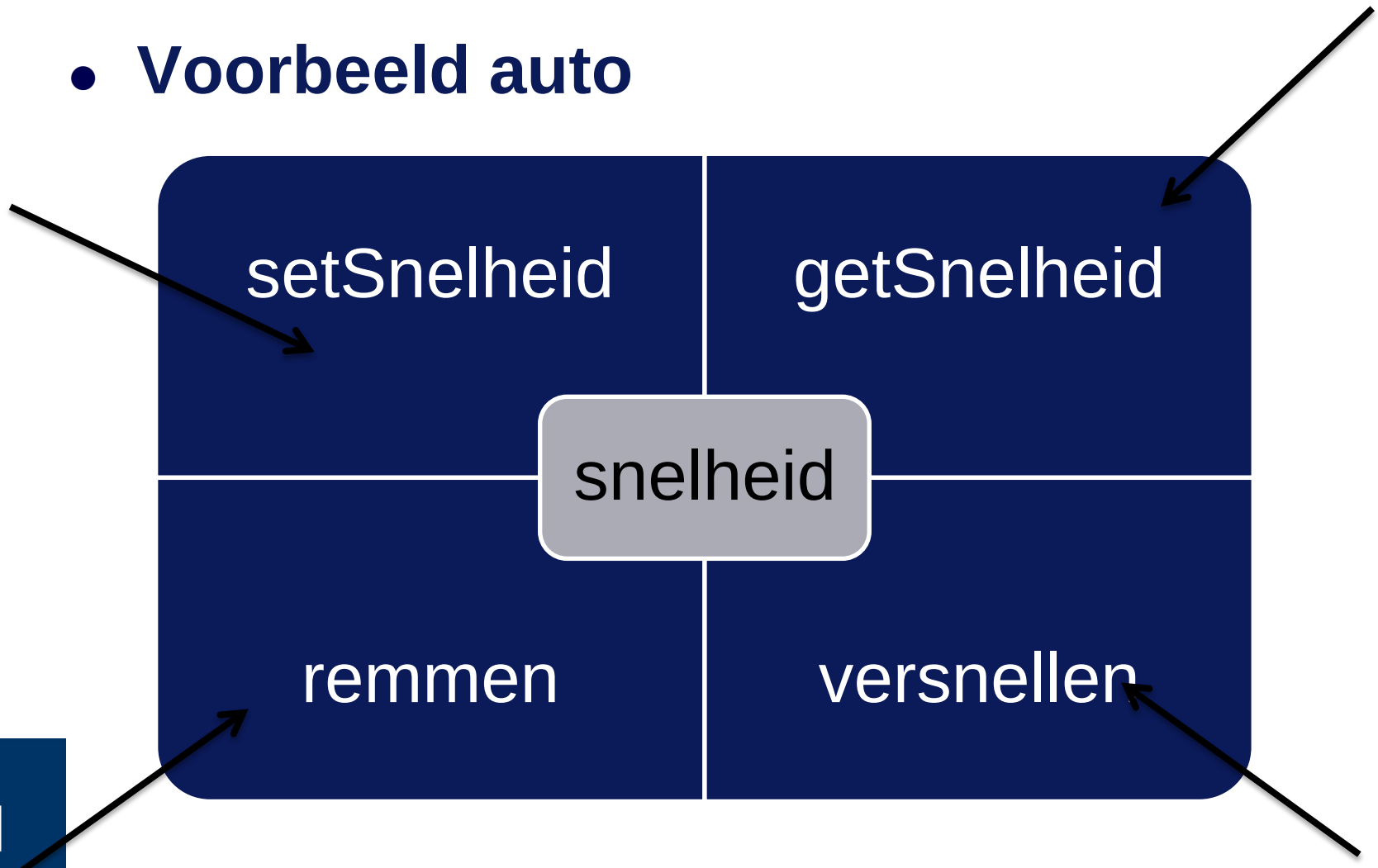
Encapsulatie



Eigen werk auteur

Encapsulatie

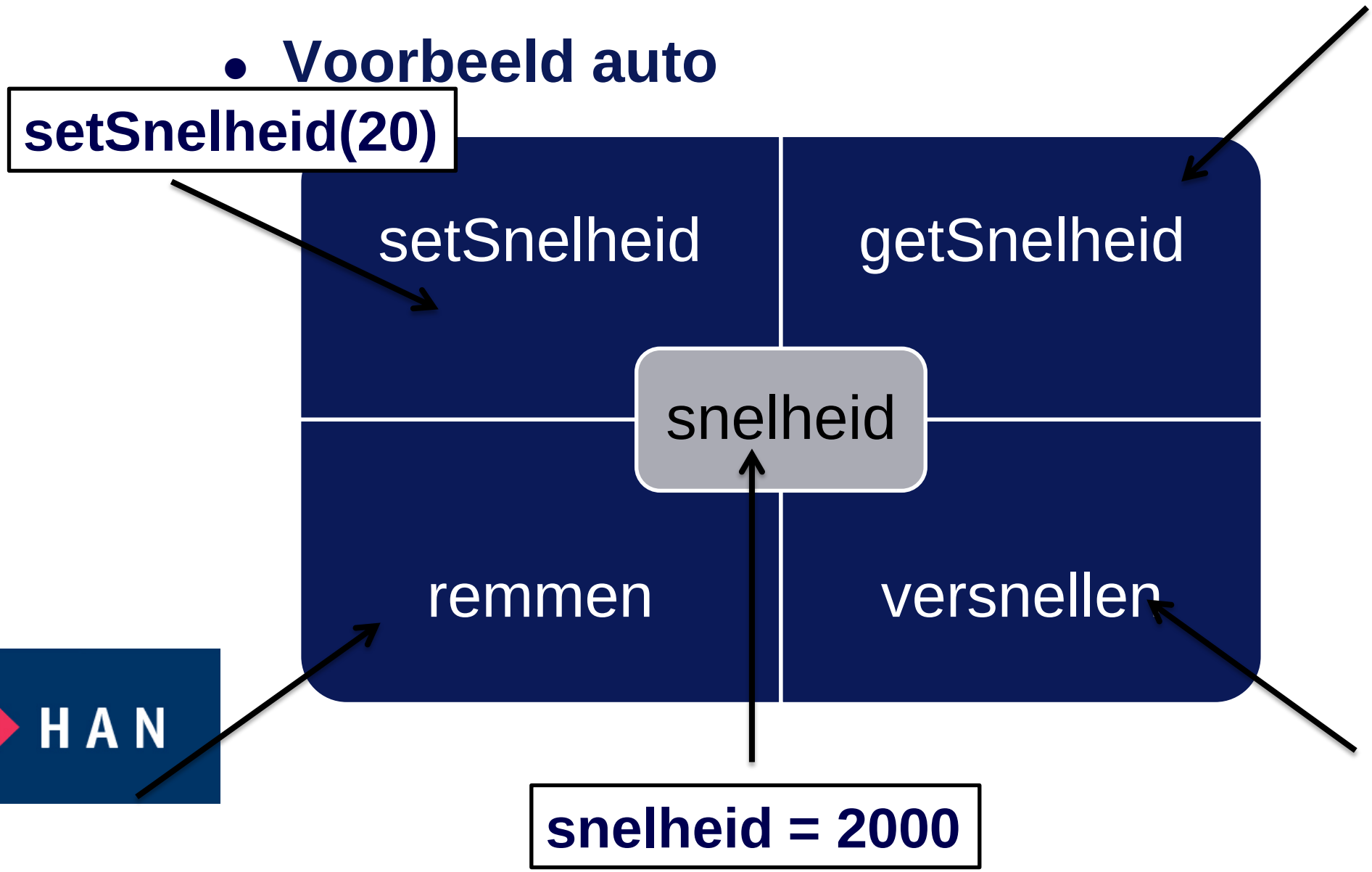
- **Voorbeeld auto**



Eigen werk auteur

Encapsulatie

- **Voorbeeld auto**



Eigen werk auteur

Encapsulatie

- Omgeven van data (variabelen) met functionaliteit (functies)
- De variabelen zijn alleen te veranderen via de functies

Voorbeeld auto 1

auto1.py - H:/courses/jaar1/owe3/informat

File Edit Format Run Options Windo

```
class auto:

    def setSnelheid(self, v):
        self.snelheid = v

    def getSnelheid(self):
        return self.snelheid
```

```
>>>
>>> a = auto()
>>> a.setSnelheid(10)
>>> a.getSnelheid()
10
>>>
```

Voorbeeld auto 1

```
class auto:

    def setSnelheid(self, v):
        self.snelheid = v

    def getSnelheid(self):
        return self.snelheid
```

Encapsulation

- Encapsulatie voorkomt dat data benaderbaar is zodat onmogelijke waardes worden voorkomen

Voorbeeld Auto 2

```
class auto:
    def setSnelheid(self, v):
        if v>0 and v<200:
            self.snelheid = v
        else:
            print ("Onmogelijk")

    def getSnelheid(self):
        return self.snelheid
```

Voorbeeld auto 2

```
class auto:

    def setSnelheid(self, v):
        if v>0 and v<200:
            self.snelheid = v
        else:
            print ("Onmogelijk")

    def getSnelheid(self):
        return self.snelheid
```

```
>>> a = auto()
>>> a.setSnelheid(10)
>>> a.setSnelheid(2000)
Onmogelijk
```

Maar...

```
>>> a = auto()
>>> a.setSnelheid(10)
>>> a.setSnelheid(2000)
Onmogelijk
>>> a.snelheid = 2000
>>> a.getSnelheid()
2000
>>>
```


Eigen werk auteur

Auto 3

```
class auto:

    def setSnelheid(self, v):
        if v>0 and v<200:
            self.__snelheid = v
        else:
            print ("Onmogelijk")

    def getSnelheid(self):
        return self.__snelheid
```

```
>>> a =auto()
>>> a = auto()
>>> a.setSnelheid(10)
>>> a.setSnelheid(2000)
Onmogelijk
>>> a.__snelheid = 2000
>>> a.getSnelheid()
10
>>>
```

Agenda

1. Object oriëntatie
2. Classes
3. Instances
4. Designing
5. Inheritance
6. Polymorfisme

Classes

- Een class is als een blauwdruk voor een huis
- Een blauwdruk kan vele malen gebruikt worden om steeds hetzelfde huis te bouwen

Auto 4

- We kunnen eindeloos veel auto's maken
- Iedere auto heeft zijn eigen snelheid

```
>>> a1 = auto()
>>> a2 = auto()
>>> a1.setSnelheid(10)
>>> a2.setSnelheid(20)
>>> a1.getSnelheid()
10
>>> a2.getSnelheid()
20
>>>
```

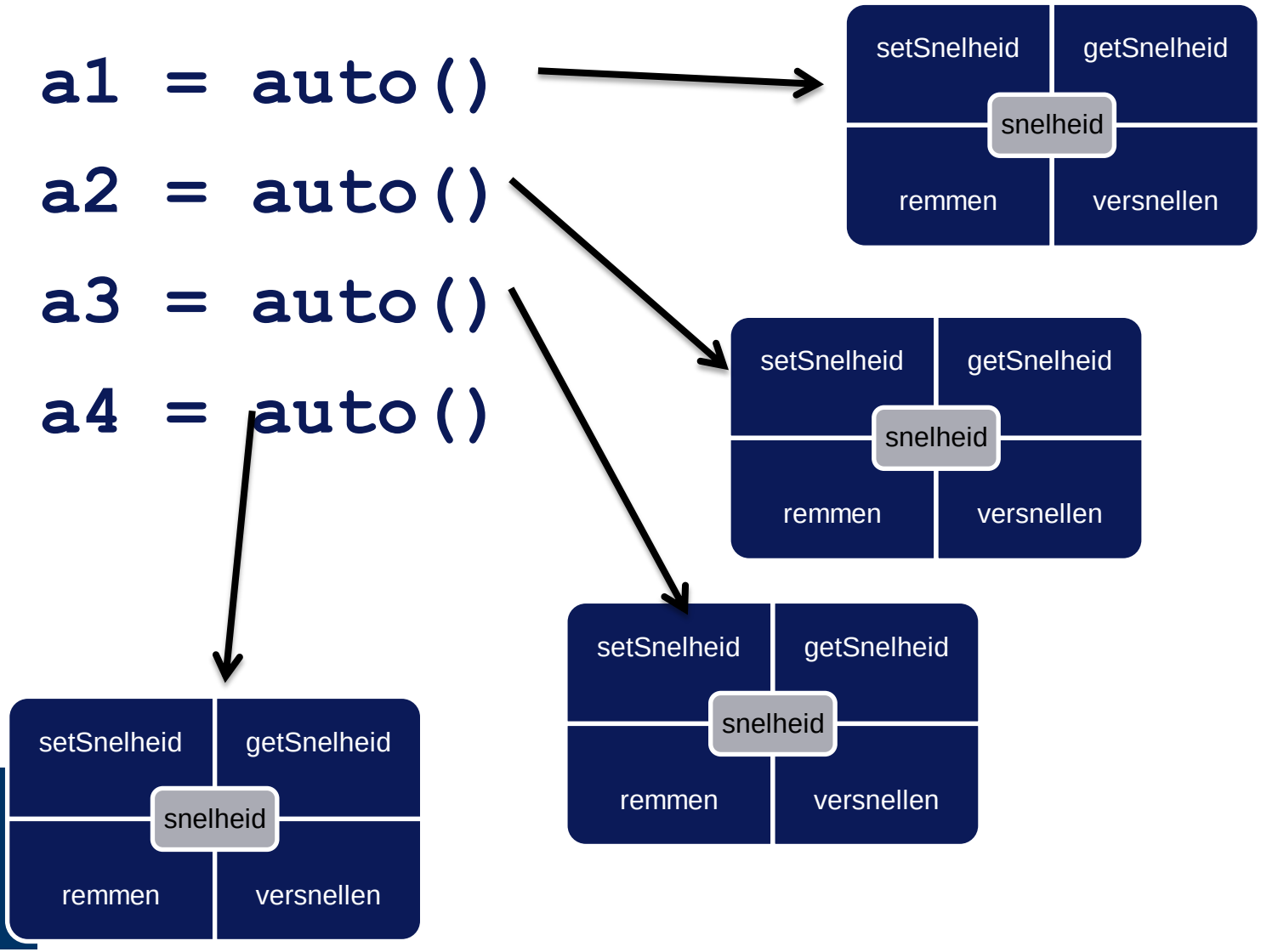

Eigen werk auteur

Agenda

1. Object oriëntatie
2. Classes
3. Instances
4. Designing
5. Inheritance
6. Polymorfisme

Instance

```
a1 = auto()
a2 = auto()
a3 = auto()
a4 = auto()
```



Eigen werk auteur

Instances, constructor

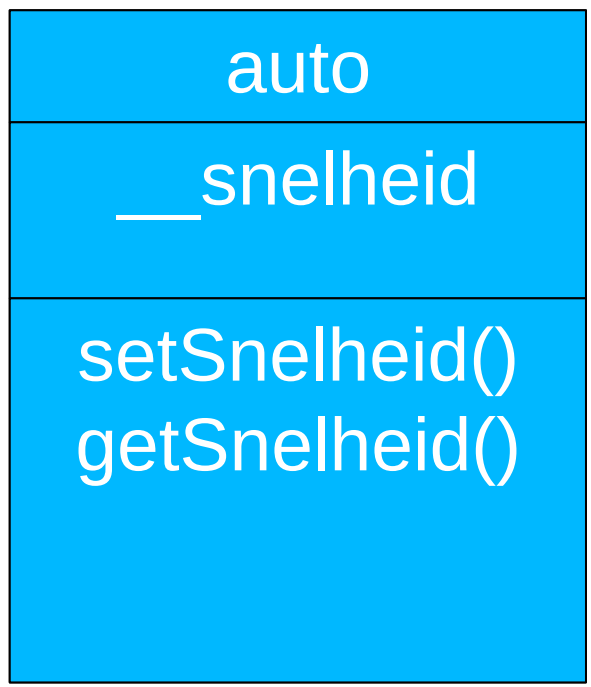
```
class auto:

    def __init__(self,v):
        self.setSnelheid(v)

    def setSnelheid(self, v):
        if v>0 and v<200:
            self.__snelheid = v
        else:
            print ("Onmogelijk")

    def getSnelheid(self):
        return self.__snelheid
```

```
>>>
>>> a1 = auto()
Traceback (most recent call last):
  File "<pyshell#42>", line 1, in <module>
    a1 = auto()
TypeError: __init__() missing 1 required positional argu
ment: 'v'
>>> a1 = auto(50)
>>>
```

Agenda

1. Object oriëntatie
2. Classes
3. Instances
4. Designing
5. Inheritance
6. Polymorfisme

OO Designing → UML



Eigen werk auteur

Agenda

1. Object oriëntatie
2. Classes
3. Instances
4. Designing
5. Inheritance
6. Polymorfisme

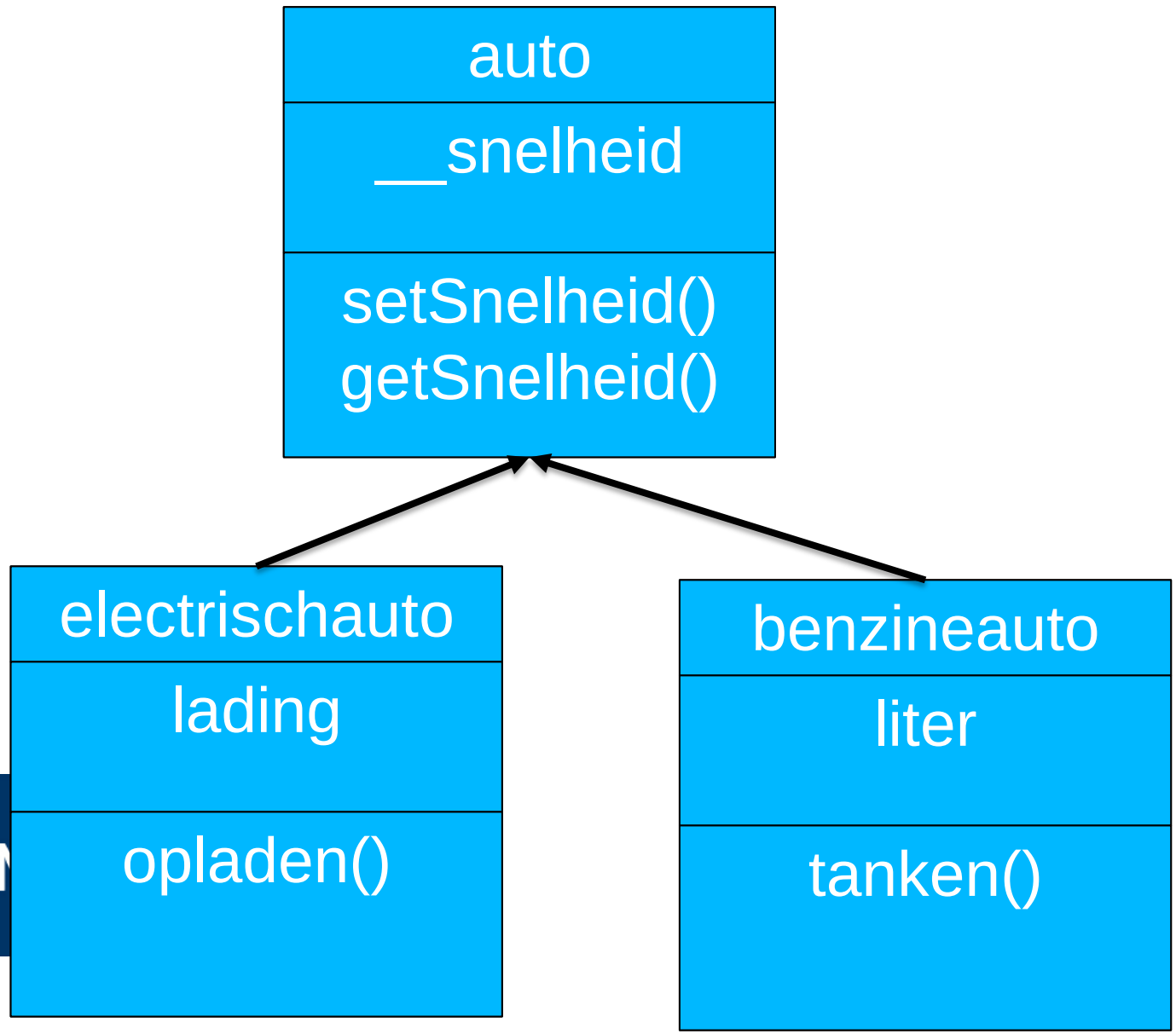
Kenmerken

- **De drie kenmerken van object oriëntatie**
  - Encapsulatie
  - Inheritance
  - Polymorfisme

Inheritance

- Een electrische auto kan overerven van auto, heeft alle kenmerken van een auto met nog extra eigenschappen

OO Designing → UML



Eigen werk auteur

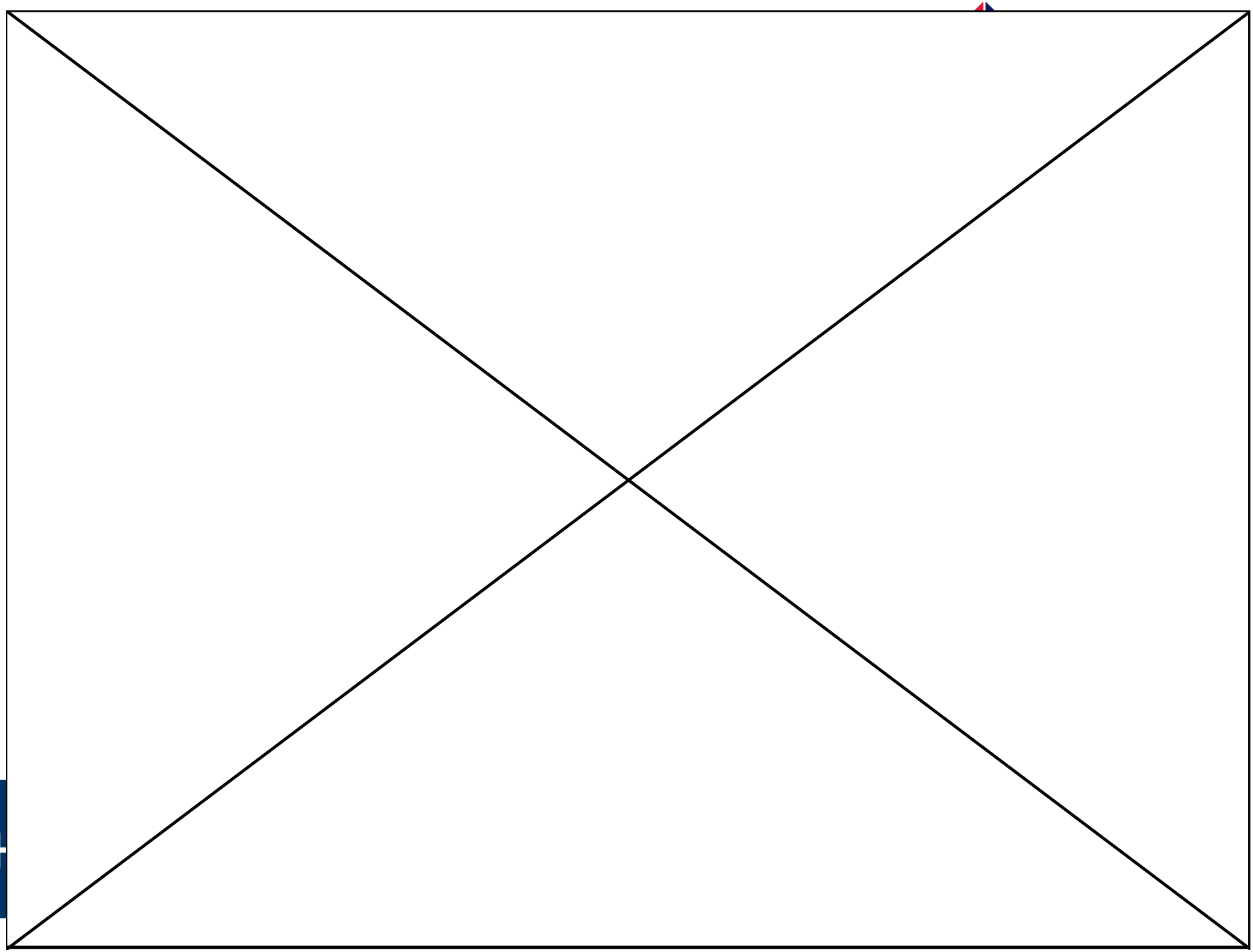
Agenda

1. Object oriëntatie
2. Classes
3. Instances
4. Designing
5. Inheritance
6. Polymorfisme

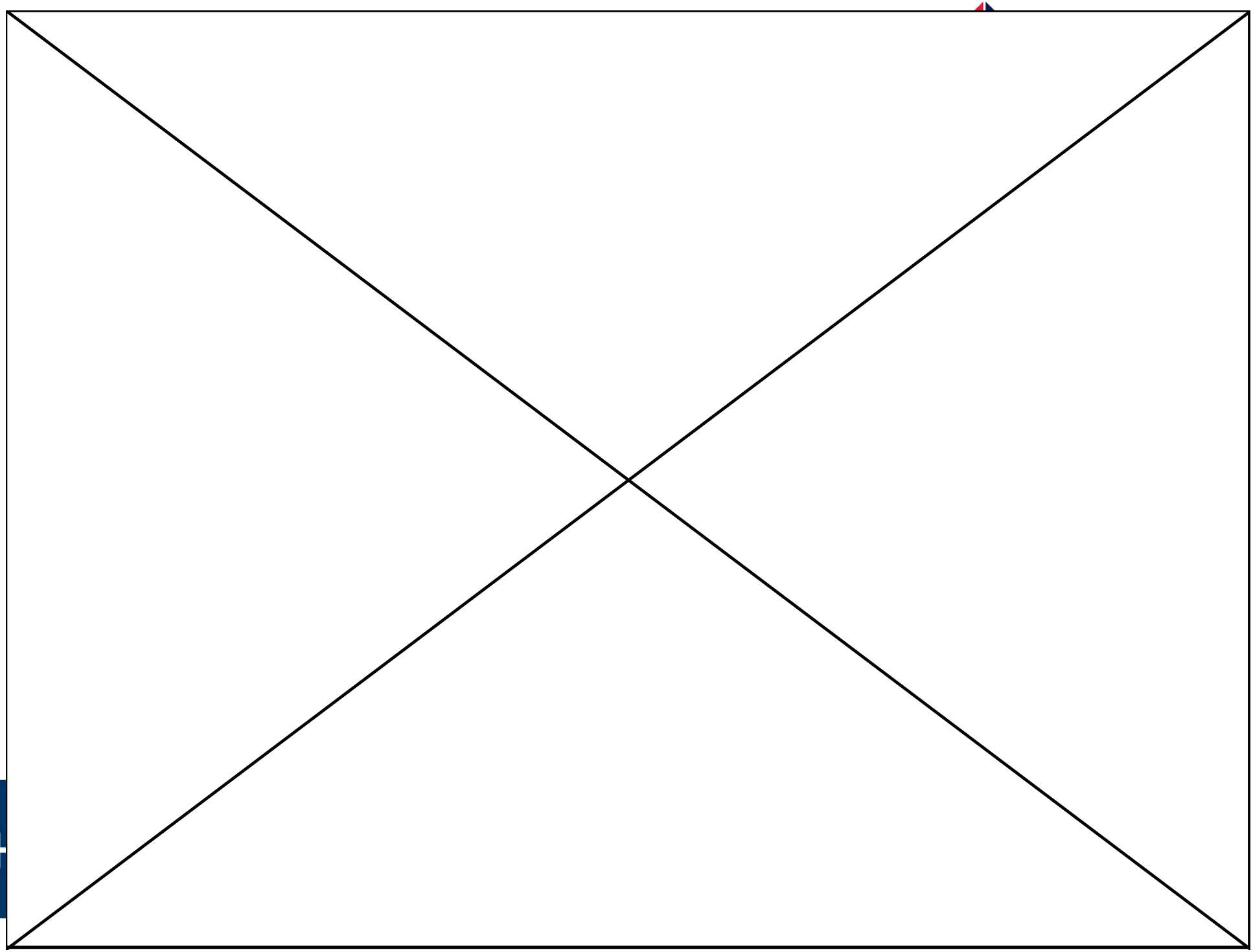
Polymorfisme

- Letterlijk veelvormig gedrag
- Iedere auto kan rijden, maar hoe die auto gaat rijden is afhankelijk van het type: benzine auto of electrische auto

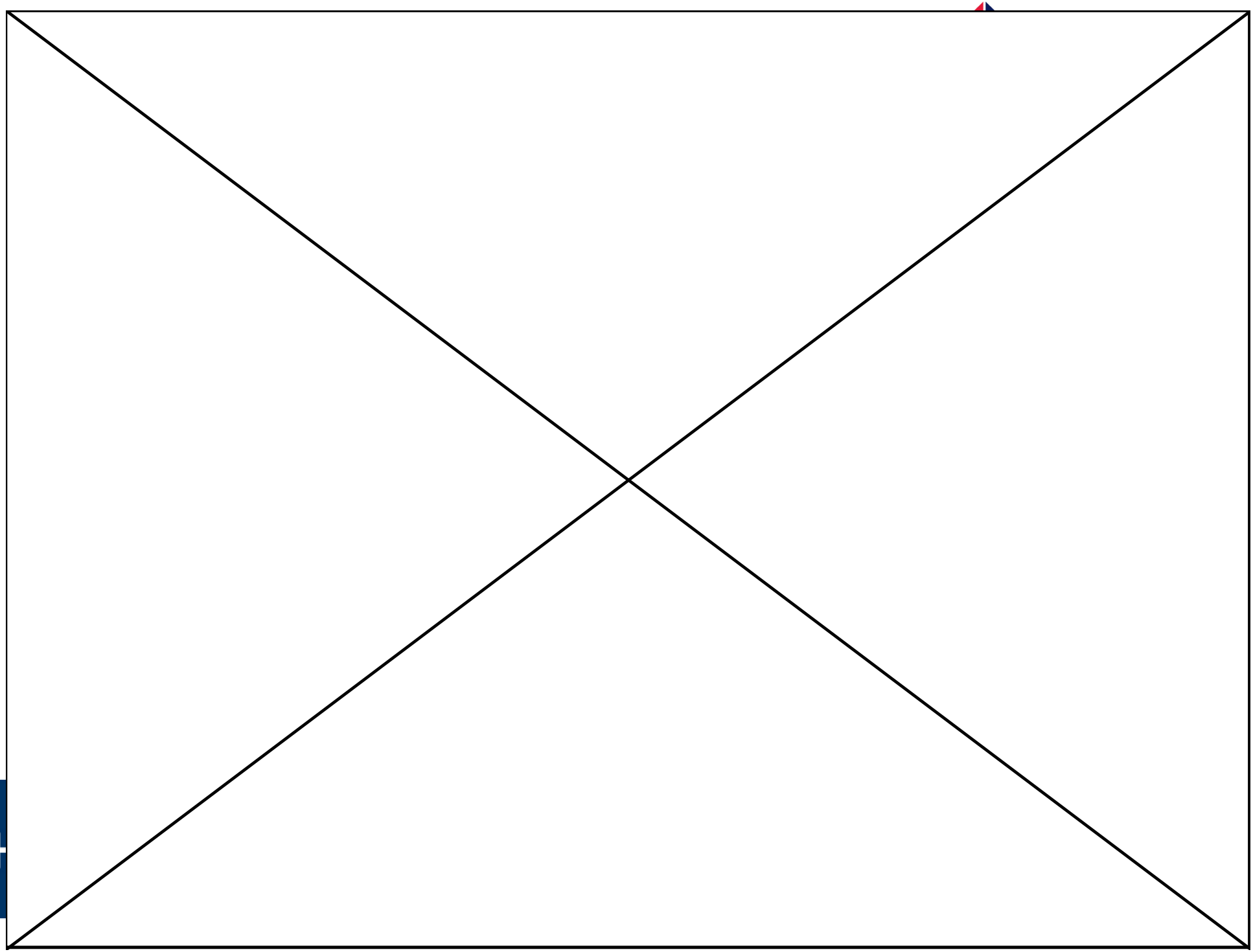
Voorbeeldvragen



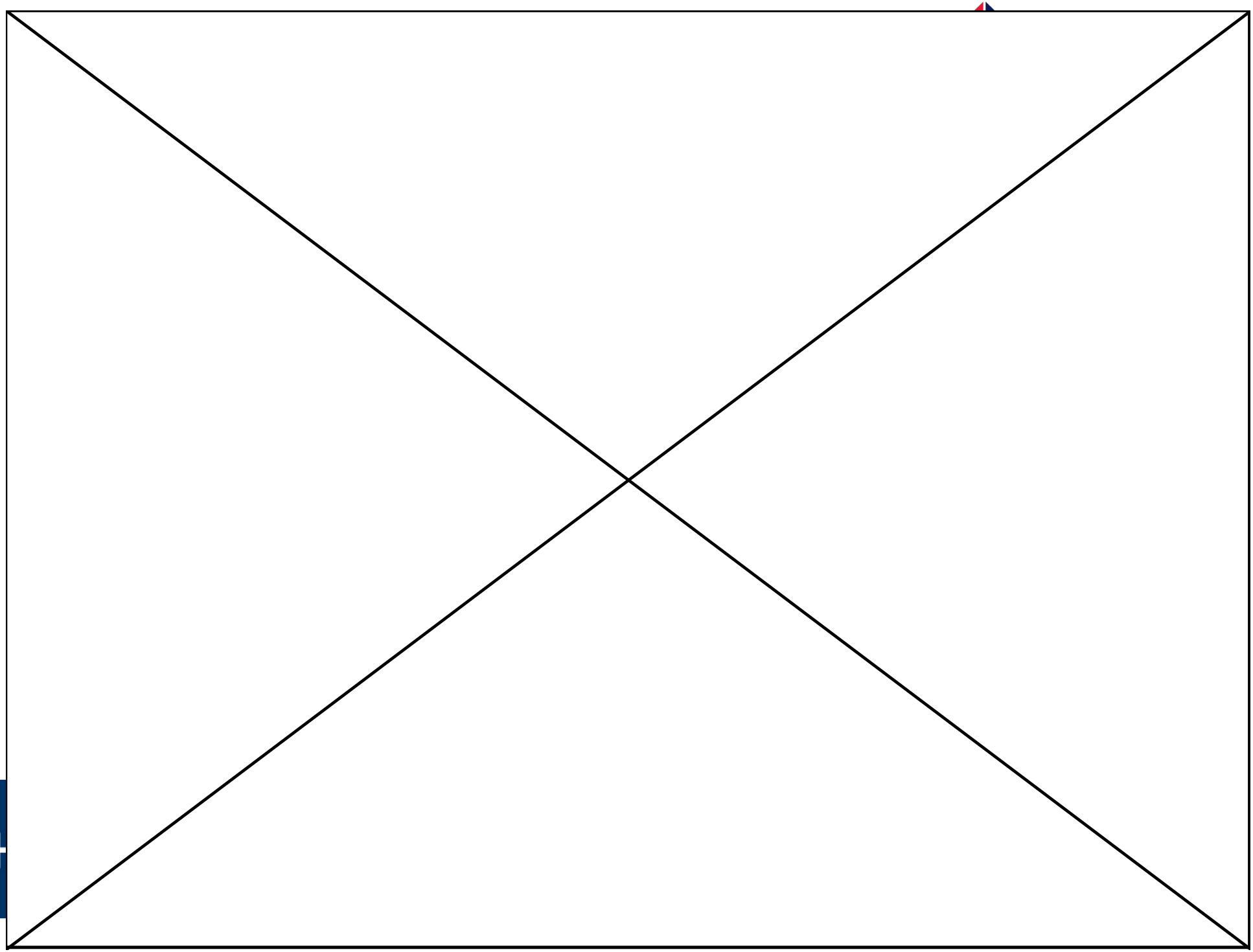
Eigen werk auteur



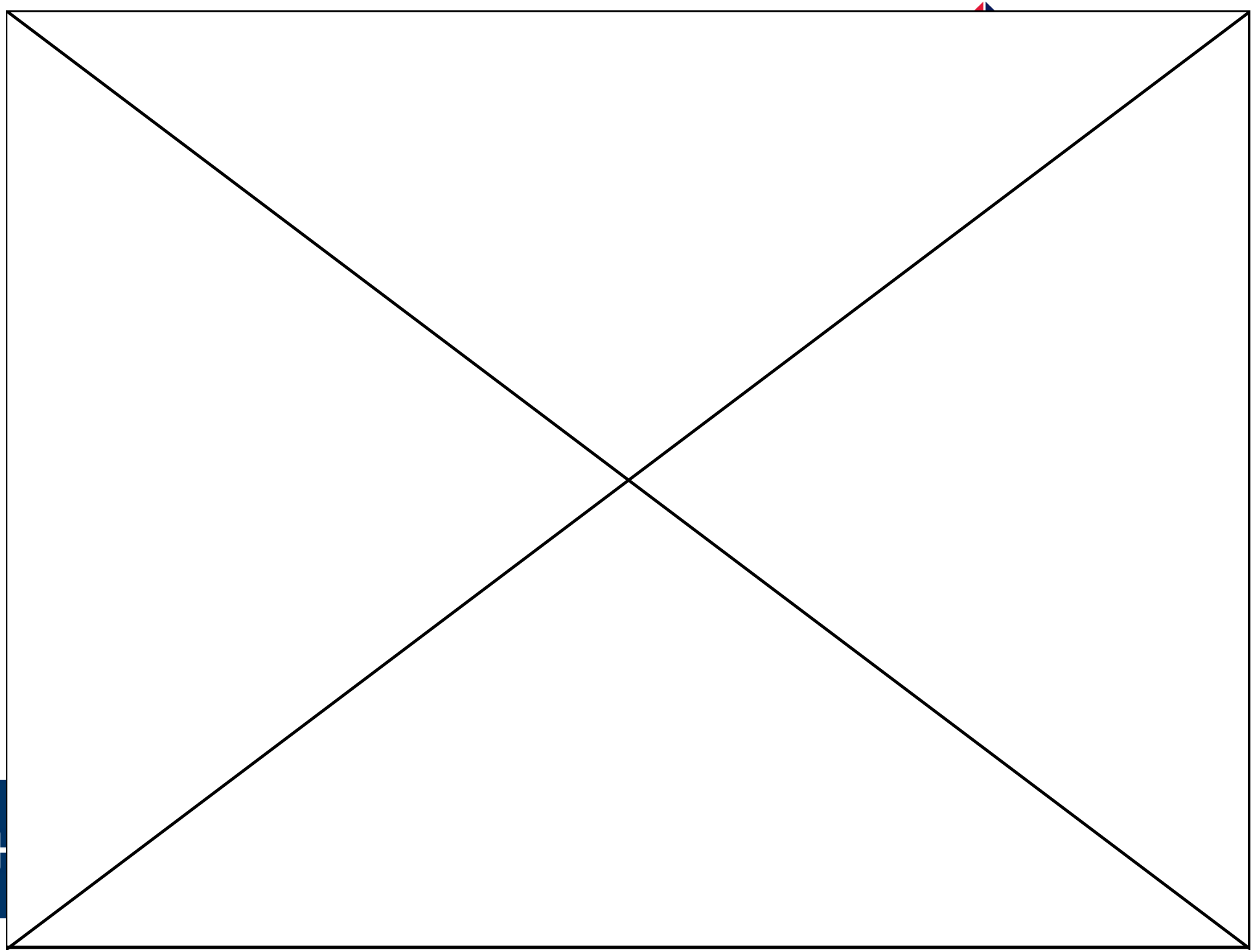
Eigen werk auteur



Eigen werk auteur



Eigen werk auteur



Eigen werk auteur

Samenvatting

- **Object-Oriëntatie is een andere manier van programmeren waardoor het dichter tegen de werkelijkheid aanligt**
- **Kenmerkende eigenschappen zijn:**
  - Encapsulation
  - Inheritance
  - Polymorfisme