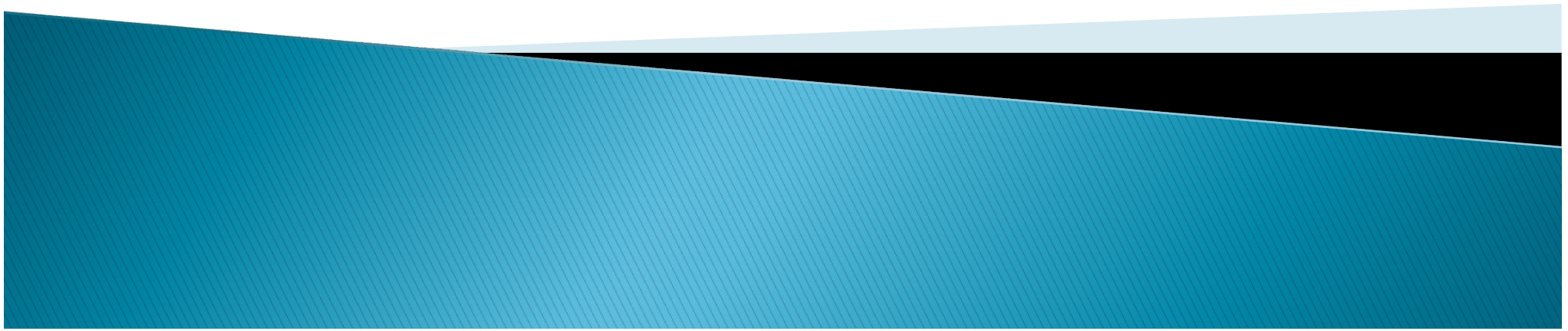
Object-oriented programming (OOP)



Definition (by Alan Kay)

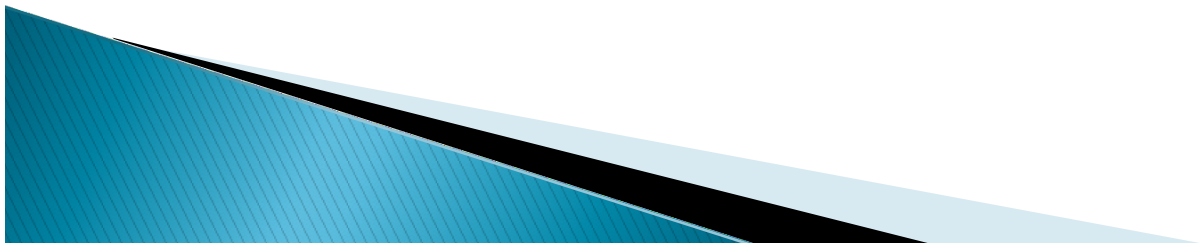
1. Everything is an object.
2. Objects communicate by sending and receiving messages.
3. Objects have their own memory.
4. Every object is an instance of a class.
5. The class holds the shared behaviour for ist instances.
6. To eval a program list, control is passed to the first object and the remainder is treated as its message.

Why do we need OOP?

Example (pseudo-code)

Beginner:

```
int a=5
int b=a*a // b=a^2
b=b*b      // b=a^4
return b*b // b=a^8
```



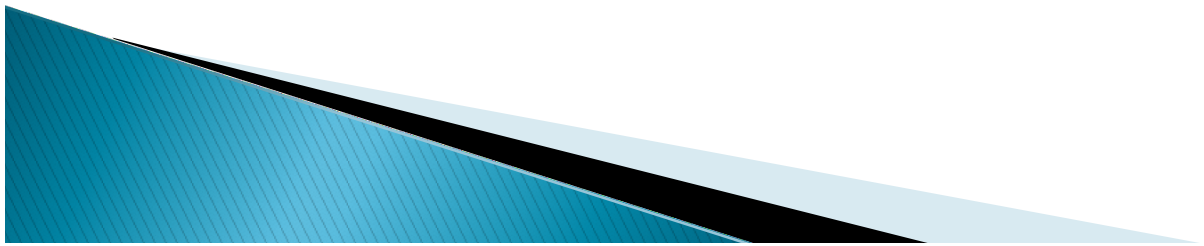
Why do we need OOP?

Example (pseudo-code)

Advanced:

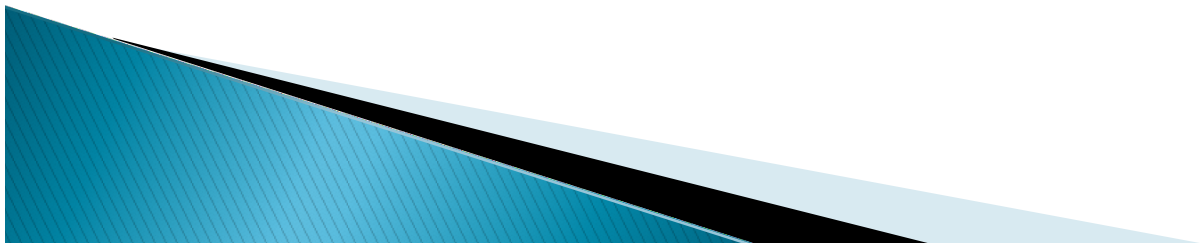
```
function power2(a)
  return a*a

int a=5
int b=power2(a)
b=power2(b)
return power2(b)
```



Why do we need OOP?

- Division of a program into different units
- Easier to understand
- Easier to add functions
- Better to reuse data structures
- Inheritance of properties



Class

- attributes

```
class car
    length
    colour
    fuel_tank
    next_service
    winter_summer
```

- methods

```
drive_a_km
change_tires
go_to_the_garage
```

Theory	Example

Object

length=5
colour=black
fuel_tank=60
winter_summer=0
next_service=10‘000

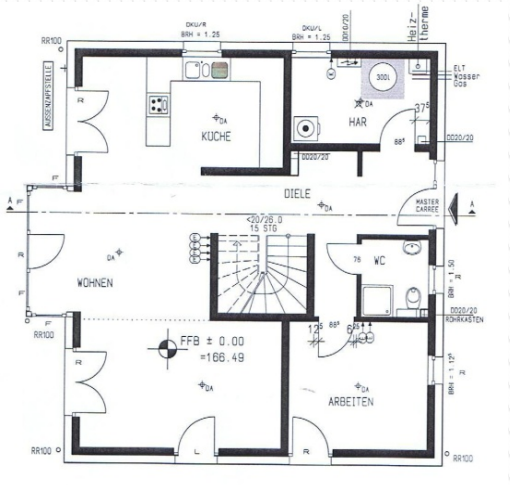
drive_a_km
  fuel_tank-=5
  next_service-=1

length=3
colour=grey
fuel_tank=45
winter_summer=1
next_service=5‘000

drive_a_km
  fuel_tank-=2
  next_service-=2

Class vs. object

- Names all the attributes and methods.
- Definition of the attributes and specific formulation of the methods.

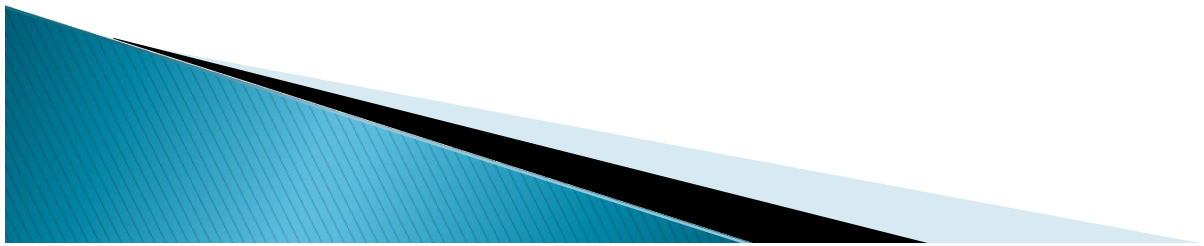


Class

object

Encapsulation

- Possibility to hide certain information or deny access.
- public / private
- Prevents the external code from being concerned with the internal workings of an object.



OOP languages

- C-languages
- Java
- **Python**
- etc.