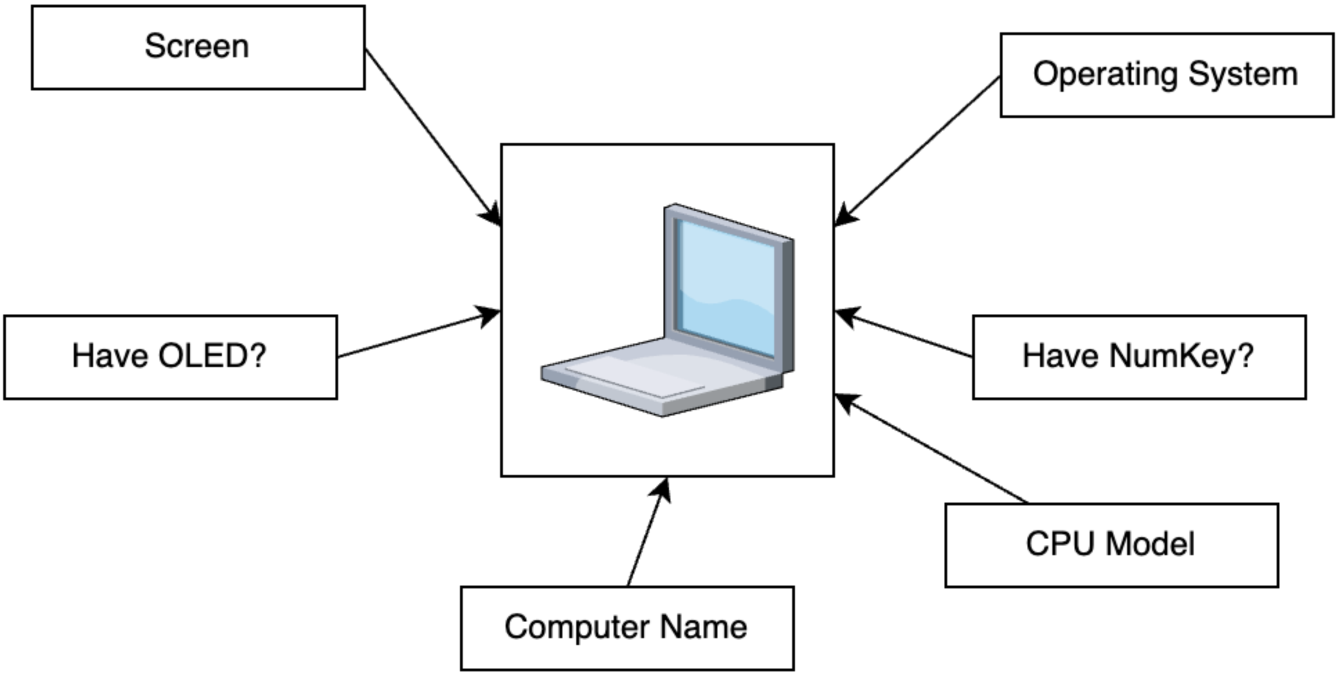
Section 1: Introduce to OOP

In this section, we will trying to provide an overview of OOP with an simple example about laptop.

Why we need OOP?

Define the issue

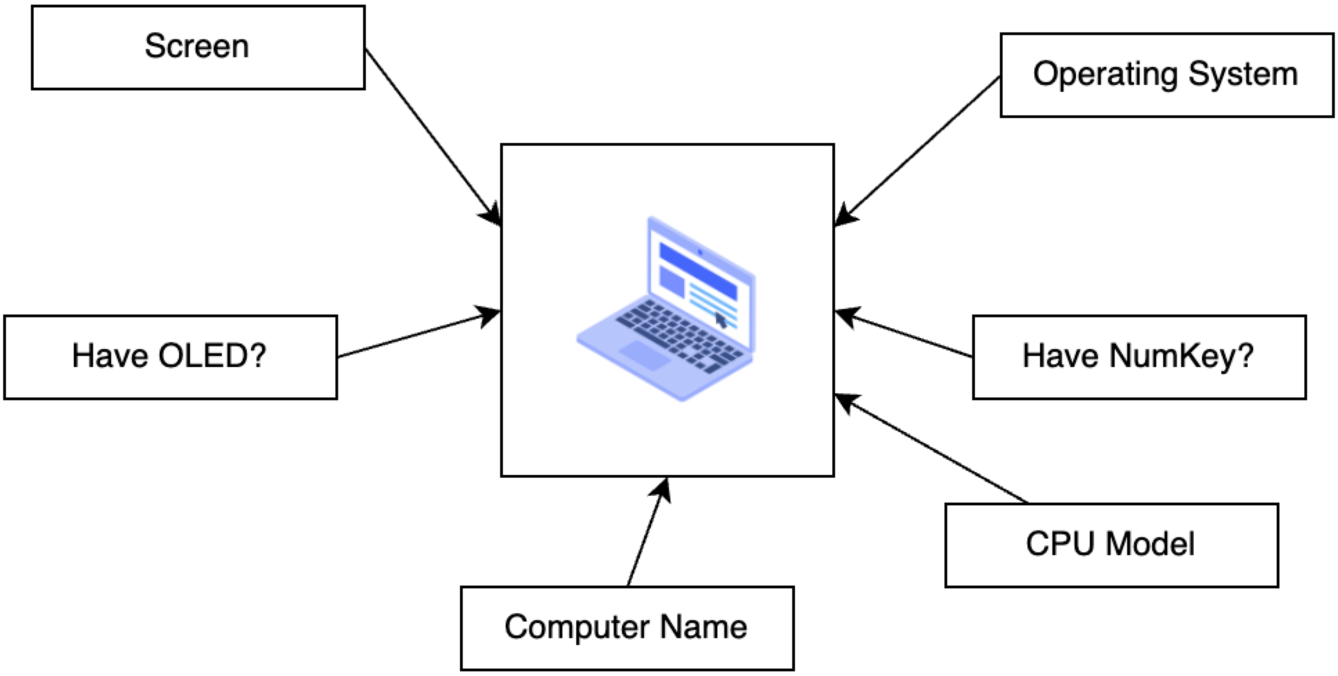
Imagine we trying to describe this laptop...



If we trying to describe this laptop, we can write a code like that:

```
int screen_pixel = 14;
bool have_OLED = true;
std::string computer_name = "It's a cool computer";
std::string cpu_model = "Letni i9-48763 4.87GHz";
bool have_numkey = true;
std::string operating_system = "Windows";
```

OK, LGTM. But if there have another new laptop...



We need more variable to describe that.

```cpp
int screen_pixel_1 = 14;
bool have_OLED_1 = true;
std::string computer_name_1 = "It's a cool computer";
std::string cpu_model_1 = "Letni i9-48763 4.87GHz";
bool have_numkey_1 = true;
std::string operating_system_1 = "Windows";

int screen_pixel_2 = 14;
bool have_OLED_2 = true;
std::string computer_name_2 = "It's an another cool computer";
std::string cpu_model_2 = "M9";
bool have_numkey_2 = false;
std::string operating_system_2 = "macOS";
```

And more computer need more variable.

```cpp
int screen_pixel_1 = 14;
bool have_OLED_1 = true;
std::string computer_name_1 = "It's a cool computer";
std::string cpu_model_1 = "Letni i9-48763 4.87GHz";
bool have_numkey_1 = true;
std::string operating_system_1 = "Windows";

int screen_pixel_2 = 14;
bool have_OLED_2 = true;
std::string computer_name_2 = "It's an another cool computer";
std::string cpu_model_2 = "M9";
bool have_numkey_2 = false;
```

```
std::string operating_system_2 = "macOS";

int screen_pixel_3 = 14;
bool have_OLED_3 = true;
std::string computer_name_2 = "It's an another another cool computer";
std::string cpu_model_3 = "M9";
bool have_numkey_3 = false;
std::string operating_system_3 = "macOS";

int screen_pixel_4 = 14;
bool have_OLED_4 = true;
std::string computer_name_4 = "It's an another another another cool computer";
std::string cpu_model_4 = "M9";
bool have_numkey_4 = false;
std::string operating_system_4 = "macOS";

// Hey! Too much variable. :angry:
```

In this case, the code should be work. But have some issue:

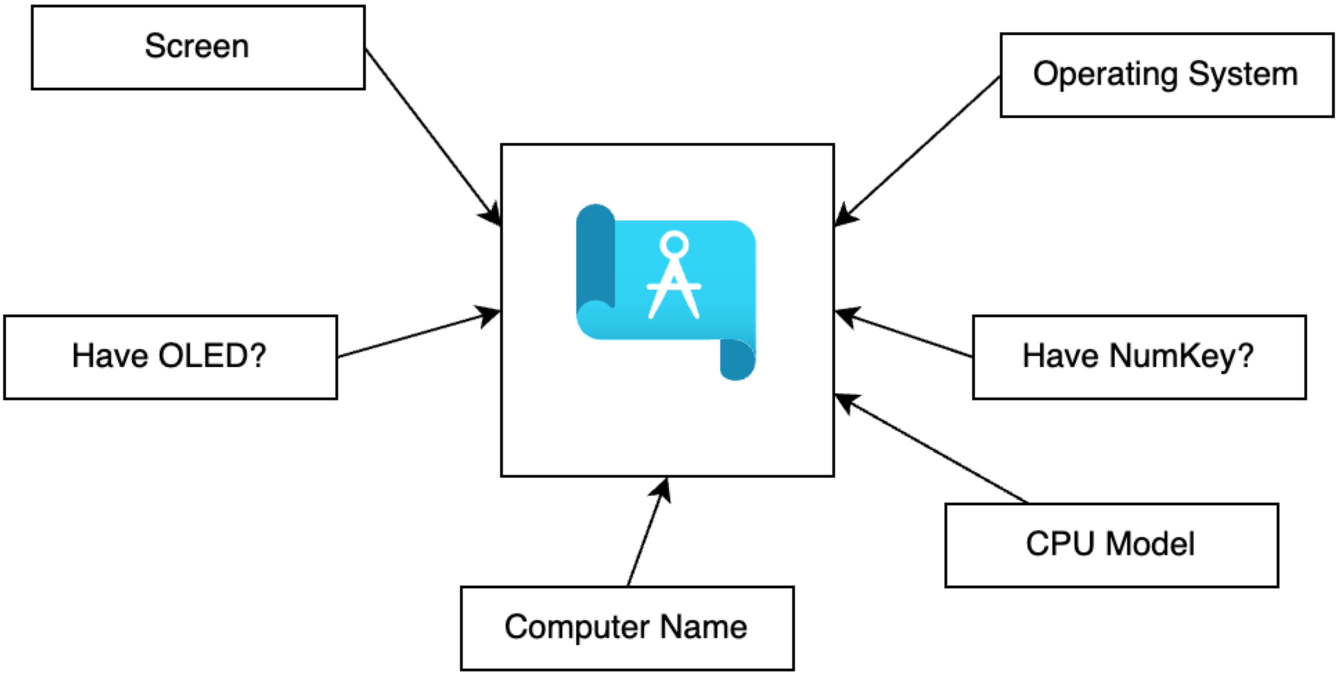
- Hard to maintain. If we have 6 computer, we need 36 lines to describe the computer. It's redundant and can be simplified.
- If we need a new computer, we need more 6 lines to describe the computer. It's redundant also.
- It seems that can be simplify. There have lot of the same attributes that can be simplify.

  For example: `operating_system_1`, `operating_system_2`, and `operating_system_3` have the same attributes `operating_system`.

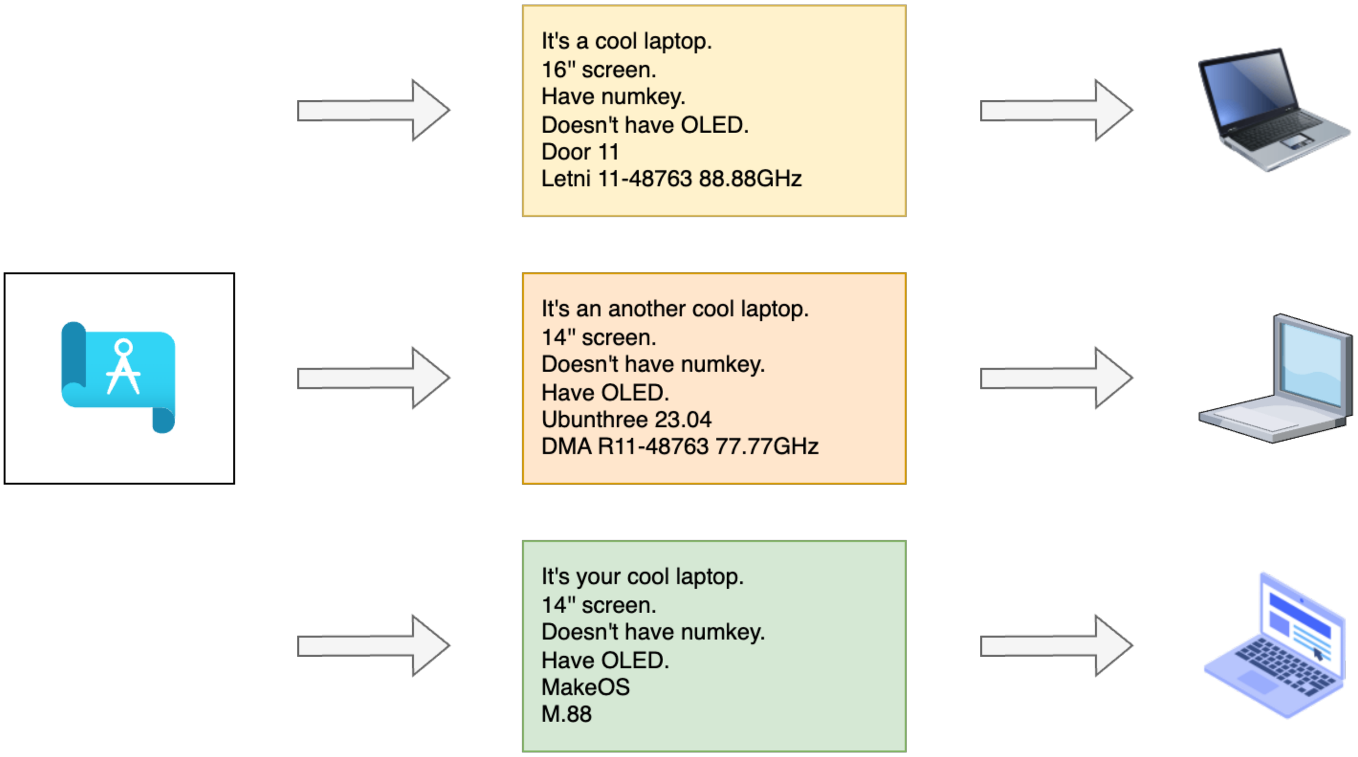
Movitation

Maybe we can extract the attribute?

In this case, we have a blueprint that can construct the laptop.



As we have a blueprint, we can setup the attribute and construct the laptop.

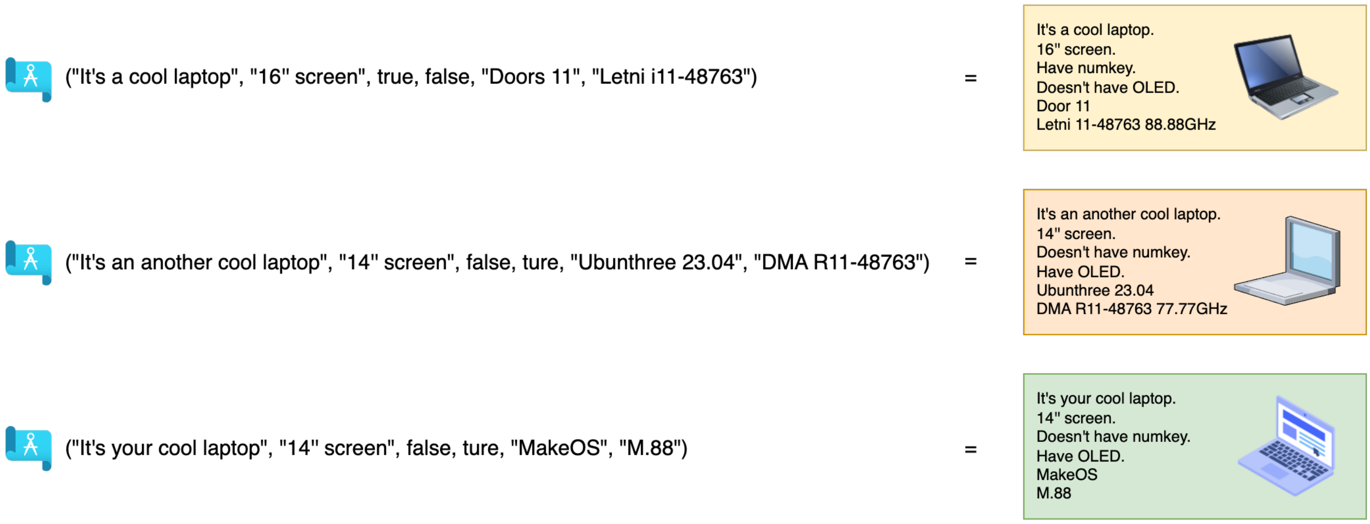


It's a cool laptop.
16" screen.
Have numkey.
Doesn't have OLED.
Door 11
Letni 11-48763 88.88GHz

It's an another cool laptop.
14" screen.
Doesn't have numkey.
Have OLED.
Ubunthree 23.04
DMA R11-48763 77.77GHz

It's your cool laptop.
14" screen.
Doesn't have numkey.
Have OLED.
MakeOS
M.88

Therefore, we can construct three computer with only three lines of code!



("It's a cool laptop", "16" screen", true, false, "Doors 11", "Letni i11-48763") =

It's a cool laptop.
16" screen.
Have numkey.
Doesn't have OLED.
Door 11
Letni 11-48763 88.88GHz

("It's an another cool laptop", "14" screen", false, ture, "Ubunthree 23.04", "DMA R11-48763") =

It's an another cool laptop.
14" screen.
Doesn't have numkey.
Have OLED.
Ubunthree 23.04
DMA R11-48763 77.77GHz

("It's your cool laptop", "14" screen", false, ture, "MakeOS", "M.88") =

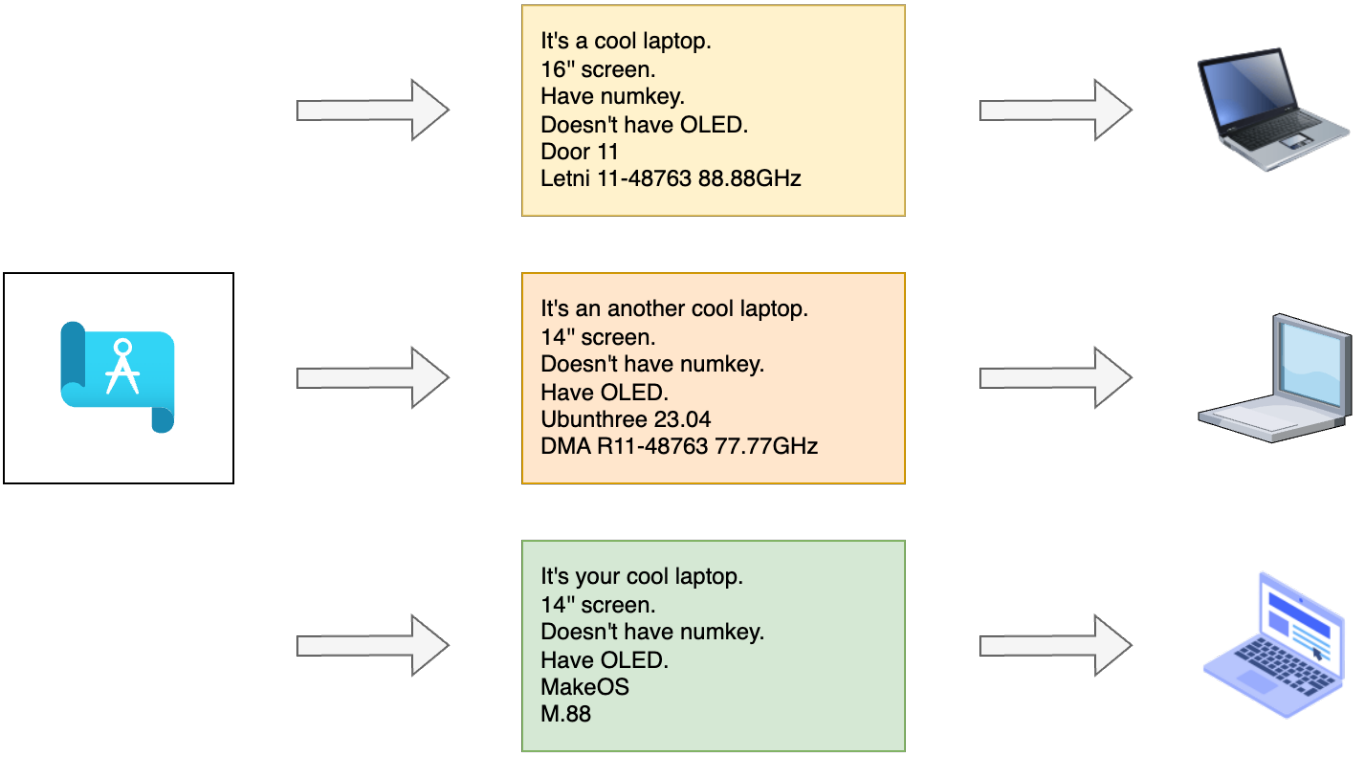
It's your cool laptop.
14" screen.
Doesn't have numkey.
Have OLED.
MakeOS
M.88

What's OOP

Introduce to OOP

OOP is a principle that have object concept.

- It's kind a blueprint. You can design a object like design a blueprint. (Design a class)
- ,You can create a object according to the blueprint. (Implement a object)



It's a cool laptop.
16" screen.
Have numkey.
Doesn't have OLED.
Door 11
Letni 11-48763 88.88GHz

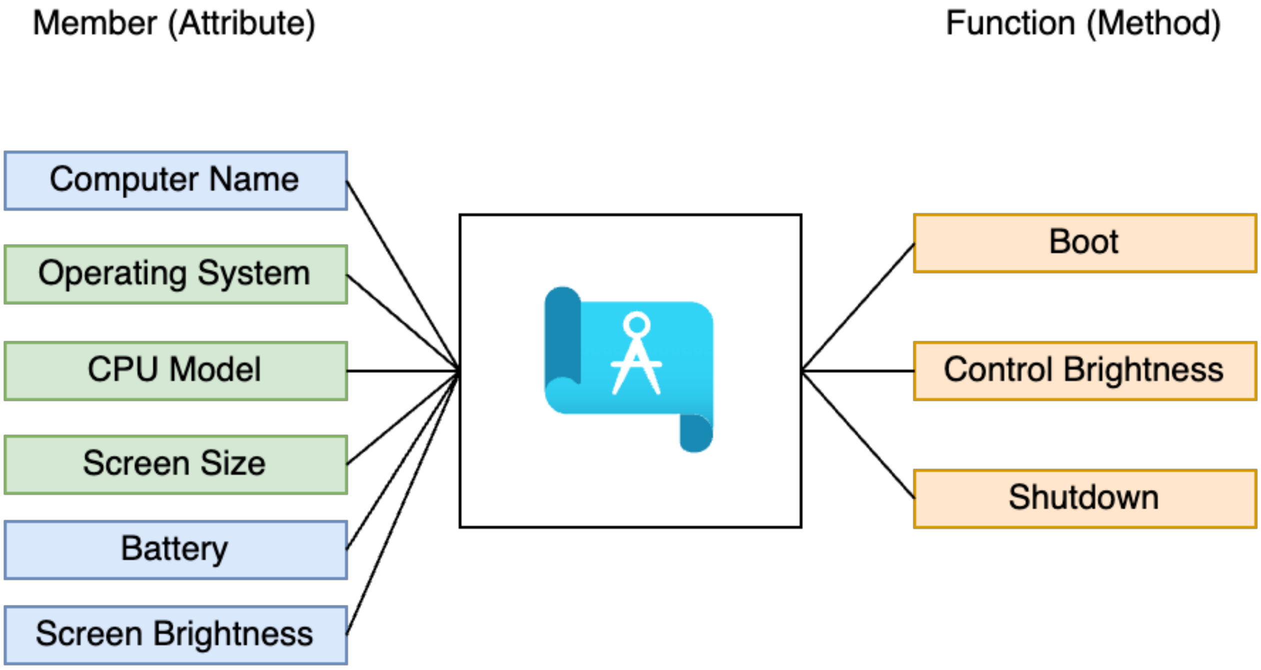
It's an another cool laptop.
14" screen.
Doesn't have numkey.
Have OLED.
Ubunthree 23.04
DMA R11-48763 77.77GHz

It's your cool laptop.
14" screen.
Doesn't have numkey.
Have OLED.
MakeOS
M.88

How do you design a class

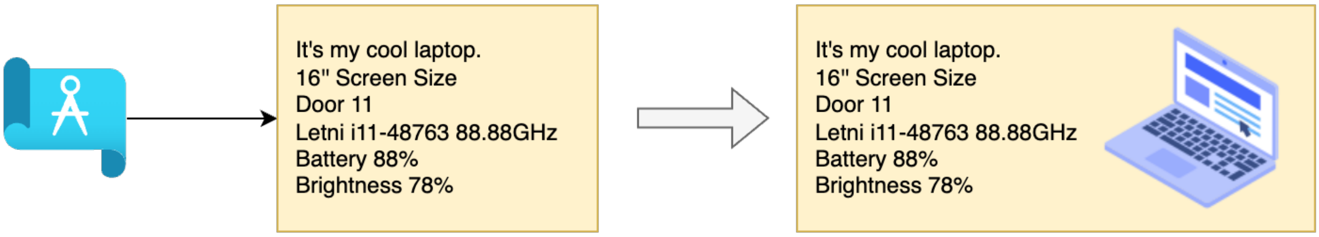
In the following image, the blueprint (class) can split into two parts. One is member, and another is function.

- For a member, we have computer name, operating system, CPU model, screen size, it record the state of laptop.
- For a function, we have some function that can boot, control the brightness, or shutdown the laptop. It change the state of laptop.



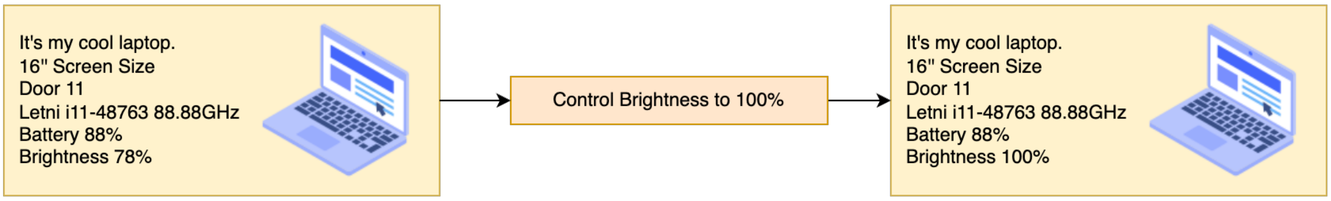
How do you implement a object

Once we have a blueprint, we can setup some attribute and implement a laptop as a object.



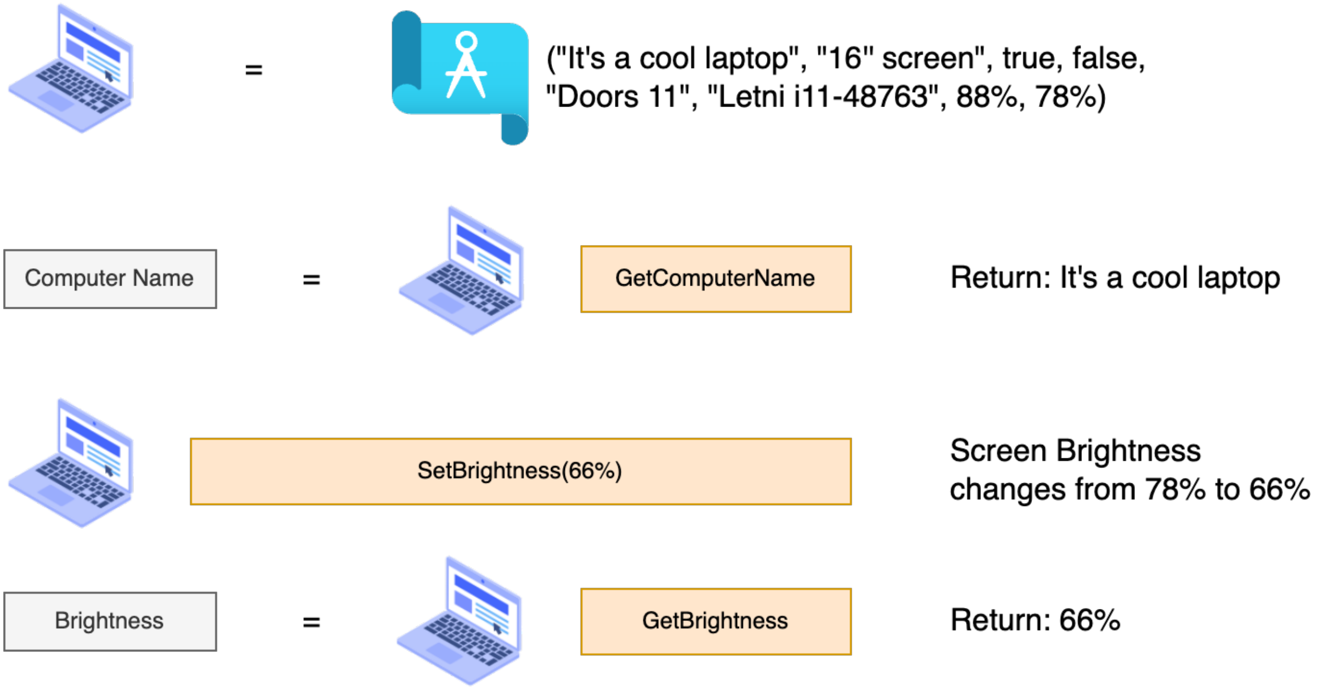
Control the object

When we have a object (that is, laptop), we can control the laptop with function!



How do we write the code

We can practice the idea with code. It will look like this:



Actually it will look like this:

```
Computer computer = Computer("It's a cool laptop", "16'' screen", true, false, "Doors 11",
"Letni i11-48763", 0.88, 0.78);

std::string name = computer.GetName(); // Return "It's a cool laptop" string

computer.setBrightness(0.66); // Set brightness from 78% into 66%

double brightness = computer.GetBrightness(); // Return 0.66 (66%)
```