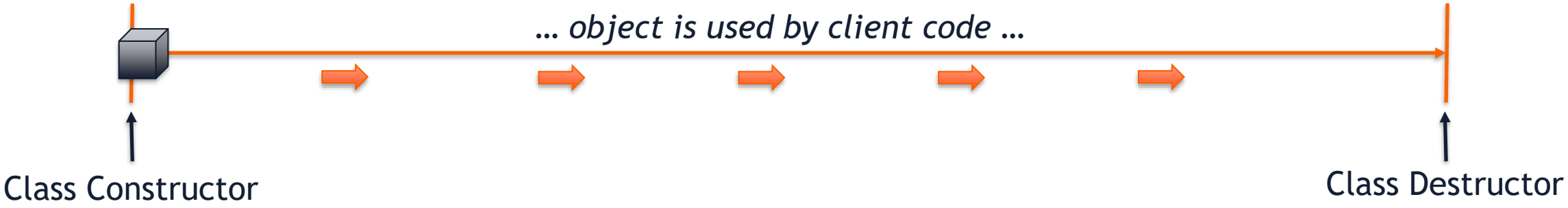
Class Destructor

Prof. Wade Fagen-Ulmschneider

When an instance of a class is cleaned up, the **class destructor** is the last call in a class's lifecycle.



Automatic Default Destructor

An **automatic default destructor** is added to your class if no other destructor is defined.

The only action of the automatic default destructor is to call the default destructor of all member objects.

Automatic Default Destructor

An destructor should never be called directly. Instead, it is automatically called when the object's memory is being reclaimed by the system:

- If the object is on the **stack**, when the function returns
- If the object is on the **heap**, when `delete` is used

Custom Destructor

To add custom behavior to the end-of-life of the function, a custom destructor can be defined as:

- A custom destructor is a member function.
- The function's destructor is the name of the class, preceded by a tilde `~`.
- All destructors have zero arguments and no return type.

```
Cube::~Cube();  // Custom destructor
```

cpp-dtor/Cube.cpp

```
14 Cube::Cube() {
15   cout << "Created $1 (default)" << endl;
16 }
17
18 Cube::Cube(double length) {
19   cout << "Created $" << getVolume() << endl;
20 }
21
22 Cube::Cube(const Cube & obj) {
23   cout << "Created $" << getVolume() << " via copy" << endl;
24 }
25
26 Cube::~Cube() {
27   cout << "Destroyed $" << getVolume() << endl;
28 }
29
30 Cube & Cube::operator=(const Cube & obj) {
31   cout << "Transformed $" << getVolume() << "-> $" << obj.getVolume() << endl;
32 }
... ...
```

cpp-dtor/main.cpp

```
11 double cube_on_stack() {
12   Cube c(3);
13   return c.getVolume();
14 }
15
16 void cube_on_heap() {
17   Cube * c1 = new Cube(10);
18   Cube * c2 = new Cube;
19   delete c1;
20 }
21
22 int main() {
23   cube_on_stack();
24   cube_on_heap();
25   cube_on_stack();
26   return 0;
27 }
```

destructor on stack

destructor on heap

memory leak on c2 since it is not destructed

Custom Destructor

A custom destructor is essential when an object allocates an external resource that must be closed or freed when the object is destroyed. Examples:

- Heap memory
- Open files
- Shared memory