API is ART : Debate

Max 5 people discuss in front of others about an API design problem or topic.

Others look at them, and finally can propose their own ideas after the talk.

I/O in std:: C++

- Apparently the ISO comittee couldn't decide :

```
 1 #include <iostream>
 2 #include <fstream>
 3
 4 int main(int argc, const char** argv) {
 5
 6   std::ifstream ifs{"somefile"};
 7
 8   int readFromFile;
 9   if (ifs.good()) {
10     ifs >> readFromFile;
11   }
12
13   try {
14     ifs.exceptions(ifs.failbit | ifs.eofbit);
15     ifs >> readFromFile;
16   } catch(const std::exception& e) {
17     std::cerr << "Error: " << e.what() << std::endl;
18   }
19
20   return 0;
21 }
```

Error reporting/handling – 30 min

- Error Handling
  - Input/Output Errors as exceptions or as known expected state ?

- How do you return values ?
  - Out parameters
  - Returnvalue, what about error return code ?
  - Getters to check for errors ?

Recovering from errors

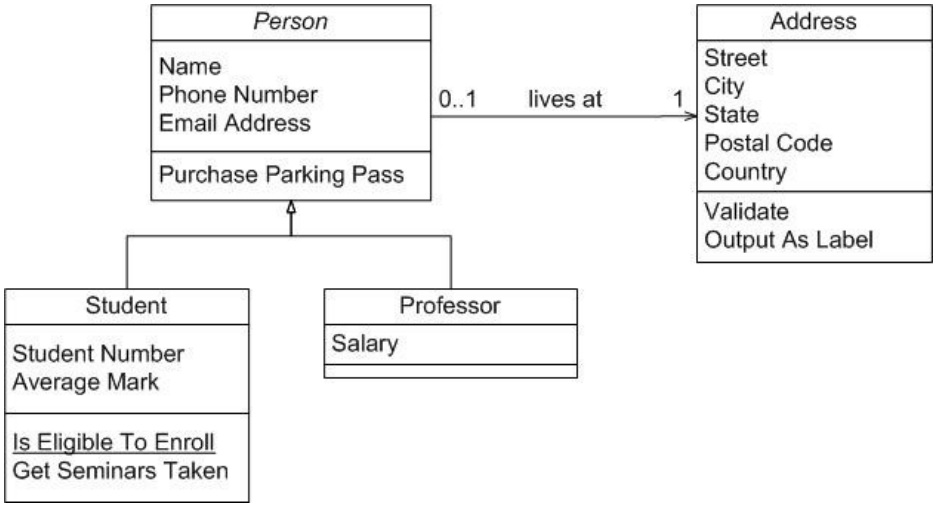
- Error Recovery
  - Handling cleanup of resources
  - Handling cleanup of own locks in multithreaded applications

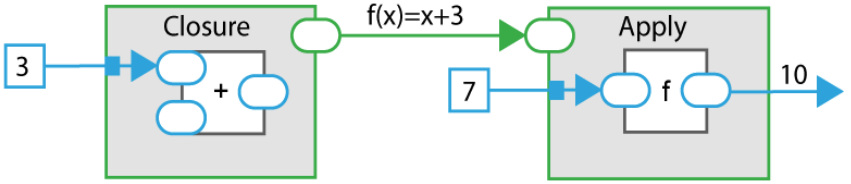
- How would one implement graceful degradation ?
  - Nullptr checks ?
  - Non available modules ?
  - Non responding APIs (timeouts, wait for comeback...)


λ
λ
THE FUNCTIONAL WAY IS THE RIGHT WAY

Object Oriented vs Functional

- Identity based computations : Typical OO
  - Types instances have identities and states, which provide methods to operate on them





- Value based computationl : Typical functional
  - Types instances are values operated with freestanding functions / functions objects.

- Could both cohabit well in a single problem domain resolution ?