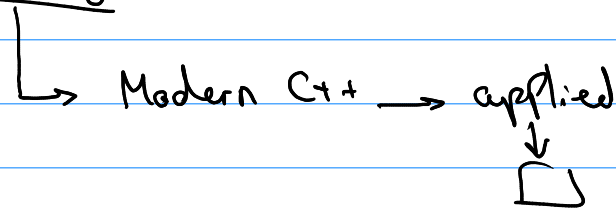
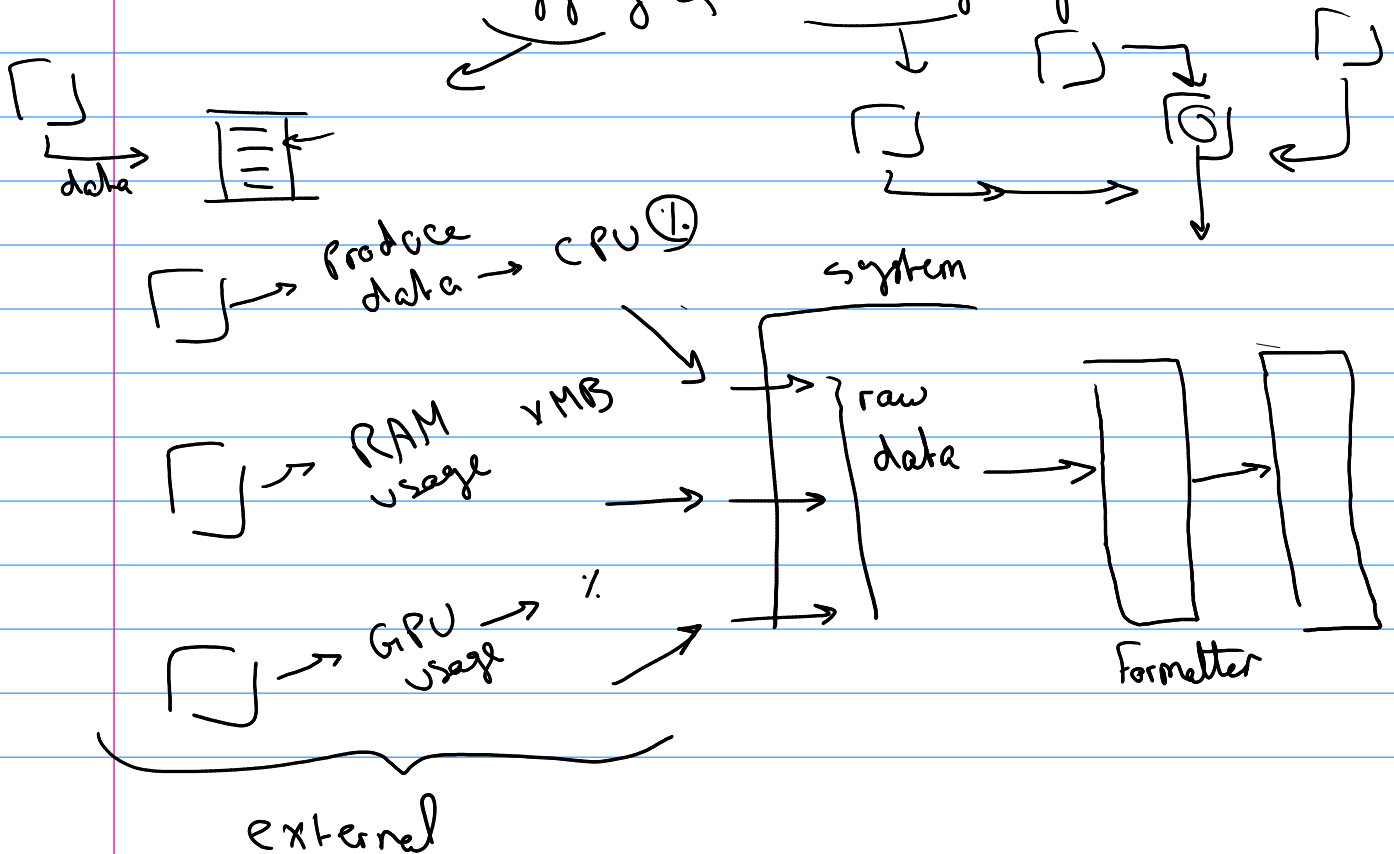
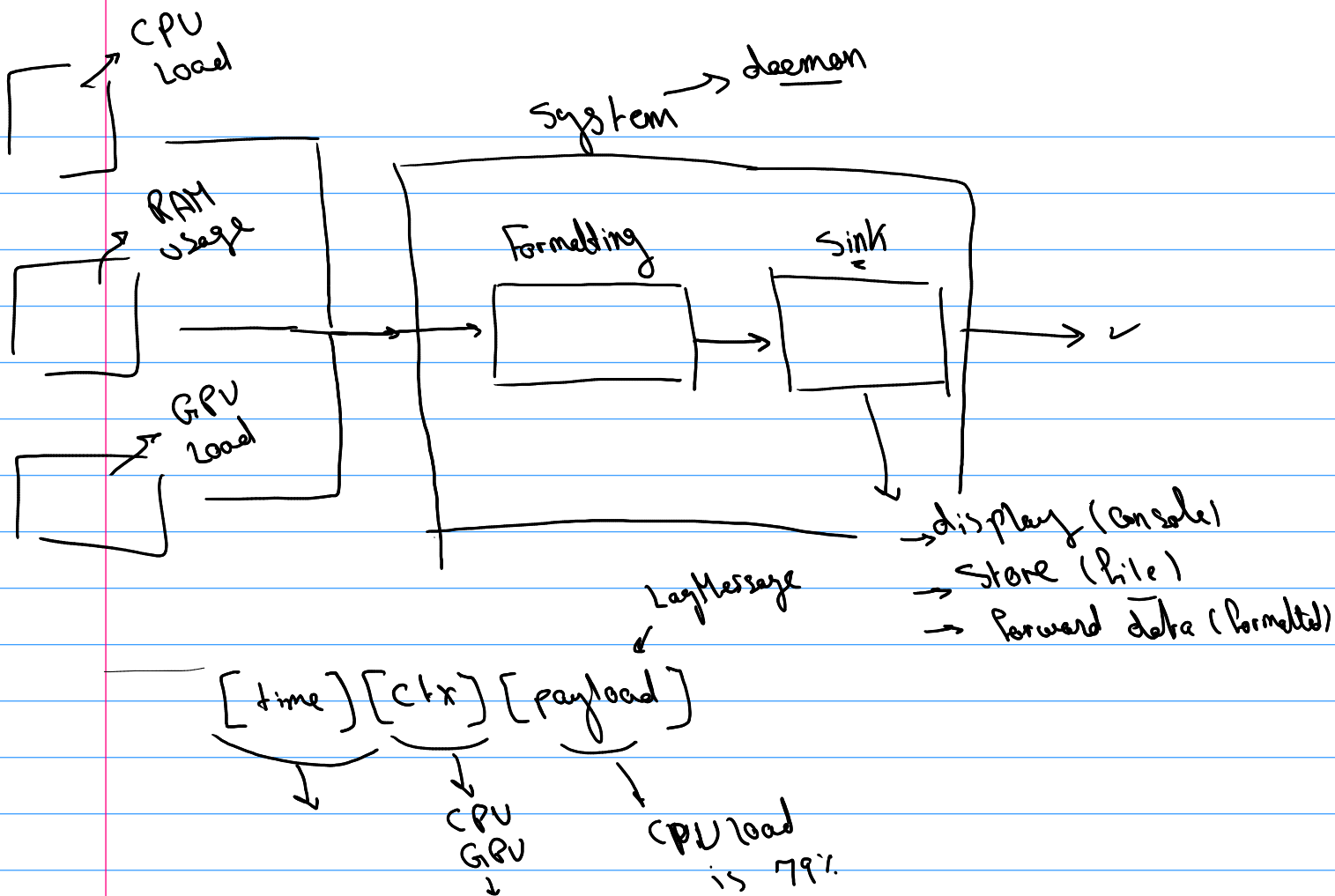


## C++ Project



## Logging & Telemetry System



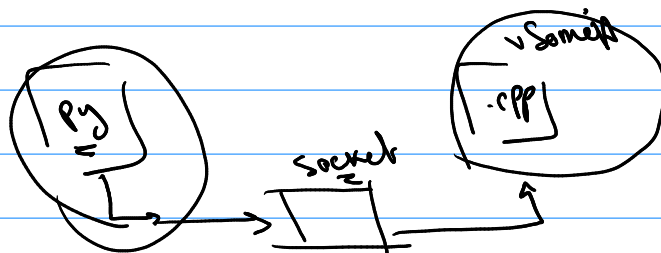
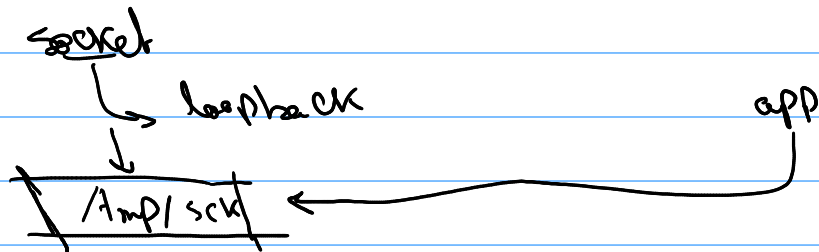


Shell script

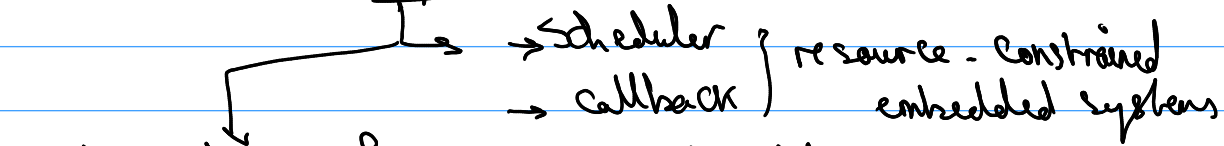
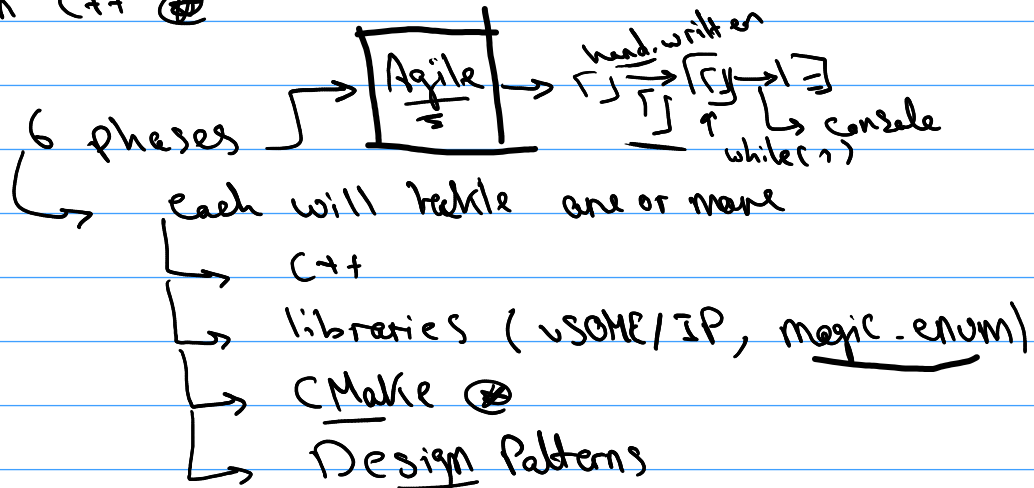
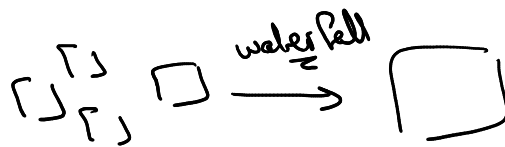
↳ randomly generated data in file / terminal

← Some ip service

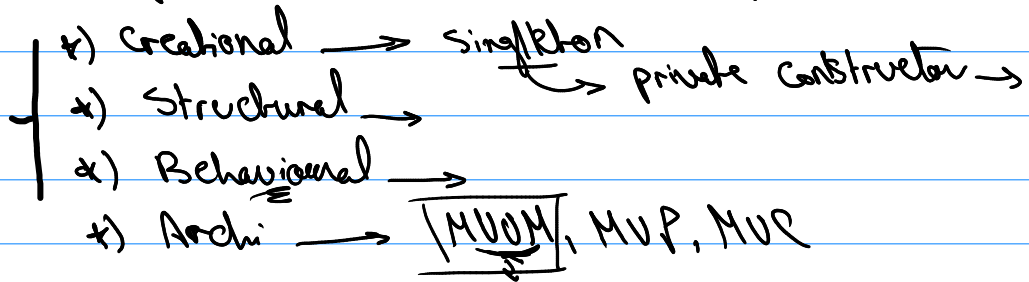
↳ other data



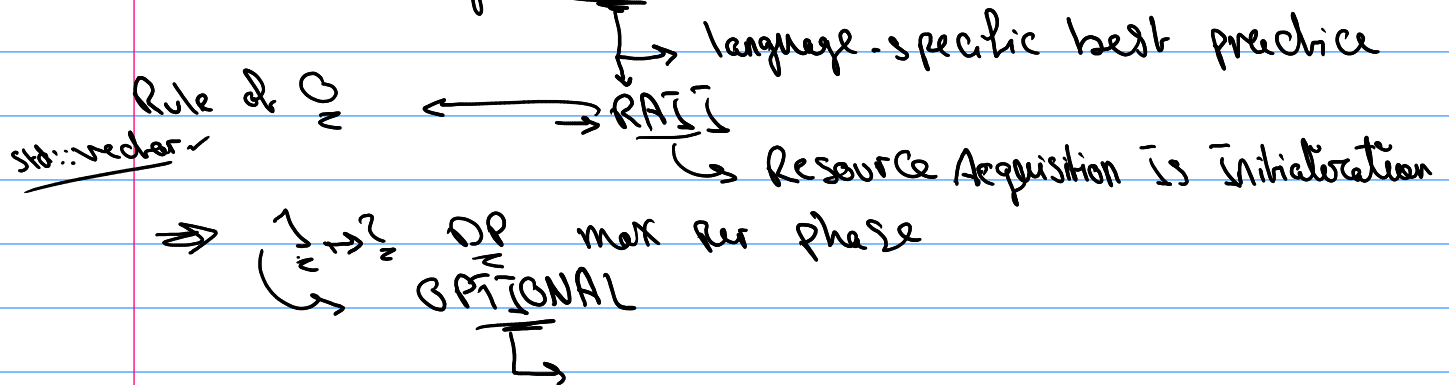
# ⊕ Modern C++ ⊕



best practices for common SW problems



## Design Idiom

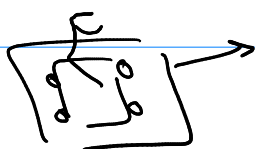


mech team -

elec team -

sw

embedded → ROS → action



# Grading & Rewards

Minimum

Per phase

C++ ✓  
CMake ✓

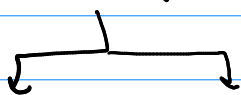
- ⊛ at the end  
↳ Doc (README + class diagram)
- ⊛ at least one  
class unit tested GTest

Herlist

↳ Score → reward

✓ .Fidl

Franga gen



✓ ← stub

✓ ← service

✓ Barzel

Optional

GTest

DP

- Modern C++ (req)
- ← → CMake (req)
- UML Diagrams ⊛ (PlantUML) (req)
- Unit testing using GTest (req)
- VSOME/IP (req)

⊛

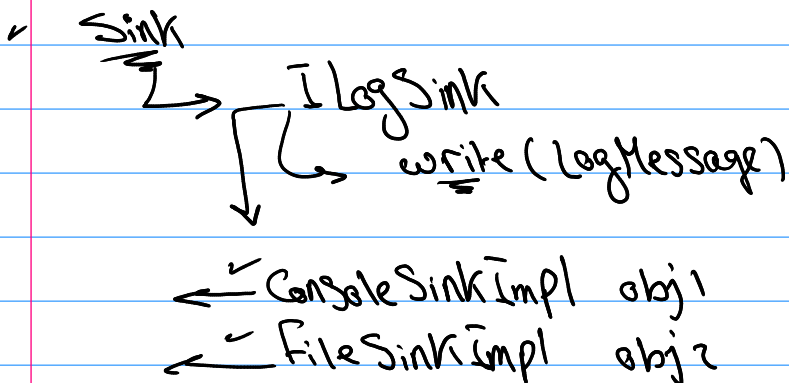
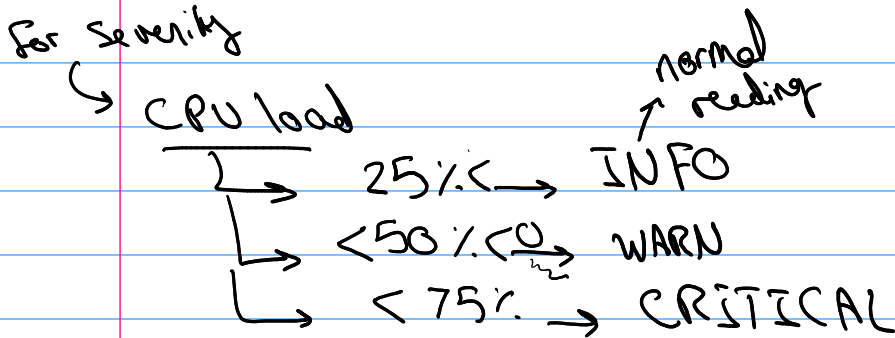
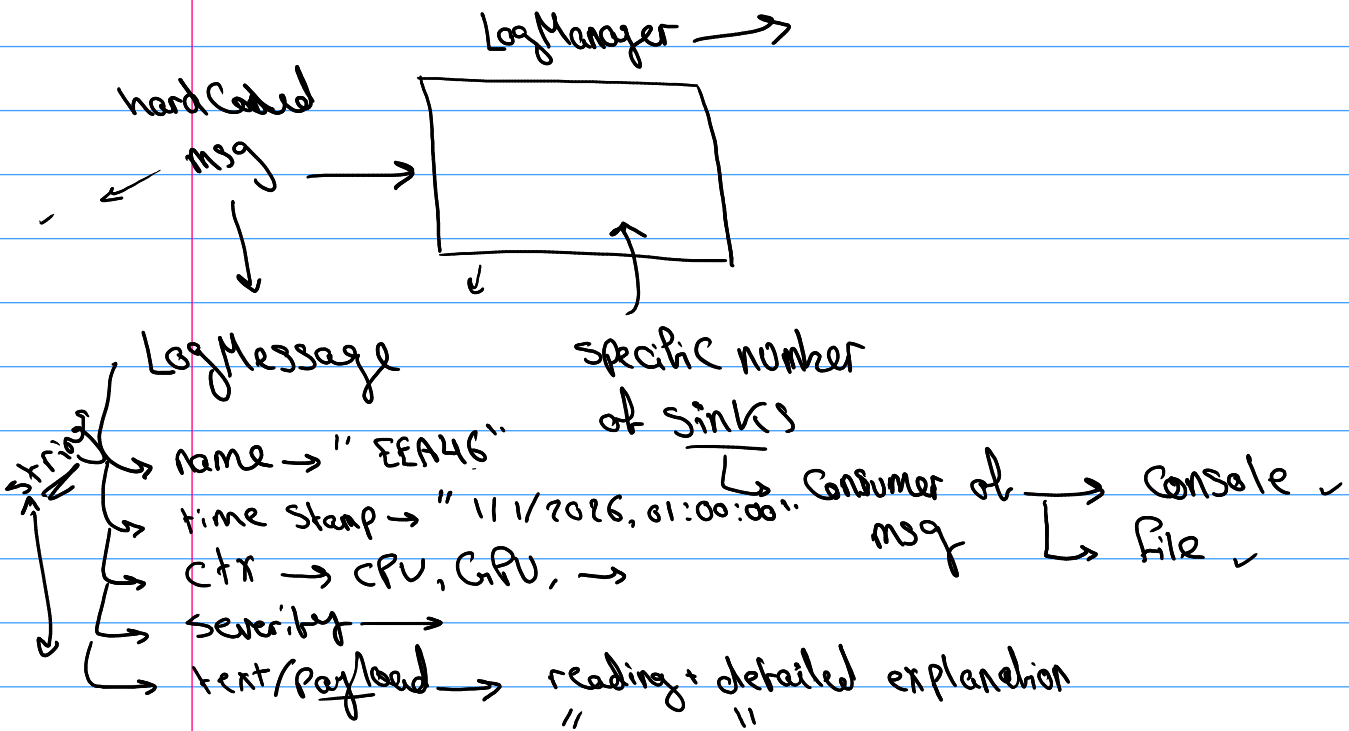
- DP
- ← → Barzel
- Common API

Omnimebron

Naming a project

# Phase One

## Core Synchronous Logging Foundation



`ILogSink* pIrr = &obj1  
obj1->write();`

LogManager

```

    addSink( )
    vector<ISink> { }
    vector<LogMessage>
    writeToAll( )

```

↳ iterate over all sinks  
& let them consume msg

src	test	inc
[ ]	[ ]	[ ]

CMakeLists.txt

Friend

[ ] operator = ( )

@ Bonus  
↳

operator<<

LogMessage msg{ };

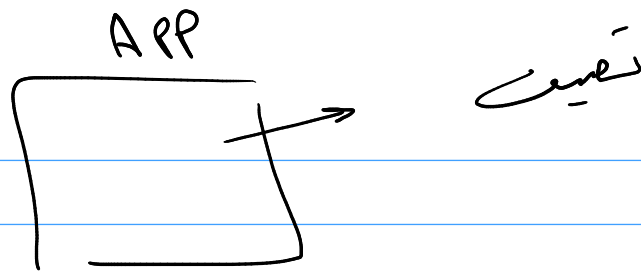
std::cout << msg << "\n";

→ [ ] [ ] [ ] [ ] [ ]

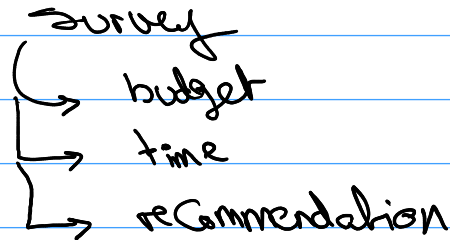
→ [EEA246][1/1/2000, 13:00:00][CPU][INFO][bad is 252]

when printed  
on console

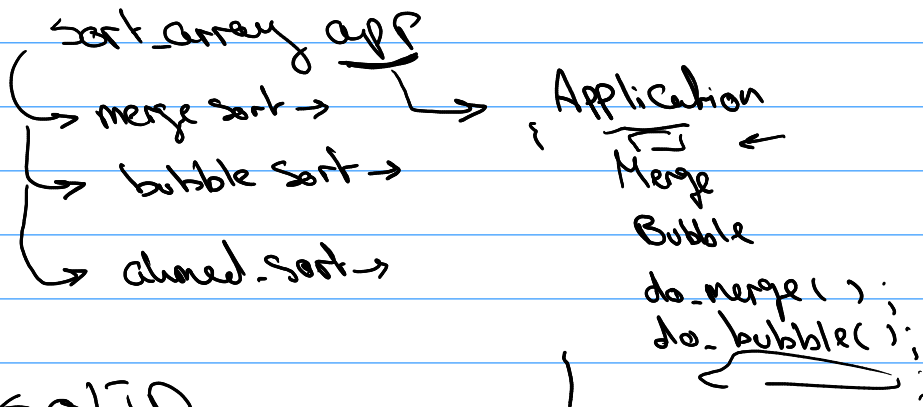
or  
written in a file



x way of transportation

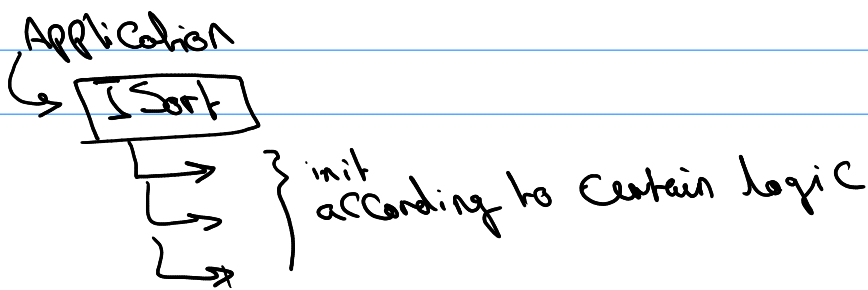
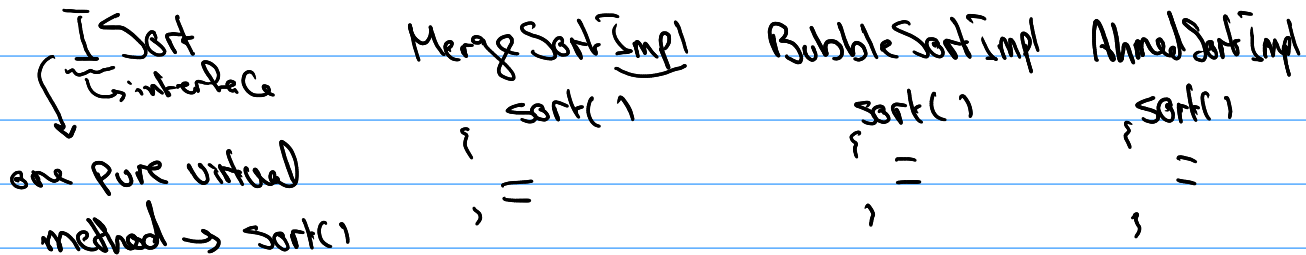


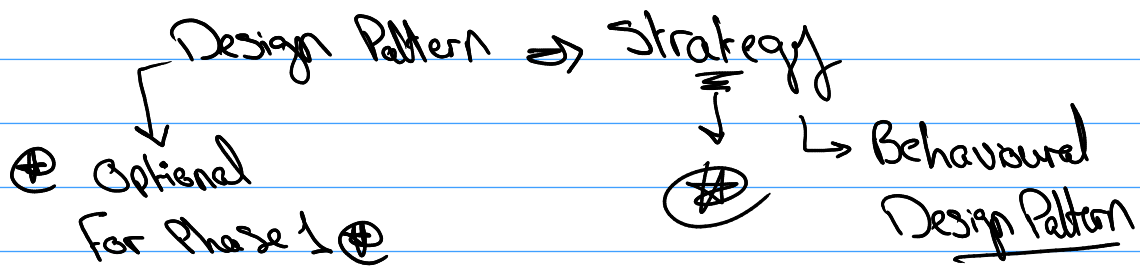
{  
 → Parameters  
 → Conditions  
 → choose right strategy of travel during runtime



SOLID

→ Open for extension  
 Closed for modification





Friend  $\rightarrow$  merge-sort;  $\rightarrow$  Merge Sort  
 $\rightarrow$  bubble-sort;  $\rightarrow$  bubble-sort;

```
class SortArray {
    BubbleSort bubble;
    MergeSort* merge;
    enum {
        SortArray(1) merge { merge }
    }
}
```

$\leftarrow$  new alg added

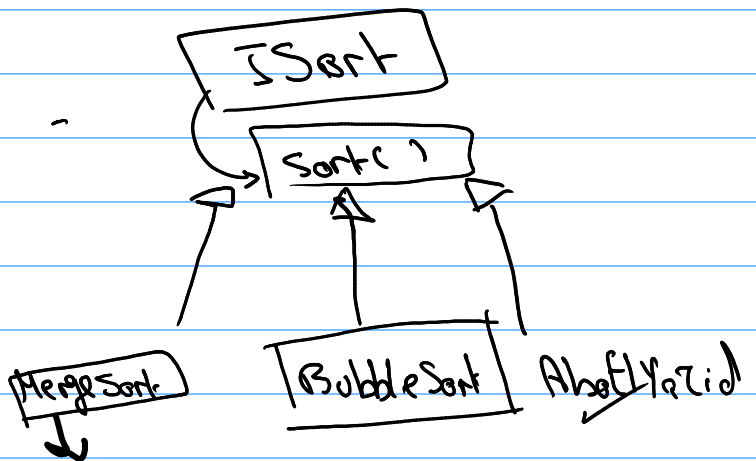
```
SortArr(enum ALG) {
    switch (ALG) {
```

$\leftarrow$  Case added

}

Strategy ✓

```
class SortArray {
    ISort* sort;
    SortArray* sort_ptr;
    SortArr(1) {
        sort_ptr -> Sort(1);
    }
}
```



⊕  $\rightarrow$  virtual bool sort(1) = 0; ⊗

