Design Patterns

Strategy Design Pattern:
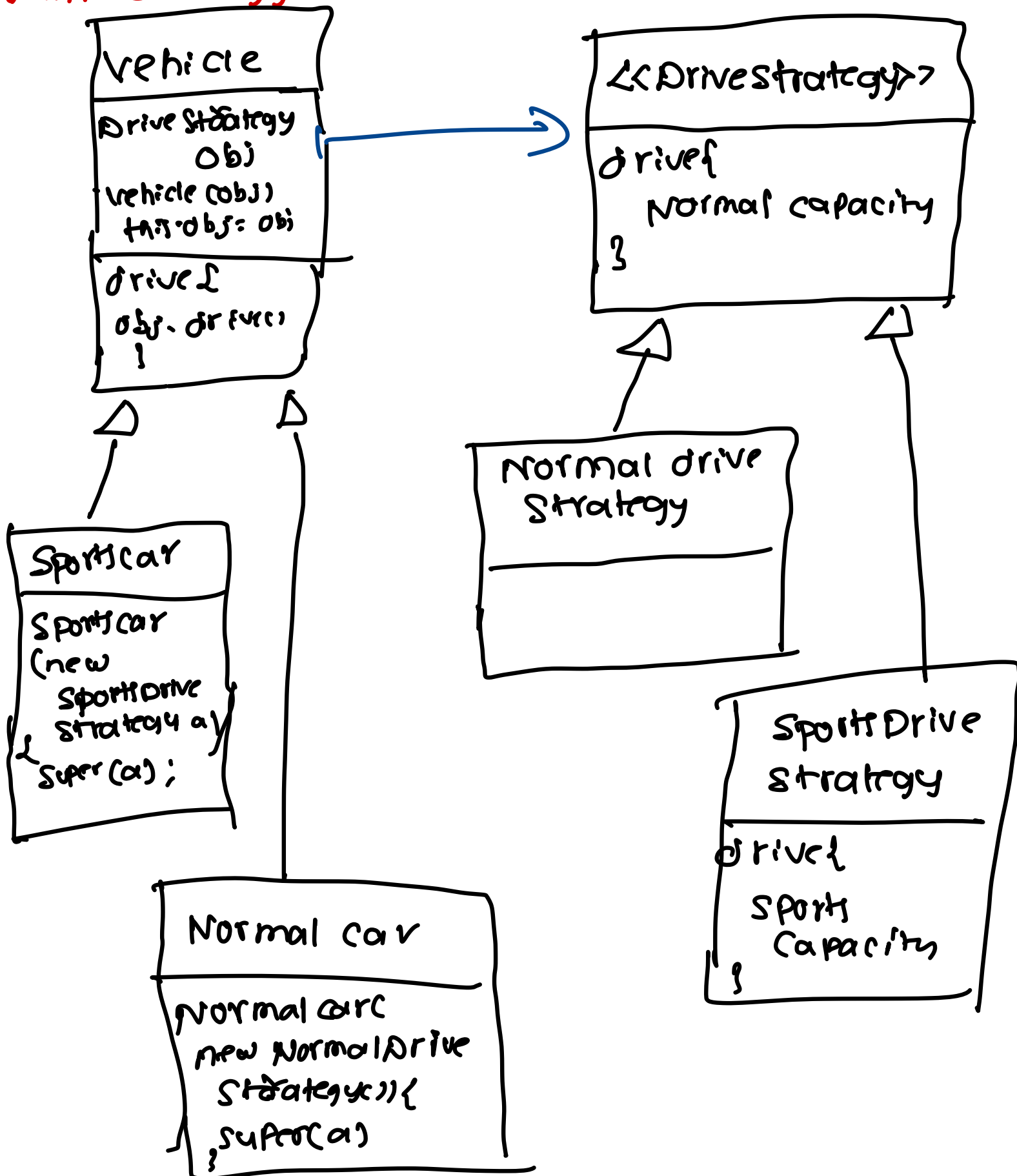
Without Strategy :

```
                    ┌─────────────────────┐
                    │      Vehicle        │
                    ├─────────────────────┤
                    │  drive{             │
                    │    Normal           │
                    │    drive            │
                    │       capacity      │
                    │  }                  │
                    └─────────────────────┘
                     △       △        △
          ┌──────────┘       │        └──────────┐
┌──────────────┐   ┌──────────────┐   ┌──────────────┐
│  Sports      │   │  Passenger   │   │   Special    │
│  Vehicle     │   │  Vehicle     │   │   Vehicle    │
├──────────────┤   ├──────────────┤   ├──────────────┤
│  drive{      │   │              │   │  drive{      │
│  sports      │   │              │   │  Sports      │
│  capacity    │   │              │   │  Capacity    │
│  }           │   │              │   │  }           │
└──────────────┘   └──────────────┘   └──────────────┘
```
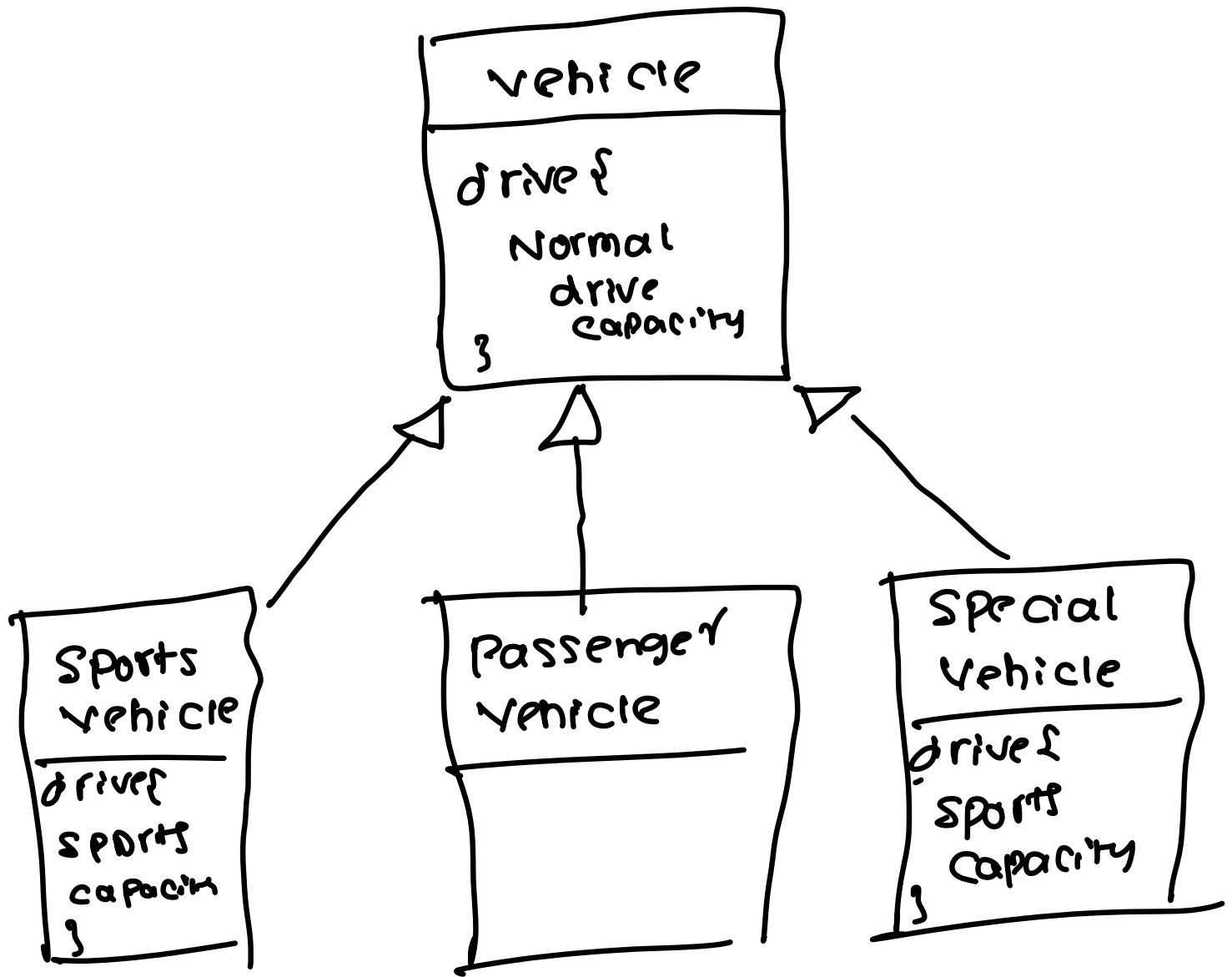
Both sports vehicle and special vehicle classes
override drive method similarly which
leads to duplication of code.

Therefore, with strategy we can eliminate
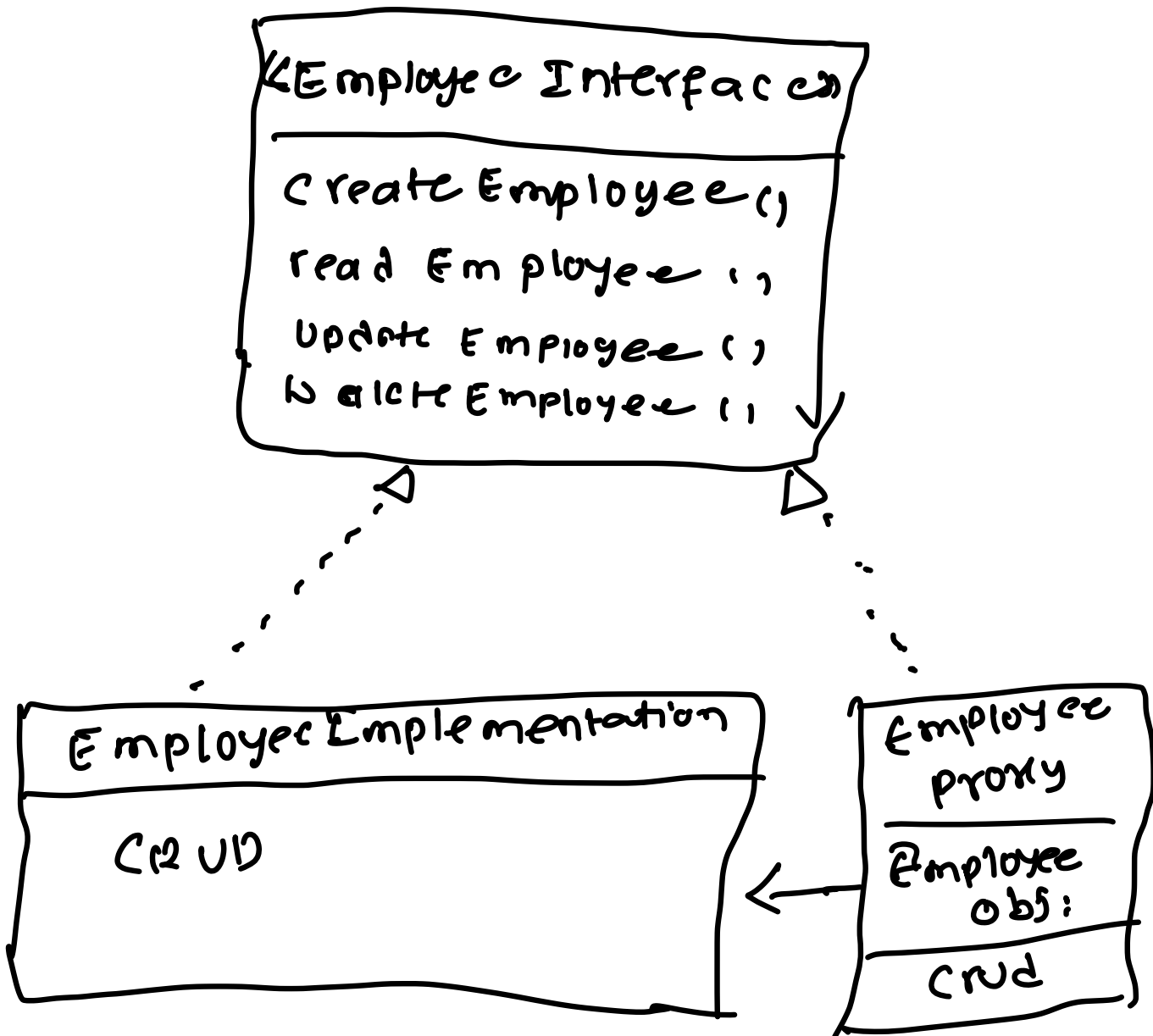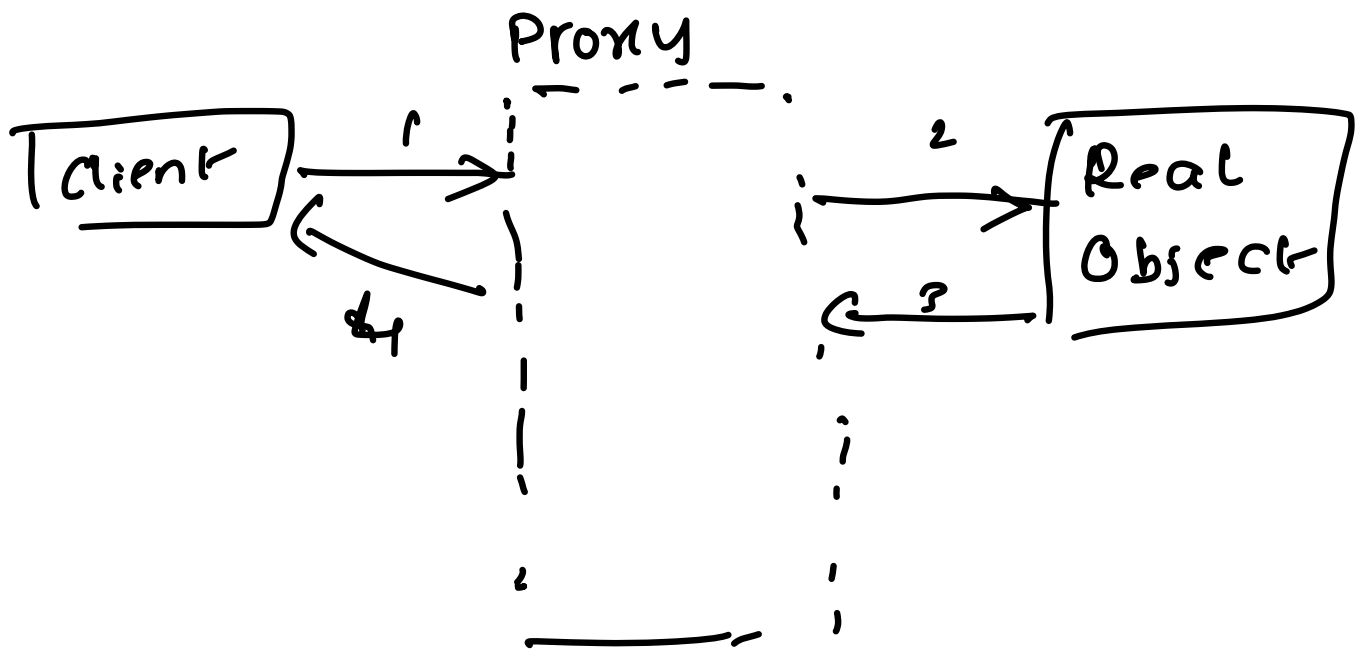duplication.

With Strategy :

```
Vehicle
-----------------------
Drive Strategy
     Obj
vehicle (obj)
this.obj: obj
-----------------------
drive {
obj. drive()
}
```

```
<<Drive Strategy>>
-----------------------
drive{
   Normal capacity
}
```

```
Sports car
-----------------------
Sports car
(new
Sports Drive
strategy a)
{
Super (a);
```

```
Normal car
-----------------------
Normal car(
new Normal Drive
strategy ())(
super(a)
)
```

```
Normal drive
Strategy
-----------------------
```

```
Sports Drive
strategy
-----------------------
drive{
   Sports
   Capacity
}
```

This way we eliminate duplication and
support open close principle

Proxy Design Pattern:

Proxy

Client → (1) → [Proxy] → (2) → Real Object

Client ← (4) ← [Proxy] ← (3) ← Real Object

```
┌─────────────────────────────────┐
│      «Employee Interface»        │
├─────────────────────────────────┤
│   Create Employee ()             │
│   read Employee ()               │
│   Update Employee ()             │
│   Delete Employee ()             │
└─────────────────────────────────┘
```

```
┌───────────────────────────────┐        ┌──────────────┐
│  Employee Implementation      │        │  Employee    │
├───────────────────────────────┤        │  proxy       │
│                               │        ├──────────────┤
│       CRUD                    │ ←───── │  Employee    │
│                               │        │  obj:        │
└───────────────────────────────┘        ├──────────────┤
                                          │  crud        │
                                          └──────────────┘
```

Why Proxy Design:

①

Proxy Server

Students —req→ Blocked Sites ⇄ Real Internet

②

Client ⇄ Cache ⇄ DB
① → Cache
② → DB
③ ←
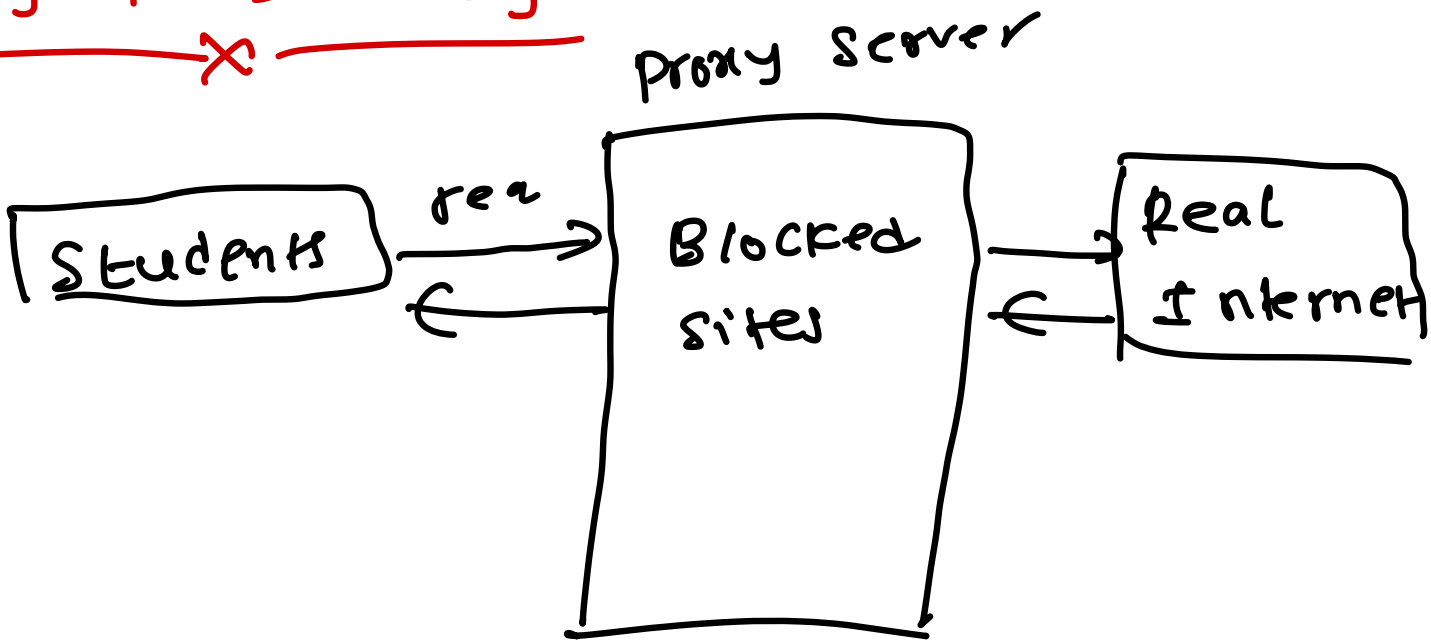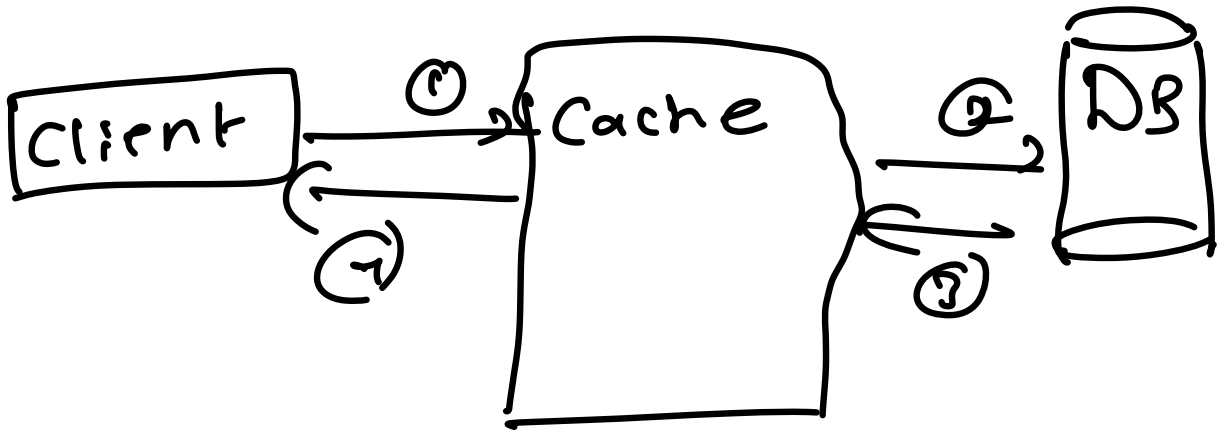
④ Pre-processing & Post-processing

Client —①→ [Pre-Pro] → logging

Code

[Post-Pro] → logging metrix update aretty