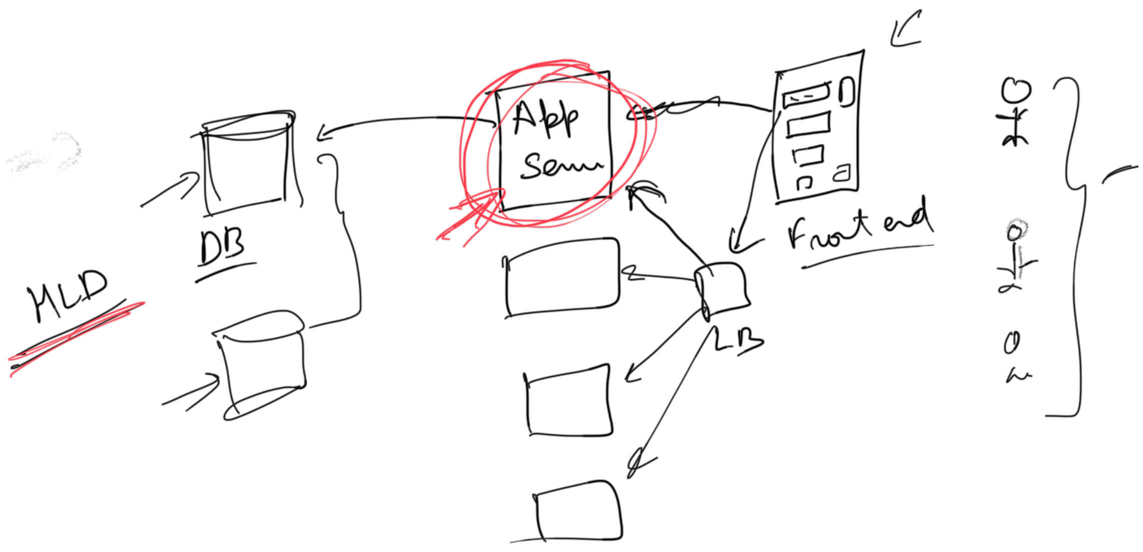


LLD

Object Oriented

Real World entity → Code entity

↓

Class

① Properties ⇐
② Actions ⇐

⇒ { → Attribut
⇒ { → Methods

Birds

↳ height
→ color
→ weight
→ flyingSpeed
→
⋮

↳ fly ⇐
eat
sleep
chirp
Hunt
lay eggs

```
class Bird {
private double ht, wt, fl
        String color;

        void fly( Str type ) :
   ⇒    // flap wings.

        }

{ String getColor(   ) {
→          return color;
  }
```

void setWon (. . .

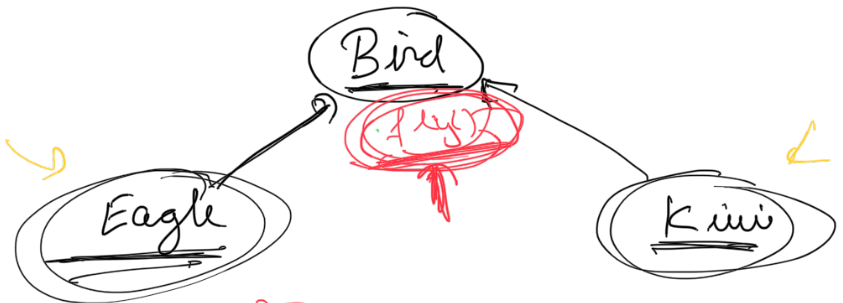
```
Bird eagle = new Bird( );
```
constructor

```
eagle.fly("eagle");   →   e

Bird parrot = new Bird()
    parrot.fly("parrot");


Bird kiwi = new Bird()
    kiwi.fly("eagle")      X
```

Inheritance



```
Class Eagle extends Bird {
    void fly () {
        Super.fly();
        // fly high
    }
}
```

```
Class Kiwi extends [
    void fly () {
        super.fly()
        // don't fly.
}
```

```
Class Parrot ~~extend~~ Bird {
        implem
    {   fly(   ){
        }
}

Parrot p = new Parrot();
```

p.fly();

Silent Killer

flap wings

Interface

Bird {
fly();
eat();
}

flyUtil {
fly()
}

Polymorphism

Bird b = new Parrot()

Bird b1 = new Eagle();

```
class Seller {

    void sellBird(Bird b) {

    Cost = getCost(b.getColor());

            b.color = '  '
}
```

b.color = 'transparent' Vs void setColor(String c
[● ● ● ●]
}

b.age = 1000000'

void setAge(int A)

}

class Boy {

```
Private
int wt, ht;
Private
int getWt(){

}

void setWt(int x) {
  if (x > --)

}
```

Encapsulation

sqrt(x)

```
Boy b = new Boy();
b.wt = 100;
b.wt = 100000;
b.ht = 100000
```

→ Inheritance
→ Polymorphism =
→ Encapsulation
→ Abstraction

→ On
→ off

```
Class X
{
public void func(A){
        A = M1(A);
        A = M2(A);
      return A;
  }

  Private M1(x){
    --
  }
```

1. Structural Exp
2. Behavioural Exp =
   → Use cases
   → functionality
   → feature

→ What it does NO
do

⇒ Search for products
↳ " "
↳ ' ______ '

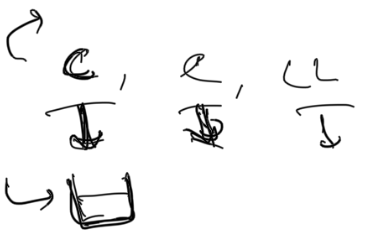
Library Sys

Functional use cases ↳
- ☆ Search a book
- ☆ Issue a book
- ☆ Retn a book
- ☆ Pay fine
- ☆ Reserve a book

Non-functional use cases } → Performance ch.

Design Coffee

① → Coffee Machn makes diff. beverages based on set ingredi



⇒

② Coffee mc → Propr
→ Actn
Beve
ingred.

① List reqr.

Cafe → coffee.

② List entities

↳ Properpeti

↳ action

③ ↓

④ Identify relation

Beverage has a recepie

↳ having multiple steps

→

ess

L

milk

Hot water.