3__09-07-2022

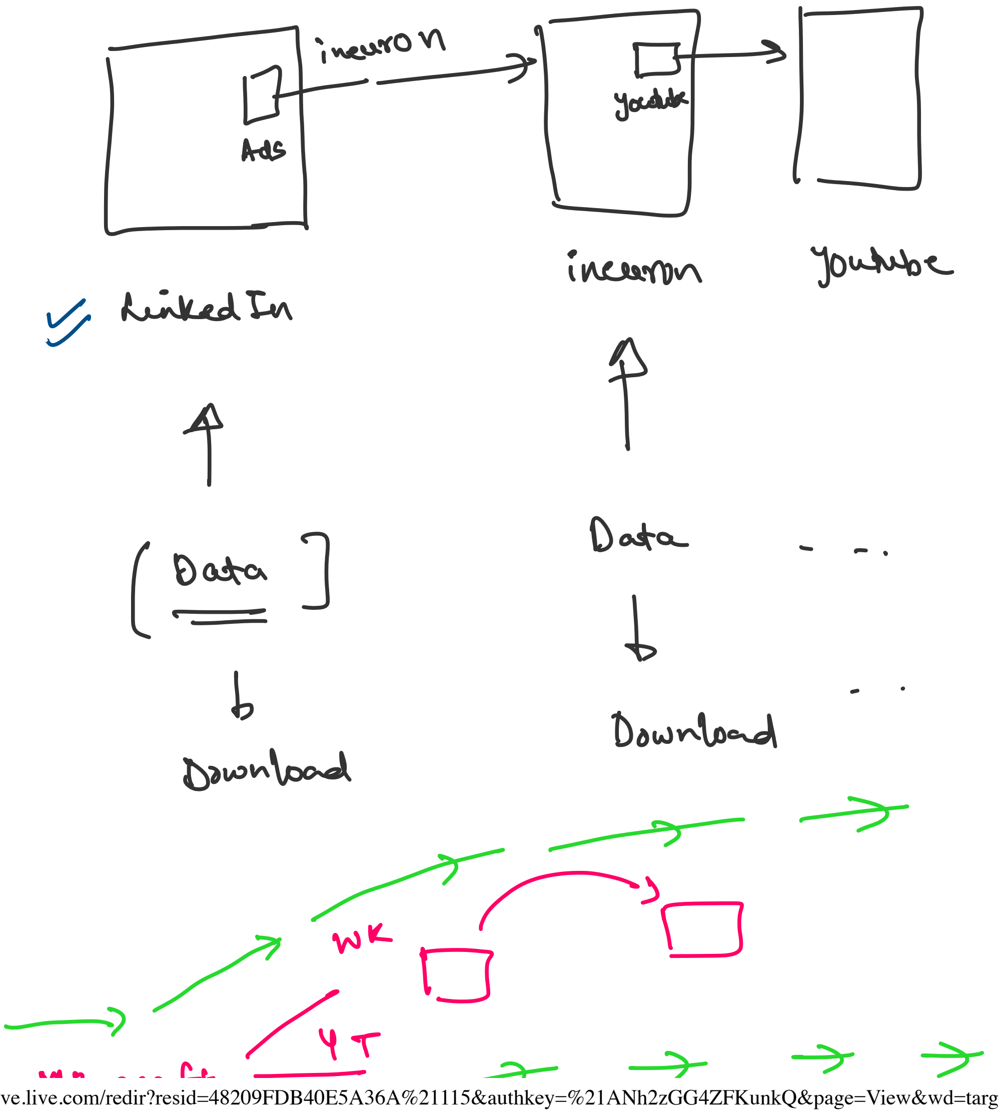
Saturday, 9 July 2022 8:03 PM

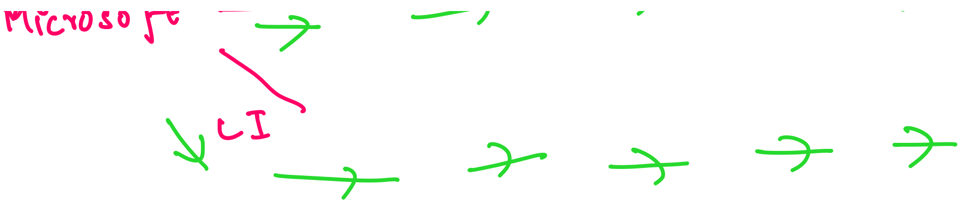
Topics to cover

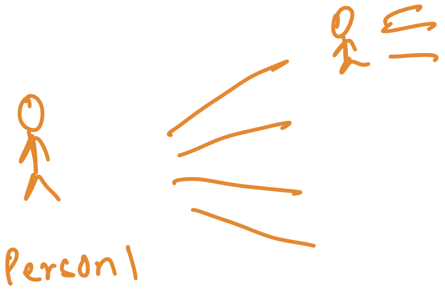
→ Steps in software development life cycle.

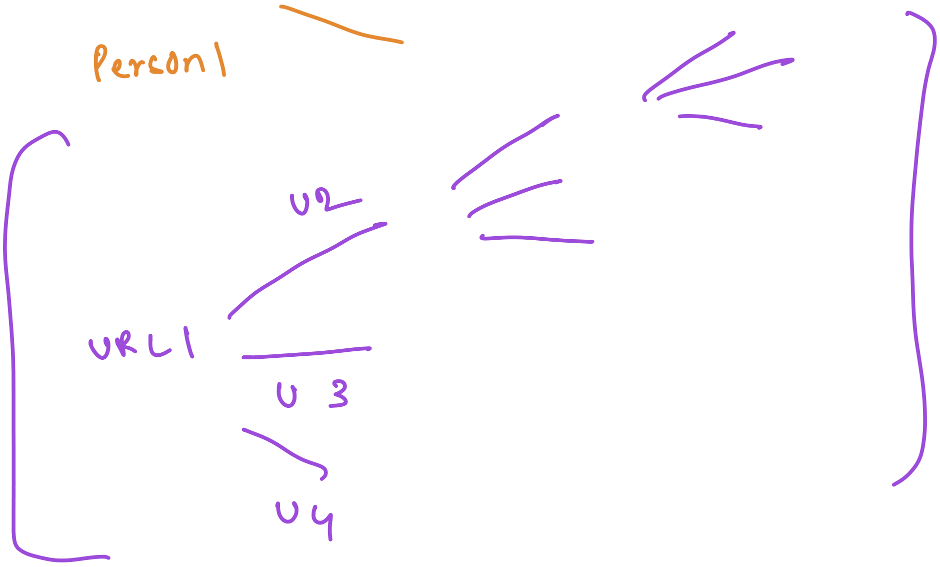
→ Object oriented programming — class
- object
- Encapsulation

Web crawler



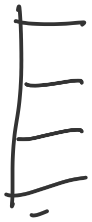






Requirement – Go to google and download all webpages

Google search
Apple



vast Data

↓

search time less

Machines? ✓

Scope - Download all content of all of the websites

Storage - How much storage required?

1.87 Billion

↓

2 B websites

2-3 pages

1000s pages

5- 50 webpages

Avg webpages = 100 per website

= 100 x 2 B

Total no of pages = 100 × 2 B

= 200 B

Size of one web page = 100 KB

Total size of content = $100 \underline{\underline{KB}} \times 200B$

$= \overset{\checkmark}{100} \times \overset{\checkmark}{10^3} \times \underline{\overset{\checkmark}{200B}}$ Bytes

$\underline{\underline{=}}\ \underline{\underline{2B \times 10^7 \text{ Bytes}}}$

Bytes
↳ KB
↳ MB
↳ GB
↳ PB

Total size = $\underline{20\ PB}$

$\boxed{1TB}$

1 machine = 1 TB

20 PB

Total machines $= \frac{20\ TB}{1\ TB}$

$$= \frac{20 \times 10^3\ TB}{TB}$$

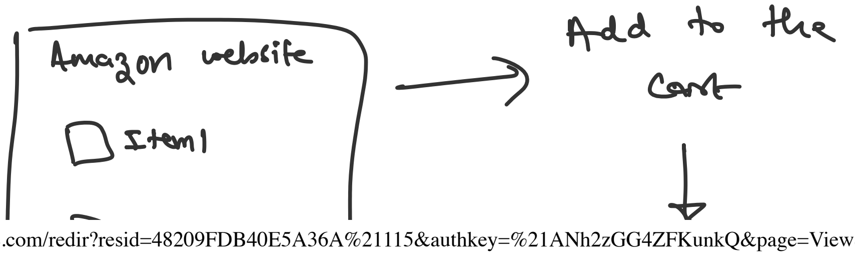
$$= \underline{20000} \text{ machines}$$

Parameters to consider for any of the project -

1. Requirement Gathering
2. Scoping
3. Estimation of data — Compute / Storage / Network
4. Breaking the bigger problem into smaller components

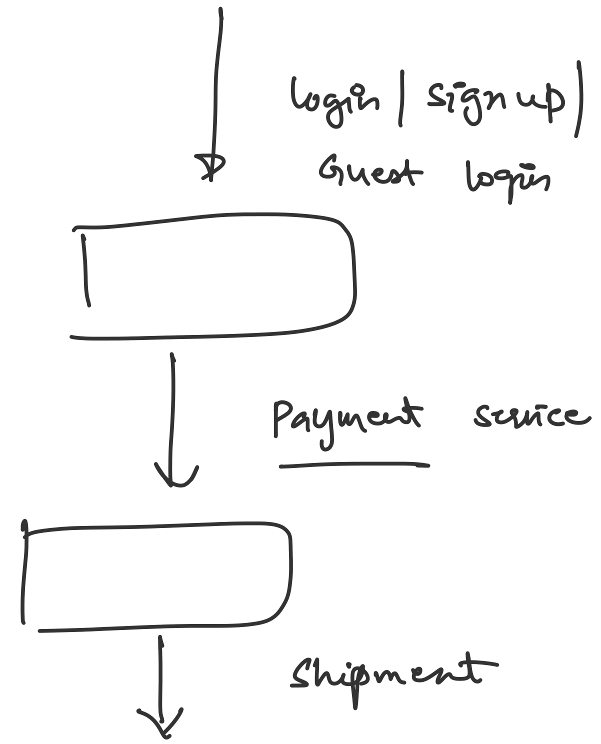
<u>Amazon</u>

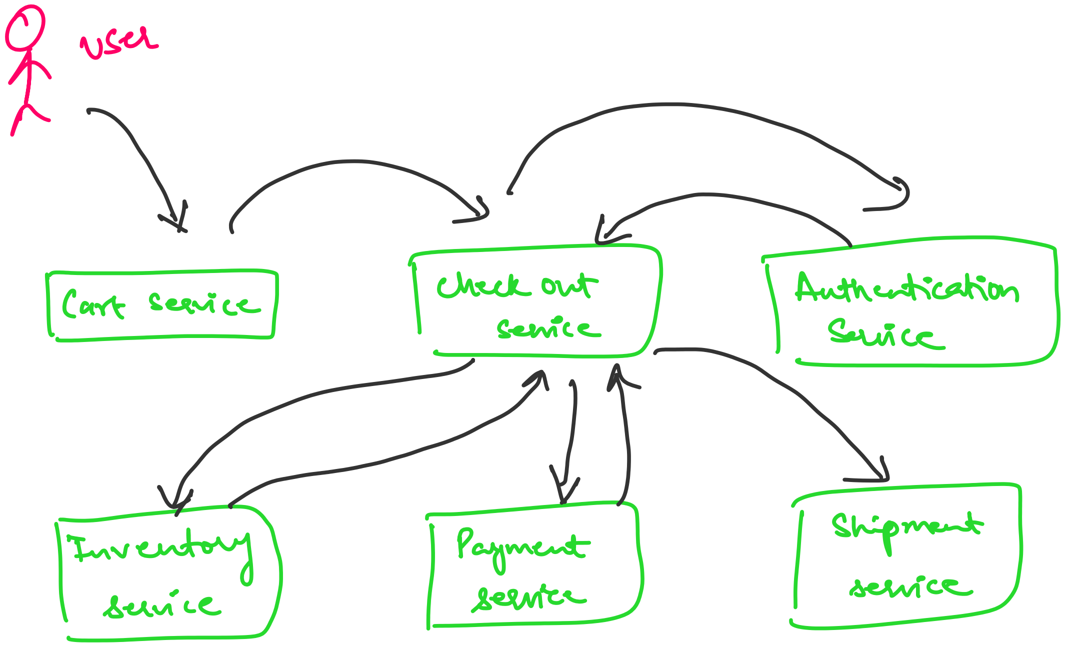
↳ user ability to buy an item



□ Item 2

Provide service for the same





SOA → Service oriented Architecture

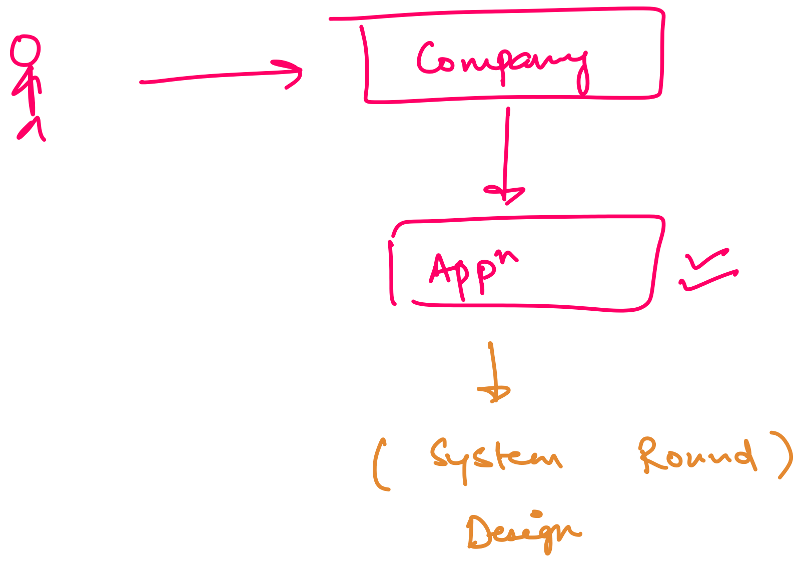
Steps -

1. Requirement Gathering
2. Scoping
3. Estimation of data
4. Break into smaller components

Planning or designing

5. Implementation of each component
6. Testing
7. Deployment

developers



Global Standards

< 2 yrs	2-7 yrs	> 7yrs
70% coding skills	50% coding	30% coding

30% system design | 50% design | 70% design

For any interview –

→ Any programming language

→ Good at DS / Algo

→ System design

- System design + fullstack
- System design + Backend
- System design ✓
  - → conceptual
  - → case studies
  - → Interview scenarios

System design

- Low level Design
- High level Design

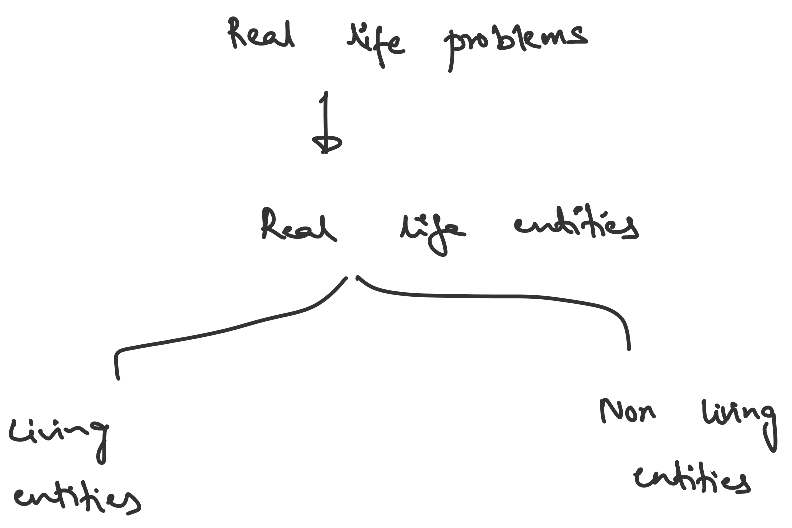
5. Implementation
6. Testing
7. Deployment

1. Rqt gathering
2. Scoping
3. Estimation
4. Breaking problem

Low level design

- object oriented principles
- Design principles (SOLID)
- GOF Design pattern
- Interview preparedness

object oriented programming



- State
- Behavior

Eg- Person

- State
  - Name
  - Age
  - Gendee
- Beh
  - sleep
  - Eat
  - walk

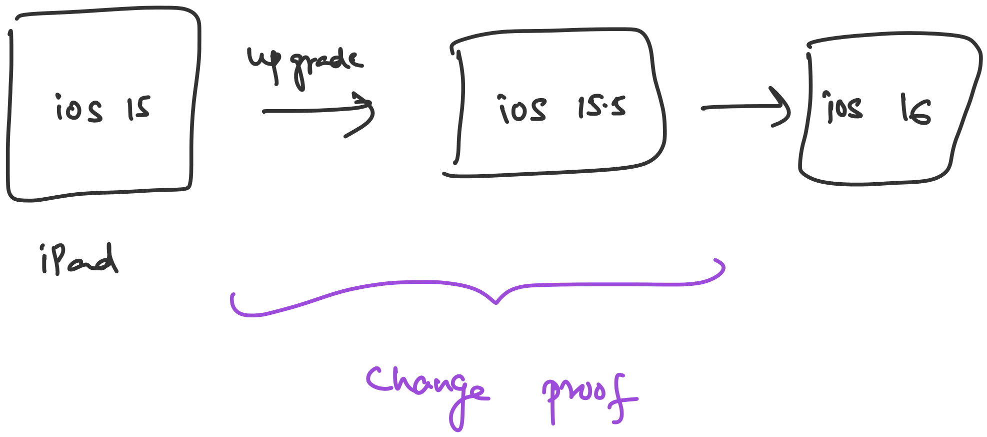
- State
- Behavior

Eg- Car

- State
  - Color
  - Brand
- Beh
  - Start
  - Stop

Real life problem

- Real life entities
- Relationship



Class and object

- Template

└ Blue print

Person

- Name
- Age
- Gender

} Variables

- Eat ()
- Sleep ()
- walk()

} Methods

Car

- Color
- Brand

} variables

- Start()
- Stop()

} methods

- Name = "abc"
- Age = 20
- Gender = Female
- Eat() – Eat more
- Sleep () – Sleep more
- walk () – walks slow

Object

↳ Instance

↳ Reference

Intellij download link - https://www.jetbrains.com/idea/download/#section=mac

```
public class Person {
   String name;
   int age;
   String gender;

   public Person(String name, int age, String gender)
   {
      this.name = name;
      this.age = age;
      this.gender = gender;
   }
}

public class Program {
   public static void main(String[] args) {
      Person person = new Person("abc", 30, "male");
      System.out.println(person.name);
      System.out.println(person.gender);
      System.out.println(person.age);
   }
}
```

Output:

```
abc
male
30
```

Encapsulation

↳ Binds state and behavior together

state ↓ variables

behavior ↓ methods

name = "abc" → name = "xyz"

age = " 30 " → age = 30

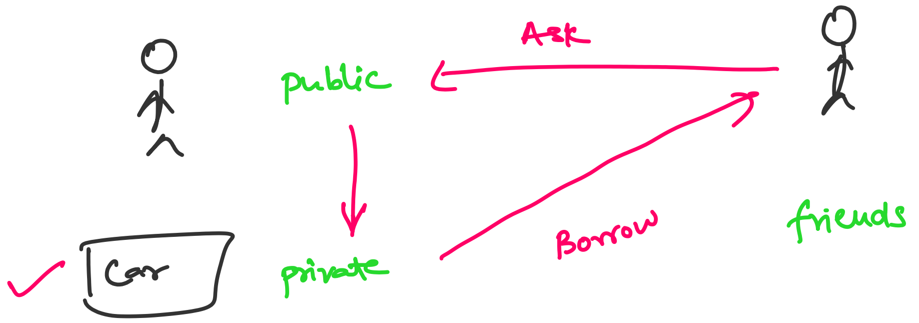
Such things can be avoided using access modifiers

Encapsulation

- → fields private] No direct modification
- → methods public

↓

Modifications can be done under control



Encapsulation

Hiding the details

↓

No one can change it

↓

Using Access modifiers

↓

"private"

Provide some way to change the state

Know the state

↓

Getting the state

↓

Get method

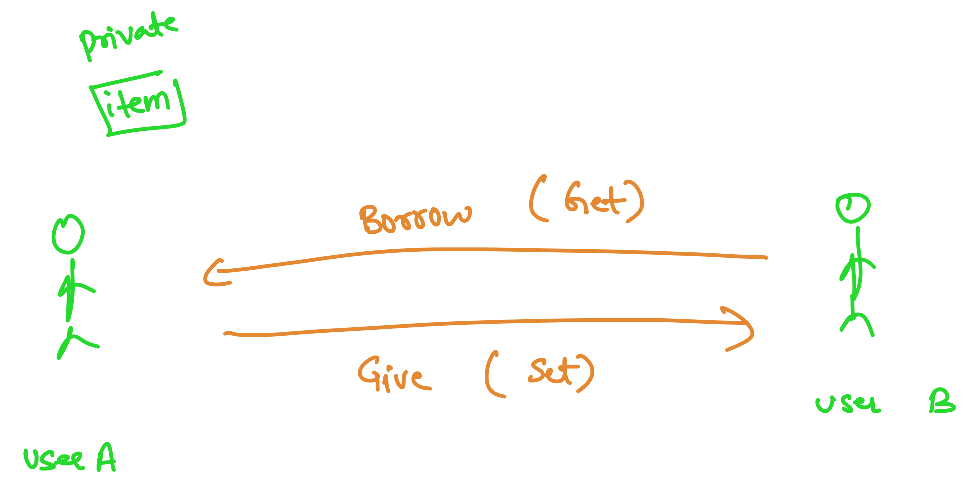
Update the state

↓

Setting the state

↓

Set method



↓

Getter method

Setter method