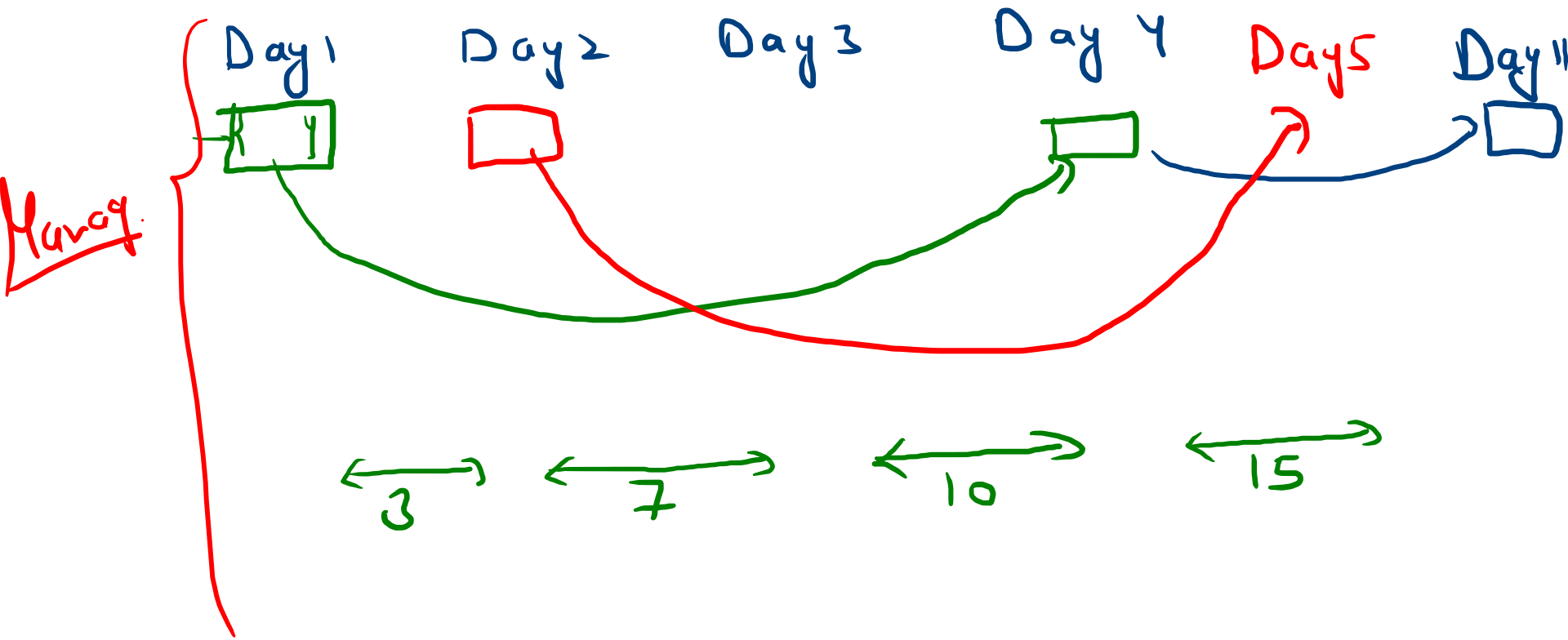


Revision.



OOP.

Procedural/Structured progr. (C, Pascal)

- * Thinking in terms of procedure or function.
- * Multiple function.

tea()

{

TakeBowl()

Add water()

Add Sugar()

Add tea()

}

functional
procedure

OOOP is thinking in terms
of objects.

(new thought process)

{ objects
data
behavior }

Thinking is OOP

Take a flight

Taking a flight.



Step 1: What different objects Involved in it?

{ Aeroplane, Airhostess, passenger }

Identify the things are involved.

Ice → liquid

1. Identify data → State ^(25 35 20 30)
~~(0, 0, 0, 0)~~
airline, make, type, position // data
~~Arrival time~~

2. Identify action → behavior
take off(), land(), cruise()

Aero plane

airline, make, type, position // data
takeoff(), land(), cruise()

Air hostess

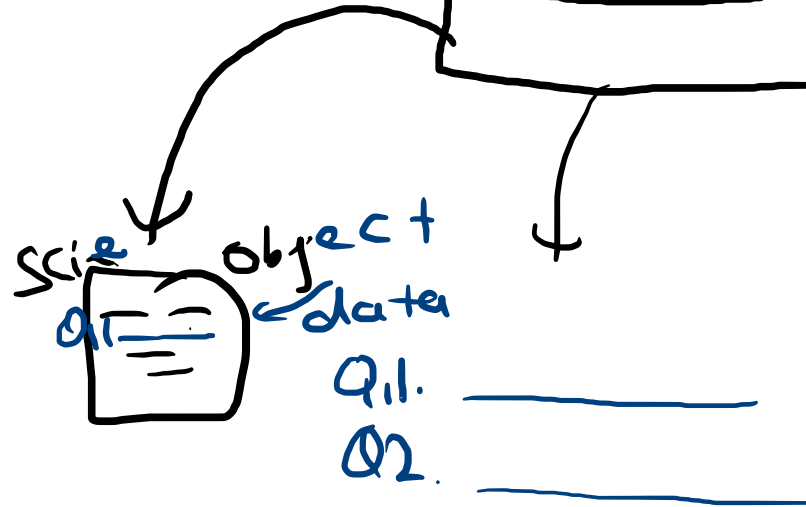
name, address // data
wish(), serve() // action

Passenger

name, address // data
checkin(), walk(), findSeat() // action



Memory
allocati.



```
class Planet {  
    name, location, dist from sun // data // state
```

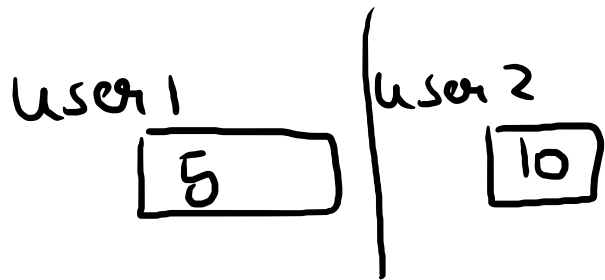
```
    revolve(), rotate() // action  
    behavior.  
}
```

```
Planet earth = new Planet();  
Planet venus = new Planet();
```

```
earth.revolve();  
earth.rotate();
```

Stock Market APP.

↓
user 1. buy Stock (5)
user 2. buy Stock (10)



How many sto.
?
5
buy

data
↓
no. of stat = ~~0~~
5

User user1 = new User ("Shubh", "era", "Paw")

User user2 = new User ("vijay", "--", "--")

Online Shopping System.

- Product

id, name, price, quantity, manufacture
update price(), order(), add disc.()

- Shopping Cart

id, items
add(item), remove(), place


- Customer

name, address, mobile

login(), logout(), select product()

Super class of every class

Object (equals ()
to String ()
hashCode ()
clone ()



Equals \rightarrow When we compare two
Objects.

Important thing equal.

equal

1. Reflexive

$x.equals(x)$

return true

2. Symmetric

$x.equals(y)$
 $y.equals(x)$

3. Transitive

$x.equals(y), y.equals(z)$
 $x.equals(z)$

4. Consistent

$x.equals(y)$

5. $x = \text{some value}$

$x.equals(\text{null})$

// false

good hashCode method properties:-

1. Obj1.equals(Obj2) is true
 $\text{Obj1.hashCode()} == \text{Obj2.hashCode()}$
2. $\text{Obj.hashCode()} \rightarrow$ return
same
value

Method Overloading

A Method having same name but
diff. parameter.

2 ways

A Constructor can only
be called by
another Constructor.