Lecture 36 : prototype design pattern.

Introduction
- simple and solves specific type of problem.
- we have used it before without knowing it's a design pattern.

* problem solved by prototype

- Now, we have created one object of class A.
- And we need multiple object of same class but there is too much similarity b/w these much created objects.

A a1 = new A();
A a2 = new A();
↓
allocation of memory in heap.

- And we know object creation in ~~java~~ are most expensive task as it may include
  1. DB connection establishment.
  2. complex calculation
  3. load big and complex file.
- These problem is fixed by prototyping design pattern.
  ↳ Let's see how with an example.

Example : NPC - Non player character. (NPC)

① Let's take Example of Non-player-character in a game which are in background or player doing certain & speed task.

② So, whenever we start game all these NPC objects need to be created so instead of making all these characters seprate objects we can reuse created objects by clonning in short using prototype design pattern.

What is and How does prototype solve problem?

1) Create one object of class suppose do1 and we need one more object then instead of creating it again we can do is copy them.

2) in short copy the existing object to reduce or save time in complex operation.

3) NPC *n1 = new NPC;
NPC *n2 = n1 → clone();

NPC
// expensive op
clone();

* Prototype pattern only used where there are only little changes bet multiple objects.

Page No.
Date

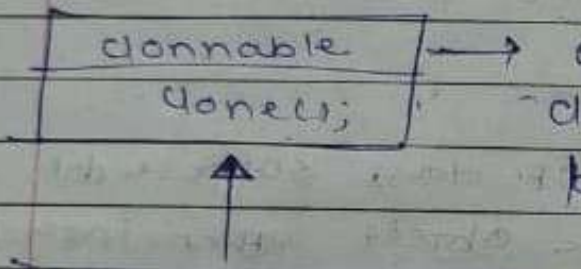
How does it clone()?

→ We know about copy constructor.

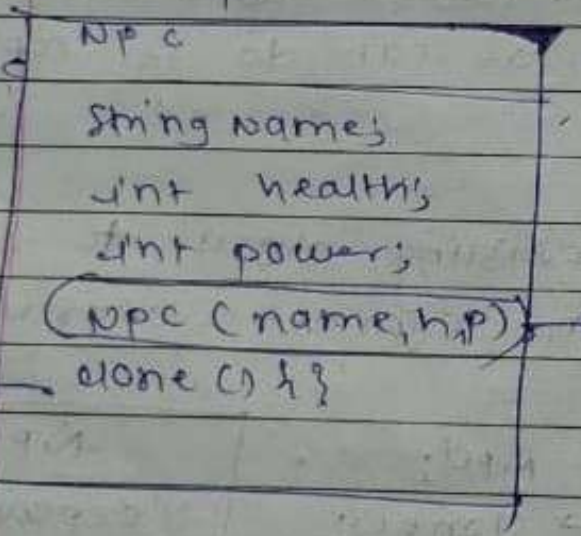
types:- default,
parameterized,
copy constructor (How object should be copied).

(5) Now, we copied all properties of n, to n2 but we do not need some properties. so we can update it using details getters setters. of that.

UML for NPC example



→ whichever class inherit clonnable class, that class / objects can be clonned.



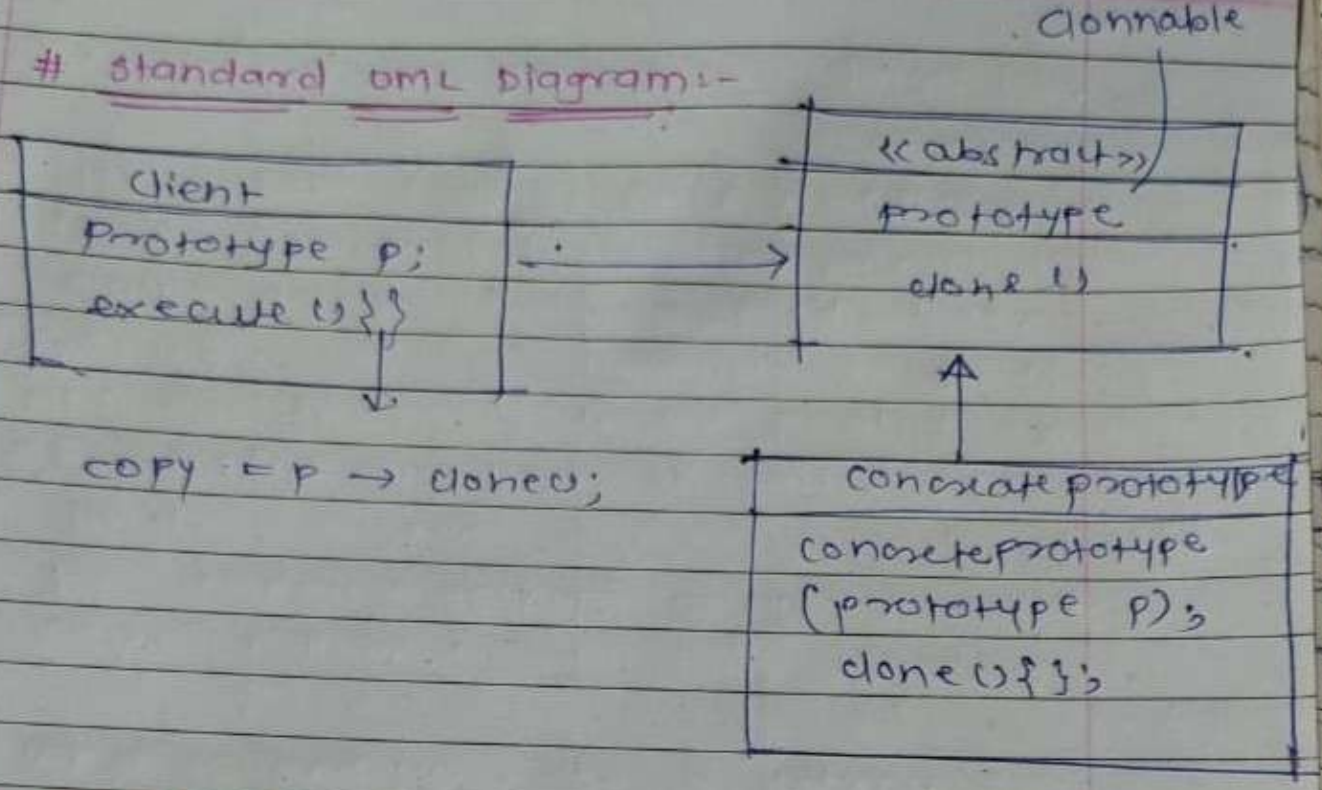
→ In java, clonnable ↑ class is known as marker interface as it's marks the class which are clonnable.

→ this has copy constructor which shows how it should be copied or helping to copy.

```
clone()
{ new Npc (*this);
}
```

```
Npc(Npc*n){
  this.name = n.name;
  this.health=n.health;
  this.power=n.power;
}
```

Standard UML Diagram:-



Standard Defn

If let you create new object by copying [clonning another instance].

Real world use case

① Games NPC characters.
② Multiple objects with less change in properties.
③ Complex / Expensive Object creation.