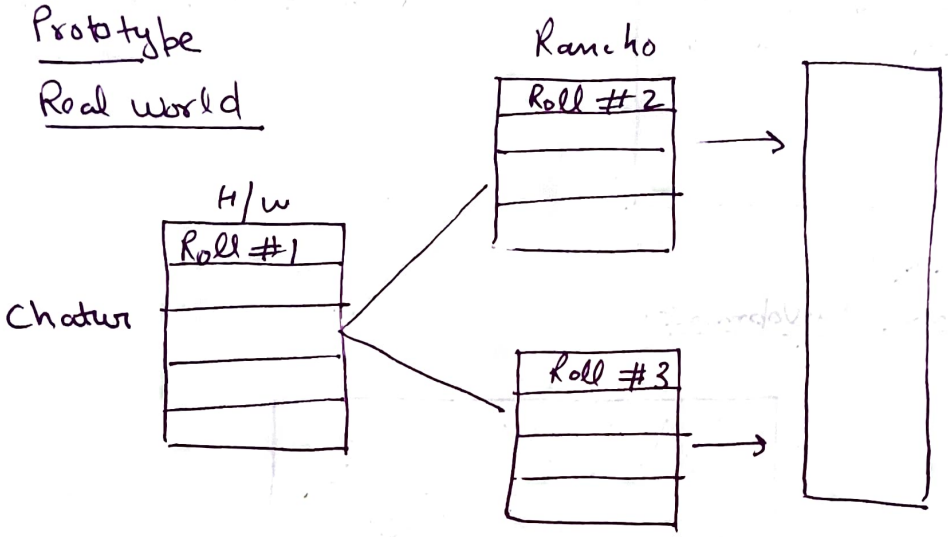
Day 04: Design Pattern: Prototype & Registry

① Prototype
- → Motivation
- → solution
- → Implement

② Prototype Registry

Prototype
Real world



User API → Create User.

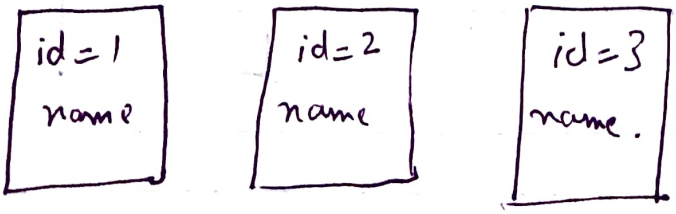
→ Load test — Jmeter.

Random API

1000

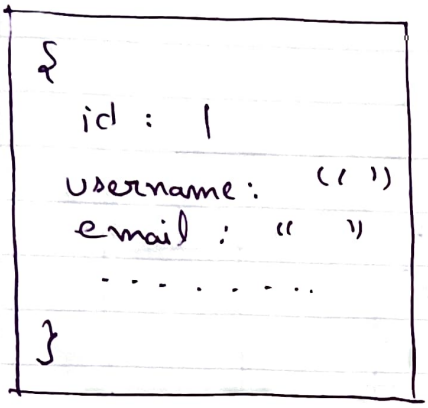
1000
- —call the API
- — create our user

Rate limited



Prototype flow

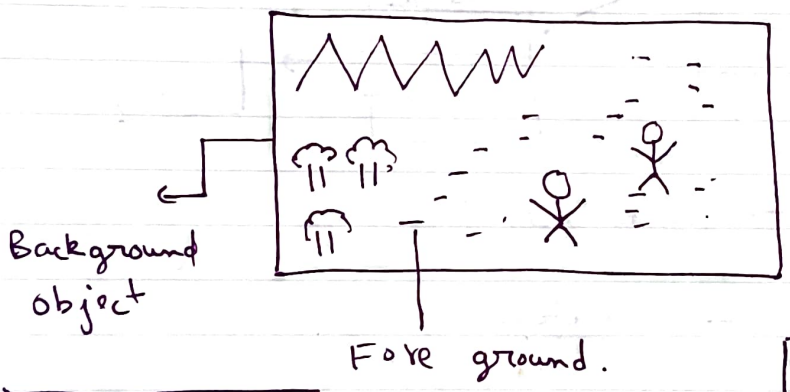
① call API once

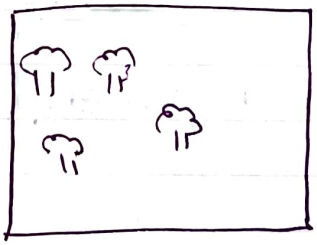


② copy user.

id = 2
email = " . . . "
name ⇒ " . . . "

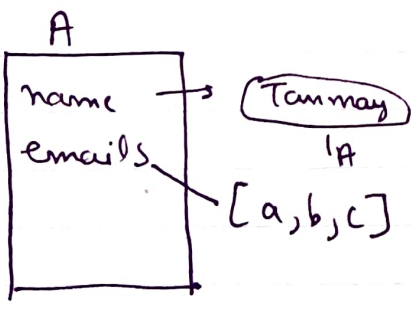
① Testing
② Game development.

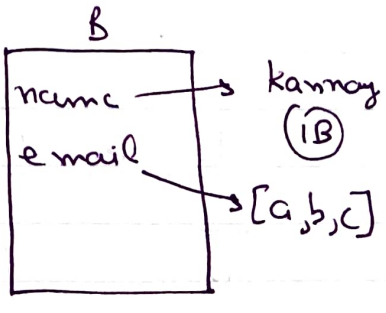




Implementation

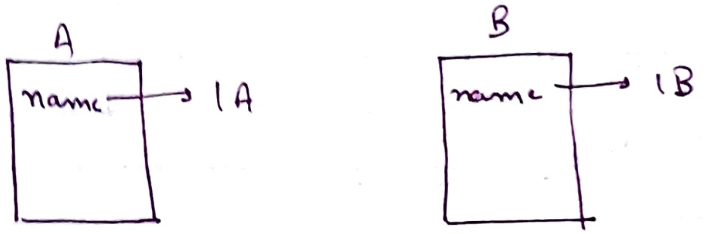
① cloning — shallow
└ deep.





String ⇒ Immutable.

Deep → copy by value.



→ Step 1 - Create a cloneable interface

```
interface  object cloneable {
    object cloneable clone();
}.
```

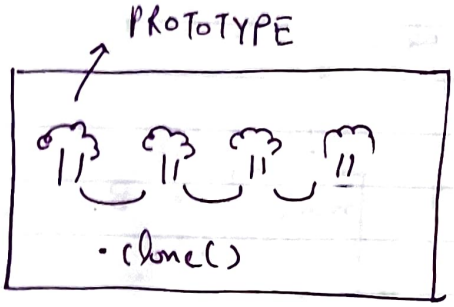
→ Step 2 - Create a concrete class

```
class user imp. object cloneable {
    @override.
    user clone() {
    ....
    }
}.
```

→ step 3 - Create a prototype

sample model blue print.

→ Step-4 - clone the prototype as required.



cloneable {

}

↓

Tree {

}

Registry

Graphical object

↓

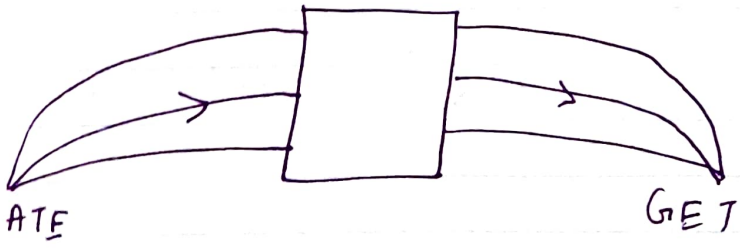
Bg object

TREE MOUNTAIN BUILDING

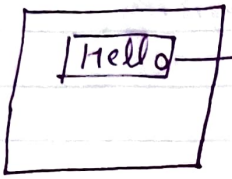
(1) We want to store multiple prototypes.

(2) store our prototype and get them when needed.

Registry



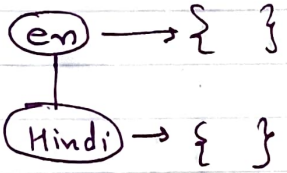
Localisation (i18n)

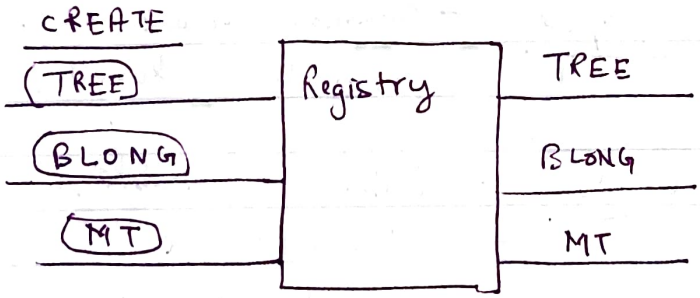


en

{
welcome text : Hello
}.

{	{
welcome text : "Namaste"	welcome text : "Vanakam"
}	}






```
interface Registry {

    store (Obj object proto, Type type);
    Obj object get (Type type);

}.
```

→ class BG Registry {

Map <Type, BGobject> registry;

→ get () {

. . . . .

}

→ store () {

. . . . .

}

}.

① Register → store, get

② Deregister. → ~~<~~

③ Modify & Get ①→

① Cloneable Interface — common for all obj. that can be cloned.

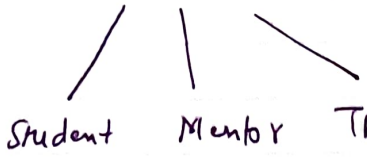
② Concreate classes — impl. cloneable.
— impl. clone
→ shallow vs deep copy.

③ Prototype Registry — • get }
— • store }

④ create prototype

⑤ clone the prototype • clone()

① User prototype ☆



— prototype
— User registry

② Make Registry generic

→ Java generics

This is assignment