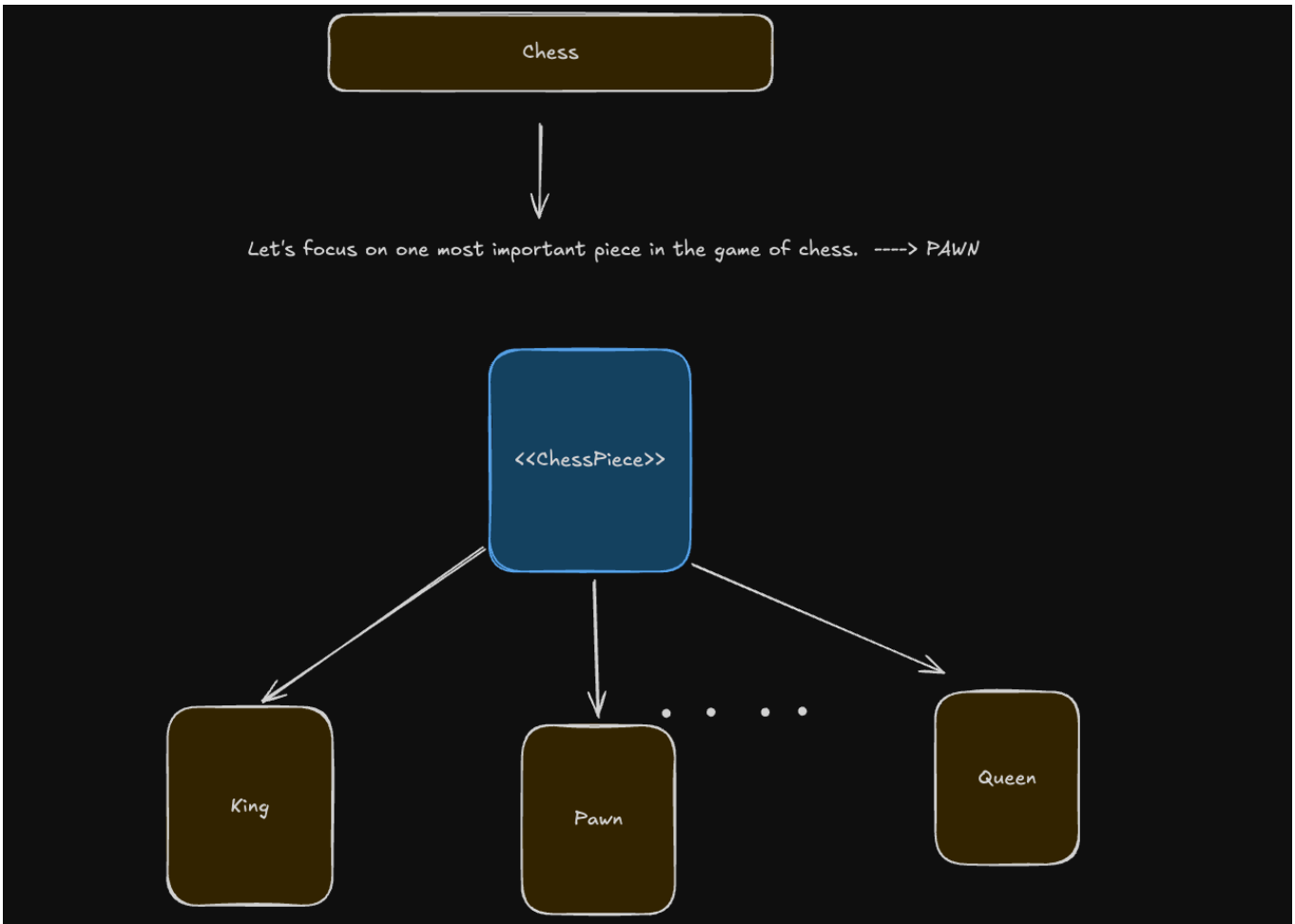



So let's say, in an interview you were able to think the best possible design for the family of chess pieces.

Now is the time to test your logic and create objects of these concrete chess piece classes.

```
Pawn p1 = new Pawn();
Pawn p2 = new Pawn();
Pawn p3 = new Pawn();
    .....
    .....
Pawn p16 = new Pawn();
```

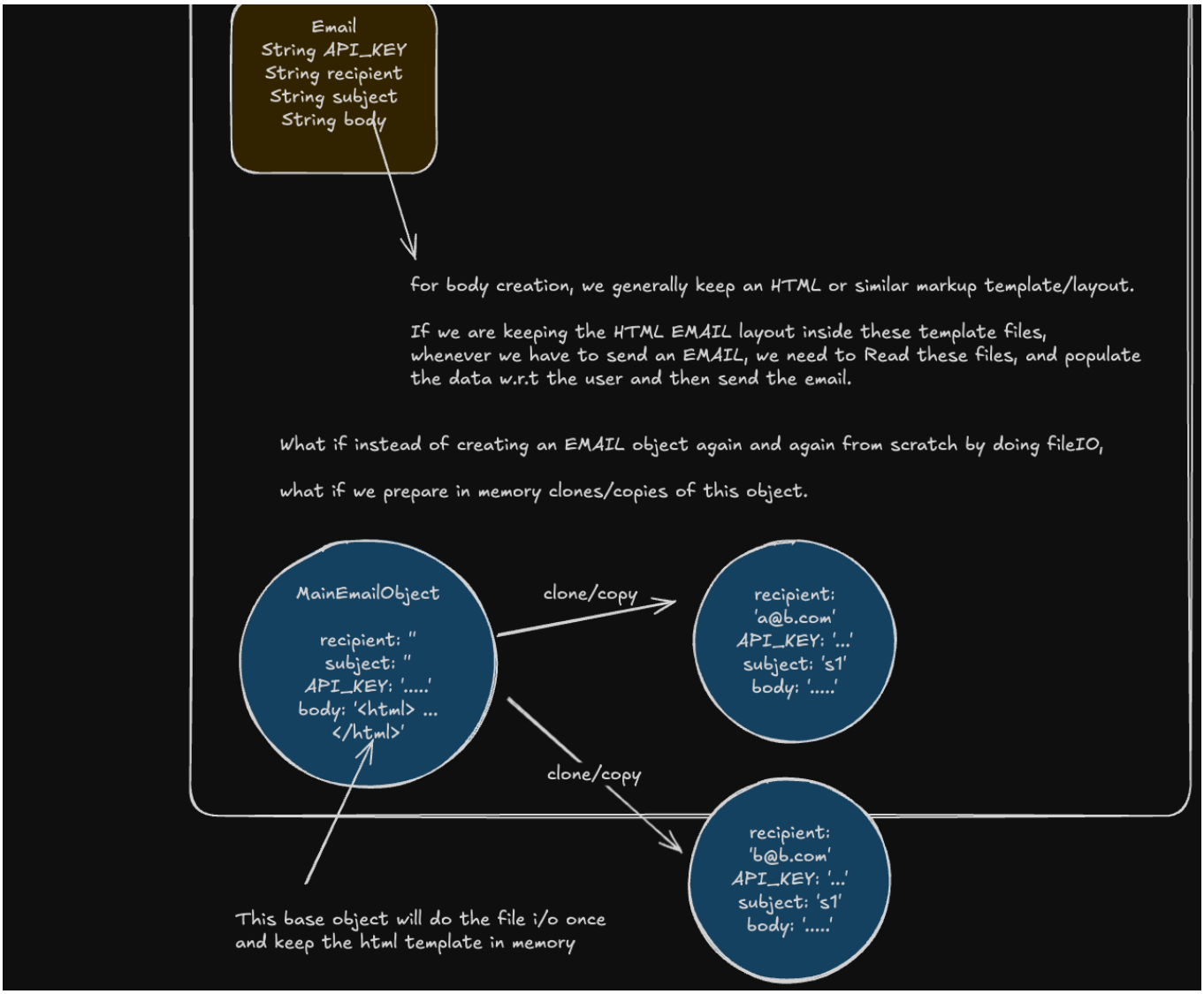
Should we go for this kind of multiple object creation one by one ?

Let's say we can't run a loop, as we might need to configure each pawn on the creation (say for coordinates a2, b2, c2 ....)

Now let's say the interviewer says that not just the configuration of every pawn is a problem but, creating a pawn object again and again can be an expensive operation also. And we need to optimise.

Real life:

In real systems, this is a very common problem of creation of multiple similar objects which are expensive to create.


Email
String API_KEY
String recipient
String subject
String body
for body creation, we generally keep an HTML or similar markup template/layout.
If we are keeping the HTML EMAIL layout inside these template files, whenever we have to send an EMAIL, we need to Read these files, and populate the data w.r.t the user and then send the email.
What if instead of creating an EMAIL object again and again from scratch by doing fileIO,
what if we prepare in memory clones/copies of this object.
MainEmailObject
recipient: ''
subject: ''
API_KEY: '....'
body: '<html> ... </html>'
clone/copy
recipient: 'a@b.com'
API_KEY: '...'
subject: 's1'
body: '....'
clone/copy
recipient: 'b@b.com'
API_KEY: '...'
subject: 's1'
body: '....'
This base object will do the file i/o once and keep the html template in memory

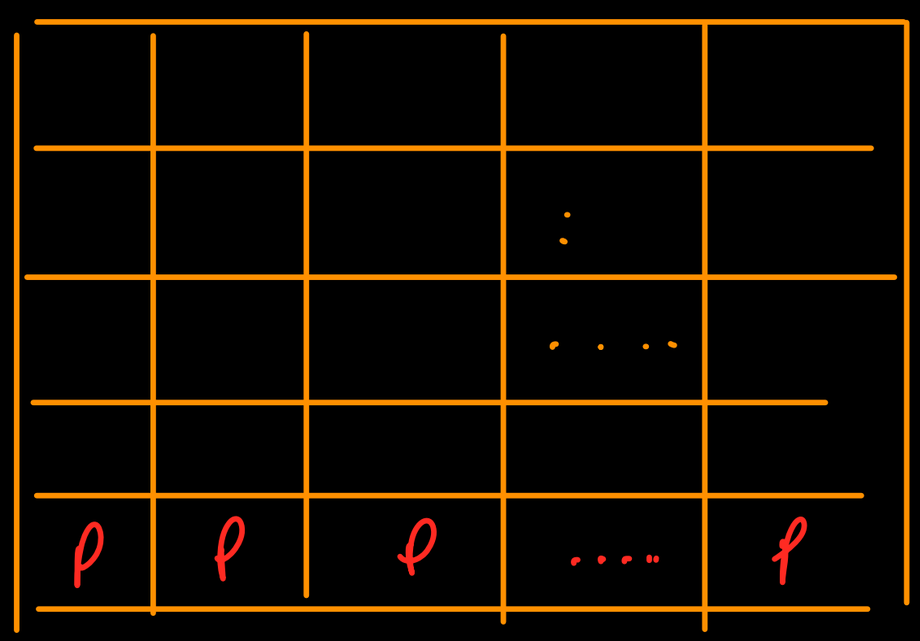
# Problem Statement →

→ Assume we have an object of a class.

→ We want to create copy of this object.

# Scenario where this problem can come up?

(1)




→ chess board

→ for this chess board, we have piece class & we might need to create multiple copies of the same piece. Ex → Pawn

→ instead of creating multiple instance of Foam, we can create copies of it.

May be the constructor call is expensive.

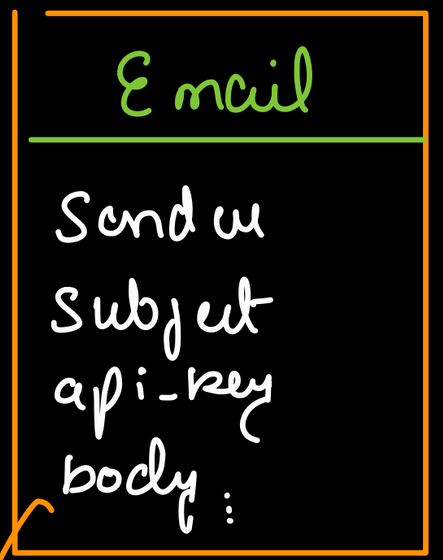
Example of expensive constructor call →

1) DB Connection object

2) Say we have an Email class, that creates a new object everytime for sending Email. for that the constructor does file IO (may be to load the template file) & then does string manipulation for

updating the template for each user.
Here file IO is expensive.

# Solution 1 → Naive Solution




```
Hi, {{ name }}
Welcom to the platform.
```

new object

```
Email e = new Email (....);
```

to create a copy lets put all the properties one by one to a new object:

```
Email copy = new Email ( );
copy.setSender (e.getSender());
copy.setSubject (e.getSubject());
copy.setBody ( e.getBody().replace
                                (—, ---))
```

Both of our code snippets are in the driver class.

Q is this solution violating anything ?

1) Dry Principle ⟶ create a function to resolve it.

But still it violates SRP & OCP.

# Another naive solution can be to use a copy constructor.

```
class Email {
    private Sender sender;
    private String body;
    private String subject;

    Email ( Email e ) {
        Email copy = new Email();
        copy.setSender (e.getSender);
        :
        return copy;
    }
}
```

→ copy constructor.

Q2) How to improve ??

→ how about we put the new copy func^ inside the class only.

Q3) what is the problem with current solution ??

```
class A {
    ≡
    A (A a) {

    }
}
```

→ copy constructor

class B extends A {

B( ) { → copy constructor

}

(if we forgot to add a copy constructor)

⟹ Driver class

```
if (sample instanceof A) {
    A copy = new A(Sample);
else if (sample instanceof B) {
    A copy = new B(sample);
}
```

OCP *

# How about the object whose copy has to be created gives it's own cloning method.

← Sample.clone()