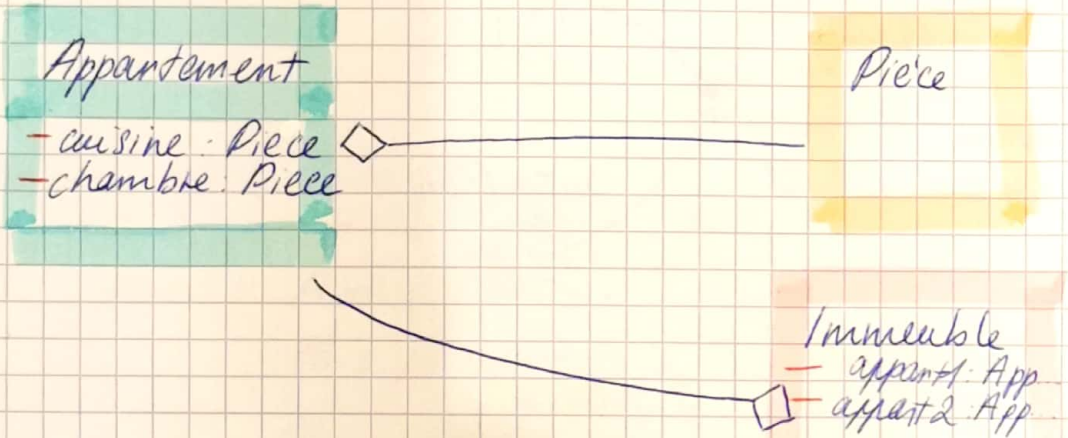
3. Composition, copie d'objets

Ex 21

1. Un appartement est composé de pièces, un immeuble est composé d'appartements

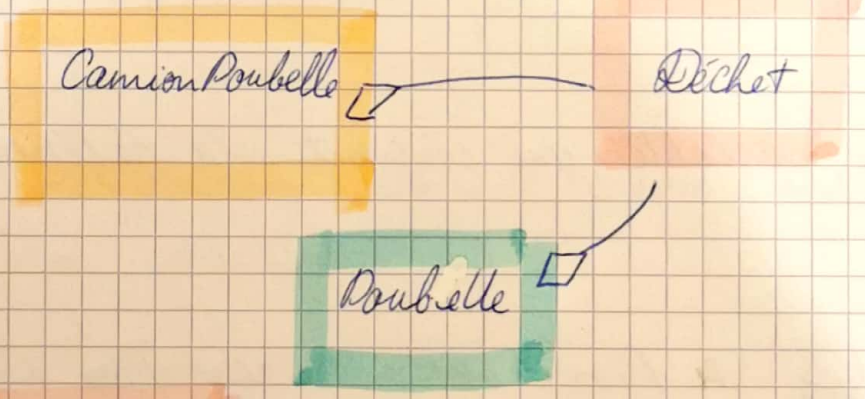


- private

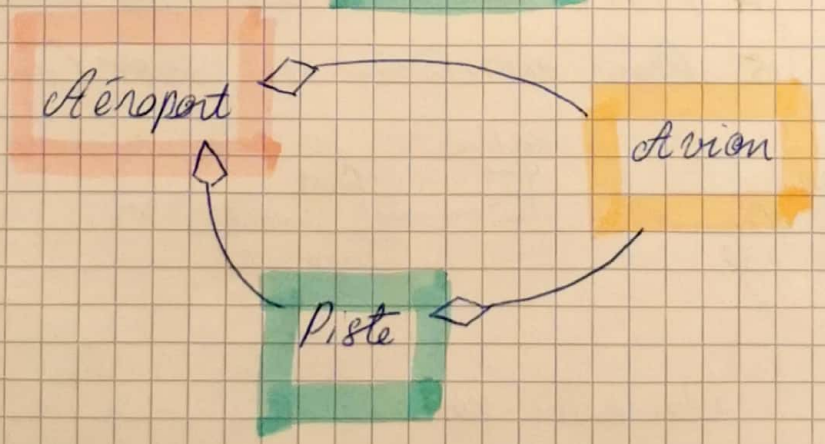
+ public

◇— Lien de composition

2.



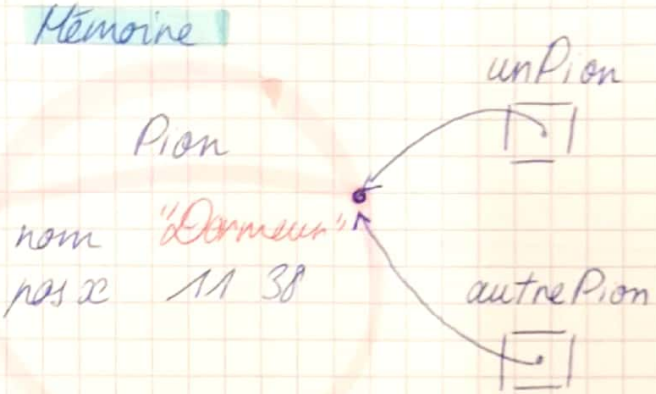
3.



- Aéroport est composé de Piste
- Aéroport est composé d'Avion
- Piste est composé d'Avion

Ex 22

Q 22.1.1 Faire un diagramme mémoire

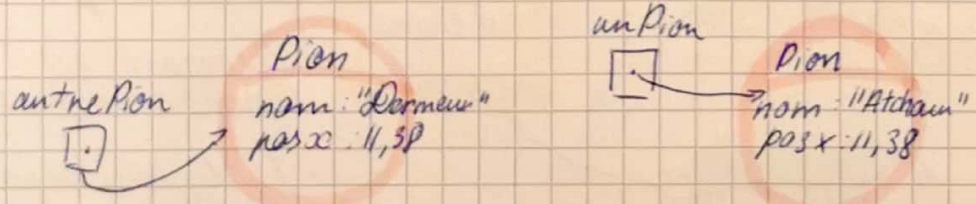


1 objet créé
2 handle

Q22.1.2
```
public class Pion {
    private String nom;
    private double posx;
    public Pion ( Pion p){
        this.posx = p.posx;
        this.nom = p.nom;
    }
}
```

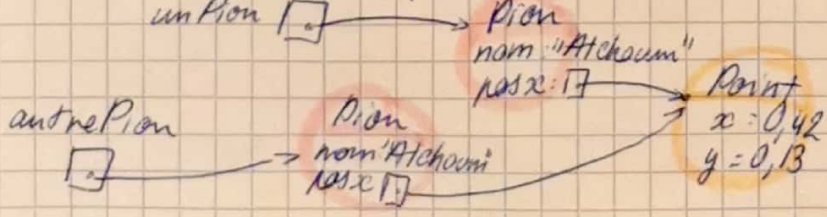
handle : variable qui contient la référence d'un objet.

Q22.1.3 14. `Pion unPion = new Pion ("Atchoum");`
15. `Pion autrePion = new Pion (unPion);`



Q22.2.1 Oui elles compilent.

Q22.2.2 des 2 pions vont se déplacer



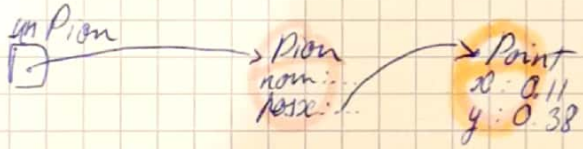
18. `autrePion.seDeplacer();`

Q 22.2.3 public class Point {

this: optionel

```
    public Point (Point p) {
        this.x = p.x;
        this.y = p.y;
    }
```

```
public class Pion {
    ... private Point posx;
    public Pion (Pion p) {
        this.posx = new Point(p.posx);
        this.nom = p.nom;
    }
```



Q 22.3 la fonction clone() renvoie une duplication d'un pion proprement.

```
public Pion clone (){
    Pion p = new Pion(nom);
    p.posx = posx.clone();
    return p;
}
```

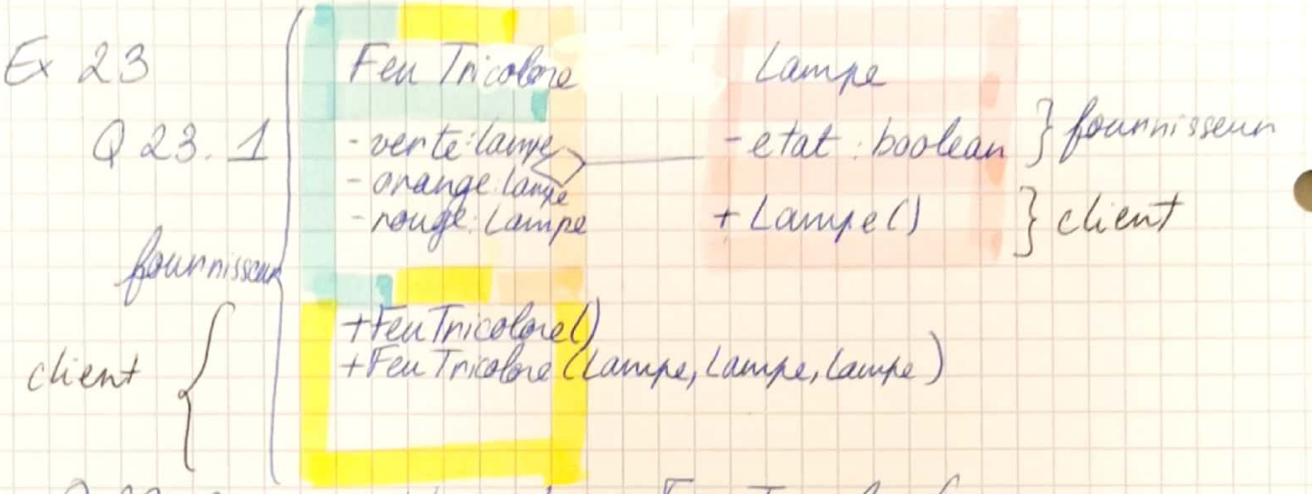
```
public Point clone (){
    Point p = new Point();
    p.x = x;
    p.y = y;
    return p; }
```

Dans le main

```
Pion unPion=
    new Pion("Atch");
Pion autrePion=
    unPion.clone();
```

Ex 23

Q 23.1



Q 23.2

```
public class FeuTricolore{
    private Lampe verte, orange, rouge;
    public FeuTricolore(){
        verte = new Lampe();
    }
}
```