1. 2. Non

I. Java

public interface I { void m(); }

A. java

pu . ~ — –

Interface I {

~~I() ...~~

}

public abstract class B implements I {

// m peut ne pas être implementé

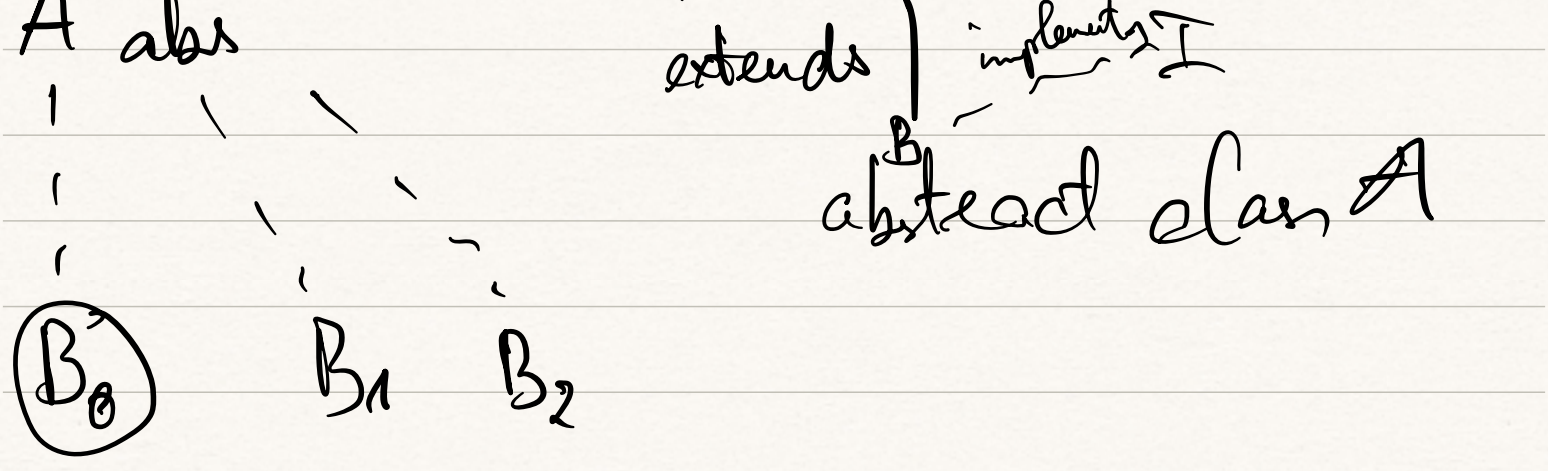
// m doit être implémenté par les héritiers concrets

2 - interface: non dans les 2 cas

abstraite : oui, avec un corps (pas sous)

3 - `A a = new B();`

classe
- B concrète
- B héritière de A
- B implémente toutes les méthodes abstraite de A



`I i = new B();`
- B concrète
- B doit implémenter les méthodes de I

4. interface

Champs : oui
mais implicitement
- static
- final } ie constantes
- public

classes abstraites

Comme une classe ordinaire

5. interfaces : méthodes statiques : oui, mais implémentées
// non statiques : oui, implémentation par défaut

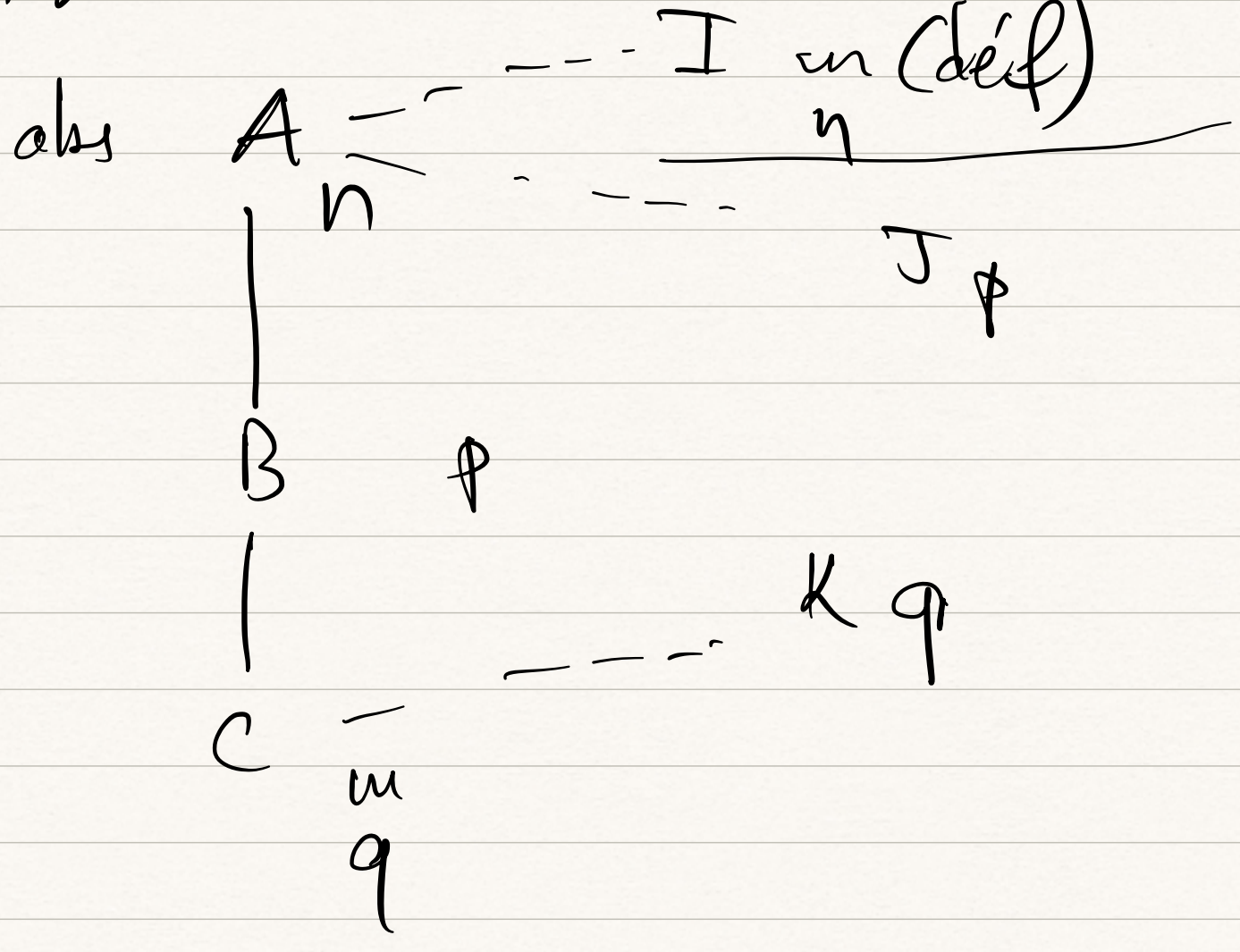
```
interface Stack {
    boolean isEmpty();
    void push(int x);
    int pop();

    default void push(int[] t) {
```

```
for (int i = 0; i < t.length; i++)
    this.push(t.Σ[i]);
}
}
```

6 : Oui, Oui ?

7 : Oui, Non elle implémente

Ex 2



~~A a = new A()~~ (abs)

B b = new B()
b.m() → T, m
b.u() → A, u
b.p() → B, p

Vrai

C c = new C()
C.u() → C, m
C, u() → A, m
C.p() → B, p
C.q() → C, q

Vrai

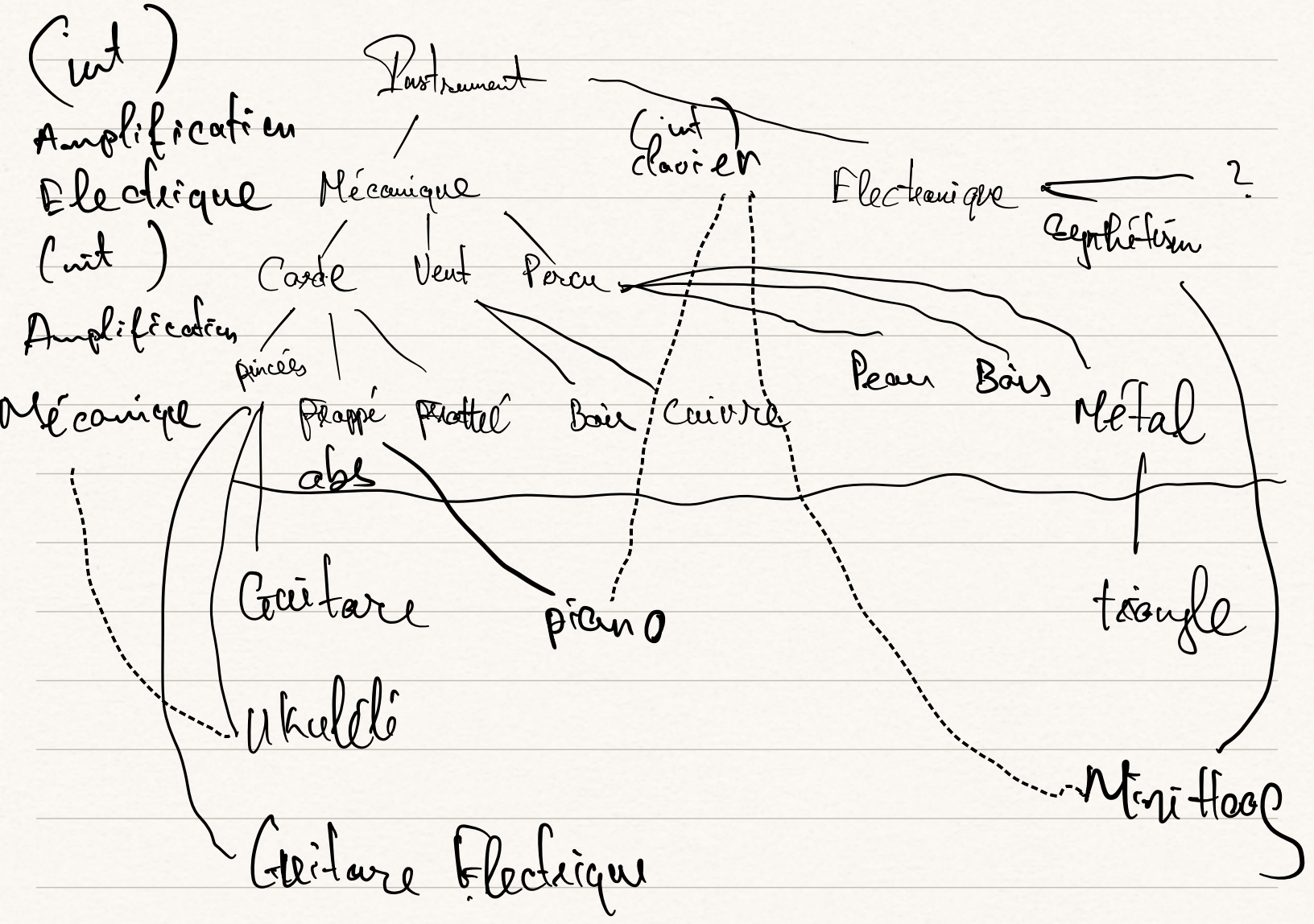
A u = b;
u.m()
u.u()
u.p()
} comme pour b
pour raisons dynamique

A v = c;
v.m()
v.u()
} comme pour c

o . p() }

~~v. q()~~ illégal

Ex 3



class A ... implements IJK {...}

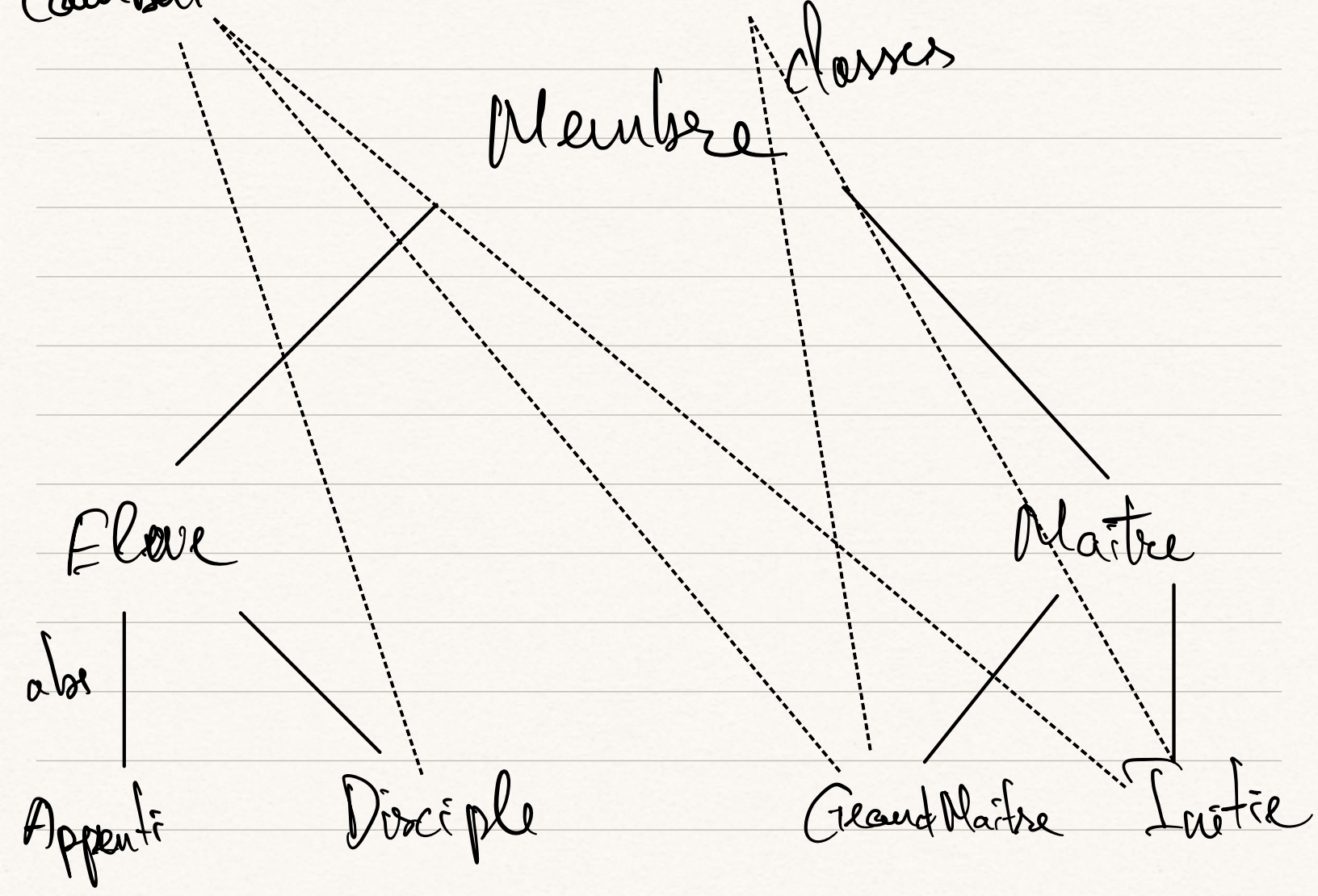
class GuitareElectrique extends Pincee
implements AmplificationElectrique

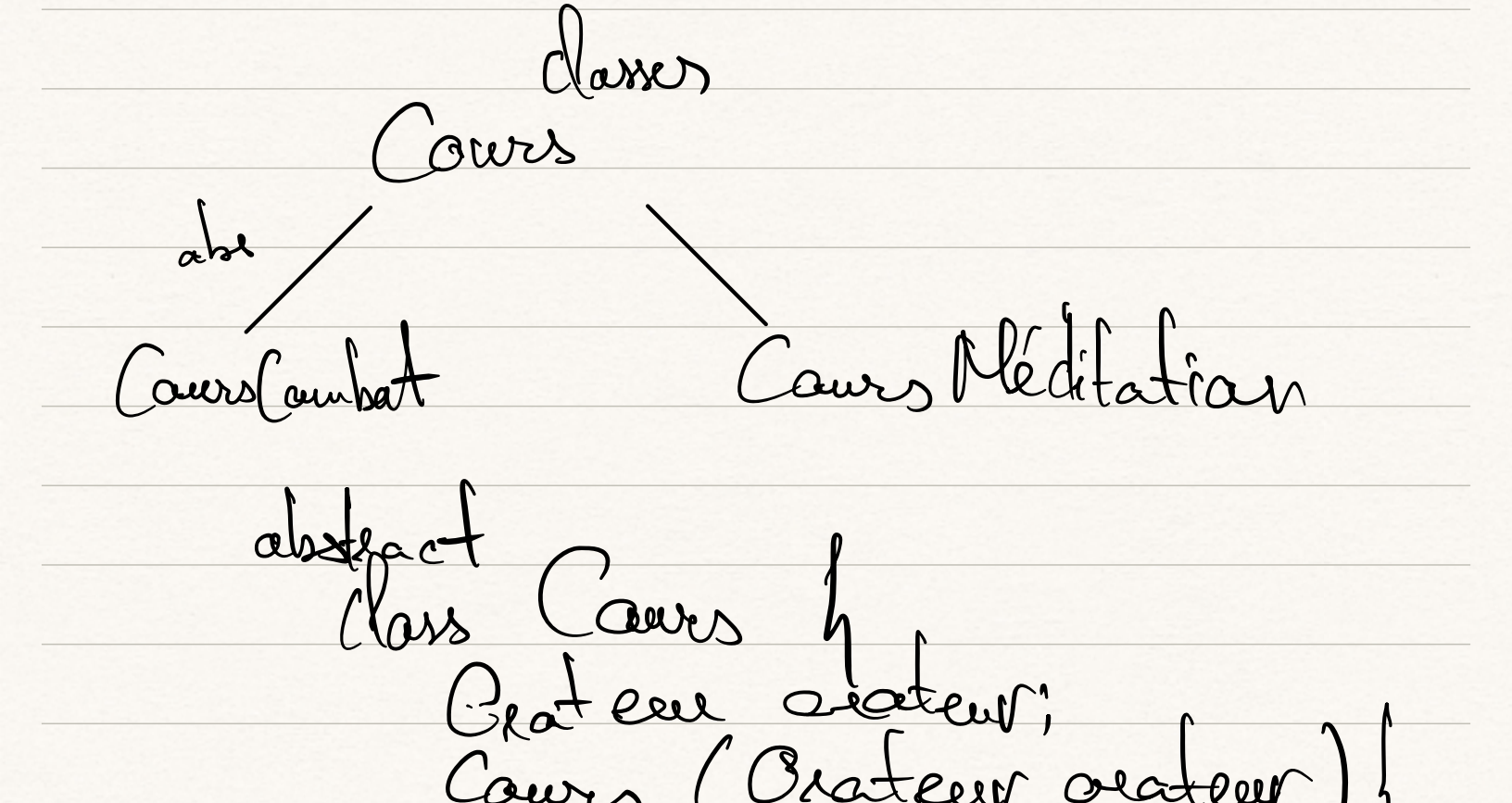
class Piano extends Frappee
implements AmplificationMecanique
Clavier

Ex 4

interface Shape {
Rectangle getBounds(),
Path getPath();
}

Createur
/ \
Combat Meditation






```
abstract
class Cours {
    Createur createur;
    Cours (Createur createur) {
```

```
        this.createur = createur;
    }
}

class CoursMeditation extends Cours {
    CoursMeditation(CreateurMeditation createur) {
        super(createur);
    }
}

class CoursCombat extends Cours {
    CoursCombat(CreateurCombat createur) {
        super(createur);
    }
}
```