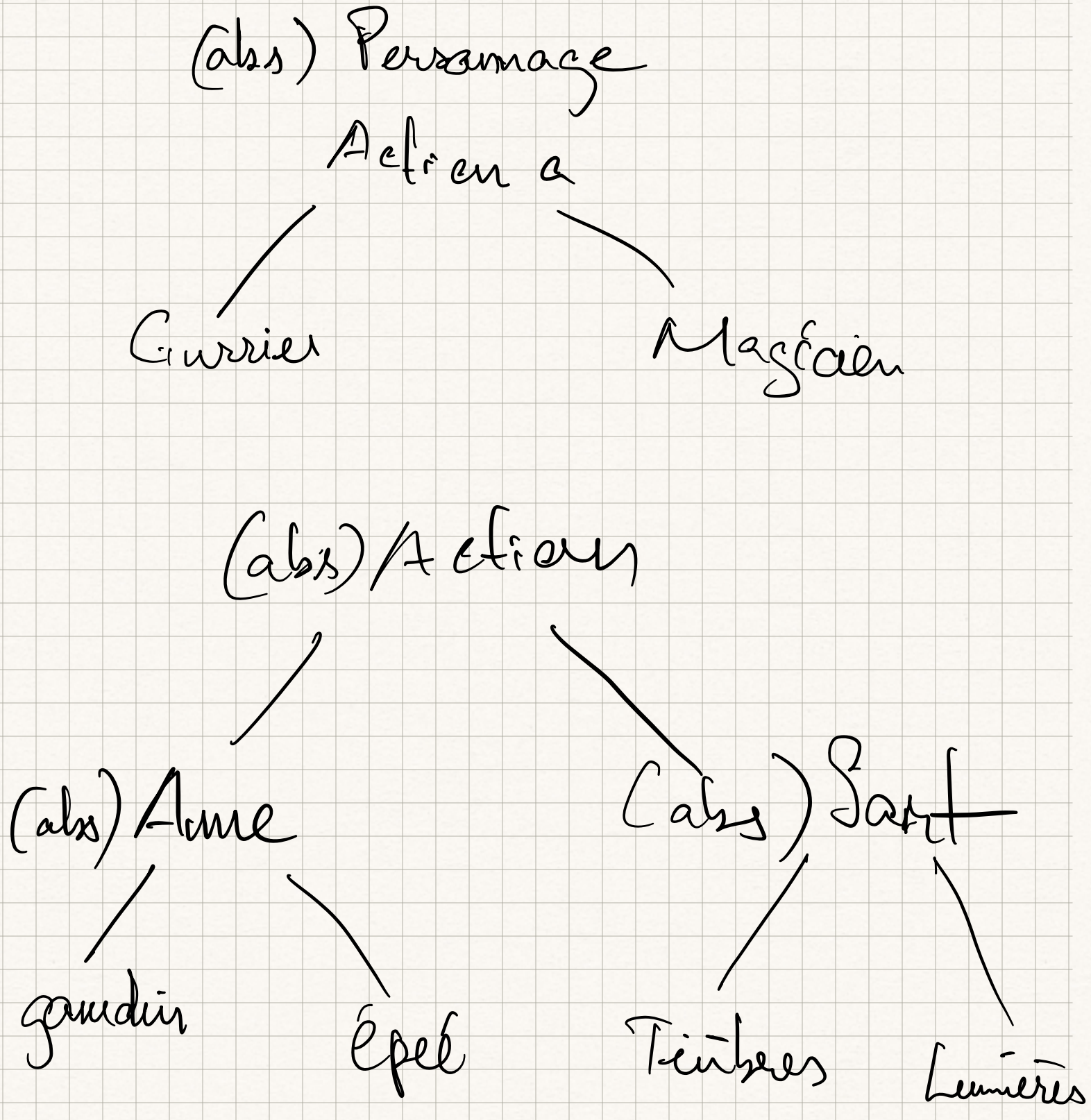
(abs) Personnage
Action
Guerrier
Magicien
(abs) Actions
(abs) Arme
(abs) Sort
gourdin
épée
Ténèbres
Lumières

```
abstract class Personnage {
  private Action a;
    Personnage (Action a)
        this.a = a;
    {
}

class Guerrier extends {
        Guerrier (Arme arme){
            super(arme);
        }
    Guerrier (Lance l ) {
        super(l);
    }
```

```
A.m();
A a = new A();
a.m();

class Base {
  Object data;
  Base(Object data) {
      this.data = data;
  }
  Object get() { return data; }
  void set(Object data) { this.data
                        = data; }
}
```

```
class Base<T> {
T data;
  Base(T data){%}
T   get(){%}
  void  set(T data){%}
}
```

```
Base<String> bs = new Base<String>
                  ("abc")
String s = bs.get();
bs.set("abc")
```

```
Base<Integer> be = new Base<>(4)
                    |||
                 Integer.value
```

```
interface Accumulator<S> {
    void accumulate(S arg);
    S read();
    boolean isOver();
}

int erface AccFunction<S,T>{
    S.apply(S acc, S exit,
            T donnée);
}
```

```
class InvalidCastedException
    extends RuntimeException {
  InvalidCastedException
      (String msg) {
    super();
  }

  // ()
}

interface Accumulator<S> {
  void accumulate(S arg);
  S read();
  boolean isOver();
}
```

final Integer tab = . . . ;

// . . .

Accumulator<Integer> a =
new Accumulator<>() {
Integer acc = 0;
int index;

public void accumulate (Integer b) {
if (!isOver()) {
acc = acc + b * tab[index++];
}}

Integer read() { return acc; }
boolean isOver() { return index

= = tab.length; }

class Matrice<T> {

T[][] data;

Matrice(T[][] data) {

if (!check(content(data))

throw new InvalidContentException();

this.data = data;

}

int nRows() { return data.length; }

int nCols() { return data[0].length

```
private class MatriceScanner<S>
  implements Accumulator<S>{
    S acc;
    int i, j, int di, dj,
    AccFunction<S,T> af;
  void accumulate(S arg){
      if(checkIndexe(i,j)){
          acc = apply(acc, arg, dat
                                [i][j]);
      i += di;
      j += dj;
    }
}
```