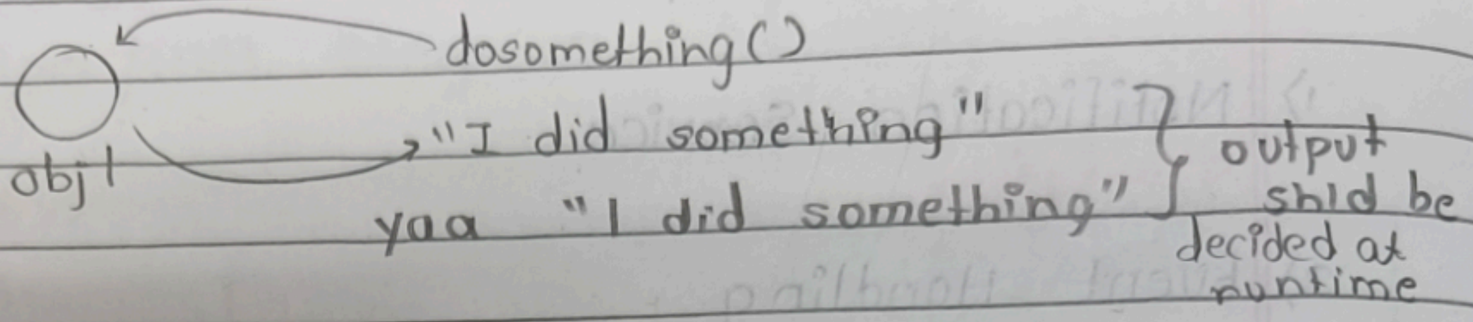
Lecture 13 : Decorator Pattern

Page No.
Date

Introduction

- Let's suppose humare pass ek object hai aur mujhe additional responsibilities / multiple task dene hai usse on runtime.

Ex:

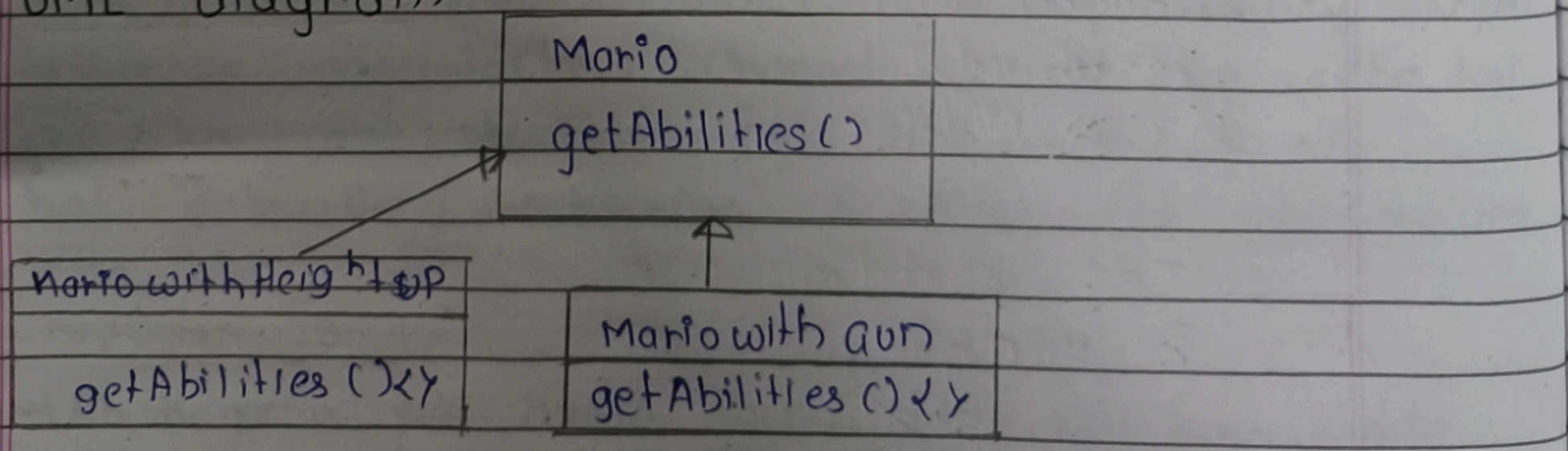


- Ab yeh functionality work karne ke liye we have inheritance but humne pattern 1 mein sikha inheritance is bad and here again is the reason

Understand using Mario Example

- Ek character hai mario jise pure game mein alag alag abilities milti hai jaise height increase; gun shooting, star abilility (for fix time jismein woh kisi ko bhi touch kare no effect).

UML diagram



- Ab yeh classes badhte rhege coz combination bahoot banegi jaise Height with Gun, Gun with Fly etc.

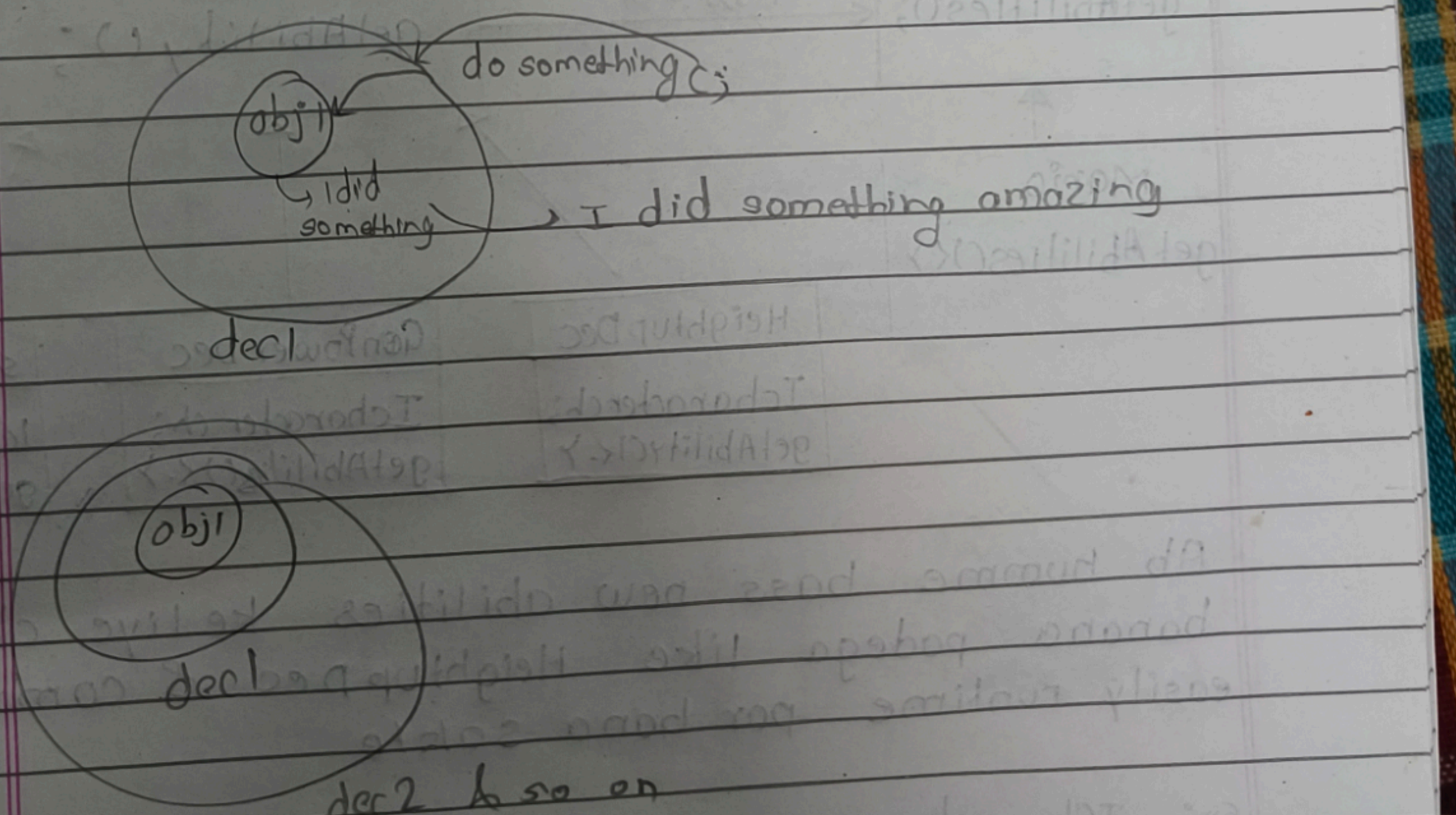
- Class Explosion occurs
Yaane ek class ke ithe child classes ban jaaye ki manage karna next to impossible ho jayega

Understanding Decorator ~~problem~~ pattern

- Hum humare object ko ek decorator/ se wrap karege yaane hum decorater ko call krege and decorator object1 ko call karega.

- And decorator response ko aur enhance kar sakta hai.

- Hum decorator ko infinetly scale up kr skte mtlb ek decobj1 ~~met~~ ko ek aur decobj2 ke andar wrap kar sakte and so on.

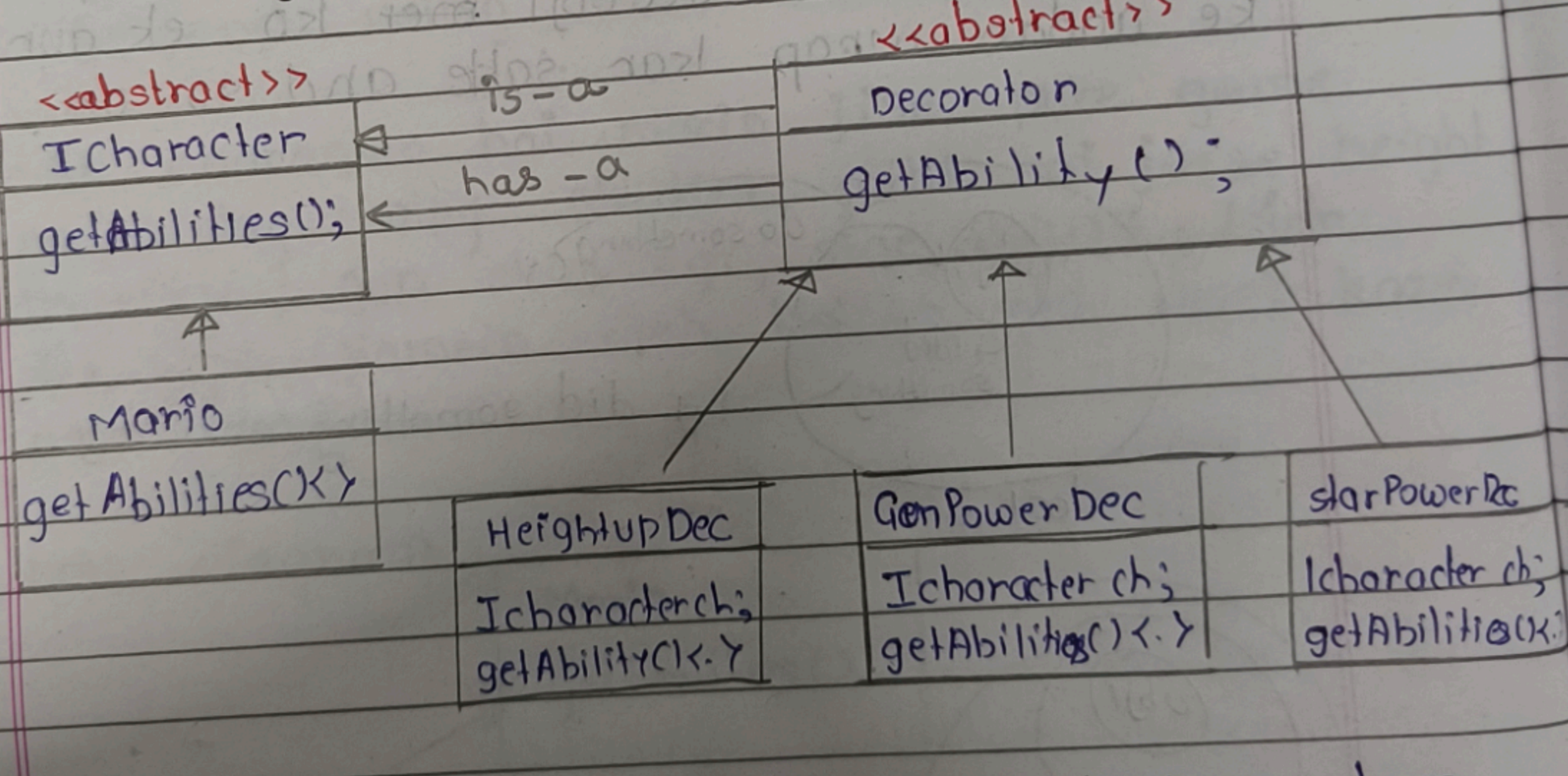


→ In short hum decorator object ko object bhi bol skte yaane inheritance yaa is-a relationship abhi bhi hai but limited to no. of abilities & not no. of combination btw abilities + no. of abilities

→ And in object mein abb composition relation has → using it to change object dynamically

⇒ In total, is-a for behaving like an object and has-a for dynamically changing that object

UML diagram for Mario Example



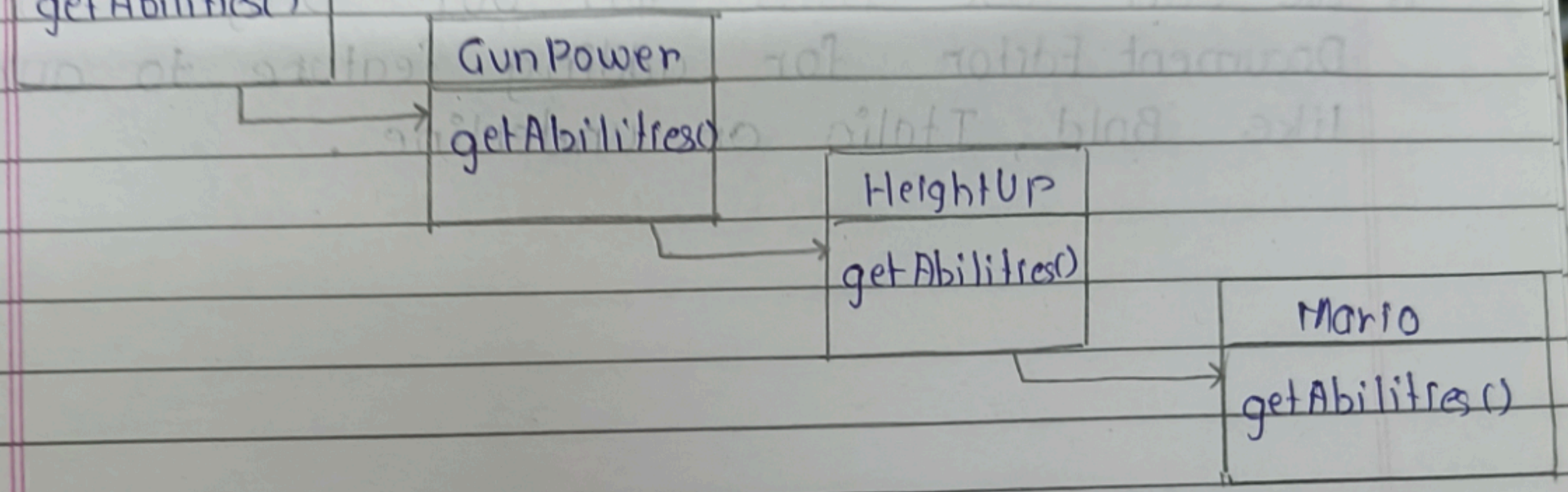
Ab humme bass new abilities ke liye class banana padega like HeightupDec & combination easily runtime par bana sakte

Ex: ICharacter mario = new HeightUpDec (new Mario());

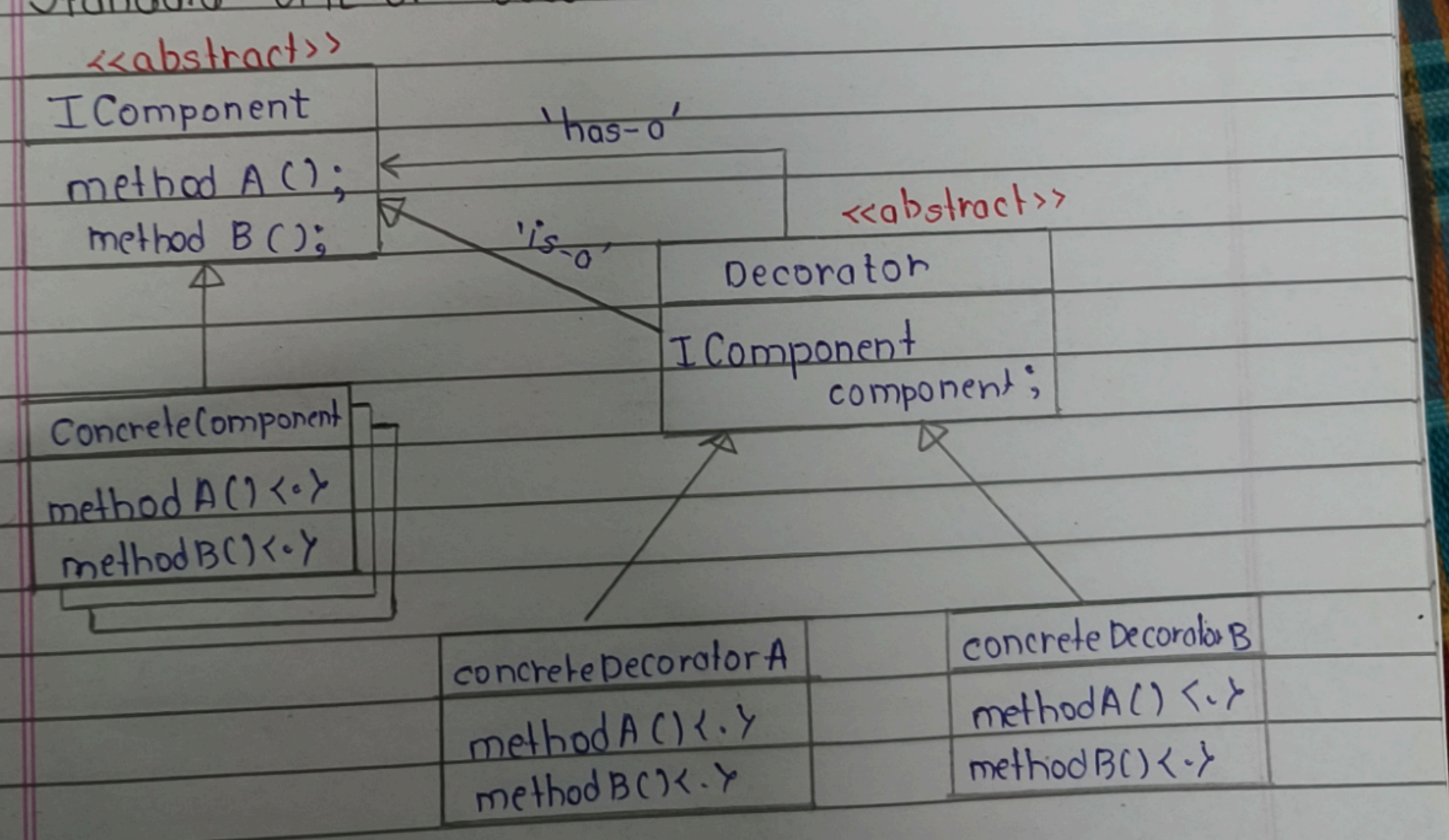
Let's d
and t
ICharac
starPo
getAbi
Sta
I

Let's design a character with all abilities and how it will executed

```
ICharacter mario = new StarPower(new GunPower(
                                 new HeightUp(
                                 new Mario())));
```



Standard UML of Decorator Pattern



Defination

- Decorator pattern attaches additional responsibilities to an object dynamically. Decorator provides a flexibile alternative to subclassing for extending functionality.

Real World Use Case

1. We can use this in our project of Document Editor for adding feature to our text like Bold, Italic or underline.