Când vreau să …		, folosesc …		, iar în cod…	13 minute
creez și populez un obiect într-un mod ușor de înțeles	a	Adapter	1	subclasa mea trebuie să îndeplinească toate așteptările de la superclasă	
am nevoie de logică asemănătoare cu ceva ce deja am implementat	b	Abstract Factory	2	scriu fiecare algoritm ca o implementare a unei interfețe comune	
las subclasele să definească anumiţi paşi dintr-un algoritm	c	Fluent Builder	3	definesc o metodă **abstract createChair():Chair** (Chair e o interfaţă)	
aleg la run-time unul dintre algoritmii dintr-o familie	d	Proxy Decorator	4	funcția mea va primi parametru o altă funcție	
subclasez corect	e	Strategy	5	Method Chaining: metode care setează câmpuri şi întorc mereu `this`	
Nu patternuri si complicatii din ziua 1. YAGNI !	f	Template Method	6	expun metode frumoase care **transformă apelurile** şi le deleagă instanţei ţintă încapsulate	
folosesc o clasă urâtă/sistem extern EVIL în logica aplicației mele	g	Pass-a-Block	7	într-o clasă/metodă rezolv o singură problematică, nu pun totul grămadă	
proiectez componente simple, ușor de înțeles și de refolosit	h	SRP	8	descompun logica existentă în metode mai mici si le apelez altfel pentru a face ce vreau	
decuplez crearea unei familii de obiecte	i	Liskov	9	scriu algoritmul generic într-o metodă concretă care invocă metode abstracte	
intermediez interacțiunile cu un obiect (de exemplu pentru a face AOP)	j	DRY	10	caut **cea mai simplă** implementare posibilă; refactorez	
execut bucăți de cod arbitrare în același loc în metoda mea	k	KISS	11	dau clientului o instanţă surogat (înlocuitor) ce implementează interfaţa subiectului de apelat	

Test holistic