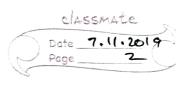
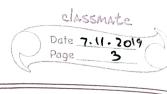
OBJECT ORIENTED PROGRAMMING

define b Colfine b	
Clambola	(mss val)
Clot	([x o])
	1
	[kg? meg show) 2]
1	(ea) msq che)
	(let (Ev (first val))
	(Set! x (+ x v)))] else (evror la "message not
Į.	else (evror la message not
	understood ~a " msg)]))))
> (6 1 sho	ω
> (61 inc	(2s)
> (b sho	ω)
8	how to do receive
	with set!
> (a 'inc 1	0
> (a show	now-field stout.
lo '	
Object-save	like environments
identifier ->	

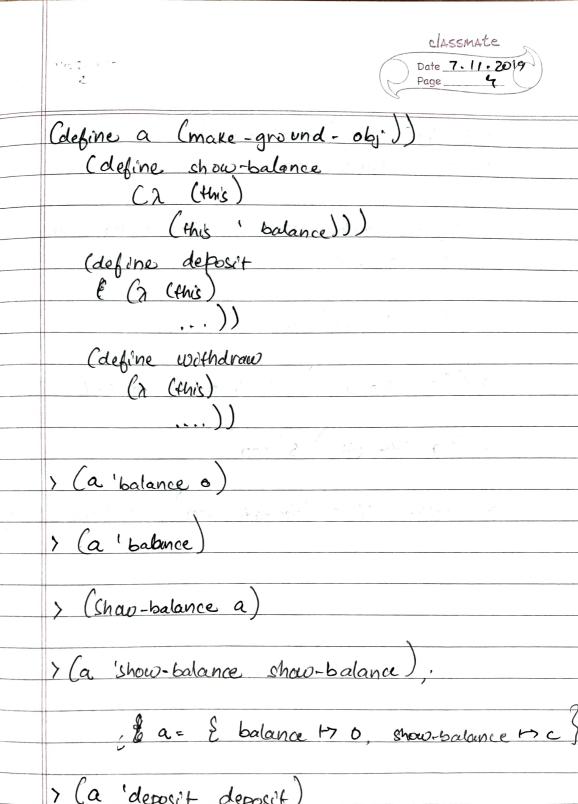
in the



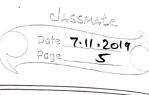
in the example, show, deposit, withdraw - balance - lexical binding three closures! Objects are implanted as procedures. Edefine make-ground-obj. (20) Clet ([table (make-hash)]) Ca mag Cimatch msg [Clist (? symbol? rej)] Chash-ref table key (λ) (ervor 'doj: kay not fond ~a" key))))] [(list (? aymbol kg) val) Chash-set (table kg val)] [else Cerror Cmake-ground-obj > edefine o (make-ground-obj)) > (0 'x 25), 0= {x +> 25} >(0 14 30), 0= \ x +> 25



> (0 12 10), 0 = { 2 +> 10, 4 +> 80} = (extend-env x 10 0,) (object - just made). > (0 12) > error > (0 'z 7); {2 H7 10, y H30, Z H> 2) Objects are enfronment. METHODS 1 (define showBalance (2 (sel+) (selt 'balance)) ch Javascropt. var showbalance = function () & return this balance. this > key reform to an identifier -> objet



> (a 'deposit deposit) > (a 'withdraw withdraw) methods no common a= { balance 1>0 State. identiffiers in the show-balance to a, methode deposit +> cz, withdraw 1-7 c3}



	1
	> Cdefine 6 (make-object))
	> (b 'balance coo)
	objects with methods should have a
-	
	Common State -
	> (Ca 'deposit) a)
	> ((a 'deposit) a 400), meshad call.
	> (la 'deposit) & soo)
	> (method coll a 'deposit 400); q.deposit (400)