X = 1 Y = 2 Z = 6	Aso	19					
class Abra {	Ahra ha - new Cadaba	Abra ho = new Cadabra()					<b></b>
int a = X, b = Y  Abra ho = new Cadabra()  Abra po = new PataDeCabra()			18 17		<u> </u>		
fun cus(int x): int {		Cadabra cir = new PataDeCabra()					
a = b + x				16 15		!	<u>L</u>
return pide(a)	print(ho.cus(X + 1)	print(ho.cus(X + 1) + po.cus(Y + 1) + cir.cus(Z + 1))				!	
}				14		1	
fun pide(int y): int {		13		1			
return a - y * b	Objeto	Metodos	Variables	12			
}	ho (Cadabra)	pide = Abra::pide()	zo (PataDeCabra())	11		!	
,		cus = Abra::cus()	a = 1	10			
			b = 2	9		'	
class Cadabra extends Abra {		cus = Abra::cus()	a = 1	8		'	
Abra zo = new PataDeCabra()	ho.zo (PataDeCabra)	pide = Abra::pide()	b = 2	7		'	
fun pide(int y): int {			c = 6	6		'	
return zo.cus(a + b) - y	po (PataDeCabra)	cus = Abra::cus()	a = 1	5		<u>'</u>	
}		pide = Abra::pide()	b = 2	4	out = 4 - 4 * 2	out = 5 - 5 * 2	out = 9 - 9
			c = 6	3	y = a	y = 5	y = 9
class PataDeCabra extends Cadabra {	cir (PataDeCabra)	cus = Abra::cus()	a = 1	2	pide(a)	pide(a)	pide(a)
int b = Y + Z, c = Z		pide = Cadabra::pide()	b = 2	1	a = 4	a = 5	a = 9
			c = 6	0	x = 2	x = 3	x = 7
fun cus(int x): int {     a = x - 3					ho.cus(X + 1) = -4	po.cus(Y + 1) = -5	cir.cus(Z + 1)
c = a + b * c						, ,	, ,
return pide(a * b + x)							
}	Output = -18						
fun pide(int y): int {							
return c - y * a							
}							

= 1 Y = 2 Z = 6	As						
class Abra {				19			
int a = X, b = Y	X. b = Y Abra ho = new Cadabra()						
for and the about the f	Abra po = new PataDeCabra()  Cadabra cir = new PataDeCabra()			17			
fun cus(int x): int {  a = b + x	Cadabia Cii - new i	16					
return pide(a)	print(ho.cus(X + 1) + po.cus(Y + 1) + cir.cus(Z + 1))			15			
}				14			
fun pide(int y): int {				13			
return a - y * b	Objeto	Metodos	Variables	12			
}		pide = Cadabra::pide()	zo (Abra())	11			
}	ho (Cadabra)	cus = Abra::cus()	a = 1	10			
			b = 2	9	out = -8 -4		
class Cadabra extends Abra {    Abra zo = new PataDeCabra()	ho.zo (PataDeCabra)	cus = Abra::cus()	a = 1	8	y = 8		
		pide = Abra::pide()	b = 2	7	pide(a)		
fun pide(int y): int {	po (PataDeCabra)	cus = PataDeCabra::cus()	a = 1	6	a = 8		
return zo.cus(a + b) - y		pide = PataDeCabra::pide()	b = 8	5	x = 6	out = 48	out = -104
class PataDeCabra extends Cadabra {   int b = Y + Z, c = Z		1 37	c = 6	4	ho.zo.cus(a + b)	y = 3	y = 39
	cir (PataDeCabra)	cus = PataDeCabra::cus()	a = 1	3	y = 4	pide(a * b + x)	pide(a * b + x)
		pide = PataDeCabra::pide()	b = 8	2	pide(a)	c = 48	c = 52
			c = 6	1	a = 4	a = 0	a = 4
				0	x = 2	x = 3	x = 7
fun cus(int x): int {     a = x - 3					ho.cus(X + 1) = -12	po.cus(Y + 1) = 48	cir.cus(Z + 1) = -1
c = a + b * c							
return pide(a * b + x)	Output = -68						
}							
fun pide(int y): int {							
return c - y * a							
}							