

Pregunta 3a

...

¡Muéstrales de qué está hecho un Saiyan!

Codigo Con Sustitucion de Constantes

```
class Abra {  
    int a = 2, b = 1  
    fun cus(int x): int {  
        a = b + x  
        return pide(a)  
    }  
    fun pide(int y): int {  
        return a - y * b  
    }  
}
```

```
class Cadabra extends Abra {  
    Abra zo = new PataDeCabra()  
    fun pide(int y): int {  
        return zo.cus(a + b) - y  
    }  
}
```

```
class PataDeCabra extends Cadabra {  
    int b = 7, c = 6  
    fun cus(int x): int {  
        a = x - 3  
        c = a + b * c  
        return pide(a * b + x)  
    }  
    fun pide(int y): int {  
        return c - y * a  
    }  
}
```

Se consideran las variables a,b, c y zo como
Variables de instancia dado que no son statics (similar a java)

Asociación Estática de Métodos

```
Abra ho = new Cadabra()
```

```
Abra po = new PataDeCabra()
```

```
Cadabra cir = new PataDeCabra()
```

```
print(ho.cus(1) + po.cus(1) + cir.cus(1))
```

Inicialmente se crean dos variables de tipo Abra y una de tipo Cadabra

Por emplearse asociación estática de métodos se tiene que ho y po tendrán los métodos de Abra. Y dado que Cadabra extiende a Abra y sobre-escribe la función pide entonces cir tiene dos métodos: cus (de Abra) y pide (de Cadabra).

Como el espacio para visualizar la ejecución es limitado se mostrará únicamente la línea print y se mostrará el método que se esté ejecutando

```
print(ho.cus(1) + po.cus(1) + cir.cus(1))
```

```
class Abra {  
    int a = 2, b = 1  
0    fun cus(int x): int {  
1        a = b + x  
2        return pide(a)  
    }  
    fun pide(int y): int {  
3        return a - y * b  
    }  
}
```

ho

cus	Abra
pide	Abra

po

cus	Abra
pide	Abra

cir

cus	Abra
pide	Cad abra

cus@Abra	x	1
	b	1
	a	2

```
print(ho.cus(1) + po.cus(1) + cir.cus(1))
```

```
class Abra {
    int a = 2, b = 1
0   fun cus(int x): int {
1       a = b + x
2       return pide(a)
    }
    fun pide(int y): int {
3       return a - y * b
    }
}
```

ho

cus	Abra
pide	Abra

po

cus	Abra
pide	Abra

cir

cus	Abra
pide	Cad abra

cus@Abra	x	1
	b	1
	a	2 2

```
print(ho.cus(1) + po.cus(1) + cir.cus(1))
```

```
class Abra {
    int a = 2, b = 1
0   fun cus(int x): int {
1       a = b + x
2       return pide(a)
    }
    fun pide(int y): int {
3       return a - y * b
    }
}
```

ho

cus	Abra
pide	Abra

po

cus	Abra
pide	Abra

cir

cus	Abra
pide	Cad abra

pide@Abr a	y	2
	b	1
	a	2
cus@Abra	x	1
	b	1
	a	2 2

```
print(ho.cus(1) + po.cus(1) + cir.cus(1))
```

```
class Abra {
    int a = 2, b = 1
0   fun cus(int x): int {
1       a = b + x
2       return pide(a)
    }
    fun pide(int y): int {
3       return a - y * b
    }
}
```

Retorna
2-2*1 = 0

ho

po

cir

cus	Abra
pide	Abra

cus	Abra
pide	Abra

cus	Abra
pide	Cad abra

pide@Abr a	y	2
	b	1
	a	2
cus@Abra	x	1
	b	1
	a	2 2

```
print(ho.cus(1) + po.cus(1) + cir.cus(1))
```

```
class Abra {
    int a = 2, b = 1
0   fun cus(int x): int {
1       a = b + x
2       return pide(a)    Retorna 0
    }
    fun pide(int y): int {
3       return a - y * b
    }
}
```

ho

cus	Abra
pide	Abra

po

cus	Abra
pide	Abra

cir

cus	Abra
pide	Cad abra

cus@Abra	x	1
	b	1
	a	2 2


```
print(ho.cus(1) + po.cus(1) + cir.cus(1))
```

```
ho.cus(1) = 0
```

```
class Abra {  
    int a = 2, b = 1  
0    fun cus(int x): int {  
1        a = b + x  
2        return pide(a)  
    }  
    fun pide(int y): int {  
3        return a - y * b  
    }  
}
```

ho

cus	Abra
pide	Abra

po

cus	Abra
pide	Abra

cir

cus	Abra
pide	Cad abra

```
print( 0 + po.cus(1) + cir.cus(1))
```

```
class Abra {  
    int a = 2, b = 1  
0    fun cus(int x): int {  
1        a = b + x  
2        return pide(a)  
    }  
    fun pide(int y): int {  
3        return a - y * b  
    }  
}
```

ho

cus	Abra
pide	Abra

po

cus	Abra
pide	Abra

cir

cus	Abra
pide	Cad abra

```
print( 0 + po.cus(1) + cir.cus(1))
```

```
class Abra {  
    int a = 2, b = 1  
0    fun cus(int x): int {  
1        a = b + x  
2        return pide(a)  
    }  
    fun pide(int y): int {  
3        return a - y * b  
    }  
}
```

ho

cus	Abra
pide	Abra

po

cus	Abra
pide	Abra

cir

cus	Abra
pide	Cad abra

```
print( 0 + po.cus(1) + cir.cus(1))
```

```
class Abra {  
    int a = 2, b = 1  
0    fun cus(int x): int {  
1        a = b + x  
2        return pide(a)  
    }  
    fun pide(int y): int {  
3        return a - y * b  
    }  
}
```

ho

cus	Abra
pide	Abra

po

cus	Abra
pide	Abra

cir

cus	Abra
pide	Cad abra

cus@Abra	x	1
	b	1
	a	2

```
print( 0 + po.cus(1) + cir.cus(1))
```

```
class Abra {  
    int a = 2, b = 1  
0    fun cus(int x): int {  
1        a = b + x  
2        return pide(a)  
    }  
    fun pide(int y): int {  
3        return a - y * b  
    }  
}
```

ho

cus	Abra
pide	Abra

po

cus	Abra
pide	Abra

cir

cus	Abra
pide	Cad abra

cus@Abra	x	1
	b	1
	a	2 2

```
print( 0 + po.cus(1) + cir.cus(1))
```

```
class Abra {
    int a = 2, b = 1
0   fun cus(int x): int {
1       a = b + x
2       return pide(a)
    }
    fun pide(int y): int {
3       return a - y * b
    }
}
```

ho

po

cir

cus	Abra
pide	Abra

cus	Abra
pide	Abra

cus	Abra
pide	Cad abra

pide@Abr a	y	2
	b	1
	a	2
cus@Abra	x	1
	b	1
	a	2 2

```
print( 0 + po.cus(1) + cir.cus(1))
```

```
class Abra {
    int a = 2, b = 1
0   fun cus(int x): int {
1       a = b + x
2       return pide(a)
    }
    fun pide(int y): int {
3       return a - y * b
    }
}
```

Retorna
2-2*1 = 0

ho

po

cir

cus	Abra
pide	Abra

cus	Abra
pide	Abra

cus	Abra
pide	Cad abra

pide@Abr a	y	2
	b	1
	a	2
cus@Abra	x	1
	b	1
	a	2 2

```
print( 0 + po.cus(1) + cir.cus(1))
```

```
class Abra {
    int a = 2, b = 1
0   fun cus(int x): int {
1       a = b + x
2       return pide(a)    Retorna 0
    }
    fun pide(int y): int {
3       return a - y * b
    }
}
```

ho

po

cir

cus	Abra
pide	Abra

cus	Abra
pide	Abra

cus	Abra
pide	Cad abra

cus@Abra	x	1
	b	1
	a	2 2


```
print( 0 + po.cus(1) + cir.cus(1))
```

```
po.cus(1) = 0
```

```
class Abra {  
    int a = 2, b = 1  
0    fun cus(int x): int {  
1        a = b + x  
2        return pide(a)  
    }  
    fun pide(int y): int {  
3        return a - y * b  
    }  
}
```

ho

cus	Abra
pide	Abra

po

cus	Abra
pide	Abra

cir

cus	Abra
pide	Cad abra

```
print( 0 + 0 + cir.cus(1))
```

```
class Abra {  
    int a = 2, b = 1  
0    fun cus(int x): int {  
1        a = b + x  
2        return pide(a)  
    }  
    fun pide(int y): int {  
3        return a - y * b  
    }  
}
```

ho

cus	Abra
pide	Abra

po

cus	Abra
pide	Abra

cir

cus	Abra
pide	Cad abra

```
print( 0 + 0 + cir.cus(1))
```

```
class Abra {  
    int a = 2, b = 1  
0    fun cus(int x): int {  
1        a = b + x  
2        return pide(a)  
    }  
    fun pide(int y): int {  
3        return a - y * b  
    }  
}
```

ho

cus	Abra
pide	Abra

po

cus	Abra
pide	Abra

cir

cus	Abra
pide	Cad abra

cus@Abra	x	1
	b	1
	a	2

```
print( 0 + 0 + cir.cus(1))
```

```
class Abra {  
    int a = 2, b = 1  
0    fun cus(int x): int {  
1        a = b + x  
2        return pide(a)  
    }  
    fun pide(int y): int {  
3        return a - y * b  
    }  
}
```

ho

cus	Abra
pide	Abra

po

cus	Abra
pide	Abra

cir

cus	Abra
pide	Cad abra

cus@Abra	x	1
	b	1
	a	2 2

```
print( 0 + 0 + cir.cus(1))
```

```
class Abra {  
    int a = 2, b = 1  
0    fun cus(int x): int {  
1        a = b + x  
2        return pide(a)  
    }  
    fun pide(int y): int {  
3        return a - y * b  
    }  
}
```

ho

cus	Abra
pide	Abra

po

cus	Abra
pide	Abra

cir

cus	Abra
pide	Cad abra

cus@Abra	x	1
	b	1
	a	2 2

```
print( 0 + 0 + cir.cus(1))
```

```
class Cadabra extends Abra {  
    Abra zo = new PataDeCabra()  
0   fun pide(int y): int {  
1       return zo.cus(a + b) - y  
    }  
}
```

ho

cus	Abra
pide	Abra

po

cus	Abra
pide	Abra

cir

cus	Abra
pide	Cad abra

zo

cus	Abra
pide	Abra

pide@Cad abra	y	2
	zo	Tipo Abra
	b	1
	a	2 2
cus@Abra	x	1
	b	1
	a	2 2

```
print( 0 + 0 + cir.cus(1))
```

```
class Cadabra extends Abra {  
    Abra zo = new PataDeCabra()  
0   fun pide(int y): int {  
1       return zo.cus(a + b) - y  
    }  
}
```

ho

cus	Abra
pide	Abra

po

cus	Abra
pide	Abra

cir

cus	Abra
pide	Cad abra

zo

cus	Abra
pide	Abra

pide@Cad abra	y	2
	zo	Tipo Abra
	b	1
	a	2 2
cus@Abra	x	1
	b	1
	a	2 2

```
print( 0 + 0 + cir.cus(1))
```

```
class Abra {  
    int a = 2, b = 1  
0    fun cus(int x): int {  
1        a = b + x  
2        return pide(a)  
    }  
    fun pide(int y): int {  
3        return a - y * b  
    }  
}
```

cus@Abra	x	3
	b	1
	a	2
pide@Cad abra	y	2
	zo	Tipo Abra
	b	1
	a	2 2
cus@Abra	x	1
	b	1
	a	2 2

ho		po		cir		zo	
cus	Abra	cus	Abra	cus	Abra	cus	Abra
pide	Abra	pide	Abra	pide	Cad abra	pide	Abra


```
print( 0 + 0 + cir.cus(1))
```

```
class Abra {
    int a = 2, b = 1
0   fun cus(int x): int {
1       a = b + x
2       return pide(a)
    }
    fun pide(int y): int {
3       return a - y * b
    }
}
```

cus@Abra	x	3
	b	1
	a	2 4
pide@Cad abra	y	2
	zo	Tipo Abra
	b	1
	a	2 2
cus@Abra	x	1
	b	1
	a	2 2

ho		po		cir		zo	
cus	Abra	cus	Abra	cus	Abra	cus	Abra
pide	Abra	pide	Abra	pide	Cad abra	pide	Abra

print(0 + 0 + cir.cus(1))

```
class Abra {  
    int a = 2, b = 1  
0    fun cus(int x): int {  
1        a = b + x  
2        return pide(a)  
    }  
    fun pide(int y): int {  
3        return a - y * b  
    }  
}
```

ho

cus	Abra
pide	Abra

po

cus	Abra
pide	Abra

cir

cus	Abra
pide	Cad abra

zo

cus	Abra
pide	Abra

cus@Abra	x	3
	b	1
	a	2 4
pide@Cad abra	y	2
	zo	Tipo Abra
	b	1
	a	2 2
cus@Abra	x	1
	b	1
	a	2 2

```
print( 0 + 0 + cir.cus(1))
```

```
class Abra {
    int a = 2, b = 1
0   fun cus(int x): int {
1       a = b + x
2       return pide(a)
    }
    fun pide(int y): int {
3       return a - y * b
    }
}
```

Retorna
4-4*1 = 0

ho

po

cir

zo

cus	Abra
pide	Abra

cus	Abra
pide	Abra

cus	Abra
pide	Cad abra

cus	Abra
pide	Abra

pide@Abra a	y	4
	b	1
	a	4
cus@Abra	x	3
	b	1
	a	2 4
pide@Cad abra	y	2
	zo	Tipo Abra
	b	1
	a	2 2
cus@Abra	x	1
	b	1
	a	2 2

```
print( 0 + 0 + cir.cus(1))
```

```
class Abra {
    int a = 2, b = 1
0   fun cus(int x): int {
1       a = b + x
2       return pide(a)
    }
    fun pide(int y): int {
3       return a - y * b
    }
}
```

Retorna
pide(a) = 0

ho

cus	Abra
pide	Abra

po

cus	Abra
pide	Abra

cir

cus	Abra
pide	Cad abra

zo

cus	Abra
pide	Abra

cus@Abra	x	3
	b	1
	a	2 4
pide@Cad abra	y	2
	zo	Tipo Abra
	b	1
	a	2 2
cus@Abra	x	1
	b	1
	a	2 2

```
print( 0 + 0 + cir.cus(1))
```

```
class Cadabra extends Abra {  
    Abra zo = new PataDeCabra()  
0   fun pide(int y): int {  
1       return zo.cus(a + b) - y  
    }  
}
```

↓
zo.cus(4) = 0

ho

cus	Abra
pide	Abra

po

cus	Abra
pide	Abra

cir

cus	Abra
pide	Cadabra

zo

cus	Abra
pide	Abra

pide@Cadabra	y	2
	zo	Tipo Abra
	b	1
	a	2 2
cus@Abra	x	1
	b	1
	a	2 2

```
print( 0 + 0 + cir.cus(1))
```

```
class Cadabra extends Abra {
    Abra zo = new PataDeCabra()
0   fun pide(int y): int {
1       return zo.cus(a + b) - y   Retorna -2
    }
}
```

ho

po

cir

zo

cus	Abra
pide	Abra

cus	Abra
pide	Abra

cus	Abra
pide	Cadabra

cus	Abra
pide	Abra

pide@Cadabra	y	2
	zo	Tipo Abra
	b	1
	a	2 2
cus@Abra	x	1
	b	1
	a	2 2

```
print( 0 + 0 + cir.cus(1))
```

```
class Abra {  
    int a = 2, b = 1  
0    fun cus(int x): int {  
1        a = b + x  
2        return pide(a)    Retorna -2  
    }  
    fun pide(int y): int {  
3        return a - y * b  
    }  
}
```

ho

cus	Abra
pide	Abra

po

cus	Abra
pide	Abra

cir

cus	Abra
pide	Cad abra

cus@Abra	x	1
	b	1
	a	2 2

```
print( 0 + 0 + cir.cus(1))
```

```
cir.cus(1) = -2
```

ho

po

cir

cus	Abra
pide	Abra

cus	Abra
pide	Abra

cus	Abra
pide	Cad abra




```
print( 0 + 0 + -2)
```

ho

cus	Abra
pide	Abra

po

cus	Abra
pide	Abra

cir

cus	Abra
pide	Cad abra

/

```
Abra ho = new Cadabra()
```

```
Abra po = new PataDeCabra()
```

```
Cadabra cir = new PataDeCabra()
```

```
print(ho.cus(1) + po.cus(1) + cir.cus(1))
```

Imprime
-2

ho

cus	Abra
pide	Abra

po

cus	Abra
pide	Abra

cir

cus	Abra
pide	Cad abra



**“Todavía no has visto todo mi poder, joven
guerrero.”**

– Cell