Examen 3

CI3641 – Lenguajes de Programación I
Enero–Marzo 2022
Estudiante: Gregory Muñoz
Carnet: 16-11313

Constantes:
X = 3
Y = 1
Z = 3

Enlace de github donde se encontrarán los programas:
https://github.com/mvgregoryj/CI3641-Lenguajes-de-Programacion-I/tree/main/Examen%203

1. Escoja algún lenguaje de programación de alto nivel, de propósito general y orientado a objetos que no comience con J, C o P.

Respuesta 1: Lenguaje escogido: Kotlin

(a) De una breve descripción del lenguaje escogido.

i. Explique la manera de crear y manipular objetos que tiene el lenguaje, incluyendo: constructores, métodos, campos, etc.

R: Se pueden definir clases con constructores, campos, métodos, visibilidad e incluso tipos genéricos

Ejemplo:

```kotlin
abstract class Person(val name: String) {
    abstract fun greet()
}

interface FoodConsumer {
    fun eat()
    fun pay(amount: Int) = println("Delicious! Here's $amount bucks!")
}

class RestaurantCustomer(name: String, val dish: String) : Person(name), FoodConsumer {
    fun order() = println("$dish, please!")
    override fun eat() = println("*Eats $dish*")
    override fun greet() = println("It's me, $name.")
}

fun main() {
    val sam = RestaurantCustomer("Sam", "Mixed salad")
    sam.greet() // Una implementación de una función abstracta
    sam.order() // Una función miembro
    sam.eat() // Una implementación de una función de interfaz
    sam.pay(10) // Una implementación predeterminada en una interfaz
}
```

Constructores:

El **constructor primario** o principal, hace parte de la cabecera de la clase. Este recibe como argumentos, aquellos datos que necesitas explicitamente para inicializar las propiedades al crear el objeto.

Constructores Secundarios: Si la lista de argumentos del constructor primario no satisface la creación de tu objeto en alguna circunstancia, entonces puedes crear un constructor secundario a la medida.

Su declaración se realiza a través de constructor al interior de la clase.

Si se tiene un constructor primario es obligatorio usar la expresión this para delegarle los parámetros que requiera. Luego se escribe la lógica de inicialización en el bloque.

Ejemplo:

Se requiere crear una clase de contactos que contenga los atributos Id y Nombre.
Se pueden crear contactos con tan solo el nombre y el ID autogenerado.
O también con ambos valores.

```
class Contact(var name: String) {
   var id: String

   init {
       id = UUID.randomUUID().toString()
   }

   constructor(id:String, name: String) : this(name){
       this.id = id
   }
}
```

Se declaró un constructor primario con el nombre y la autogeneración del Id en el bloque de inicialización.

Luego en el constructor secundario se reciben ambos parámetros. Se delegas hacia el constructor primario el nombre y el id se reemplaza.

De esta forma se puede crear los dos tipos de contactos:

```
Contact("Gregory")
Contact("20-11313", "Fulanito")
```

Campos (propiedades):

En Kotlin, no existe el concepto de campo tal como se conoce; en su lugar, emplea el concepto de "propiedades". Veamos el siguiente ejemplo, con dos propiedades mutables (lectura-escritura) declaradas con la palabra reservada var: title y isbn en la clase Book.

```
class Book {
   var title: String
   var isbn: Long
   constructor(title: String, isbn: Long) {
       this.title = title
       this.isbn = isbn
   }
}
```

Visibilidad:

▸ **public —** Accesible en cualquier parte
▸ **private —** Accesible sólo en el alcance actual
    ▸ Para clases se re ere a la clase que se está teniendo
    ▸ Para declaraciones top–level se re ere al archivo actual
▸ **protected —** Accesible sólo en la clase actual y demás clases que hereden de ésta
▸ **internal —** Accesible sólo en el módulo actual

Nota: Si anula (override) un miembro protegido (protected) en la clase derivada sin especificar su visibilidad, su visibilidad también estará protegida (protected).

```
open class Base() {
   var a = 1                 // publica por defecto
   private var b = 2         // privado a clase Base
   protected open val c = 3  // visible para la Base y la clase Derived
   internal val d = 4        // visible dentro del mismo módulo
   protected fun e() { }     // visible para la Base y la clase Derived
}

class Derived: Base() {

   // a, c, d y e() de la clase Base son visibles
   // b no es visible

   override val c = 9        // c está protegido
}

fun main(args: Array<String>) {
   val base = Base()

   // base.a y base.d son visibles
   // base.b, base.c y base.e() no son visibles

   val derived = Derived()
   // derived.c no es visible
}
```

Tipos Genericos:

Los tipos genéricos son tipos de parámetros especificados para las definiciones de clases, funciones e interfaces que escribe Kotlin en su código fuente.

Los tipos genéricos están marcados con corchetes angulares **<>** después del nombre de la construcción. El siguiente ejemplo muestra una clase de Kotlin llamada Car que tiene un tipo genérico:

```
class Car<T>
```

El tipo genérico <T> anterior es en realidad un tipo de marcador de posición, y puede reemplazar la T con cualquier tipo de Kotlin válido cuando crea una instancia de un nuevo objeto de la clase.

Los siguientes ejemplos son instancias válidas de Car de tipo String e Int respectivamente:

```
val car = Car<String>()
val anotherCar = Car<Int>()
```

La ventaja de usar tipos genéricos en Kotlin es que permite crear una clase más flexible.

Para ilustrar los beneficios de usar un tipo genérico, imagine que tiene una clase Car con un parámetro name para su constructor principal, como se muestra a continuación:

```
class Car(val name: String)
```

Ahora, cada vez que cree una instancia de la clase Car, siempre tendrá una propiedad de nombre de tipo String

```
val car = Car("Tesla")
println(car.name) // Tesla
```

Si se intenta pasar un Int al parámetro name, se generará un error durante el tiempo de compilación de la siguiente manera:

```
val anotherCar = Car(2)
// El literal entero no se ajusta al tipo esperado String
```

Dado que los parámetros de clase deben tener anotaciones de tipo, sólo puede definir un tipo para que la clase lo acepte. Es un parámetro String o un parámetro Int.

Aquí es donde un tipo genérico puede ayudar. Al agregar un tipo de marcador de posición para la clase, se puede definir el tipo de los parámetros de la clase más adelante cuando se cree el objeto.

Con la siguiente definición de clase, el parámetro de nombre tendría el tipo que defina en la creación de instancias:

```
class Car<T>(val name: T)
```

Ahora la T se puede reemplazar con un String o un Int según se requiera:

```
val car = Car<String>("Tesla")
println(car.name) // Tesla

val anotherCar = Car<Int>(2)
println(anotherCar.name) // 2
```

La <T> no es ninguna sintaxis especial de Kotlin. Es solo una convención utilizada al definir tipos genéricos.

Las convenciones para los genéricos de Kotlin son las siguientes:

E para el elemento
T para tipo
K para clave
V por valor

ii. Describa el funcionamiento del manejo de memoria, ya sea explícito (new/delete) o implícito (recolector de basura).

R:

Crear Objetos:

Cuando se define la clase, solo se define la especificación para el objeto; no se asigna memoria ni almacenamiento.

Para acceder a los miembros definidos dentro de la clase, es necesario crear objetos.

Ejemplo:

```
class Lamp {

   // property (data member)
   private var isOn: Boolean = false

   // member function
   fun turnOn() {
       isOn = true
   }

   // member function
   fun turnOff() {
       isOn = false
   }
}

fun main(args: Array<String>) {

   val l1 = Lamp() // crear objeto l1 de la clase Lamp
   val l2 = Lamp() // crear objeto l2 de la clase Lamp
}
```

Eliminar Objetos:

-Kotlin/JVM:

No es necesario preocuparse por eliminar objetos como en otros lenguajes como c++/c... el recolector de basura de la JVM se encarga de ello, todo lo que se necesita saber para eliminar un objeto es no mantener referencias en el objeto

Entonces, si se tiene una colección (lista, mapa ...) donde se pone el objeto, también se debe sacarlo si la colección es una propiedad de una clase longeva como un modelo o algo así ... esa es la única posibilidad de meterse en problemas dentro de kotlin, poner una referencia en una colección a la que hace referencia un objeto estático o de larga duración.

Dentro de una función no es necesario eliminar los objetos creados con ella. Se debe tener en cuenta que el recolector de basura (GC) no se ejecuta instantáneamente después de finalizar el método. Existen diferentes estrategias dependiendo de la antigüedad del objeto y del recolector de basura en sí.

-Kotlin/Native:

Kotlin / Native proporciona un esquema de administración de memoria automatizado, similar al que proporcionan Java o Swift. La implementación actual incluye un contador de referencia automatizado con un colector de ciclo para recoger basura cíclica.

iii. Diga si el lenguaje usa asociación estática o dinámica de métodos y si hay forma de alterar la elección por defecto del lenguaje.

R:

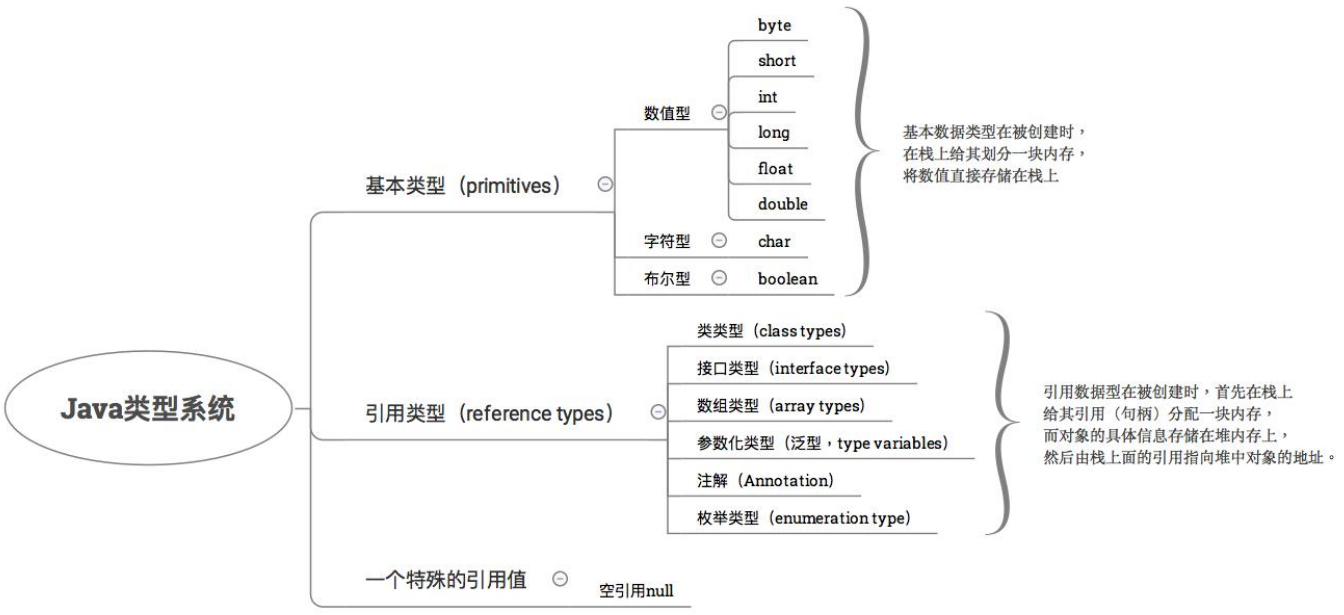
► Se usa asociación estática de métodos y, más aún, no se pueden sobrecargar métodos ni propiedades

► Para que una clase admita sobrecargas en clases que lo heredan, la misma debe estar declarada como open

- ► Los métodos y propiedades de igual forma
- ► En la clase hija deben establecer la sobrecarga explícitamente

iv. Describa la jerarquía de tipos, incluyendo mecanismos de herencia múltiple (de haberlos), polimorfismo paramétrico (de tenerlo) y manejo de varianzas.

R: En Kotlin, el tipo más alto en la **jerarquía de tipos** se llama Any. Es equivalente al tipo Object en Java. Esto significa que todas las clases en Kotlin heredan explícitamente del tipo Any, incluyendo String, Int, Double, y así sucesivamente. El tipo Any contiene tres métodos: equals, toString, y hashcode.



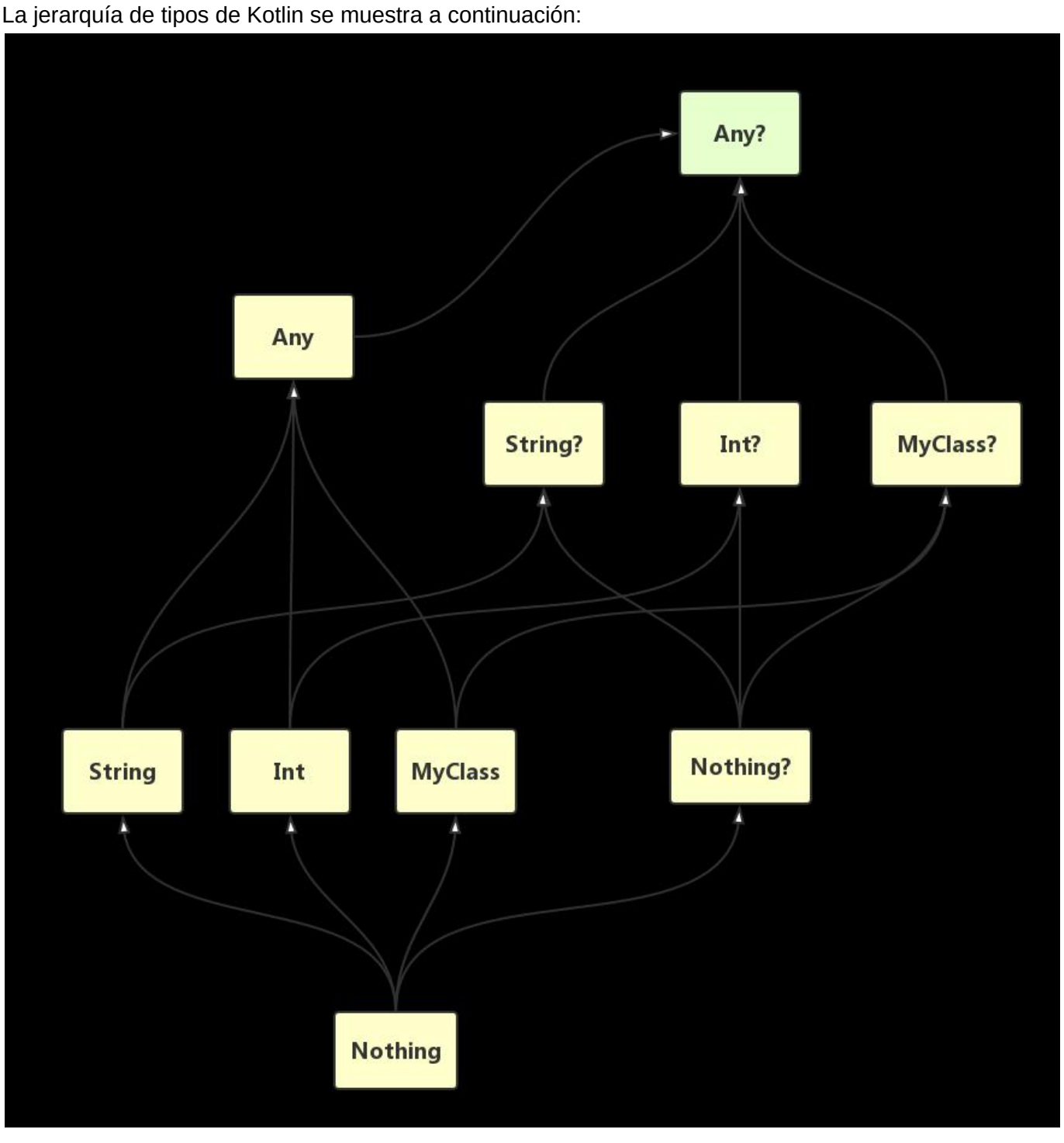
En se introducen tipos de datos básicos, pero para poder operar estos tipos de datos básicos como objetos, Java introduce los tipos de empaquetado correspondientes para cada tipo de datos básicos (clase de contenedor), la clase de contenedor de int es Integer, y desde Java 5 introdujo el mecanismo de autoboxing / unboxing, para que los dos puedan convertirse entre sí.

Java proporciona un tipo de contenedor para cada tipo primitivo:
Tipos primitivos: boolean, char, byte, short, int, long, float, double
Tipo de embalaje: booleano, carácter, byte, corto, entero, largo, flotante, doble

Kotlin elimina el tipo original, solo "tipo envuelto". Cuando el compilador compila el código, optimiza automáticamente el rendimiento y desempaqueta el tipo de empaquetado correspondiente al tipo original.

Los tipos de sistema Kotlin se dividen en tipos anulables y tipos no anulables. Kotlin introduce tipos anulables y usa tipos anulables para representar valores que pueden ser nulos. Esto establece un "límite" claro y explícito entre referencias anulables y no anulables.

La jerarquía de tipos de Kotlin se muestra a continuación:



Mediante el uso explícito de tipos anulables y la verificación de tipos en tiempo de compilación, la probabilidad de excepciones de puntero nulo se reduce considerablemente.

Para los tipos de números de Kotlin, los tipos no anulables corresponden a los tipos de números primitivos en Java. Como se muestra en la tabla a continuación

Kotlin	**Java**
Int	int
Long	long
Float	float
Double	double

El tipo de número anulable correspondiente en Kotlin es equivalente al tipo de número en caja en Java. Como se muestra en la tabla a continuación

Kotlin	**Java**
Int?	Integer
Long?	Long
Float?	Float
Double?	Double

- ► Kotlin tiene clases abstractas
  - ► Estas son abiertas por defecto
  - ► Aún así, los métodos que sobrecargan métodos y propiedades abstractas deben anotarse con override
- ► Kotlin tiene interfaces
  - ► Como la clases abstractas, pero no pueden mantener estado
  - ► Pueden proveer funcionalidad
    - ► Definir funciones por defecto

- ► Kotlin no tiene **herencia múltiple** para clases
  - ► Para interfaces, sin embargo, si está disponible

```
interface A {
   fun f() { print("hola") }
}

interface B {
   fun f() { print("chao") }
}

class C : A, B {
   override fun f() {
       super<A>.f()
       super<B>.f()
   }
}
```

- ► Kotlin cuenta con **polimorfismo** general
  - ► Similar al de Java

Manejo de varianza:

La varianza es la relación entre tipos. En general, el término varianza básicamente se refiere a la relación entre tipos genéricos que tienen la misma clase base pero diferentes argumentos de tipo. La varianza nos da la respuesta a preguntas como: "¿Están en la misma jerarquía de clases?" o “¿Es uno de ellos un subtipo del otro?”.

NOTA: el subtipo aquí se refiere al hecho de que una instancia del tipo derivado se puede usar donde se espera una instancia del tipo base, por lo que son intercambiables.

“La varianza establece la relación entre elementos genéricos que comparten la misma clase base”

La varianza es crucial para evitar inconsistencias en los tipos de datos y evitar bloqueos durante la ejecución del programa..

Dependiendo de la relación entre tipos genéricos, tenemos 3 escenarios posibles:

- **Invariancia:** los componentes genéricos no tienen ninguna relación.
- **Covarianza:** el componente genérico con un parámetro de tipo derivado se considera hijo del componente con un parámetro de tipo base.
- **Contravarianza:** el componente genérico con un parámetro de tipo derivado se considera un padre del otro.

Invariante:

```
val ints: Array<Int> = arrayOf(1, 2, 3)
val any = Array<Any>(3) { "" }
copy(ints, any)
//   ^  el tipo es Array<Int > pero se esperaba Array<Any>
```

Aquí se encuentra con un problema: Array<T> es **invariante** en T, por lo que ni Array<Int> ni Array<Any> son un subtipo del otro. ¿Por qué no? De nuevo, esto se debe a que copy podría tener un comportamiento inesperado, por ejemplo, puede intentar escribir una cadena en from, y si realmente pasa una arreglo de Int allí, se lanzará una *ClassCastException* más adelante.

En Kotlin se tiene una forma de definir genéricos para que sean **covariantes o contravariantes.**

Out:
Si observa la definición de la lista inmutable en Kotlin, verá esta línea:

```
interface List<out E> {
 fun get(index: Int): E
}
```

La palabra clave **out** dice que los métodos en una lista solo pueden devolver el tipo E y no pueden tomar ningún tipo E como argumento.

Esta limitación nos permite hacer que la lista sea **covariante.**

In:
Cuando revisas la definición de Comparar, puedes ver algo diferente:

```
interface Compare<in T> {
 fun compare(first: T, second: T): Int
}
```

En este caso, hay una palabra clave **in** junto al parámetro.
Esto significa que todos los métodos de Compare pueden tener T como argumento, pero no pueden devolver el tipo T.
Esto hace que la comparación sea **contravariante**.

En Kotlin, el desarrollador que escribe la declaración de clase debe considerar la varianza y no el desarrollador que usa el código. Es por eso que se llama varianza en el sitio de la declaración.

Pregunta 3.a (Asociación estática de métodos)

X = 3, Y = 1, Z = 3

```
1     class Bebida {
2        int a = 3, b = 1
3
4        fun s(int x) : int {
5            a = b - x
6            return t(a)
7        }
8
9        fun t(int y) : int {
10           return a * y + b
11       }
12    }
13
14    class Cafe extends Bebida {
15       Bebida caliente = new Marron()
16
17       fun q(int y) : int {
18           return caliente.s(a + b) - y
19       }
20    }
21
22    class Marron extends Cafe {
23       int c = 3
24
25       fun s(int x) : int {
26           a = x - 4
27           c = a + b * x
28           return t(a*b + c)
29       }
30
31       fun t(int y) : int {
32           return c - y
33       }
34    }
35
36    Bebida e = new Cafe()
37    Bebida pres = new Marron()
38    Cafe o = new Marron()
39
40    print(e.s(1) + pres.s(1) + o.s(1))
```

Nombre (Objeto)	Métodos	Valores Variables
e (Cafe())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1.
pres (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
o (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
e.caliente (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
pres.caliente (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
o.caliente (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.

IMPRIME

```
+------+------+----------------------------------------+
|global|  pc  | 40
+------+------+----------------------------------------+
```

Pregunta 3.a (Asociación estática de métodos)

X = 3, Y = 1, Z = 3

```
1     class Bebida {
2        int a = 3, b = 1
3
4        fun s(int x) : int {
5            a = b - x
6            return t(a)
7        }
8
9        fun t(int y) : int {
10           return a * y + b
11       }
12    }
13
14    class Cafe extends Bebida {
15       Bebida caliente = new Marron()
16
17       fun q(int y) : int {
18           return caliente.s(a + b) - y
19       }
20    }
21
22    class Marron extends Cafe {
23       int c = 3
24
25       fun s(int x) : int {
26           a = x - 4
27           c = a + b * x
28           return t(a*b + c)
29       }
30
31       fun t(int y) : int {
32           return c - y
33       }
34    }
35
36    Bebida e = new Cafe()
37    Bebida pres = new Marron()
38    Cafe o = new Marron()
39
40    print(e.s(1) + pres.s(1) + o.s(1))
```

Nombre (Objeto)	Métodos	Valores Variables
e (Cafe())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1.
pres (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
o (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
e.caliente (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
pres.caliente (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
o.caliente (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.

IMPRIME

```
+------+-----------------------------------------------+
|      |return| -
|      +-----------------------------------------------+
| e.s  |  x   | 1
|      +-----------------------------------------------+
|      |  pc  | 5
+------+------+-----------------------------------------+
|global|  pc  | 40
+------+------+-----------------------------------------+
```

Pregunta 3.a (Asociación estática de métodos)

X = 3, Y = 1, Z = 3

```
1     class Bebida {
2        int a = 3, b = 1
3
4        fun s(int x) : int {
5            a = b - x
6            return t(a)
7        }
8
9        fun t(int y) : int {
10           return a * y + b
11       }
12    }
13
14    class Cafe extends Bebida {
15       Bebida caliente = new Marron()
16
17       fun q(int y) : int {
18           return caliente.s(a + b) - y
19       }
20    }
21
22    class Marron extends Cafe {
23       int c = 3
24
25       fun s(int x) : int {
26           a = x - 4
27           c = a + b * x
28           return t(a*b + c)
29       }
30
31       fun t(int y) : int {
32           return c - y
33       }
34    }
35
36    Bebida e = new Cafe()
37    Bebida pres = new Marron()
38    Cafe o = new Marron()
39
40    print(e.s(1) + pres.s(1) + o.s(1))
```

Nombre (Objeto)	Métodos	Valores Variables
e (Cafe())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 1-1=0, b = 1.
pres (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
o (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
e.caliente (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
pres.caliente (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
o.caliente (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.

IMPRIME

```
+------+-----------------------------------------------+
|      |return| t(0)
|      +-----------------------------------------------+
| e.s  |  x   | 1
|      +-----------------------------------------------+
|      |  pc  | 6
+------+------+---------------------------------------+
|global|  pc  | 40
+------+------+---------------------------------------+
```

Pregunta 3.a (Asociación estática de métodos)

X = 3, Y = 1, Z = 3

```
1     class Bebida {
2        int a = 3, b = 1
3
4        fun s(int x) : int {
5            a = b - x
6            return t(a)
7        }
8
9        fun t(int y) : int {
10           return a * y + b
11       }
12    }
13
14    class Cafe extends Bebida {
15       Bebida caliente = new Marron()
16
17       fun q(int y) : int {
18           return caliente.s(a + b) - y
19       }
20    }
21
22    class Marron extends Cafe {
23       int c = 3
24
25       fun s(int x) : int {
26           a = x - 4
27           c = a + b * x
28           return t(a*b + c)
29       }
30
31       fun t(int y) : int {
32           return c - y
33       }
34    }
35
36    Bebida e = new Cafe()
37    Bebida pres = new Marron()
38    Cafe o = new Marron()
39
40    print(e.s(1) + pres.s(1) + o.s(1))
```

Nombre (Objeto)	Métodos	Valores Variables
e (Cafe())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a =0, b = 1.
pres (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
o (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
e.caliente (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
pres.caliente (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
o.caliente (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.

IMPRIME

```
+------+-----------------------------------------------+
|      |return| 0*0+1
|      +-----------------------------------------------+
| e.t  |  y   | 0
|      +-----------------------------------------------+
|      |  pc  | 10
+------+-----------------------------------------------+
|      |return| t(0)
|      +-----------------------------------------------+
| e.s  |  x   | 1
|      +-----------------------------------------------+
|      |  pc  | 6
+------+------+----------------------------------------+
|global|  pc  | 40
+------+------+----------------------------------------+
```

Pregunta 3.a (Asociación estática de métodos)

X = 3, Y = 1, Z = 3

```
1     class Bebida {
2        int a = 3, b = 1
3
4        fun s(int x) : int {
5            a = b - x
6            return t(a)
7        }
8
9        fun t(int y) : int {
10           return a * y + b
11       }
12    }
13
14    class Cafe extends Bebida {
15       Bebida caliente = new Marron()
16
17       fun q(int y) : int {
18           return caliente.s(a + b) - y
19       }
20    }
21
22    class Marron extends Cafe {
23       int c = 3
24
25       fun s(int x) : int {
26           a = x - 4
27           c = a + b * x
28           return t(a*b + c)
29       }
30
31       fun t(int y) : int {
32           return c - y
33       }
34    }
35
36    Bebida e = new Cafe()
37    Bebida pres = new Marron()
38    Cafe o = new Marron()
39
40    print(e.s(1) + pres.s(1) + o.s(1))
```

Nombre (Objeto)	Métodos	Valores Variables
e (Cafe())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a =0, b = 1.
pres (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
o (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
e.caliente (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
pres.caliente (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
o.caliente (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.

IMPRIME

```
+------+-----------------------------------------------+
|      |return| t(0) = 1
|      +-----------------------------------------------+
| e.s  |  x   | 1
|      +-----------------------------------------------+
|      |  pc  | 6
+------+------+---------------------------------------+
|global|  pc  | 40
+------+------+---------------------------------------+
```

Pregunta 3.a (Asociación estática de métodos)

X = 3, Y = 1, Z = 3

```
1     class Bebida {
2        int a = 3, b = 1
3
4        fun s(int x) : int {
5            a = b - x
6            return t(a)
7        }
8
9        fun t(int y) : int {
10           return a * y + b
11       }
12    }
13
14    class Cafe extends Bebida {
15       Bebida caliente = new Marron()
16
17       fun q(int y) : int {
18           return caliente.s(a + b) - y
19       }
20    }
21
22    class Marron extends Cafe {
23       int c = 3
24
25       fun s(int x) : int {
26           a = x - 4
27           c = a + b * x
28           return t(a*b + c)
29       }
30
31       fun t(int y) : int {
32           return c - y
33       }
34    }
35
36    Bebida e = new Cafe()
37    Bebida pres = new Marron()
38    Cafe o = new Marron()
39
40    print(e.s(1) + pres.s(1) + o.s(1))
```

1

Nombre (Objeto)	Métodos	Valores Variables
e (Cafe())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a =0, b = 1.
pres (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
o (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
e.caliente (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
pres.caliente (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
o.caliente (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.

IMPRIME

```
+------+------+----------------------------------------+
|global|  pc  | 40
+------+------+----------------------------------------+
```

Pregunta 3.a (Asociación estática de métodos)

X = 3, Y = 1, Z = 3

```
1     class Bebida {
2        int a = 3, b = 1
3
4        fun s(int x) : int {
5            a = b - x
6            return t(a)
7        }
8
9        fun t(int y) : int {
10           return a * y + b
11       }
12    }
13
14    class Cafe extends Bebida {
15       Bebida caliente = new Marron()
16
17       fun q(int y) : int {
18           return caliente.s(a + b) - y
19       }
20    }
21
22    class Marron extends Cafe {
23       int c = 3
24
25       fun s(int x) : int {
26           a = x - 4
27           c = a + b * x
28           return t(a*b + c)
29       }
30
31       fun t(int y) : int {
32           return c - y
33       }
34    }
35
36    Bebida e = new Cafe()
37    Bebida pres = new Marron()
38    Cafe o = new Marron()
39
40    print(e.s(1) + pres.s(1) + o.s(1))
```

1

Nombre (Objeto)	Métodos	Valores Variables
e (Cafe())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a =0, b = 1.
pres (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
o (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
e.caliente (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
pres.caliente (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
o.caliente (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.

IMPRIME

```
+------+-----------------------------------------------+
|      |return| -
|      +-----------------------------------------------+
|pres.s|  x   | 1
|      +-----------------------------------------------+
|      |  pc  | 5
+------+------+-----------------------------------------+
|global|  pc  | 40
+------+------+-----------------------------------------+
```

Pregunta 3.a (Asociación estática de métodos)

X = 3, Y = 1, Z = 3

```
1     class Bebida {
2        int a = 3, b = 1
3
4        fun s(int x) : int {
5            a = b - x
6            return t(a)
7        }
8
9        fun t(int y) : int {
10           return a * y + b
11       }
12    }
13
14    class Cafe extends Bebida {
15       Bebida caliente = new Marron()
16
17       fun q(int y) : int {
18           return caliente.s(a + b) - y
19       }
20    }
21
22    class Marron extends Cafe {
23       int c = 3
24
25       fun s(int x) : int {
26           a = x - 4
27           c = a + b * x
28           return t(a*b + c)
29       }
30
31       fun t(int y) : int {
32           return c - y
33       }
34    }
35
36    Bebida e = new Cafe()
37    Bebida pres = new Marron()
38    Cafe o = new Marron()
39
40    print(e.s(1) + pres.s(1) + o.s(1))
```

1

Nombre (Objeto)	Métodos	Valores Variables
e (Cafe())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a =0, b = 1.
pres (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 1 - 1=0, b = 1, c = 3.
o (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
e.caliente (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
pres.caliente (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
o.caliente (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.

IMPRIME

```
+------+-----------------------------------------------+
|      |return| -
|      +-----------------------------------------------+
|pres.s|  x   | 1
|      +-----------------------------------------------+
|      |  pc  | 6
+------+------+-----------------------------------------+
|global|  pc  | 40
+------+------+-----------------------------------------+
```

Pregunta 3.a (Asociación estática de métodos)

X = 3, Y = 1, Z = 3

```
1     class Bebida {
2        int a = 3, b = 1
3
4        fun s(int x) : int {
5            a = b - x
6            return t(a)
7        }
8
9        fun t(int y) : int {
10           return a * y + b
11       }
12    }
13
14    class Cafe extends Bebida {
15       Bebida caliente = new Marron()
16
17       fun q(int y) : int {
18           return caliente.s(a + b) - y
19       }
20    }
21
22    class Marron extends Cafe {
23       int c = 3
24
25       fun s(int x) : int {
26           a = x - 4
27           c = a + b * x
28           return t(a*b + c)
29       }
30
31       fun t(int y) : int {
32           return c - y
33       }
34    }
35
36    Bebida e = new Cafe()
37    Bebida pres = new Marron()
38    Cafe o = new Marron()
39
40    print(e.s(1) + pres.s(1) + o.s(1))
```

1

Nombre (Objeto)	Métodos	Valores Variables
e (Cafe())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 0, b = 1.
pres (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 0, b = 1, c = 3.
o (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
e.caliente (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
pres.caliente (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
o.caliente (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.

IMPRIME

```
+------+-----------------------------------------------+
|      |return| t(0)
|      +-----------------------------------------------+
|pres.s|  x   | 1
|      +-----------------------------------------------+
|      |  pc  | 6
+------+------+---------------------------------------+
|global|  pc  | 40
+------+------+---------------------------------------+
```

Pregunta 3.a (Asociación estática de métodos)

X = 3, Y = 1, Z = 3

```
1     class Bebida {
2        int a = 3, b = 1
3
4        fun s(int x) : int {
5            a = b - x
6            return t(a)
7        }
8
9        fun t(int y) : int {
10           return a * y + b
11       }
12    }
13
14    class Cafe extends Bebida {
15       Bebida caliente = new Marron()
16
17       fun q(int y) : int {
18           return caliente.s(a + b) - y
19       }
20    }
21
22    class Marron extends Cafe {
23       int c = 3
24
25       fun s(int x) : int {
26           a = x - 4
27           c = a + b * x
28           return t(a*b + c)
29       }
30
31       fun t(int y) : int {
32           return c - y
33       }
34    }
35
36    Bebida e = new Cafe()
37    Bebida pres = new Marron()
38    Cafe o = new Marron()
39
40    print(e.s(1) + pres.s(1) + o.s(1))
```

1

Nombre (Objeto)	Métodos	Valores Variables
e (Cafe())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 0, b = 1.
pres (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 0, b = 1, c = 3.
o (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
e.caliente (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
pres.caliente (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
o.caliente (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.

IMPRIME

```
+------+-----------------------------------------------+
|      |return| 0*0+1
|      +-----------------------------------------------+
|pres.t|  y   | 0
|      +-----------------------------------------------+
|      |  pc  | 10
+------+-----------------------------------------------+
|      |return| t(0)
|      +-----------------------------------------------+
|pres.s|  x   | 1
|      +-----------------------------------------------+
|      |  pc  | 6
+------+------+----------------------------------------+
|global|  pc  | 40
+------+------+----------------------------------------+
```

Pregunta 3.a (Asociación estática de métodos)

X = 3, Y = 1, Z = 3

```
1     class Bebida {
2        int a = 3, b = 1
3
4        fun s(int x) : int {
5            a = b - x
6            return t(a)
7        }
8
9        fun t(int y) : int {
10           return a * y + b
11       }
12    }
13
14    class Cafe extends Bebida {
15       Bebida caliente = new Marron()
16
17       fun q(int y) : int {
18           return caliente.s(a + b) - y
19       }
20    }
21
22    class Marron extends Cafe {
23       int c = 3
24
25       fun s(int x) : int {
26           a = x - 4
27           c = a + b * x
28           return t(a*b + c)
29       }
30
31       fun t(int y) : int {
32           return c - y
33       }
34    }
35
36    Bebida e = new Cafe()
37    Bebida pres = new Marron()
38    Cafe o = new Marron()
39
40    print(e.s(1) + pres.s(1) + o.s(1))
```

1

Nombre (Objeto)	Métodos	Valores Variables
e (Cafe())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 0, b = 1.
pres (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 0, b = 1, c = 3.
o (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
e.caliente (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
pres.caliente (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
o.caliente (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.

IMPRIME

```
+------+-----------------------------------------------+
|      |return| 1
|      +-----------------------------------------------+
|pres.s|  x   | 1
|      +-----------------------------------------------+
|      |  pc  | 6
+------+------+-----------------------------------------+
|global|  pc  | 40
+------+------+-----------------------------------------+
```

Pregunta 3.a (Asociación estática de métodos)

X = 3, Y = 1, Z = 3

```
1     class Bebida {
2        int a = 3, b = 1
3
4        fun s(int x) : int {
5            a = b - x
6            return t(a)
7        }
8
9        fun t(int y) : int {
10           return a * y + b
11       }
12    }
13
14    class Cafe extends Bebida {
15       Bebida caliente = new Marron()
16
17       fun q(int y) : int {
18           return caliente.s(a + b) - y
19       }
20    }
21
22    class Marron extends Cafe {
23       int c = 3
24
25       fun s(int x) : int {
26           a = x - 4
27           c = a + b * x
28           return t(a*b + c)
29       }
30
31       fun t(int y) : int {
32           return c - y
33       }
34    }
35
36    Bebida e = new Cafe()
37    Bebida pres = new Marron()
38    Cafe o = new Marron()
39
40    print(e.s(1) + pres.s(1) + o.s(1))
```

1 1

Nombre (Objeto)	Métodos	Valores Variables
e (Cafe())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 0, b = 1.
pres (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 0, b = 1, c = 3.
o (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
e.caliente (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
pres.caliente (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
o.caliente (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.

IMPRIME

```
+------+------+----------------------------------------+
|global|  pc  | 40
+------+------+----------------------------------------+
```

Pregunta 3.a (Asociación estática de métodos)

X = 3, Y = 1, Z = 3

```
1     class Bebida {
2        int a = 3, b = 1
3
4        fun s(int x) : int {
5            a = b - x
6            return t(a)
7        }
8
9        fun t(int y) : int {
10           return a * y + b
11       }
12    }
13
14    class Cafe extends Bebida {
15       Bebida caliente = new Marron()
16
17       fun q(int y) : int {
18           return caliente.s(a + b) - y
19       }
20    }
21
22    class Marron extends Cafe {
23       int c = 3
24
25       fun s(int x) : int {
26           a = x - 4
27           c = a + b * x
28           return t(a*b + c)
29       }
30
31       fun t(int y) : int {
32           return c - y
33       }
34    }
35
36    Bebida e = new Cafe()
37    Bebida pres = new Marron()
38    Cafe o = new Marron()
39
40    print(e.s(1) + pres.s(1) + o.s(1))
```

1 1

Nombre (Objeto)	Métodos	Valores Variables
e (Cafe())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a =0, b = 1.
pres (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 0, b = 1, c = 3.
o (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
e.caliente (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
pres.caliente (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
o.caliente (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.

IMPRIME

```
+------+-----------------------------------------------+
|      |return| -
|      +-----------------------------------------------+
| o.s  |  x   | 1
|      +-----------------------------------------------+
|      |  pc  | 5
+------+------+-----------------------------------------+
|global|  pc  | 40
+------+------+-----------------------------------------+
```

Pregunta 3.a (Asociación estática de métodos)

X = 3, Y = 1, Z = 3

```
1     class Bebida {
2        int a = 3, b = 1
3
4        fun s(int x) : int {
5            a = b - x
6            return t(a)
7        }
8
9        fun t(int y) : int {
10           return a * y + b
11       }
12    }
13
14    class Cafe extends Bebida {
15       Bebida caliente = new Marron()
16
17       fun q(int y) : int {
18           return caliente.s(a + b) - y
19       }
20    }
21
22    class Marron extends Cafe {
23       int c = 3
24
25       fun s(int x) : int {
26           a = x - 4
27           c = a + b * x
28           return t(a*b + c)
29       }
30
31       fun t(int y) : int {
32           return c - y
33       }
34    }
35
36    Bebida e = new Cafe()
37    Bebida pres = new Marron()
38    Cafe o = new Marron()
39
40    print(e.s(1) + pres.s(1) + o.s(1))
```

1 1

Nombre (Objeto)	Métodos	Valores Variables
e (Cafe())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a =0, b = 1.
pres (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a =0, b = 1, c = 3.
o (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 1-1=0, b = 1, c = 3.
e.caliente (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
pres.caliente (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
o.caliente (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.

IMPRIME

```
+------+-----------------------------------------------+
|      |return| -
|      +-----------------------------------------------+
| o.s  |  x   | 1
|      +-----------------------------------------------+
|      |  pc  | 6
+------+------+---------------------------------------+
|global|  pc  | 40
+------+------+---------------------------------------+
```

Pregunta 3.a (Asociación estática de métodos)

X = 3, Y = 1, Z = 3

```
1     class Bebida {
2        int a = 3, b = 1
3
4        fun s(int x) : int {
5            a = b - x
6            return t(a)
7        }
8
9        fun t(int y) : int {
10           return a * y + b
11       }
12    }
13
14    class Cafe extends Bebida {
15       Bebida caliente = new Marron()
16
17       fun q(int y) : int {
18           return caliente.s(a + b) - y
19       }
20    }
21
22    class Marron extends Cafe {
23       int c = 3
24
25       fun s(int x) : int {
26           a = x - 4
27           c = a + b * x
28           return t(a*b + c)
29       }
30
31       fun t(int y) : int {
32           return c - y
33       }
34    }
35
36    Bebida e = new Cafe()
37    Bebida pres = new Marron()
38    Cafe o = new Marron()
39
40    print(e.s(1) + pres.s(1) + o.s(1))
```

1 1

Nombre (Objeto)	Métodos	Valores Variables
e (Cafe())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 0, b = 1.
pres (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 0, b = 1, c = 3.
o (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 0, b = 1, c = 3.
e.caliente (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
pres.caliente (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
o.caliente (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.

IMPRIME

```
+------+-----------------------------------------------+
|      |return| t(0)
|      +-----------------------------------------------+
|pres.s|  x   | 1
|      +-----------------------------------------------+
|      |  pc  | 6
+------+------+---------------------------------------+
|global|  pc  | 40
+------+------+---------------------------------------+
```

Pregunta 3.a (Asociación estática de métodos)

X = 3, Y = 1, Z = 3

```
1     class Bebida {
2        int a = 3, b = 1
3
4        fun s(int x) : int {
5            a = b - x
6            return t(a)
7        }
8
9        fun t(int y) : int {
10           return a * y + b
11       }
12    }
13
14    class Cafe extends Bebida {
15       Bebida caliente = new Marron()
16
17       fun q(int y) : int {
18           return caliente.s(a + b) - y
19       }
20    }
21
22    class Marron extends Cafe {
23       int c = 3
24
25       fun s(int x) : int {
26           a = x - 4
27           c = a + b * x
28           return t(a*b + c)
29       }
30
31       fun t(int y) : int {
32           return c - y
33       }
34    }
35
36    Bebida e = new Cafe()
37    Bebida pres = new Marron()
38    Cafe o = new Marron()
39
40    print(e.s(1) + pres.s(1) + o.s(1))
```

1 1

Nombre (Objeto)	Métodos	Valores Variables
e (Cafe())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 0, b = 1.
pres (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 0, b = 1, c = 3.
o (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 0, b = 1, c = 3.
e.caliente (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
pres.caliente (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
o.caliente (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.

IMPRIME

```
+------+-----------------------------------------------+
|      |return| 0*0+1
|      +-----------------------------------------------+
| o.t  |  y   | 0
|      +-----------------------------------------------+
|      |  pc  | 10
+------+-----------------------------------------------+
|      |return| t(0)
|      +-----------------------------------------------+
| o.s  |  x   | 1
|      +-----------------------------------------------+
|      |  pc  | 6
+------+------+----------------------------------------+
|global|  pc  | 40
+------+------+----------------------------------------+
```

Pregunta 3.a (Asociación estática de métodos)

X = 3, Y = 1, Z = 3

```
1     class Bebida {
2        int a = 3, b = 1
3
4        fun s(int x) : int {
5            a = b - x
6            return t(a)
7        }
8
9        fun t(int y) : int {
10           return a * y + b
11       }
12    }
13
14    class Cafe extends Bebida {
15       Bebida caliente = new Marron()
16
17       fun q(int y) : int {
18           return caliente.s(a + b) - y
19       }
20    }
21
22    class Marron extends Cafe {
23       int c = 3
24
25       fun s(int x) : int {
26           a = x - 4
27           c = a + b * x
28           return t(a*b + c)
29       }
30
31       fun t(int y) : int {
32           return c - y
33       }
34    }
35
36    Bebida e = new Cafe()
37    Bebida pres = new Marron()
38    Cafe o = new Marron()
39
40    print(e.s(1) + pres.s(1) + o.s(1))
```

1 1

Nombre (Objeto)	Métodos	Valores Variables
e (Cafe())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 0, b = 1.
pres (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 0, b = 1, c = 3.
o (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 0, b = 1, c = 3.
e.caliente (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
pres.caliente (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
o.caliente (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.

IMPRIME

```
+------+-----------------------------------------------+
|      |return| 1
|      +-----------------------------------------------+
| o.s  |  x   | 1
|      +-----------------------------------------------+
|      |  pc  | 6
+------+------+---------------------------------------+
|global|  pc  | 40
+------+------+---------------------------------------+
```

Pregunta 3.a (Asociación estática de métodos)

X = 3, Y = 1, Z = 3

```
1     class Bebida {
2        int a = 3, b = 1
3
4        fun s(int x) : int {
5            a = b - x
6            return t(a)
7        }
8
9        fun t(int y) : int {
10           return a * y + b
11       }
12    }
13
14    class Cafe extends Bebida {
15       Bebida caliente = new Marron()
16
17       fun q(int y) : int {
18           return caliente.s(a + b) - y
19       }
20    }
21
22    class Marron extends Cafe {
23       int c = 3
24
25       fun s(int x) : int {
26           a = x - 4
27           c = a + b * x
28           return t(a*b + c)
29       }
30
31       fun t(int y) : int {
32           return c - y
33       }
34    }
35
36    Bebida e = new Cafe()
37    Bebida pres = new Marron()
38    Cafe o = new Marron()
39
40    print(e.s(1) + pres.s(1) + o.s(1))
```

1 1 1

Nombre (Objeto)	Métodos	Valores Variables
e (Cafe())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 0, b = 1.
pres (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 0, b = 1, c = 3.
o (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 0, b = 1, c = 3.
e.caliente (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
pres.caliente (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
o.caliente (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.

IMPRIME

```
+------+------+---------------------------------------+
|global|  pc  | 40
+------+------+---------------------------------------+
```

Pregunta 3.a (Asociación estática de métodos)

X = 3, Y = 1, Z = 3

```
1     class Bebida {
2        int a = 3, b = 1
3
4        fun s(int x) : int {
5            a = b - x
6            return t(a)
7        }
8
9        fun t(int y) : int {
10           return a * y + b
11       }
12    }
13
14    class Cafe extends Bebida {
15       Bebida caliente = new Marron()
16
17       fun q(int y) : int {
18           return caliente.s(a + b) - y
19       }
20    }
21
22    class Marron extends Cafe {
23       int c = 3
24
25       fun s(int x) : int {
26           a = x - 4
27           c = a + b * x
28           return t(a*b + c)
29       }
30
31       fun t(int y) : int {
32           return c - y
33       }
34    }
35
36    Bebida e = new Cafe()
37    Bebida pres = new Marron()
38    Cafe o = new Marron()
39
40    print(e.s(1) + pres.s(1) + o.s(1))
```

Nombre (Objeto)	Métodos	Valores Variables
e (Cafe())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 0, b = 1.
pres (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 0, b = 1, c = 3.
o (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 0, b = 1, c = 3.
e.caliente (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
pres.caliente (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
o.caliente (Marron())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.

IMPRIME

3

Pregunta 3.b (Asociación dinámica de métodos)

X = 3, Y = 1, Z = 3

```
1     class Bebida {
2        int a = 3, b = 1
3
4        fun s(int x) : int {
5            a = b - x
6            return t(a)
7        }
8
9        fun t(int y) : int {
10           return a * y + b
11       }
12    }
13
14    class Cafe extends Bebida {
15       Bebida caliente = new Marron()
16
17       fun q(int y) : int {
18           return caliente.s(a + b) - y
19       }
20    }
21
22    class Marron extends Cafe {
23       int c = 3
24
25       fun s(int x) : int {
26           a = x - 4
27           c = a + b * x
28           return t(a*b + c)
29       }
30
31       fun t(int y) : int {
32           return c - y
33       }
34    }
35
36    Bebida e = new Cafe()
37    Bebida pres = new Marron()
38    Cafe o = new Marron()
39
40    print(e.s(1) + pres.s(1) + o.s(1))
```

Nombre (Objeto)	Métodos	Valores Variables
e (Cafe())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1.
pres (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
o (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
e.caliente (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
pres.caliente (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
o.caliente (Marron())	s -> Marron :: s, t -> Maroon :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.

IMPRIME

```
+------+------+----------------------------------------+
|global|  pc  | 40
+------+------+----------------------------------------+
```

Pregunta 3.b (Asociación dinámica de métodos)

X = 3, Y = 1, Z = 3

```
1     class Bebida {
2        int a = 3, b = 1
3
4        fun s(int x) : int {
5            a = b - x
6            return t(a)
7        }
8
9        fun t(int y) : int {
10           return a * y + b
11       }
12    }
13
14    class Cafe extends Bebida {
15       Bebida caliente = new Marron()
16
17       fun q(int y) : int {
18           return caliente.s(a + b) - y
19       }
20    }
21
22    class Marron extends Cafe {
23       int c = 3
24
25       fun s(int x) : int {
26           a = x - 4
27           c = a + b * x
28           return t(a*b + c)
29       }
30
31       fun t(int y) : int {
32           return c - y
33       }
34    }
35
36    Bebida e = new Cafe()
37    Bebida pres = new Marron()
38    Cafe o = new Marron()
39
40    print(e.s(1) + pres.s(1) + o.s(1))
```

Nombre (Objeto)	Métodos	Valores Variables
e (Cafe())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 3, b = 1.
pres (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
o (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
e.caliente (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
pres.caliente (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
o.caliente (Marron())	s -> Marron :: s, t -> Maroon :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.

IMPRIME

```
+------+-----------------------------------------------+
|      |return| -
|      +-----------------------------------------------+
| e.s  |  x   | 1
|      +-----------------------------------------------+
|      |  pc  | 5
+------+------+-----------------------------------------+
|global|  pc  | 40
+------+------+-----------------------------------------+
```

Pregunta 3.b (Asociación dinámica de métodos)

X = 3, Y = 1, Z = 3

```
1     class Bebida {
2        int a = 3, b = 1
3
4        fun s(int x) : int {
5            a = b - x
6            return t(a)
7        }
8
9        fun t(int y) : int {
10           return a * y + b
11       }
12    }
13
14    class Cafe extends Bebida {
15       Bebida caliente = new Marron()
16
17       fun q(int y) : int {
18           return caliente.s(a + b) - y
19       }
20    }
21
22    class Marron extends Cafe {
23       int c = 3
24
25       fun s(int x) : int {
26           a = x - 4
27           c = a + b * x
28           return t(a*b + c)
29       }
30
31       fun t(int y) : int {
32           return c - y
33       }
34    }
35
36    Bebida e = new Cafe()
37    Bebida pres = new Marron()
38    Cafe o = new Marron()
39
40    print(e.s(1) + pres.s(1) + o.s(1))
```

Nombre (Objeto)	Métodos	Valores Variables
e (Cafe())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 1-1 = 0, b = 1.
pres (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
o (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
e.caliente (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
pres.caliente (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
o.caliente (Marron())	s -> Marron :: s, t -> Maroon :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.

IMPRIME

```
+------+-----------------------------------------+
|      |return| t(0)
|      +-----------------------------------------+
| e.s  |  x   | 1
|      +-----------------------------------------+
|      |  pc  | 6
+------+------+-----------------------------------+
|global|  pc  | 40
+------+------+-----------------------------------+
```

Pregunta 3.b (Asociación dinámica de métodos)

X = 3, Y = 1, Z = 3

```
1     class Bebida {
2        int a = 3, b = 1
3
4        fun s(int x) : int {
5            a = b - x
6            return t(a)
7        }
8
9        fun t(int y) : int {
10           return a * y + b
11       }
12    }
13
14    class Cafe extends Bebida {
15       Bebida caliente = new Marron()
16
17       fun q(int y) : int {
18           return caliente.s(a + b) - y
19       }
20    }
21
22    class Marron extends Cafe {
23       int c = 3
24
25       fun s(int x) : int {
26           a = x - 4
27           c = a + b * x
28           return t(a*b + c)
29       }
30
31       fun t(int y) : int {
32           return c - y
33       }
34    }
35
36    Bebida e = new Cafe()
37    Bebida pres = new Marron()
38    Cafe o = new Marron()
39
40    print(e.s(1) + pres.s(1) + o.s(1))
```

Nombre (Objeto)	Métodos	Valores Variables
e (Cafe())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 0, b = 1.
pres (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
o (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
e.caliente (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
pres.caliente (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
o.caliente (Marron())	s -> Marron :: s, t -> Maroon :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.

IMPRIME

```
+------+-----------------------------------------+
|      |return| 0*0+1
|      +-----------------------------------------+
| e.t  |  y   | 0
|      +-----------------------------------------+
|      |  pc  | 10
+------+-----------------------------------------+
|      |return| t(0)
|      +-----------------------------------------+
| e.s  |  x   | 1
|      +-----------------------------------------+
|      |  pc  | 6
+------+------+----------------------------------+
|global|  pc  | 40
+------+------+----------------------------------+
```

Pregunta 3.b (Asociación dinámica de métodos)

X = 3, Y = 1, Z = 3

```
1     class Bebida {
2        int a = 3, b = 1
3
4        fun s(int x) : int {
5            a = b - x
6            return t(a)
7        }
8
9        fun t(int y) : int {
10           return a * y + b
11       }
12    }
13
14    class Cafe extends Bebida {
15       Bebida caliente = new Marron()
16
17       fun q(int y) : int {
18           return caliente.s(a + b) - y
19       }
20    }
21
22    class Marron extends Cafe {
23       int c = 3
24
25       fun s(int x) : int {
26           a = x - 4
27           c = a + b * x
28           return t(a*b + c)
29       }
30
31       fun t(int y) : int {
32           return c - y
33       }
34    }
35
36    Bebida e = new Cafe()
37    Bebida pres = new Marron()
38    Cafe o = new Marron()
39
40    print(e.s(1) + pres.s(1) + o.s(1))
```

Nombre (Objeto)	Métodos	Valores Variables
e (Cafe())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 0, b = 1.
pres (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
o (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
e.caliente (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
pres.caliente (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
o.caliente (Marron())	s -> Marron :: s, t -> Maroon :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.

IMPRIME

```
+------+-----------------------------------------------+
|      |return| 1
|      +-----------------------------------------------+
| e.s  |  x   | 1
|      +-----------------------------------------------+
|      |  pc  | 6
+------+------+-----------------------------------------+
|global|  pc  | 40
+------+------+-----------------------------------------+
```

Pregunta 3.b (Asociación dinámica de métodos)

X = 3, Y = 1, Z = 3

```
1     class Bebida {
2        int a = 3, b = 1
3
4        fun s(int x) : int {
5            a = b - x
6            return t(a)
7        }
8
9        fun t(int y) : int {
10           return a * y + b
11       }
12    }
13
14    class Cafe extends Bebida {
15       Bebida caliente = new Marron()
16
17       fun q(int y) : int {
18           return caliente.s(a + b) - y
19       }
20    }
21
22    class Marron extends Cafe {
23       int c = 3
24
25       fun s(int x) : int {
26           a = x - 4
27           c = a + b * x
28           return t(a*b + c)
29       }
30
31       fun t(int y) : int {
32           return c - y
33       }
34    }
35
36    Bebida e = new Cafe()
37    Bebida pres = new Marron()
38    Cafe o = new Marron()
39
40    print(e.s(1) + pres.s(1) + o.s(1))
```

1

Nombre (Objeto)	Métodos	Valores Variables
e (Cafe())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 0, b = 1.
pres (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
o (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
e.caliente (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
pres.caliente (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
o.caliente (Marron())	s -> Marron :: s, t -> Maroon :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.

IMPRIME

```
+------+------+----------------------------------------+
|global|  pc  | 40
+------+------+----------------------------------------+
```

Pregunta 3.b (Asociación dinámica de métodos)

X = 3, Y = 1, Z = 3

```
1     class Bebida {
2        int a = 3, b = 1
3
4        fun s(int x) : int {
5            a = b - x
6            return t(a)
7        }
8
9        fun t(int y) : int {
10           return a * y + b
11       }
12    }
13
14    class Cafe extends Bebida {
15       Bebida caliente = new Marron()
16
17       fun q(int y) : int {
18           return caliente.s(a + b) - y
19       }
20    }
21
22    class Marron extends Cafe {
23       int c = 3
24
25       fun s(int x) : int {
26           a = x - 4
27           c = a + b * x
28           return t(a*b + c)
29       }
30
31       fun t(int y) : int {
32           return c - y
33       }
34    }
35
36    Bebida e = new Cafe()
37    Bebida pres = new Marron()
38    Cafe o = new Marron()
39
40    print(e.s(1) + pres.s(1) + o.s(1))
```

1

Nombre (Objeto)	Métodos	Valores Variables
e (Cafe())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 0, b = 1.
pres (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
o (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
e.caliente (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
pres.caliente (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
o.caliente (Marron())	s -> Marron :: s, t -> Maroon :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.

IMPRIME

```
+------+-----------------------------------------------+
|      |return| -
|      +-----------------------------------------------+
|pres.s|  x   | 1
|      +-----------------------------------------------+
|      |  pc  | 26
+------+------+-----------------------------------------+
|global|  pc  | 40
+------+------+-----------------------------------------+
```

Pregunta 3.b (Asociación dinámica de métodos)

X = 3, Y = 1, Z = 3

```
1     class Bebida {
2        int a = 3, b = 1
3
4        fun s(int x) : int {
5            a = b - x
6            return t(a)
7        }
8
9        fun t(int y) : int {
10           return a * y + b
11       }
12    }
13
14    class Cafe extends Bebida {
15       Bebida caliente = new Marron()
16
17       fun q(int y) : int {
18           return caliente.s(a + b) - y
19       }
20    }
21
22    class Marron extends Cafe {
23       int c = 3
24
25       fun s(int x) : int {
26           a = x - 4
27           c = a + b * x
28           return t(a*b + c)
29       }
30
31       fun t(int y) : int {
32           return c - y
33       }
34    }
35
36    Bebida e = new Cafe()
37    Bebida pres = new Marron()
38    Cafe o = new Marron()
39
40    print(e.s(1) + pres.s(1) + o.s(1))
```

1

Nombre (Objeto)	Métodos	Valores Variables
e (Cafe())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 0, b = 1.
pres (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = 1-4 = -3 b = 1, c = 3.
o (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
e.caliente (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
pres.caliente (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
o.caliente (Marron())	s -> Marron :: s, t -> Maroon :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.

IMPRIME

```
+------+-----------------------------------------------+
|      |return| -
|      +-----------------------------------------------+
|pres.s|  x   | 1
|      +-----------------------------------------------+
|      |  pc  | 26 27
+------+------+---------------------------------------+
|global|  pc  | 40
+------+------+---------------------------------------+
```

Pregunta 3.b (Asociación dinámica de métodos)

X = 3, Y = 1, Z = 3

```
1     class Bebida {
2        int a = 3, b = 1
3
4        fun s(int x) : int {
5            a = b - x
6            return t(a)
7        }
8
9        fun t(int y) : int {
10           return a * y + b
11       }
12    }
13
14    class Cafe extends Bebida {
15       Bebida caliente = new Marron()
16
17       fun q(int y) : int {
18           return caliente.s(a + b) - y
19       }
20    }
21
22    class Marron extends Cafe {
23       int c = 3
24
25       fun s(int x) : int {
26           a = x - 4
27           c = a + b * x
28           return t(a*b + c)
29       }
30
31       fun t(int y) : int {
32           return c - y
33       }
34    }
35
36    Bebida e = new Cafe()
37    Bebida pres = new Marron()
38    Cafe o = new Marron()
39
40    print(e.s(1) + pres.s(1) + o.s(1))
```

1

Nombre (Objeto)	Métodos	Valores Variables
e (Cafe())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 0, b = 1.
pres (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = -3, b = 1, c=-3+1*1=-2
o (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
e.caliente (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
pres.caliente (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
o.caliente (Marron())	s -> Marron :: s, t -> Maroon :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.

IMPRIME

```
+------+-----------------------------------------------+
|      |return| t(-3*1 + -2)
|      +-----------------------------------------------+
|pres.s|  x   | 1
|      +-----------------------------------------------+
|      |  pc  | 26 27 28
+------+------+---------------------------------------+
|global|  pc  | 40
+------+------+---------------------------------------+
```

Pregunta 3.b (Asociación dinámica de métodos)

X = 3, Y = 1, Z = 3

```
1     class Bebida {
2        int a = 3, b = 1
3
4        fun s(int x) : int {
5            a = b - x
6            return t(a)
7        }
8
9        fun t(int y) : int {
10           return a * y + b
11       }
12    }
13
14    class Cafe extends Bebida {
15       Bebida caliente = new Marron()
16
17       fun q(int y) : int {
18           return caliente.s(a + b) - y
19       }
20    }
21
22    class Marron extends Cafe {
23       int c = 3
24
25       fun s(int x) : int {
26           a = x - 4
27           c = a + b * x
28           return t(a*b + c)
29       }
30
31       fun t(int y) : int {
32           return c - y
33       }
34    }
35
36    Bebida e = new Cafe()
37    Bebida pres = new Marron()
38    Cafe o = new Marron()
39
40    print(e.s(1) + pres.s(1) + o.s(1))
```

1

Nombre (Objeto)	Métodos	Valores Variables
e (Cafe())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 0, b = 1.
pres (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = -3, b = 1, c = -2
o (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
e.caliente (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
pres.caliente (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
o.caliente (Marron())	s -> Marron :: s, t -> Maroon :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.

IMPRIME

```
+------+-----------------------------------------------+
|      |return| t(-2-(-5))
|      +-----------------------------------------------+
|pres.t|  y   | -5
|      +-----------------------------------------------+
|      |  pc  | 32
+------+-----------------------------------------------+
|      |return| t(-5)
|      +-----------------------------------------------+
|pres.s|  x   | 1
|      +-----------------------------------------------+
|      |  pc  | 26 27 28
+------+------+----------------------------------------+
|global|  pc  | 40
+------+------+----------------------------------------+
```

Pregunta 3.b (Asociación dinámica de métodos)

X = 3, Y = 1, Z = 3

```
1     class Bebida {
2        int a = 3, b = 1
3
4        fun s(int x) : int {
5            a = b - x
6            return t(a)
7        }
8
9        fun t(int y) : int {
10           return a * y + b
11       }
12    }
13
14    class Cafe extends Bebida {
15       Bebida caliente = new Marron()
16
17       fun q(int y) : int {
18           return caliente.s(a + b) - y
19       }
20    }
21
22    class Marron extends Cafe {
23       int c = 3
24
25       fun s(int x) : int {
26           a = x - 4
27           c = a + b * x
28           return t(a*b + c)
29       }
30
31       fun t(int y) : int {
32           return c - y
33       }
34    }
35
36    Bebida e = new Cafe()
37    Bebida pres = new Marron()
38    Cafe o = new Marron()
39
40    print(e.s(1) + pres.s(1) + o.s(1))
```

1

Nombre (Objeto)	Métodos	Valores Variables
e (Cafe())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 0, b = 1.
pres (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = -3, b = 1, c = -2
o (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
e.caliente (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
pres.caliente (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
o.caliente (Marron())	s -> Marron :: s, t -> Maroon :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.

IMPRIME

```
+------+-----------------------------------------------+
|      |return| 3
|      +-----------------------------------------------+
|pres.s|  x   | 1
|      +-----------------------------------------------+
|      |  pc  | ~~26~~ ~~27~~ 28
+------+------+---------------------------------------+
|global|  pc  | 40
+------+------+---------------------------------------+
```

Pregunta 3.b (Asociación dinámica de métodos)

X = 3, Y = 1, Z = 3

```
1     class Bebida {
2        int a = 3, b = 1
3
4        fun s(int x) : int {
5            a = b - x
6            return t(a)
7        }
8
9        fun t(int y) : int {
10           return a * y + b
11       }
12    }
13
14    class Cafe extends Bebida {
15       Bebida caliente = new Marron()
16
17       fun q(int y) : int {
18           return caliente.s(a + b) - y
19       }
20    }
21
22    class Marron extends Cafe {
23       int c = 3
24
25       fun s(int x) : int {
26           a = x - 4
27           c = a + b * x
28           return t(a*b + c)
29       }
30
31       fun t(int y) : int {
32           return c - y
33       }
34    }
35
36    Bebida e = new Cafe()
37    Bebida pres = new Marron()
38    Cafe o = new Marron()
39
40    print(e.s(1) + pres.s(1) + o.s(1))
```

1 3

Nombre (Objeto)	Métodos	Valores Variables
e (Cafe())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 0, b = 1.
pres (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = -3, b = 1, c = -2
o (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
e.caliente (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
pres.caliente (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
o.caliente (Marron())	s -> Marron :: s, t -> Maroon :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.

IMPRIME

```
+------+------+---------------------------------------+
|global|  pc  | 40
+------+------+---------------------------------------+
```

Pregunta 3.b (Asociación dinámica de métodos)

X = 3, Y = 1, Z = 3

```
1     class Bebida {
2        int a = 3, b = 1
3
4        fun s(int x) : int {
5            a = b - x
6            return t(a)
7        }
8
9        fun t(int y) : int {
10           return a * y + b
11       }
12    }
13
14    class Cafe extends Bebida {
15       Bebida caliente = new Marron()
16
17       fun q(int y) : int {
18           return caliente.s(a + b) - y
19       }
20    }
21
22    class Marron extends Cafe {
23       int c = 3
24
25       fun s(int x) : int {
26           a = x - 4
27           c = a + b * x
28           return t(a*b + c)
29       }
30
31       fun t(int y) : int {
32           return c - y
33       }
34    }
35
36    Bebida e = new Cafe()
37    Bebida pres = new Marron()
38    Cafe o = new Marron()
39
40    print(e.s(1) + pres.s(1) + o.s(1))
```

1 **3**

Nombre (Objeto)	Métodos	Valores Variables
e (Cafe())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 0, b = 1.
pres (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = -3, b = 1, c = -2
o (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
e.caliente (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
pres.caliente (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
o.caliente (Marron())	s -> Marron :: s, t -> Maroon :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.

IMPRIME

```
+------+-----------------------------------------------+
|      |return| -
|      +-----------------------------------------------+
| o.s  |  x   | 1
|      +-----------------------------------------------+
|      |  pc  | 26
+------+------+-----------------------------------------+
|global|  pc  | 40
+------+------+-----------------------------------------+
```

Pregunta 3.b (Asociación dinámica de métodos)

X = 3, Y = 1, Z = 3

```
1     class Bebida {
2        int a = 3, b = 1
3
4        fun s(int x) : int {
5            a = b - x
6            return t(a)
7        }
8
9        fun t(int y) : int {
10           return a * y + b
11       }
12    }
13
14    class Cafe extends Bebida {
15       Bebida caliente = new Marron()
16
17       fun q(int y) : int {
18           return caliente.s(a + b) - y
19       }
20    }
21
22    class Marron extends Cafe {
23       int c = 3
24
25       fun s(int x) : int {
26           a = x - 4
27           c = a + b * x
28           return t(a*b + c)
29       }
30
31       fun t(int y) : int {
32           return c - y
33       }
34    }
35
36    Bebida e = new Cafe()
37    Bebida pres = new Marron()
38    Cafe o = new Marron()
39
40    print(e.s(1) + pres.s(1) + o.s(1))
```

1 **3**

Nombre (Objeto)	Métodos	Valores Variables
e (Cafe())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 0, b = 1.
pres (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = -3, b = 1, c = -2
o (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = 1-4 = -3, b = 1, c = 3.
e.caliente (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
pres.caliente (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
o.caliente (Marron())	s -> Marron :: s, t -> Maroon :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.

IMPRIME

```
+------+-----------------------------------------+
|      |return| -
|      +-----------------------------------------+
| o.s  |  x   | 1
|      +-----------------------------------------+
|      |  pc  | 26 27
+------+------+-----------------------------------+
|global|  pc  | 40
+------+------+-----------------------------------+
```

Pregunta 3.b (Asociación dinámica de métodos)

X = 3, Y = 1, Z = 3

```
1     class Bebida {
2        int a = 3, b = 1
3
4        fun s(int x) : int {
5            a = b - x
6            return t(a)
7        }
8
9        fun t(int y) : int {
10           return a * y + b
11       }
12    }
13
14    class Cafe extends Bebida {
15       Bebida caliente = new Marron()
16
17       fun q(int y) : int {
18           return caliente.s(a + b) - y
19       }
20    }
21
22    class Marron extends Cafe {
23       int c = 3
24
25       fun s(int x) : int {
26           a = x - 4
27           c = a + b * x
28           return t(a*b + c)
29       }
30
31       fun t(int y) : int {
32           return c - y
33       }
34    }
35
36    Bebida e = new Cafe()
37    Bebida pres = new Marron()
38    Cafe o = new Marron()
39
40    print(e.s(1) + pres.s(1) + o.s(1))
```

1 **3**

Nombre (Objeto)	Métodos	Valores Variables
e (Cafe())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 0, b = 1.
pres (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = -3, b = 1, c = -2
o (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = -3, b = 1, c=-3+1*1=-2
e.caliente (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
pres.caliente (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
o.caliente (Marron())	s -> Marron :: s, t -> Maroon :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.

IMPRIME

```
+------+-----------------------------------------------+
|      |return| t(-3*1+-2)
|      +-----------------------------------------------+
| o.s  |  x   | 1
|      +-----------------------------------------------+
|      |  pc  | 26 27 28
+------+------+---------------------------------------+
|global|  pc  | 40
+------+------+---------------------------------------+
```

Pregunta 3.b (Asociación dinámica de métodos)

X = 3, Y = 1, Z = 3

```
1     class Bebida {
2        int a = 3, b = 1
3
4        fun s(int x) : int {
5            a = b - x
6            return t(a)
7        }
8
9        fun t(int y) : int {
10           return a * y + b
11       }
12    }
13
14    class Cafe extends Bebida {
15       Bebida caliente = new Marron()
16
17       fun q(int y) : int {
18           return caliente.s(a + b) - y
19       }
20    }
21
22    class Marron extends Cafe {
23       int c = 3
24
25       fun s(int x) : int {
26           a = x - 4
27           c = a + b * x
28           return t(a*b + c)
29       }
30
31       fun t(int y) : int {
32           return c - y
33       }
34    }
35
36    Bebida e = new Cafe()
37    Bebida pres = new Marron()
38    Cafe o = new Marron()
39
40    print(e.s(1) + pres.s(1) + o.s(1))
```

1 **3**

Nombre (Objeto)	Métodos	Valores Variables
e (Cafe())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 0, b = 1.
pres (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = -3, b = 1, c = -2
o (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = -3, b = 1, c = -2
e.caliente (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
pres.caliente (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
o.caliente (Marron())	s -> Marron :: s, t -> Maroon :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.

IMPRIME

```
+------+-----------------------------------------------+
|      |return| -2-(-5)
|      +-----------------------------------------------+
| o.t  |  y   | -5
|      +-----------------------------------------------+
|      |  pc  | 32
+------+-----------------------------------------------+
|      |return| t(-5)
|      +-----------------------------------------------+
| o.s  |  x   | 1
|      +-----------------------------------------------+
|      |  pc  | 26 27 28
+------+------+----------------------------------------+
|global|  pc  | 40
+------+------+----------------------------------------+
```

Pregunta 3.b (Asociación dinámica de métodos)

X = 3, Y = 1, Z = 3

```
1     class Bebida {
2        int a = 3, b = 1
3
4        fun s(int x) : int {
5            a = b - x
6            return t(a)
7        }
8
9        fun t(int y) : int {
10           return a * y + b
11       }
12    }
13
14    class Cafe extends Bebida {
15       Bebida caliente = new Marron()
16
17       fun q(int y) : int {
18           return caliente.s(a + b) - y
19       }
20    }
21
22    class Marron extends Cafe {
23       int c = 3
24
25       fun s(int x) : int {
26           a = x - 4
27           c = a + b * x
28           return t(a*b + c)
29       }
30
31       fun t(int y) : int {
32           return c - y
33       }
34    }
35
36    Bebida e = new Cafe()
37    Bebida pres = new Marron()
38    Cafe o = new Marron()
39
40    print(e.s(1) + pres.s(1) + o.s(1))
```

1 **3**

Nombre (Objeto)	Métodos	Valores Variables
e (Cafe())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 0, b = 1.
pres (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = -3, b = 1, c = -2
o (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = -3, b = 1, c = -2
e.caliente (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
pres.caliente (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
o.caliente (Marron())	s -> Marron :: s, t -> Maroon :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.

IMPRIME

```
+------+-----------------------------------------------+
|      |return| 3
|      +-----------------------------------------------+
| o.s  |  x   | 1
|      +-----------------------------------------------+
|      |  pc  | 26 27 28
+------+------+---------------------------------------+
|global|  pc  | 40
+------+------+---------------------------------------+
```

Pregunta 3.b (Asociación dinámica de métodos)

X = 3, Y = 1, Z = 3

```
1     class Bebida {
2        int a = 3, b = 1
3
4        fun s(int x) : int {
5            a = b - x
6            return t(a)
7        }
8
9        fun t(int y) : int {
10           return a * y + b
11       }
12    }
13
14    class Cafe extends Bebida {
15       Bebida caliente = new Marron()
16
17       fun q(int y) : int {
18           return caliente.s(a + b) - y
19       }
20    }
21
22    class Marron extends Cafe {
23       int c = 3
24
25       fun s(int x) : int {
26           a = x - 4
27           c = a + b * x
28           return t(a*b + c)
29       }
30
31       fun t(int y) : int {
32           return c - y
33       }
34    }
35
36    Bebida e = new Cafe()
37    Bebida pres = new Marron()
38    Cafe o = new Marron()
39
40    print(e.s(1) + pres.s(1) + o.s(1))
```

1 3 3

Nombre (Objeto)	Métodos	Valores Variables
e (Cafe())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 0, b = 1.
pres (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = -3, b = 1, c = -2
o (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = -3, b = 1, c = -2
e.caliente (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
pres.caliente (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
o.caliente (Marron())	s -> Marron :: s, t -> Maroon :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.

IMPRIME

```
+------+------+----------------------------------------+
|global|  pc  | 40
+------+------+----------------------------------------+
```

Pregunta 3.b (Asociación dinámica de métodos)

X = 3, Y = 1, Z = 3

```
1     class Bebida {
2        int a = 3, b = 1
3
4        fun s(int x) : int {
5            a = b - x
6            return t(a)
7        }
8
9        fun t(int y) : int {
10           return a * y + b
11       }
12    }
13
14    class Cafe extends Bebida {
15       Bebida caliente = new Marron()
16
17       fun q(int y) : int {
18           return caliente.s(a + b) - y
19       }
20    }
21
22    class Marron extends Cafe {
23       int c = 3
24
25       fun s(int x) : int {
26           a = x - 4
27           c = a + b * x
28           return t(a*b + c)
29       }
30
31       fun t(int y) : int {
32           return c - y
33       }
34    }
35
36    Bebida e = new Cafe()
37    Bebida pres = new Marron()
38    Cafe o = new Marron()
39
40    print(e.s(1) + pres.s(1) + o.s(1))
```

Nombre (Objeto)	Métodos	Valores Variables
e (Cafe())	s -> Bebida :: s, t -> Bebida :: t, q-> Cafe :: q.	a = 0, b = 1.
pres (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = -3, b = 1, c = -2
o (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = -3, b = 1, c = -2
e.caliente (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
pres.caliente (Marron())	s -> Marron :: s, t -> Marron :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.
o.caliente (Marron())	s -> Marron :: s, t -> Maroon :: t, q-> Cafe :: q.	a = 3, b = 1, c = 3.

IMPRIME

7