**Punto 1**

Para este punto se utiliza el método getIdentificación, debido a que el otro método no estaba implementado en el código.

Graphical user interface, text

Description automatically generated

Text

Description automatically generated

El método que inició toda la ejecución fue el método pedirDatosPropietario de la clase Directorio.

**Punto 2**

Text

Description automatically generated

Se contaba inicialmente con este propietario

Graphical user interface, text

Description automatically generated

Se modificó dentro del debugger el número de identificación donde paso de ser 800 a 300.

Lo que en el caso propuesto por la profe sería cambiar el nombre a “PEDRO PEREZ”

**Punto 3**

**Graphical user interface, application

Description automatically generated**

Condición establecida

**Text

Description automatically generated**

Breakout condicional en el método eliminarPropietario(int identificación)

**Text

Description automatically generated**

Desde el directorio, se ingresa desde el terminal el valor de Identificación; para este caso, va a 100

**A screenshot of a computer

Description automatically generated with medium confidence**

Como resultado, se entra al breakout condicional

**Punto 4**

Se trabaja con el mismo método del punto anterior. En este caso, se define una expresión como identificacion ==100 y como el propietario con esta identificación había sido previamente agregado, la función da como resultado true para eliminarlo.

**Text

Description automatically generated**

En el otro caso, se define una expresión como identificacion ==500 y como el propietario con esta identificación no existe en el sistema, no es posible eliminarlo y la función da como resultado false.

**Graphical user interface, text, application

Description automatically generated**

**Punto 5**

Para este punto se define un watch que tenga en la “mira” a la variable identificación, para ver cómo va ir cambiando a lo largo que se corre el programa.

**Graphical user interface, text

Description automatically generated**

**Text

Description automatically generated**

**Text

Description automatically generated**

Se puede ver como fue cambiando la variable identificación durante el flujo del código. Para, finalmente, tener un valor de 500 debido a la modificación que se realizó, desde el terminal, a la identificación del propietario.

**Punto 6**

A lo largo del trabajo con del debugger, pude notar la diferencia que tiene el step over(F8) y el step into(F7). El primero ayuda a recorrer paso a paso las líneas de código, pero sin la necesidad de entrar tanto a detalle de lo que hay detrás de cierta función o implementación. Por otro lado, el step into brinda muchos detalles de lo que conforma a cierta función o variable, de tal manera que había momentos en los que incluso se baja tanto en el nivel de abstracción, que se llegaba a ver cómo cierta parte del código está implementada en lenguaje ensamblador.

Por ejemplo, cuando se pone un breakout desde el archivo MainVeterinaria y se comienzan a utilizar step into, el debugger te lleva a recorrer las líneas de código donde cierto método fue previamente definido o incluso la clase a donde pertenece un método en particular.

Otro ejemplo, es que, cd en el código de clínica veterinaria, se planteó un método que pide los datos del propietario y con ellos crea una instancia de tipo propietario a partir del constructor con parámetros; por esto, cuando se utiliza el step into, el debugger lo lleva a uno hasta el archivo de la clase donde está definido ese constructor en particular. No obstante, lo mismo no sucedía con el step over, donde simplemente no se mostraba ese nivel de detalle.