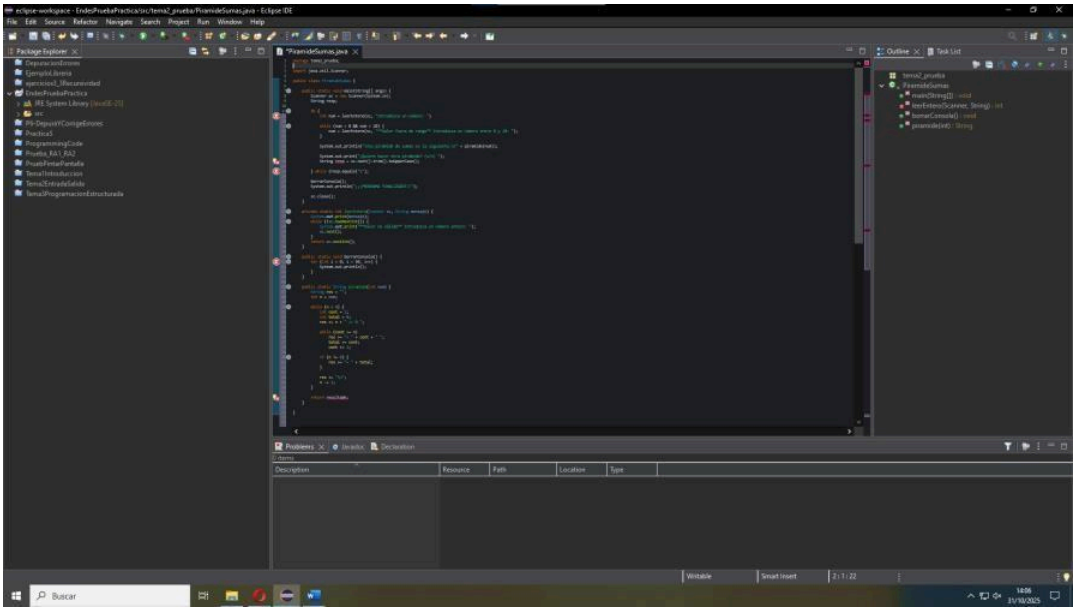
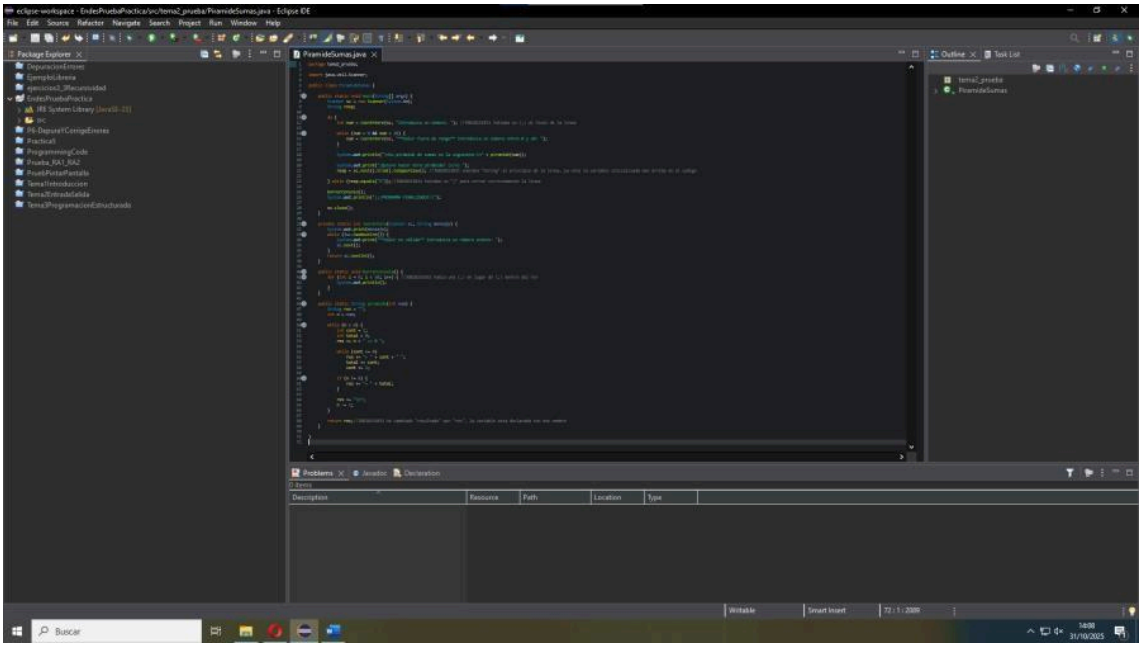
Jorge González Burgos – ENDES Prueba Práctica 1 – Depuración



(Captura del código son corregir)

2.2 Corregir errores sintácticos (y documentarlos)



(Captura del código corregido (sintaxis))

- **¿Qué es un error sintáctico?**

  Un error sintáctico es cuando hay algo mal escrito en el código, puede ser que falte un paréntesis, un punto y coma, o que una variable esté inicializada con un nombre y luego se escriba mal.

- **¿Cuándo los visualizamos?**

  En Eclipse los vemos subrayados con una línea roja. Si hacemos clic en la cruz roja donde pone el número de línea nos dará más información sobre el error que ha encontrado en dicha línea.

- **¿Podemos depurar con errores de sintaxis?**

No, cuando le damos a depurar sí que cambia la perspectiva y entra en modo depuración, pero da un error de compilación.

2.3 Corregir errores lógicos usando el Depurador

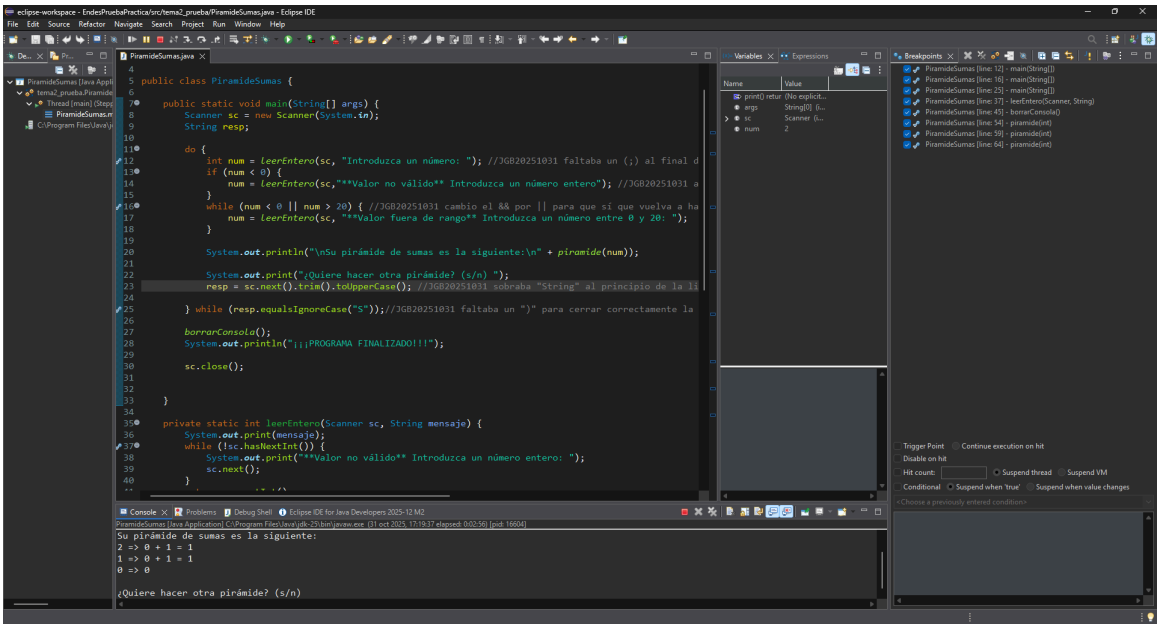
He conseguido hacer que el programa se ejecute y que vuelva a preguntar por un número cada vez que no sea válido, aunque no he
conseguido que la suma se muestre correctamente por consola. Veo que hay algo mal con cómo se hacen las sumas dentro del *while*, pero no sé distinguir el qué.

Cuando el código pregunta si quieres introducir otro número, funciona correctamente, aunque no he sabido cómo hacer que cualquier cosa que sea diferente a s o S sea considerado como una respuesta negativa, y se termine de ejecutar.

- **Proporciona una mini guía de cómo realizar la depuración de un programa (si te es más fácil, elige uno de los fallos lógicos y úsalo de ejemplo)**

Para realizar la depuración en Eclipse basta con pulsar el botón *Debug* de java, aunque también disponemos de *breakpoints*, que sirven para detener la ejecución en puntos clave del código y dejarnos ver de forma más clara los posibles errores que haya.

- **Agrega capturas con los puntos de ruptura y las vistas que consideres más importantes a la hora de depurar un programa y por qué.**



En la captura enseño alguno de los breakpoints que he puesto para hacer más fácil la depuración. En la parte derecha tengo desplegada la vista de "variables", donde puedo ir comprobando los valores que estas toman a medida que se ejecuta el código. Más a la derecha aún tengo desplegada la vista de los *breakpoints*, donde puedo consultar rápidamente dónde están colocados, el número de línea y en la función en la que se encuentran.

- **¿Dónde colocaste los breakpoints y qué valores viste que confirmaron el fallo?**

He colocado los puntos de ruptura antes de estructuras condicionales, para comprobar si era correcta la condición para entrar a ellos. También los he colocado al empezar o dentro de un *do while* o un *while* para comprobar que los valores de las variables correspondientes fuesen correctos con cada iteración.

- **¿En qué situación debemos usar Step Into (F5), Step Over (F6), Step Return (F7) y Resume (F8)?**

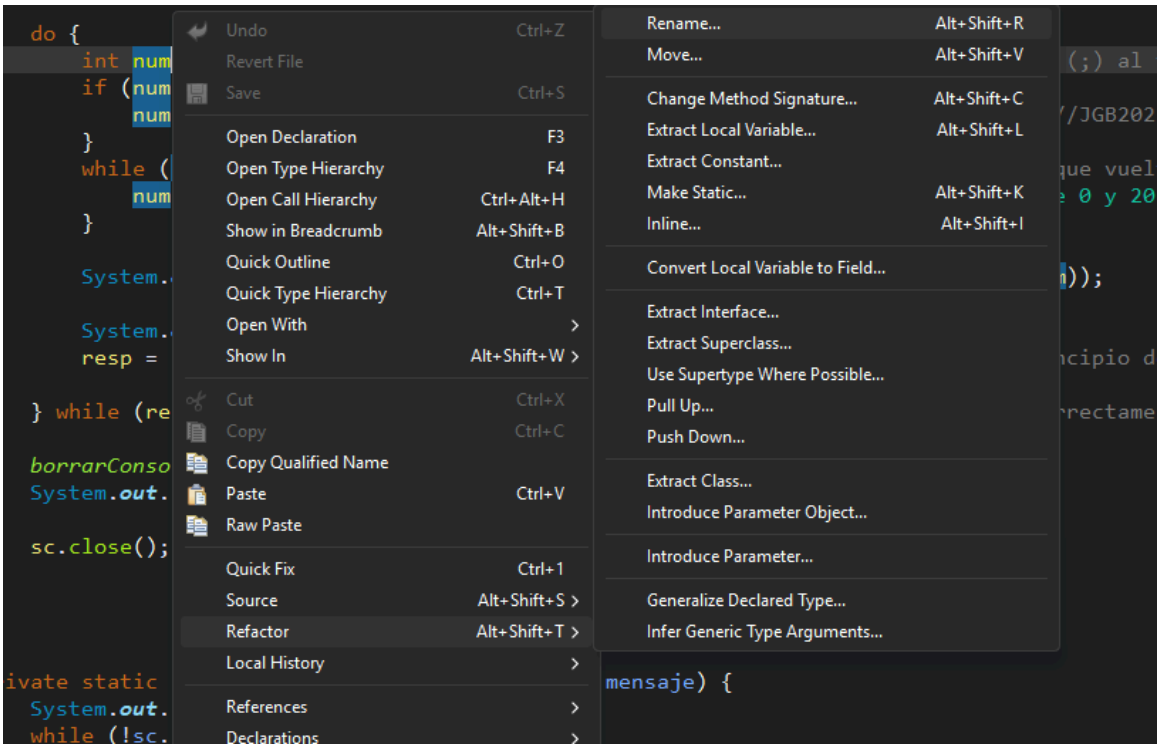
Step into se usa cuando quieres entrar dentro de la función a la que se está llamando. En el ejemplo del ejercicio, utilizo el *step into* para entrar a las funciones *leerEntero*, *borrarConsola* y *piramide*.

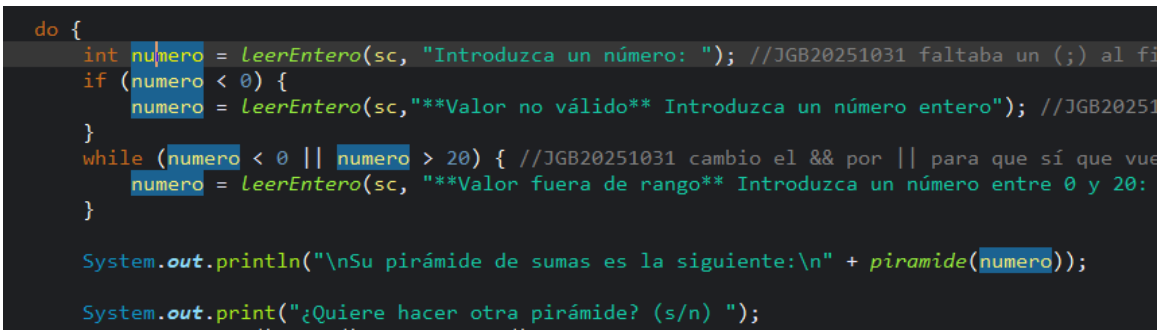
Step Over lo uso cuando no necesito comprobar la ejecución de la función, porque ya sé que no hay ningún error dentro de ella.

Step Return lo utilizo cuando estoy dentro de una función y ya he comprobado lo que quería. Entonces utilizo el *step return* para volver a la línea en la que se llamó la función, y se ejecuta.

Resume se utiliza cuando estás en un punto del código que ya has comprobado y quieres saltar al siguiente *breakpoint* que hayas puesto.

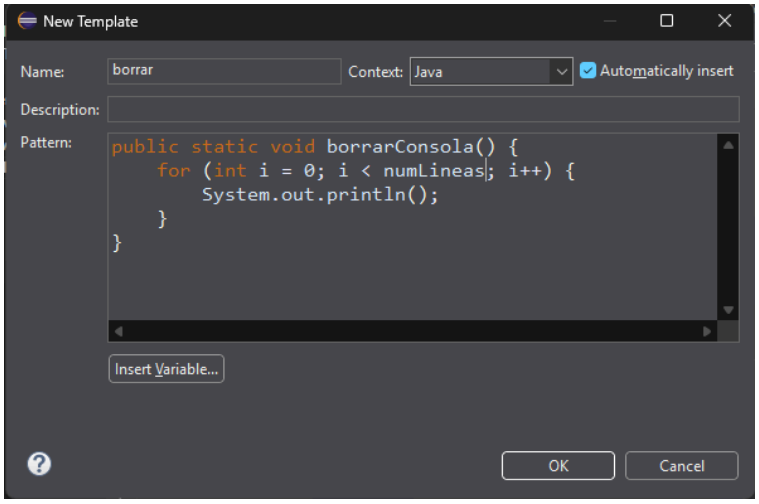
2.4 Refactorización: renombrar num → numero

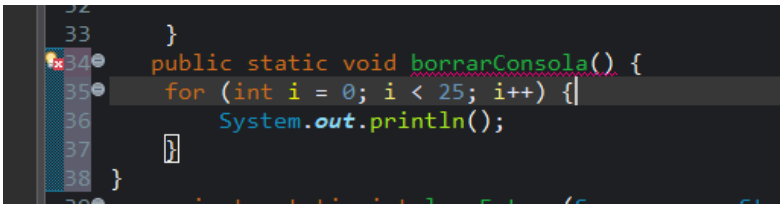




Renombrar todas las coincidencias en el código en Eclipse es tan fácil como hacer clic derecho sobre el nombre que queramos cambiar y pulsar *Refactor* > *Rename*. Después, simplemente escribo “numero” y *Enter* para confirmar.

2.5 Crear una plantilla (template) “borrar”





Desde *Window* > *Preferences* > *Java* > *Editor* > *Templates*, podemos crear plantillas de código que luego podremos usar de forma rápida y cómoda. Hacemos clic en *new* y escribimos el código que vayamos a usar como plantilla. Una vez la tengamos, le damos a *apply and close* y ya podremos usarla. Será tan fácil como escribir el nombre que le hayamos puesto a la plantilla y pulsar *Ctrl + Espacio*. Pulsamos *Enter* para confirmar y se completará el código de forma automática.

Las plantillas son útiles para reciclar código de forma cómoda y rápida. Si tienes un código con una función que se pueda utilizar de forma general en muchos contextos, puedes hacer una plantilla con la función para tenerla a mano de forma más eficiente que ir copiándola y pegándola de un proyecto a otro.