2021

Depuración

ENTORNOS DE DESARROLLO

IVAN VIDAL

Índice

Crear los breakpoints en el programa

Tipo línea:

Se creo uno en la línea 17 de la clase que tiene el Main.

```
1  package testing;
2
3  public class PruebaPresonasAlumnosProfesores
4  {
5      public static void main(String[] args)
6      {
7              Profesor[] arrayProfesores = new Profesor[5];
8              Alumno[] arrayAlumnos = new Alumno[5];
9
10             for (int i = 0; i < arrayProfesores.length; i++)
11             {
12                 arrayProfesores[i] = new Profesor();
13             }
14
15             for (int i = 0; i < arrayProfesores.length; i++)
16             {
17                 System.out.println(arrayProfesores[i].toString());
18             }
19
20             for (int i = 0; i < arrayAlumnos.length; i++)
21             {
22                     arrayAlumnos[i] = (i <= 2) ? new AlumnoPresencial() : new AlumnoLibre() ;
23             }
24
25
26             for (int i = 0; i < arrayAlumnos.length; i++)
27             {
28                 System.out.println(arrayAlumnos[i].toString());
29             }
30     }
31
32 }
33
```

Tipo clase:

Se creó uno en la clase abstracta alumno:

```
1  package testing;
2
3
4
5  public abstract class Alumno extends Persona
6  {
7      private int curso;
8      private String nivelAcademico;
9
10     public void cambiarcurso(int curso)
11     {
12         this.curso = ((int) Math.random() * curso);
13     }
14
15     public String toString()
16     {
17         return super.toString()+" curso: "+this.curso+" nivel acad@mico: "+this.nivelAcademico;
18     }
19     public abstract double pagoMensual();
20     public abstract String  mostrarAsignaturas();
21 }
22
```

Tipo campo:

Se hizo uno en la clase "AlumnoPresencial":

```java
package testing;


public class AlumnoPresencial extends Alumno
{
    private int matriculaCurso;
    private int noConvocatoria;
    private int plusPorConvocatoria;

    @Override
    public double pagoMensual()
    {
        return (this.matriculaCurso+this.plusPorConvocatoria*this.noConvocatoria/12);
    }
    @Override
    public String mostrarAsignaturas()
    {
        return "Todas las asignaturas del curso "+this.matriculaCurso;
    }

}

```

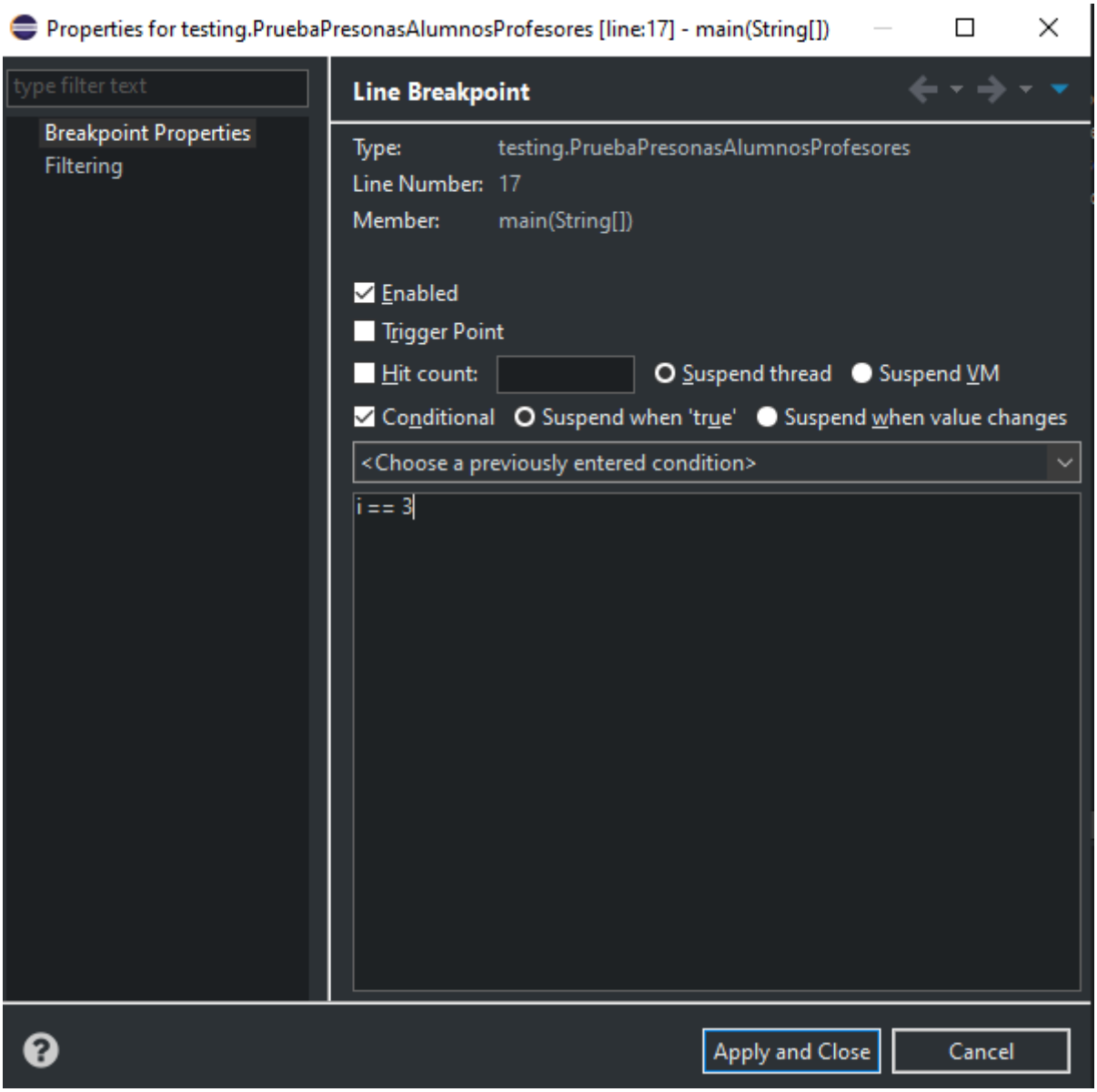
Tipo método:

Se hizo uno en la clase "AlumnoLibre":

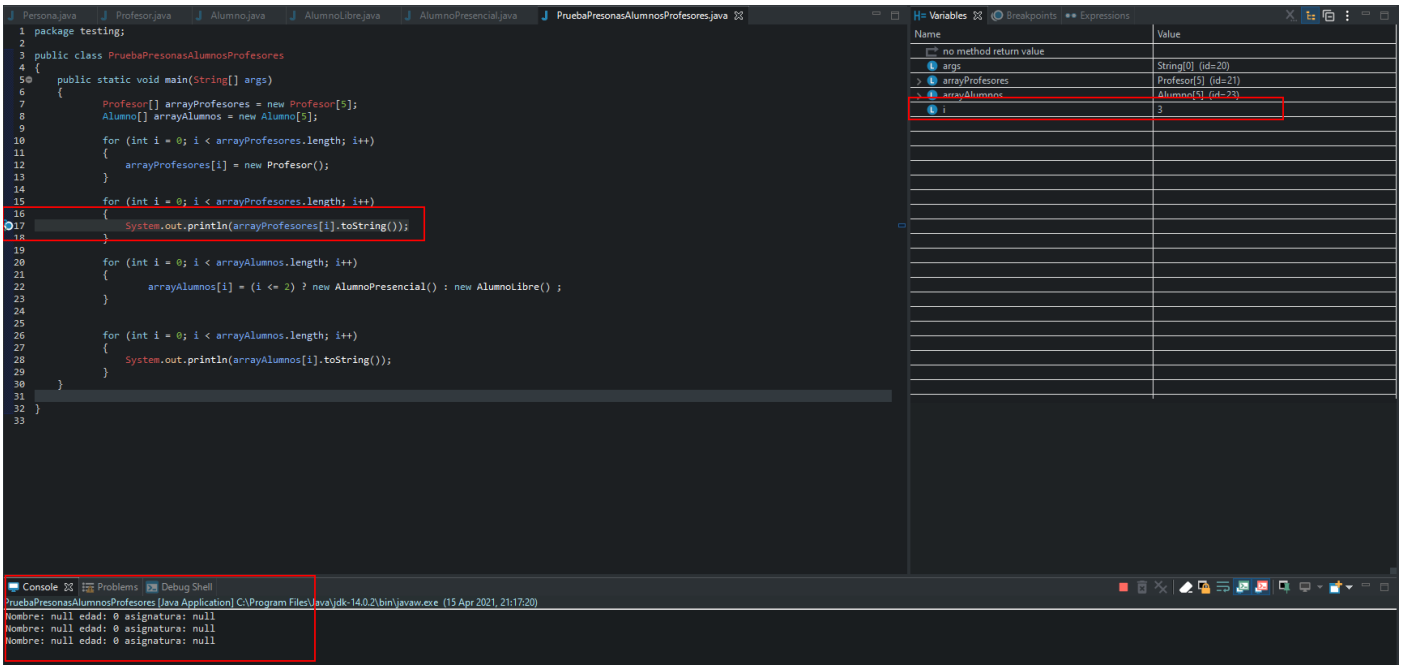
```java
package testing;


public class AlumnoLibre extends Alumno
{
    private String[] listaAsignaturas;
    private float precioPorHora;
    private int noHorasDiarias;

    public AlumnoLibre()
    {
        this.precioPorHora = (float) Math.random() * 10;
    }
    public void pedirAsignaturas( String[] asignaturas)
    {
        listaAsignaturas = new String[asignaturas.length];
        for (int i = 0; i < asignaturas.length; i++)
        {
            listaAsignaturas[i] = new String(asignaturas[i]);
        }
    }
    public double pagoMensual()
    {
        return this.precioPorHora*this.noHorasDiarias+30;
    }
    public String mostrarAsignaturas()
    {
        String asignaturas = "";
        for (int i = 0; i < listaAsignaturas.length; i++)
        {
            asignaturas += listaAsignaturas[i];
        }
        return asignaturas;
    }

}

```

Propiedades de los breakpoints

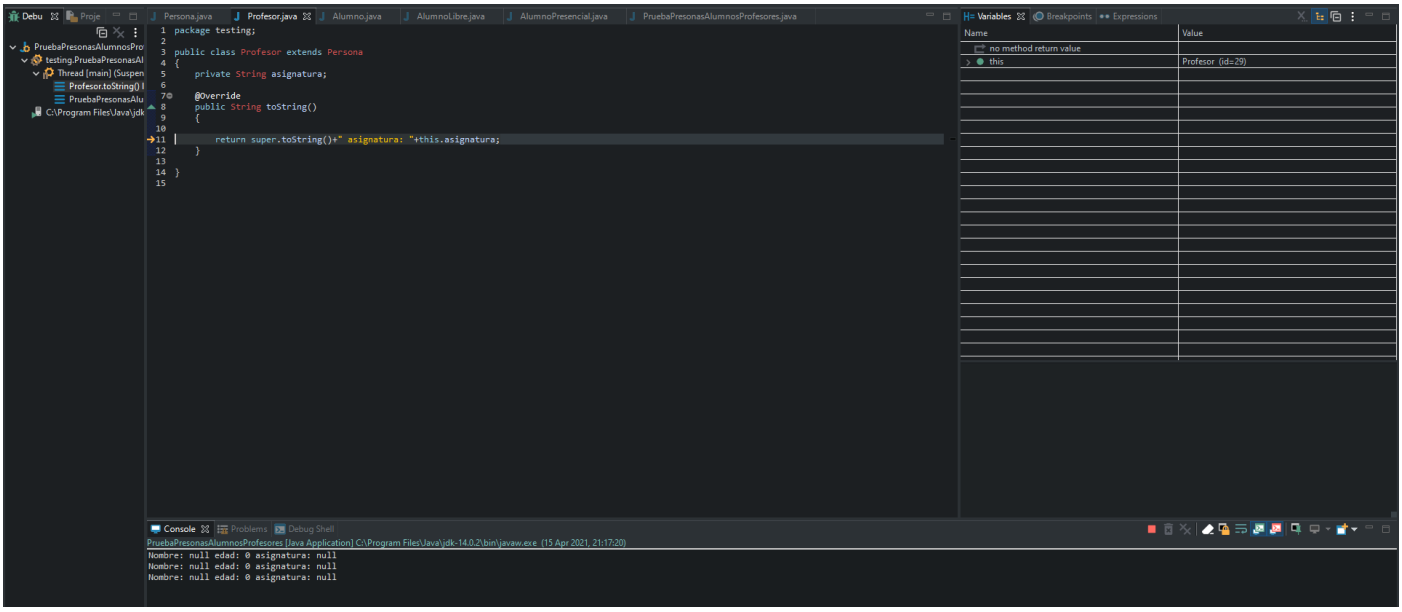
Al breakpoint de tipo línea se le dio una condicional que se detenga solo cuando i == 3:



Vemos que cuando se ejecuta el debugger se muestra que se para al llegar a la tercera línea:

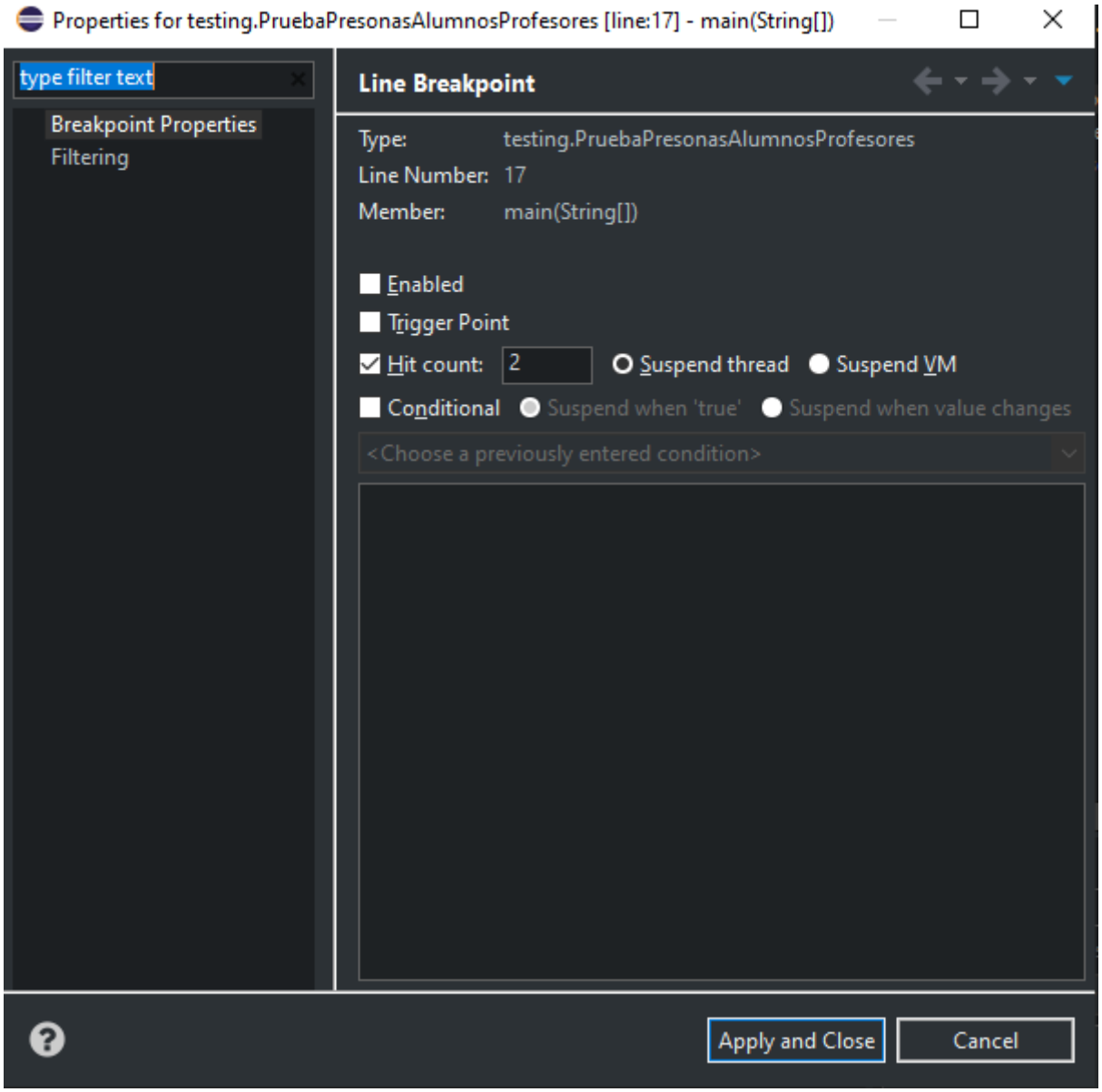


Al dar "F5" para seguir seguiría continuando el programa

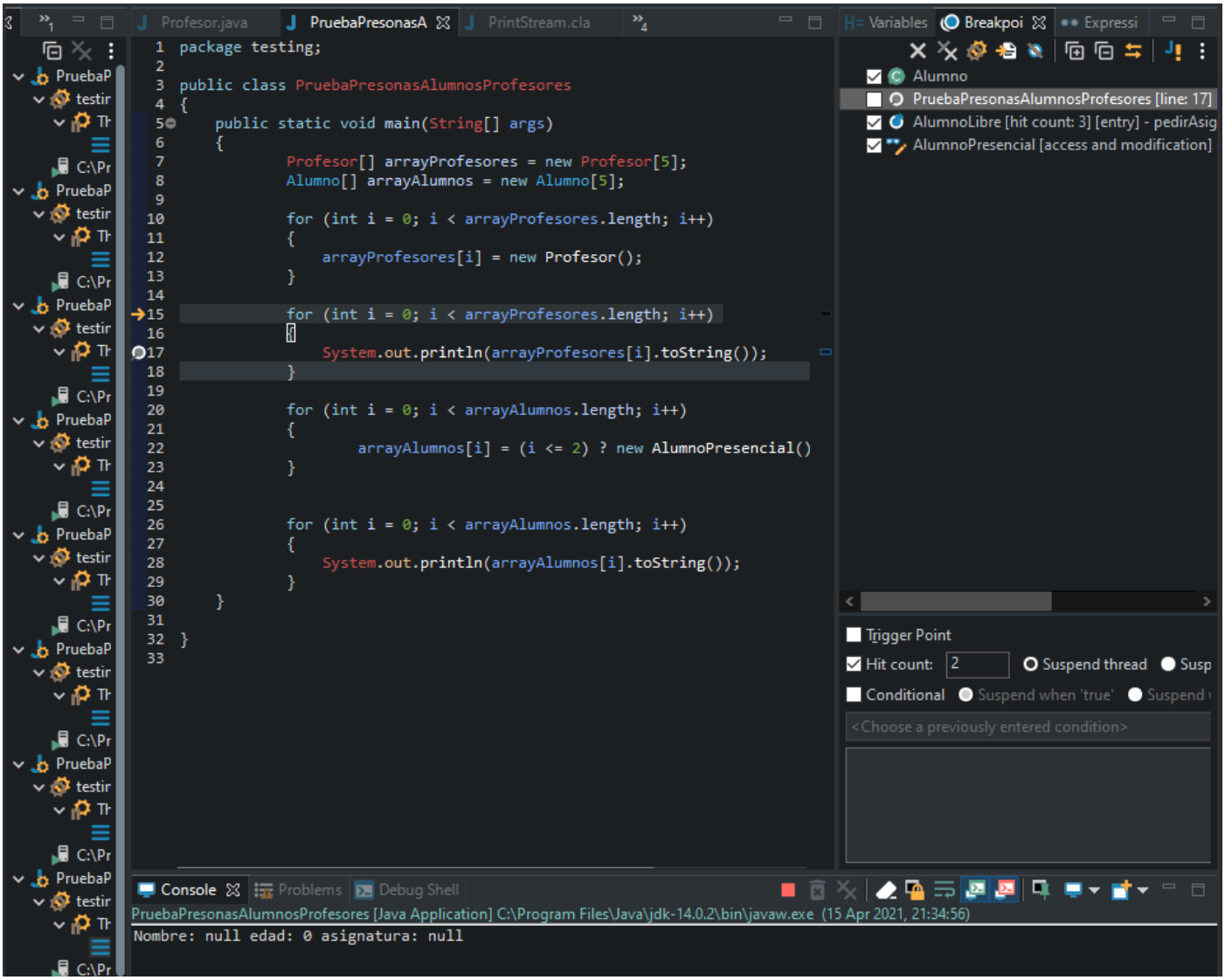


Como se ve, pasaría a la clase Profesor para ver el método "toString()" y al final volvería a sacar la línea.

Ahora para lo de los hit count:



Esto significa que no se parará el breakpoint hasta que se llegué por segunda vez ahí:



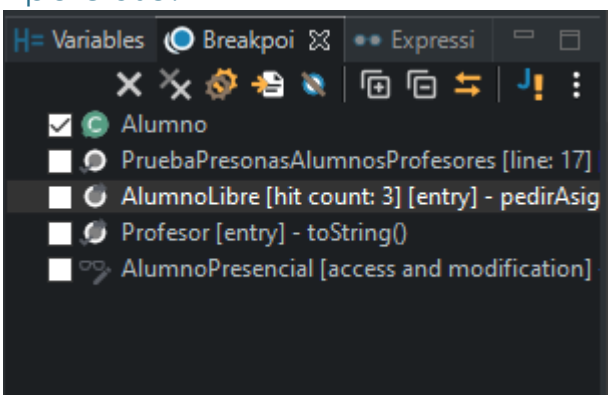

```
1  package testing;
2
3  public class PruebaPresonasAlumnosProfesores
4  {
5      public static void main(String[] args)
6      {
7              Profesor[] arrayProfesores = new Profesor[5];
8              Alumno[] arrayAlumnos = new Alumno[5];
9
10             for (int i = 0; i < arrayProfesores.length; i++)
11             {
12                 arrayProfesores[i] = new Profesor();
13             }
14
15             for (int i = 0; i < arrayProfesores.length; i++)
16             {
17                 System.out.println(arrayProfesores[i].toString());
18             }
19
20             for (int i = 0; i < arrayAlumnos.length; i++)
21             {
22                     arrayAlumnos[i] = (i <= 2) ? new AlumnoPresencial()
23             }
24
25
26             for (int i = 0; i < arrayAlumnos.length; i++)
27             {
28                 System.out.println(arrayAlumnos[i].toString());
29             }
30     }
31
32 }
33
```

Como se ve aquí se detiene justo en la segunda vez que llega al breakpoint.

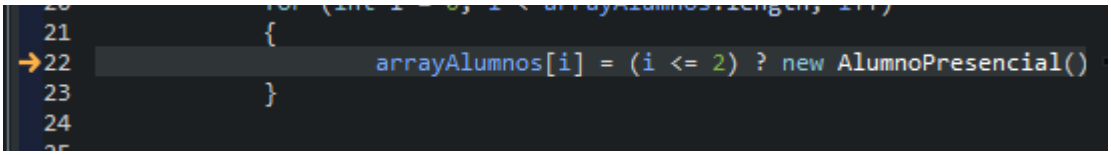
Probando los otros breakpoints

Para demostrar que los otros breakpoint funcionan se desactivan los demás breakpoints hasta probar todos:

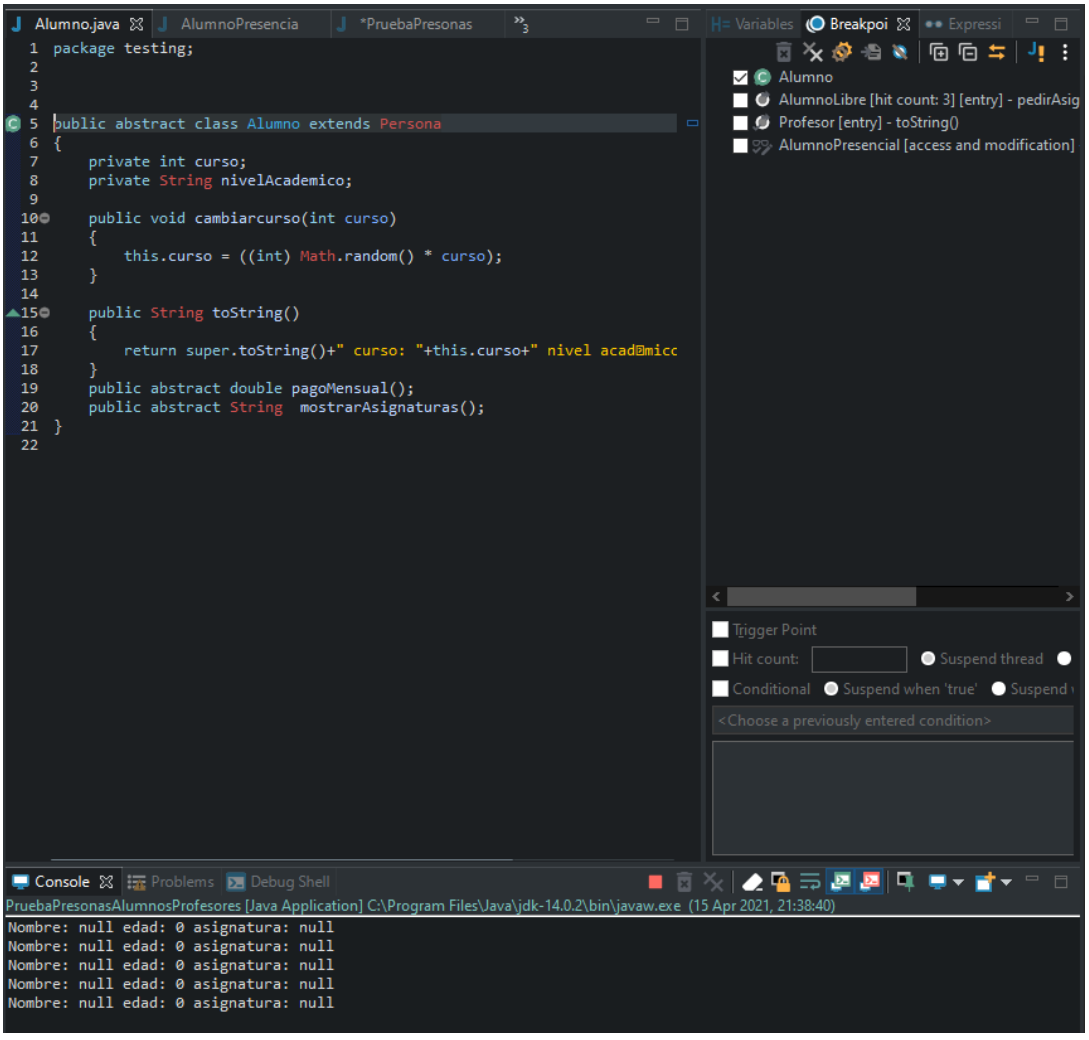
Tipo clase:



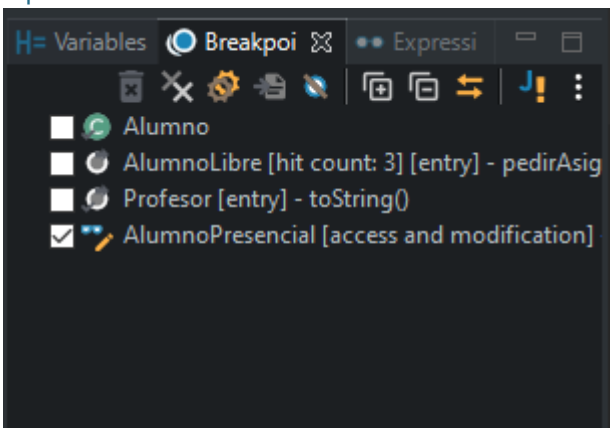
Al ejecutar el debugger se para justo al momento de llegar a la clase Alumno:



Se detiene en la línea 22 de la clase que tiene el Main, esto se debe a que va a crear un objeto de una clase hija de la clase alumno.

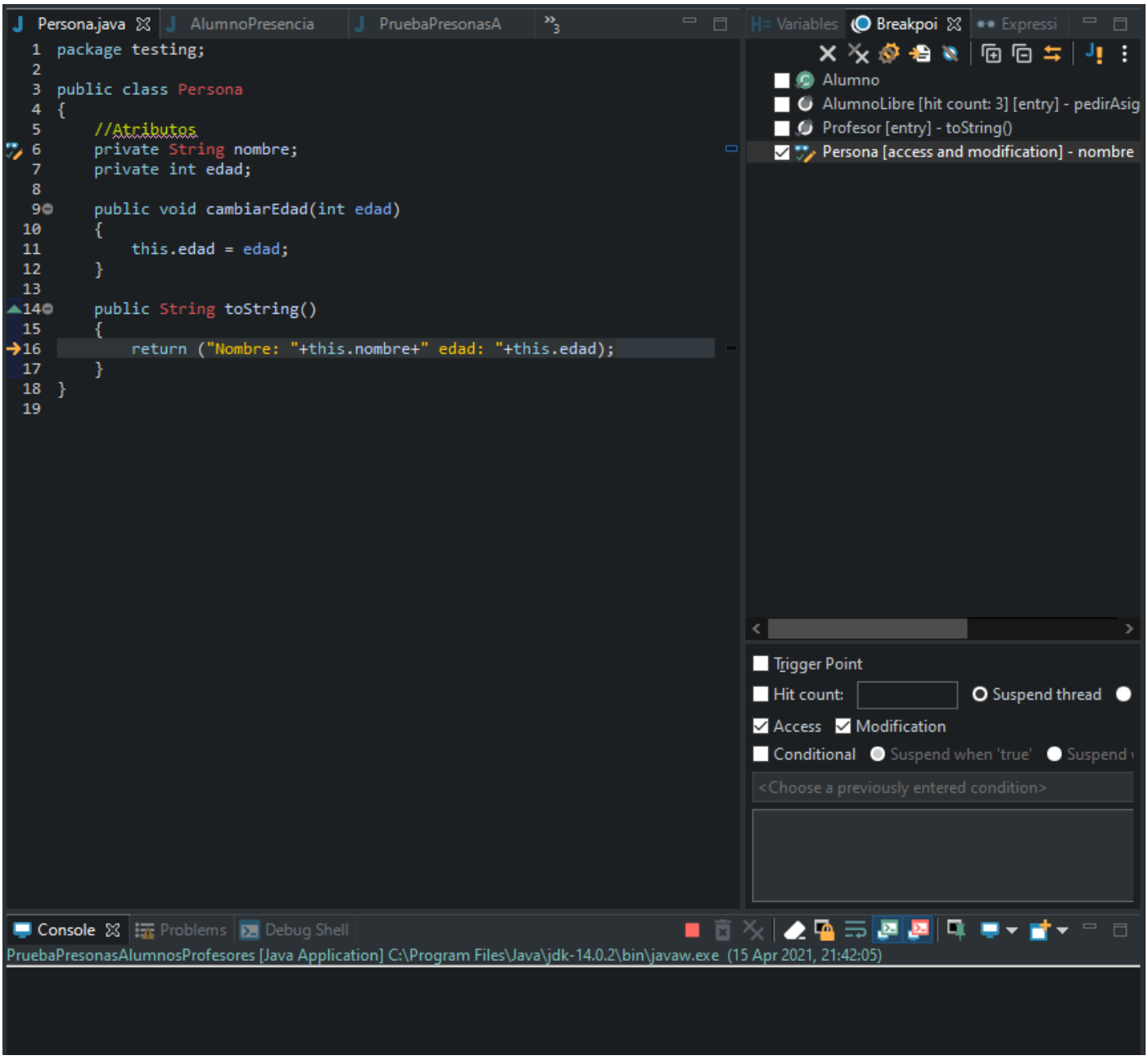


Tipo atributo



Cambiamos al tipo atributo por lo cual al ejecutar el debugger se detiene al momento de entrar al parámetro.

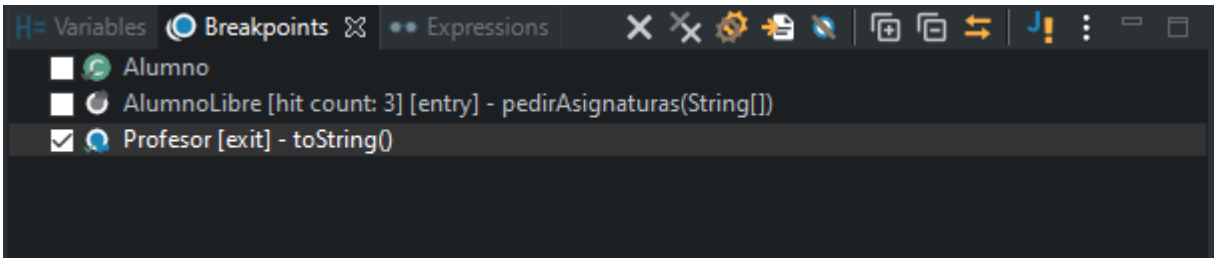
Se cambió el breakpoint a uno de la clase Persona para que se detenga cuando acceda a ese parámetro:



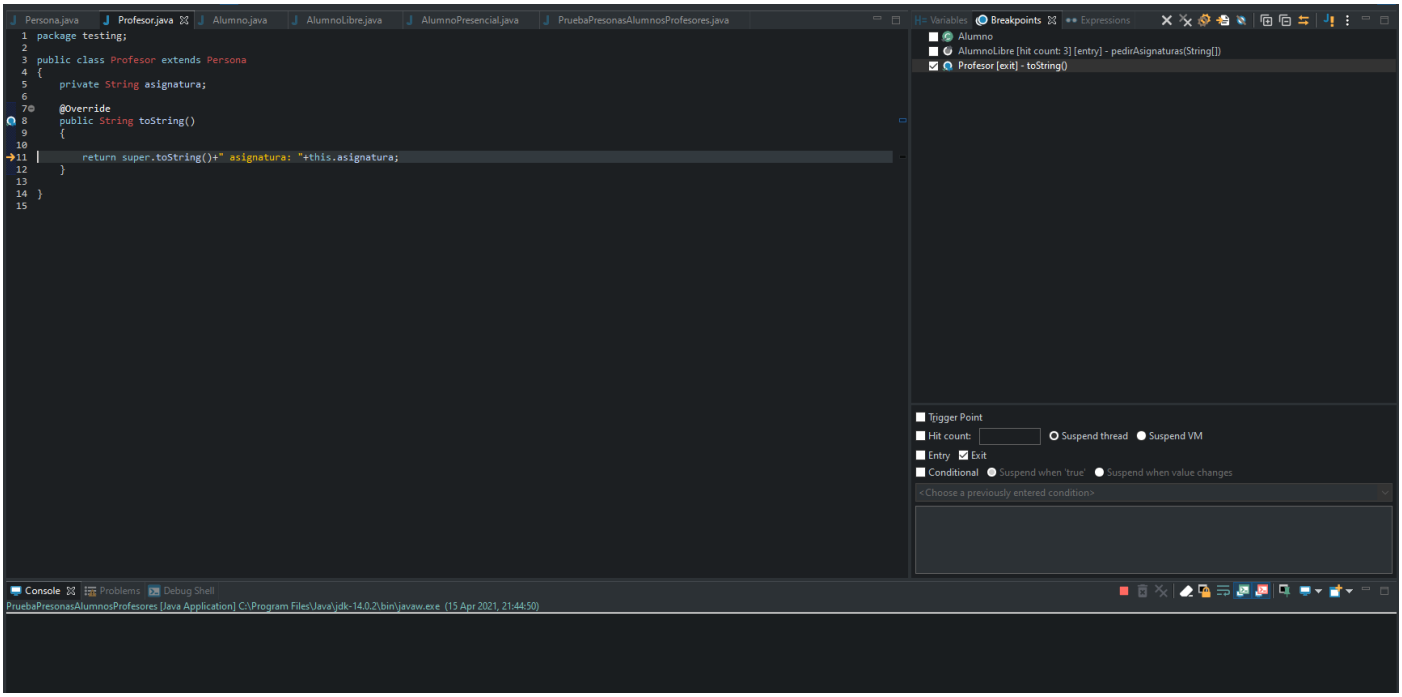
Efectivamente se detuvo al momento de realizar el "toString()" de la clase Persona ya que accede a ese parámetro.

Tipo método

Se cambio el breakpoint al método "toString()" de "Profesor" y se puso de salida:



Y efectivamente el programa se detiene al momento de terminar el método:



Si se da "F5" para seguir pasa directamente a la clase ejecutable: