

# Depuración de Programas en Java con el IDE NetBeans

La mayoría de las herramientas de desarrollo de software modernas, el **IDE NetBeans para Java** incluido, tienen integrado un depurador. Un depurador es un módulo de software que permite ejecutar un programa instrucción por instrucción, ejecutarlo hasta cierta instrucción o en su totalidad. También nos permite observar cómo cambia el valor de las variables del programa conforme se van ejecutando sus instrucciones.

## Ejecución de un Programa Instrucción por Instrucción

Para ejecutar un programa instrucción por instrucción se sigue el siguiente procedimiento:

1. Utilizando la herramienta de desarrollo **NetBeans**, creé un proyecto llamado **telegrama**.
2. Haga que este proyecto sea el proyecto principal y establezca el nombre de la clase principal a **DemoTelegrama** dentro del paquete **pruebas**.
3. Cree una clase llamada **Telegrama** dentro del paquete **objetosNegocio** y edite el código de la clase **Telegrama** como se muestra en el siguiente listado:

### Telegrama.java

```
/*
 * Telegrama.java
 */
package objetosNegocio;

/**
 * Esta clase permite calcular el costo de un telegrama.
 *
 * @author mdomitsu
 */
public class Telegrama {

    private final double COSTO_ORDINARIO = 25.0;
    private final double COSTO_URGENTE = 40.0;
    private final double COSTO_ADICIONAL_ORDINARIO = 2.5;
    private final double COSTO_ADICIONAL_URGENTE = 4.0;
    private String tipoTelegrama;
    private int numPalabras;
    private double costo;

    /**
```

```

* Constructor de la clase. Inicializa los atributos de la clase al
* valor de sus parametros
* @param tipoTelegrama Tipo de telegrama: "Ordinario" o "Urgente"
* @param numPalabras Numero de palabras del telegrama
*/
public Telegrama(String tipoTelegrama, int numPalabras) {
    this.tipoTelegrama = tipoTelegrama;
    this.numPalabras = numPalabras;
}

/**
* Esta funcion regresa el numero de palabras del telegrama.
* @return Numero de palabras del telegrama
*/
public int getNumPalabras() {
    return numPalabras;
}

/**
* Esta funcion establece el numero de palabras del telegrama.
* @param numPalabras Numero de palabras del telegrama
*/
public void setNumPalabras(int numPalabras) {
    this.numPalabras = numPalabras;
}

/**
* Esta funcion regresa el tipo del telegrama
* @return Tipo de telegrama: "Ordinario" o "Urgente"
*/
public String getTipoTelegrama() {
    return tipoTelegrama;
}

/**
* Esta funcion establece el tipo del telegrama
* @param tipoTelegrama Tipo de telegrama: "Ordinario" o "Urgente"
*/
public void setTipoTelegrama(String tipoTelegrama) {
    this.tipoTelegrama = tipoTelegrama;
}

/**
* Esta funcion regresa el costo del telegrama
* @return Costo del telegrama
*/
public double getCosto() {
    return costo;
}

/**
* Esta funcion calcula el costo de un telegrama en funcion de su
* tipo y numero de palabras
*/
public void calculaCosto() {
    // Si el telegrama es ordinario
    if (tipoTelegrama.charAt(0) == 'O' ||

```

```

        tipoTelegrama.charAt(0) == 'o') {
        // Si el telegrama tiene hasta 10 palabras
        if (numPalabras <= 10) {
            costo = COSTO_ORDINARIO;
        }
        // Si el numero de palabras excede a 10 palabras
        else {
            costo = COSTO_ORDINARIO +
                COSTO_ADICIONAL_ORDINARIO * (numPalabras - 10);
        }
    }
    // Si el telegrama es urgente
    else if (tipoTelegrama.charAt(0) == 'U' ||
            tipoTelegrama.charAt(0) == 'u') {
        // Si el telegrama tiene hasta 10 palabras
        if (numPalabras <= 10) {
            costo = COSTO_URGENTE;
        }
        // Si el numero de palabras excede a 10 palabras
        else {
            costo = COSTO_URGENTE +
                COSTO_ADICIONAL_URGENTE * (numPalabras - 10);
        }
    }
    // Si el telegrama no es ordinario ni urgente
    else {
        costo = 0;
    }
}

/**
 * Esta función regresa una cadena con el tipo de telegrama, su
 * numero de palabras y su costo.
 * @return Cadena con el tipo de telegrama, su numero de palabras y
 * su costo.
 */
public String toString() {
    return tipoTelegrama + ", " + numPalabras + ", " + costo;
}
}

```

4. Edite la clase principal **DemoTelegrama** como se muestra en el siguiente listado:

### DemoTelegrama.java

```

/**
 * DemoTelegrama.java
 */
package pruebas;

import objetosNegocio.Telegrama;

/**
 * Esta clase permite probar la clase Telegrama

```

```
*
* @author mdomitsu
*/
public class DemoTelegrama {

    /**
     * Esta funcion prueba los metodos de la clase Telegrama
     * @param args Argumentos en la linea de comando
     */
    public static void main(String[] args) {
        Telegrama telegrama1 = new Telegrama("Ordinario", 8);
        telegrama1.calculaCosto();
        System.out.println(telegrama1);

        telegrama1.setNumPalabras(12);
        telegrama1.calculaCosto();
        System.out.println(telegrama1);

        Telegrama telegrama2 = new Telegrama("Urgente", 8);
        telegrama2.calculaCosto();
        System.out.println(telegrama2);

        telegrama2.setNumPalabras(12);
        telegrama2.calculaCosto();
        System.out.println(telegrama2);
    }
}
```

5. Para iniciar la ejecución del programa instrucción por instrucción seleccione del menú principal la opción **Debug/Step Into** o presione la tecla **F7** como se ve en la Figura 1.
6. NetBeans iniciará la ejecución del programa y detendrá la ejecución en la línea con la primera instrucción del programa, resaltando esa línea de color verde como se muestra en la Figura 2. La instrucción resaltada es la siguiente instrucción a ejecutarse.
7. Para ejecutar la instrucción del programa resaltada seleccione del menú principal la opción **Debug/Step Over** o presione la tecla **F8** como se ve en la Figura 3. Otra forma de realizar esta tarea es hacer clic en el icono **Step Over** de la barra de tareas, Figura 4.

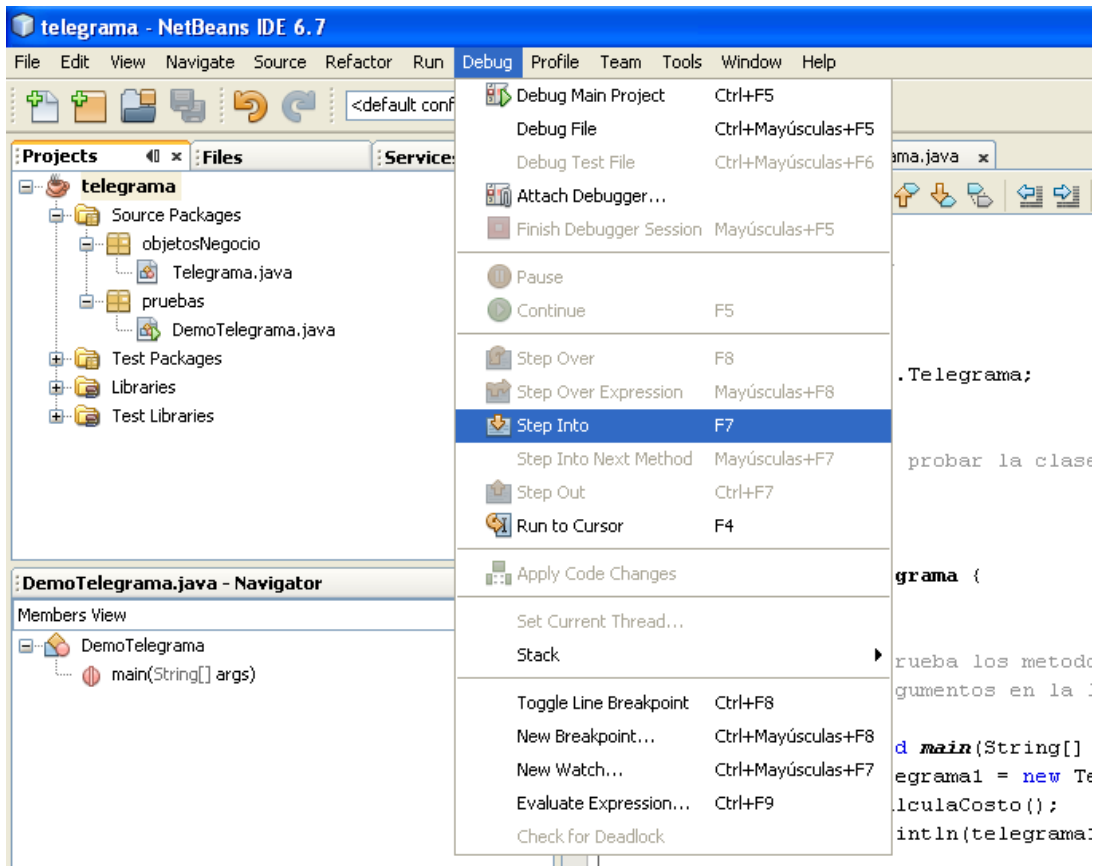


Figura 1

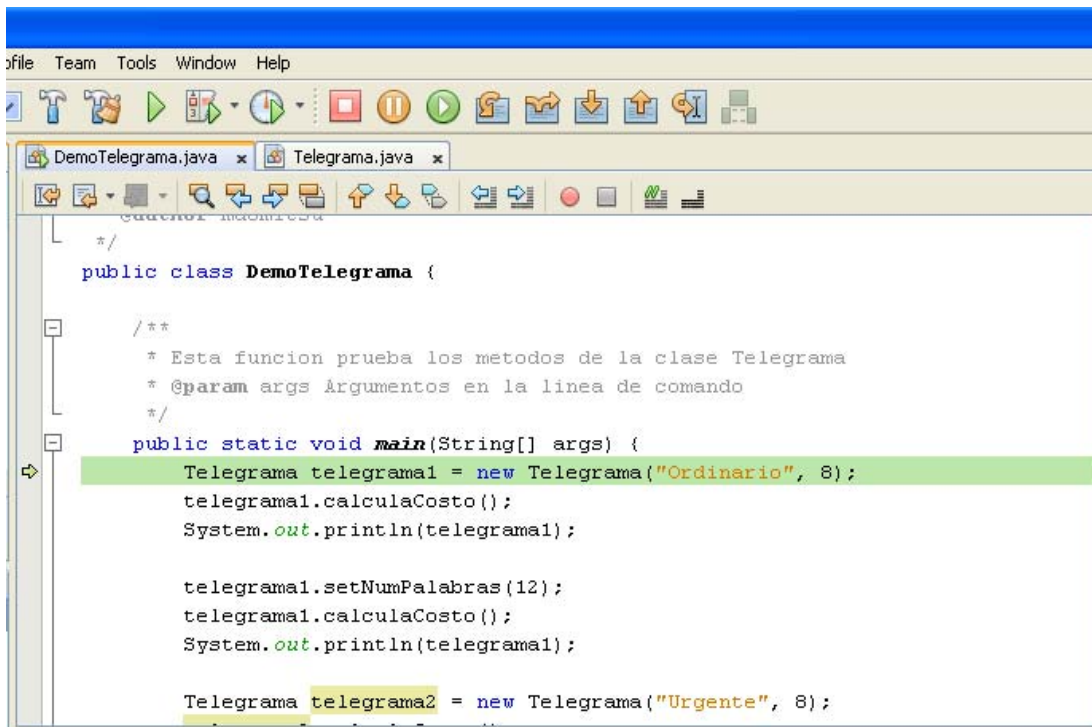


Figura 2

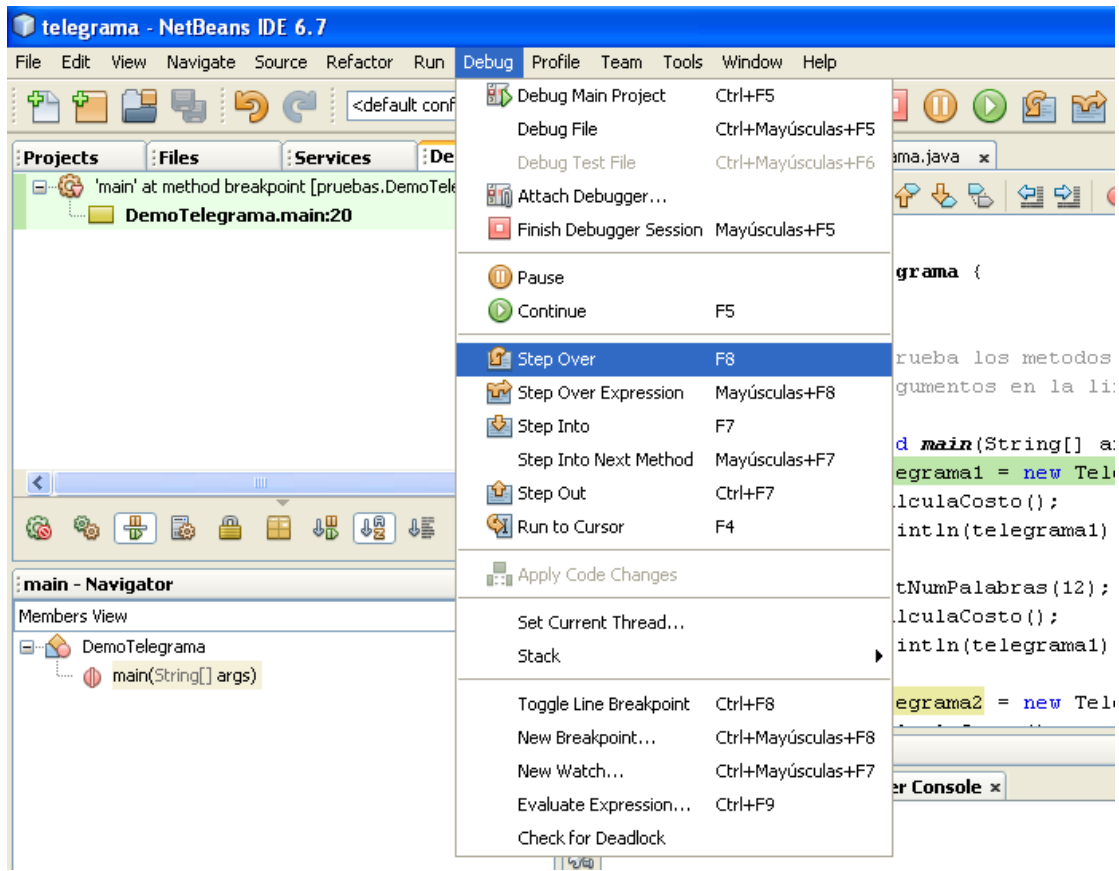


Figura 3



Figura 4

8. Note que aunque la instrucción aparte de crear el objeto invoca al constructor de la clase, NetBeans ejecuta la instrucción como si fuera una instrucción simple y detiene la ejecución del programa en la línea con la siguiente instrucción del programa, resaltando esa línea de color verde, Figura 5.

9. Como la instrucción ejecutada:

```
Telegrama telegrama1 = new Telegrama("Ordinario", 8);
```

crea el objeto `telegrama1`, NetBeans abre un panel llamado **Variables** debajo del panel de edición y ahí nos muestra las variables creadas, hasta este momento, en el método en que se encuentra el programa así como los parámetros del método (el método `main()`, en este caso), Figura 6.

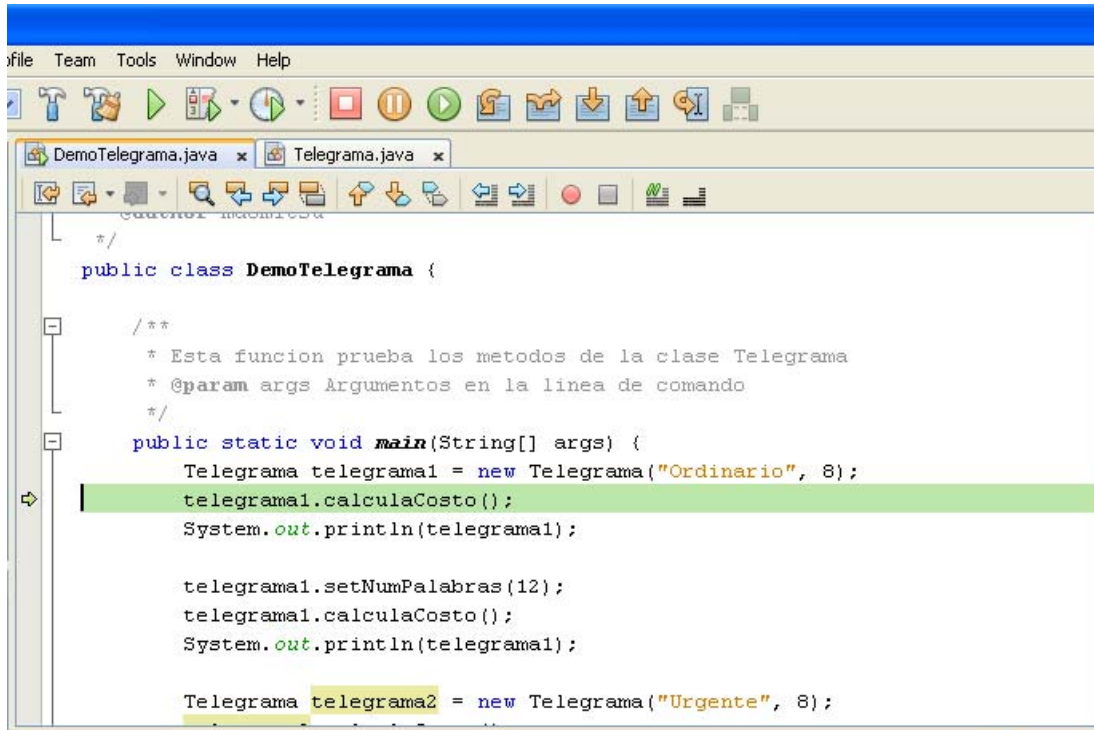


Figura 5

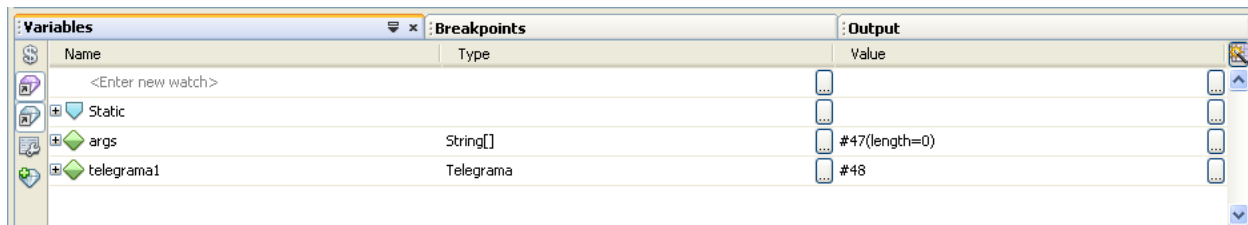


Figura 6

10. Si no aparece el panel **Variables** seleccione del menú principal la opción **Window/Debugging/Variables** o presione las teclas **Alt + Mayúsculas + 1**, como se ve en la Figura 7. Aparecerá el panel **Variables**.
11. Como la variable `telegrama1` es un objeto, podemos inspeccionar los valores de sus atributos haciendo clic en el icono + a la izquierda del nombre de la variable. Al hacerlo el nodo de la variable se expandirá mostrando sus atributos y sus valores, Figura 8.
12. Presiona la tecla F8 de nuevo para ejecutar la siguiente instrucción:

```
telegrama1.calculaCosto();
```

Note que de nuevo, Netbeans ejecuta la invocación del método `calculacosto()` como una sola instrucción. Inspeccione en el panel Variables el valor del atributo `costo` del objeto `telegrama1`.

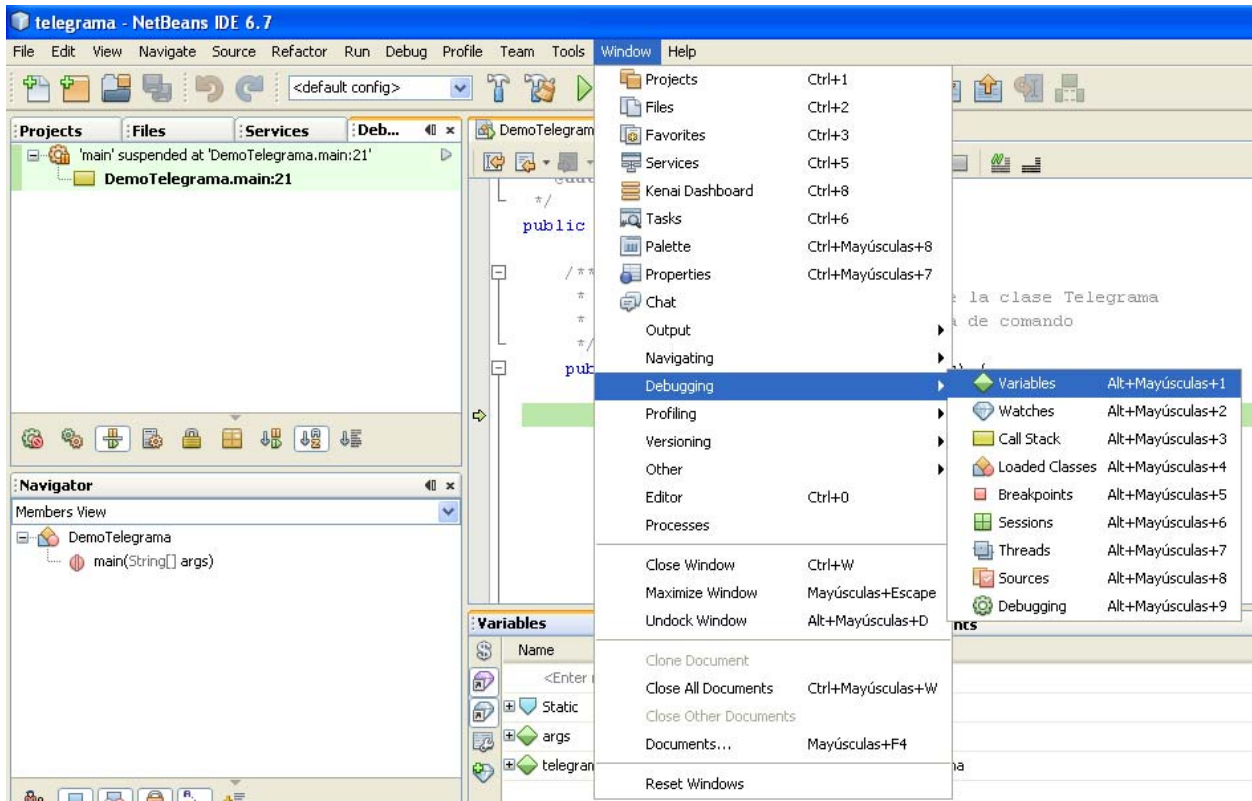


Figura 7

Variables		Breakpoints	Output
Name	Type		Value
Static			
args	String[]		#47(length=0)
telegrama1	Telegrama		#48
COSTO_ORDINARIO	double		25.0
COSTO_URGENTE	double		40.0
COSTO_ADICIONAL_ORDINARIO	double		2.5
COSTO_ADICIONAL_URGENTE	double		4.0
tipoTelegrama	String		"Ordinario"
numPalabras	int		8
costo	double		0.0

Figura 8

## Ejecución de un Programa Hasta una Instrucción Dada

Hay ocasiones en las que nos interesa ejecutar las instrucciones de un programa hasta una instrucción dada sin detenernos en las instrucciones intermedias. Una forma de hacerlo es la siguiente:

1. Coloque el cursor al principio de la línea con la instrucción en la que se desee detener la ejecución. En este caso coloque el cursor en la línea con la instrucción:



```
Telegrama telegrama2 = new Telegrama("Urgente", 8);
```

- Para ejecutar las instrucciones del programa hasta antes de la instrucción en la que está el cursor, seleccione del menú principal la opción **Debug/Run to Cursor** o presione la tecla **F4** como se ve en la Figura 9. Otra forma de realizar esta tarea es hacer clic en el icono **Step Over** de la barra de tareas, Figura 10.

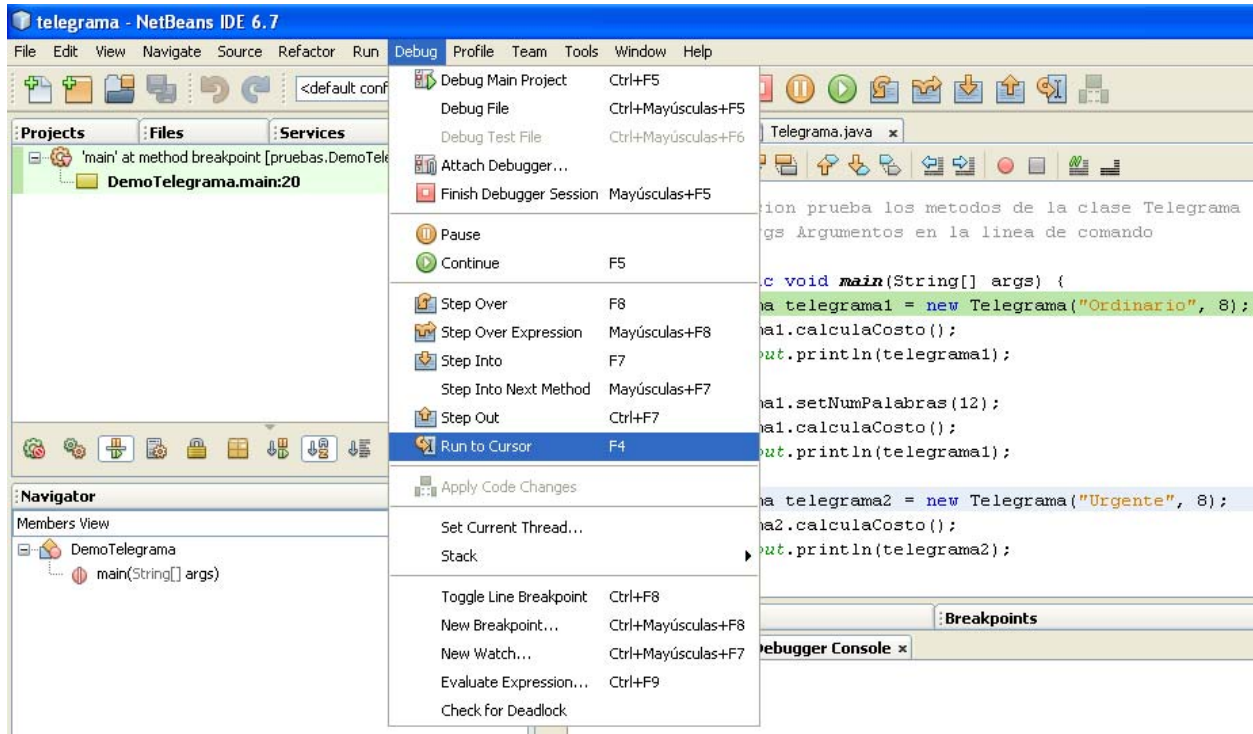


Figura 9



Figura 10

- El programa se detendrá en la línea seleccionada resaltándola. Inspeccione los nuevos valores de los atributos de `telegrama1`. También inspeccione la salida producida por el programa en el panel de depuración que aparece debajo de el panel de edición, traslapando al panel **Variables**, Figura 11.

A continuación podemos ejecutar el programa instrucción por instrucción presionando la tecla **F8**, ejecutar el programa hasta otra instrucción colocando el cursor al inicio de la línea que tiene la instrucción y presionando la tecla **F4**, ejecutar el programa hasta el final o terminar la sesión de depurado como se verán más adelante.

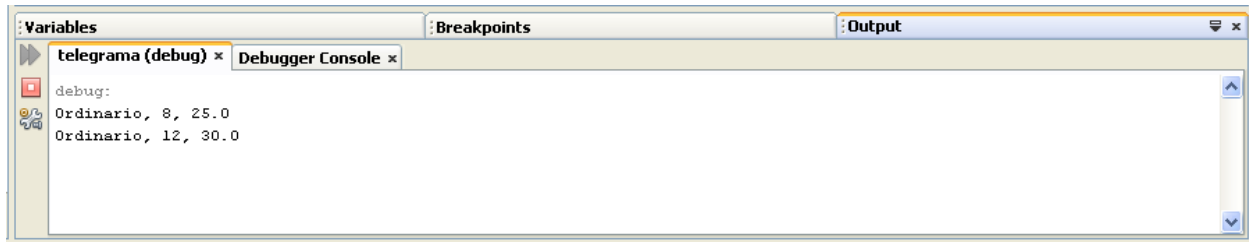


Figura 11

## Depurado de un Método

Si al ejecutar un programa que tiene uno o más métodos queremos depurar el código de un método y no sólo ejecutar el código del método como si fuese una sola instrucción se sigue el siguiente procedimiento:

1. El programa se encuentra detenido en la línea con la instrucción:

```
Telegrama telegrama2 = new Telegrama("Urgente", 8);
```

Para entrar al código del método (el constructor) para ejecutar su código en el depurador seleccione del menú principal la opción **Debug/Step Into** o presione la tecla **F7** como se ve en la Figura 12. Otra forma de realizar esta tarea es hacer clic en el icono **Step Into** de la barra de tareas, Figura 13.

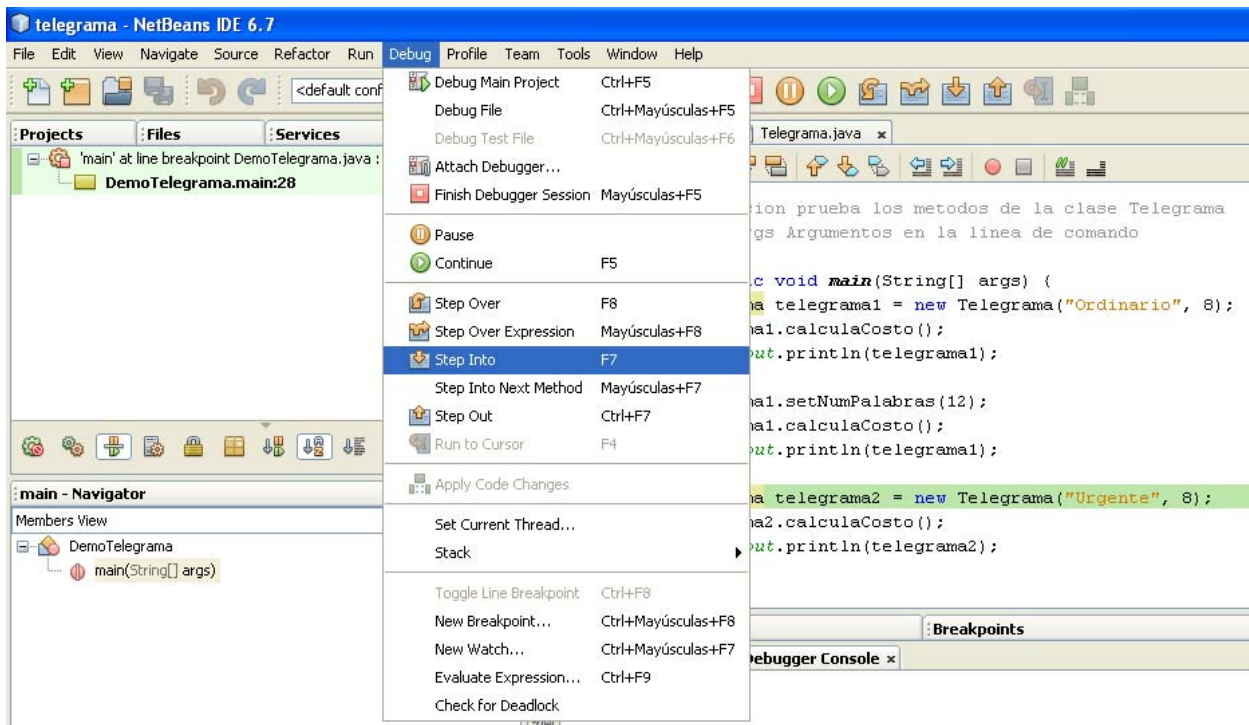


Figura 12

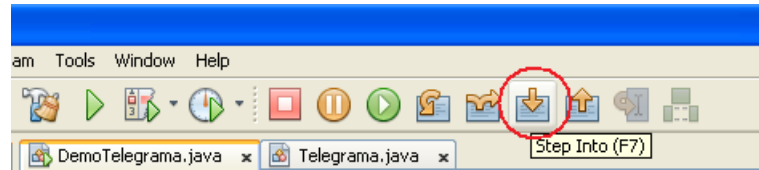


Figura 13

- Al hacerlo, el depurador se detendrá en la línea en la que se encuentra el encabezado del método, como se muestra en la figura 14. Note que en este caso, como el método se encuentra en la clase Telegrama, en el panel de edición, el panel de esa clase es el activo. Si el archivo de la clase hubiera estado cerrado, NetBeans lo habría abierto:

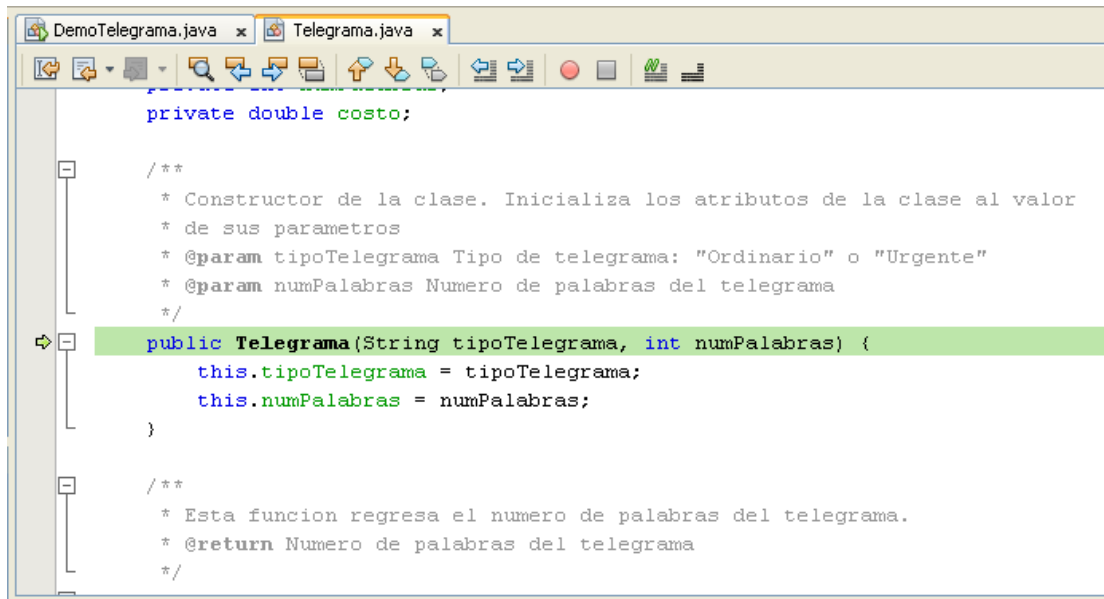


Figura 14

- Como NetBeans entró en el código de una clase, en el panel Variables aparece el nodo `this` que si lo expandimos veremos que aparecen nodos para cada uno de los atributos de la clase, Figura 15. También como esta en el encabezado del método, en el panel variables están los nodos que representan los parámetros de la función
- Para empezar a ejecutar el código del constructor presione la tecla **F8**. Como en este caso en la clase hay unos atributos que se inicializan al declararse y como esta inicialización ocurre al construir un objeto de la clase, el depurador se detendrá en la línea de la primera declaración, Figura 16.
- Presione la tecla F8 cuatro veces para ejecutar cada una de las inicializaciones. El depurador se detendrá en la primera instrucción del método, Figura 17

:Variables		:Breakpoints	:Output
Name	Type		Value
<Enter new watch>			
◆ this	Telegrama		#91
◆ COSTO_ORDINARIO	double		0.0
◆ COSTO_URGENTE	double		0.0
◆ COSTO_ADICIONAL_ORDINARIO	double		0.0
◆ COSTO_ADICIONAL_URGENTE	double		0.0
◆ tipoTelegrama			null
◆ numPalabras	int		0
◆ costo	double		0.0
◆ tipoTelegrama	String		"Urgente"
◆ numPalabras	int		8

Figura 15

```

/**
 * Esta clase permite calcular el costo de un telegrama.
 *
 * @author mdomitsu
 */
public class Telegrama {

    private final double COSTO_ORDINARIO = 25.0;
    private final double COSTO_URGENTE = 40.0;
    private final double COSTO_ADICIONAL_ORDINARIO = 2.5;
    private final double COSTO_ADICIONAL_URGENTE = 4.0;
    private String tipoTelegrama;
    private int numPalabras;
    private double costo;
  
```

Figura 16

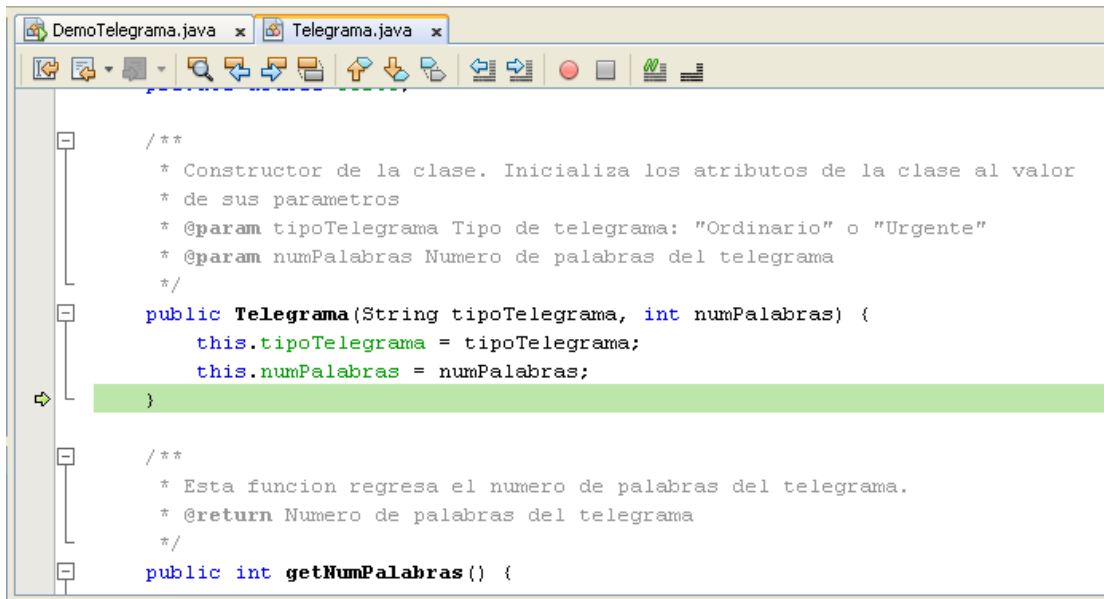
```

/**
 * Constructor de la clase. Inicializa los atributos de la clase al valor
 * de sus parametros
 * @param tipoTelegrama Tipo de telegrama: "Ordinario" o "Urgente"
 * @param numPalabras Numero de palabras del telegrama
 */
public Telegrama(String tipoTelegrama, int numPalabras) {
    this.tipoTelegrama = tipoTelegrama;
    this.numPalabras = numPalabras;
}

/**
 * Esta funcion regresa el numero de palabras del telegrama.
 * @return Numero de palabras del telegrama
  
```

Figura 17

6. Dentro del cuerpo del método puede ejecutar las instrucciones una por una presionando la tecla **F8** o ejecutar hasta una instrucción colocando el cursor en la instrucción deseada y presionando la tecla **F4**.
7. Presione la tecla **F8** dos veces observando lo que sucede en cada caso. Note que NetBeans se detiene en la línea que contiene la llave que cierra el cuerpo del método, Figura 18.



```

DemoTelegrama.java x Telegrama.java x
[Icons] [Tools] [Run] [Debug] [Test] [Find] [Copy] [Paste] [Undo] [Redo] [Home] [End] [List] [Help]

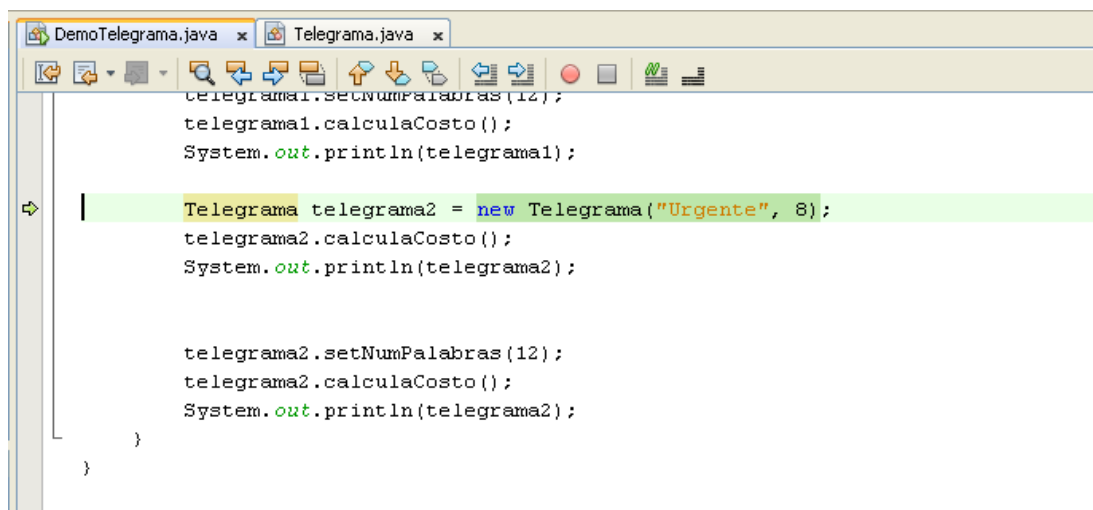
/**
 * Constructor de la clase. Inicializa los atributos de la clase al valor
 * de sus parametros
 * @param tipoTelegrama Tipo de telegrama: "Ordinario" o "Urgente"
 * @param numPalabras Numero de palabras del telegrama
 */
public Telegrama(String tipoTelegrama, int numPalabras) {
    this.tipoTelegrama = tipoTelegrama;
    this.numPalabras = numPalabras;
}

/**
 * Esta funcion regresa el numero de palabras del telegrama.
 * @return Numero de palabras del telegrama
 */
public int getNumPalabras () {

```

Figura 18

8. Presione de nuevo la tecla **F8** para terminar la ejecución del método. El depurador termina la ejecución del método y el programa regresa a la línea de código en la que ocurrió la llamada al método, Figura 19.



```

DemoTelegrama.java x Telegrama.java x
[Icons] [Tools] [Run] [Debug] [Test] [Find] [Copy] [Paste] [Undo] [Redo] [Home] [End] [List] [Help]

telegrama1.setNumPalabras(12);
telegrama1.calculaCosto();
System.out.println(telegrama1);

Telegrama telegrama2 = new Telegrama("Urgente", 8);
telegrama2.calculaCosto();
System.out.println(telegrama2);

telegrama2.setNumPalabras(12);
telegrama2.calculaCosto();
System.out.println(telegrama2);
}
}

```

Figura 19

## Terminar el Depurado de un Método

Si estando depurando un método ya no nos interesa continuar su ejecución instrucción por instrucción podemos ejecutar el código hasta el final y regresar a donde fue llamado, siguiendo el siguiente procedimiento:

1. Seleccione del menú principal la opción **Debug/Step Out** o presione la tecla **Ctrl + F7** como se ve en la Figura 20. Otra forma de realizar esta tarea es hacer clic en el icono **Step Into** de la barra de tareas, Figura 21.

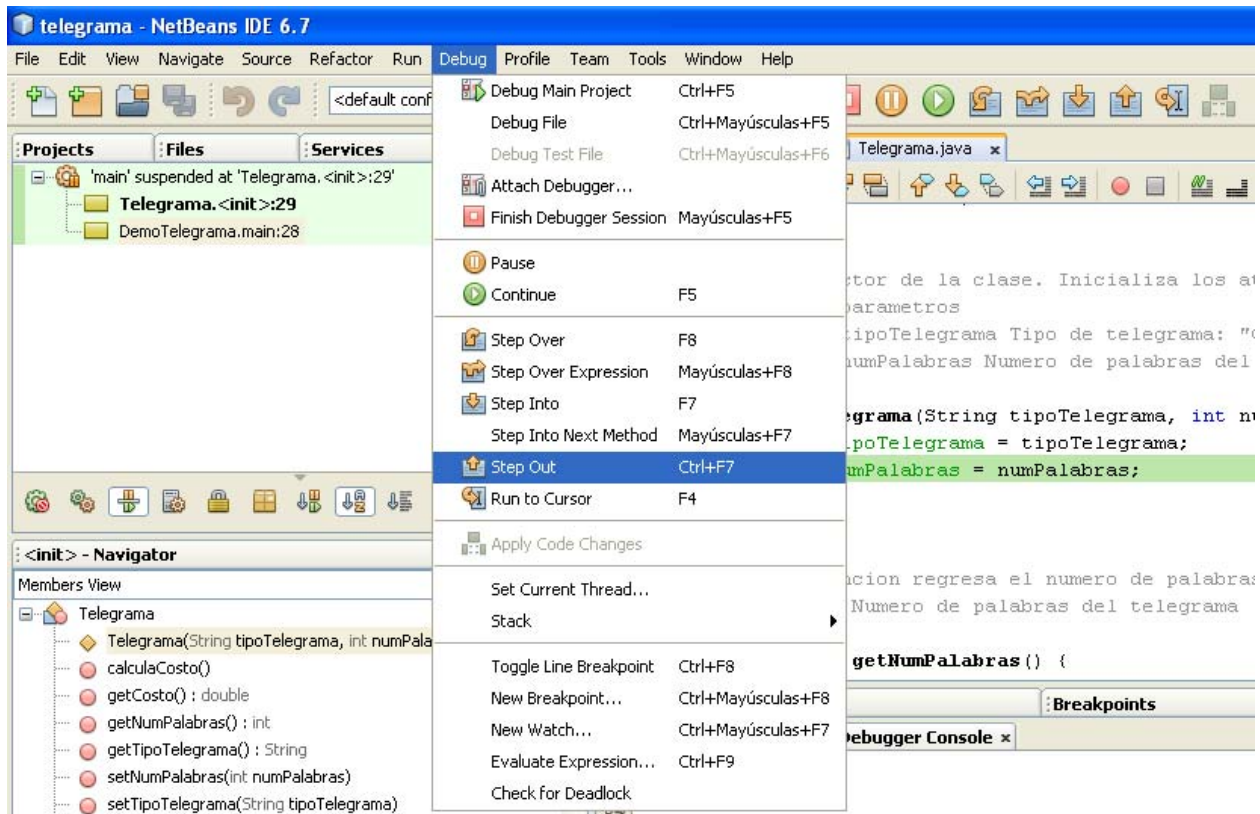


Figura 20

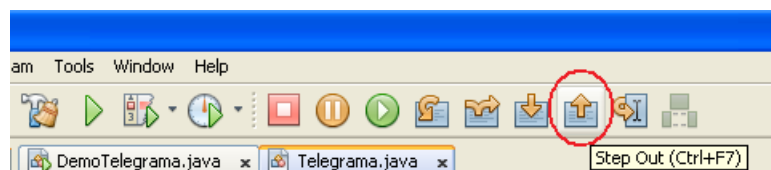


Figura 21

2. El depurador termina la ejecución del método y el programa regresa a la línea de código en la que ocurrió la llamada al método, Figura 19:
3. Podemos seguir ejecutando el programa instrucción por instrucción presionando la tecla **F8**, ejecutar el programa hasta otra instrucción colocando el cursor al



inicio de la línea que tiene la instrucción y presionando la tecla **F4**, ejecutar el programa hasta el final o terminar la sesión de depurado como se verán a continuación.

## Ejecución de un Programa Hasta el Final

Hay ocasiones en las que nos interesa ejecutar las instrucciones de un programa hasta el final sin detenernos en las instrucciones intermedias. Una forma de hacerlo es la siguiente:

1. Seleccione del menú principal la opción **Debug/Continue** o presione la tecla **F5** como se ve en la Figura 22. Otra forma de realizar esta tarea es hacer clic en el icono **Continue** de la barra de tareas, Figura 23.

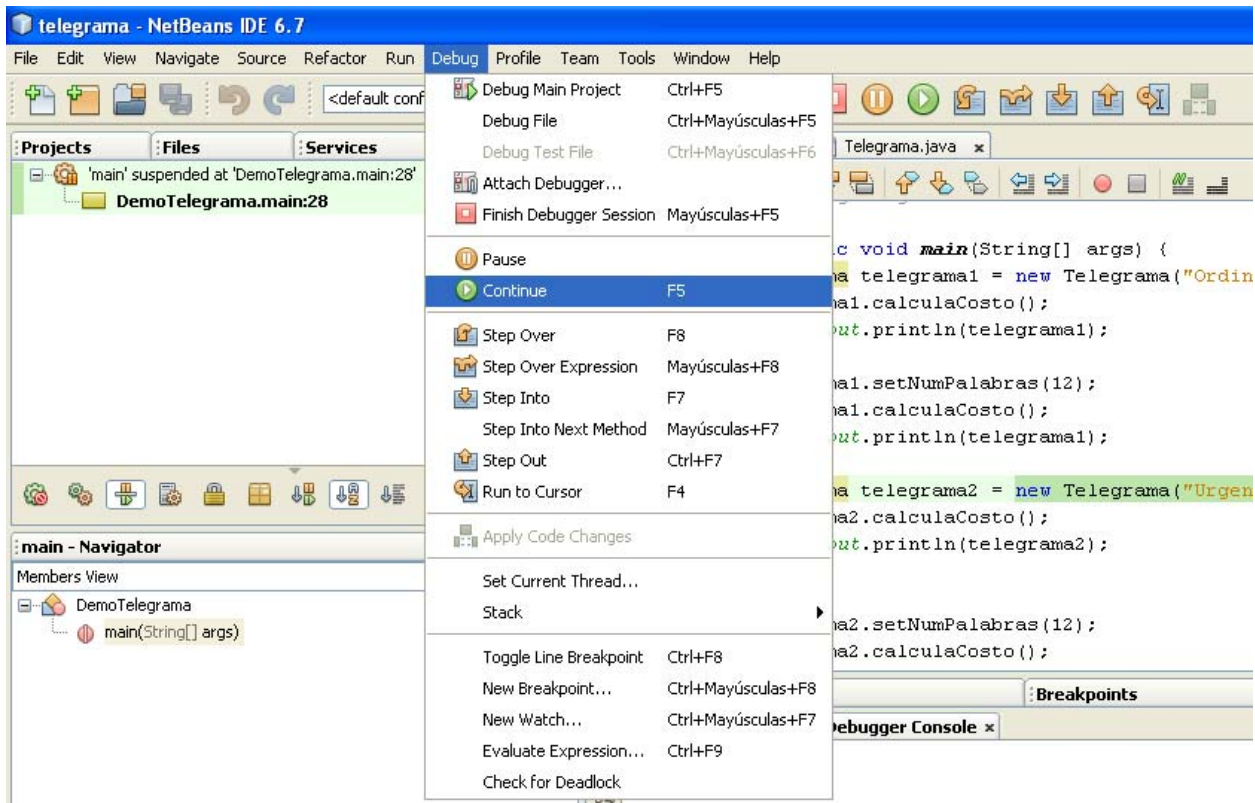


Figura 22



Figura 23

2. Al hacerlo, el programa ejecutará hasta el final, terminando la sesión de depuración.

## Terminación de la Sesión de Depurado de un Programa

Si ya no nos interesa ejecutar las instrucciones de un programa hasta el final, podemos terminar la sesión de depurado del programa. Una forma de hacerlo es la siguiente:

1. Seleccione del menú principal la opción **Debug/Finish Debugger Session** o presione las teclas **Mayúsculas + F5** como se ve en la Figura 24. Otra forma de realizar la misma tarea es hacer clic sobre el icono **Finish Debugger Session** de la barra de tareas, como se muestra en la Figura 25.

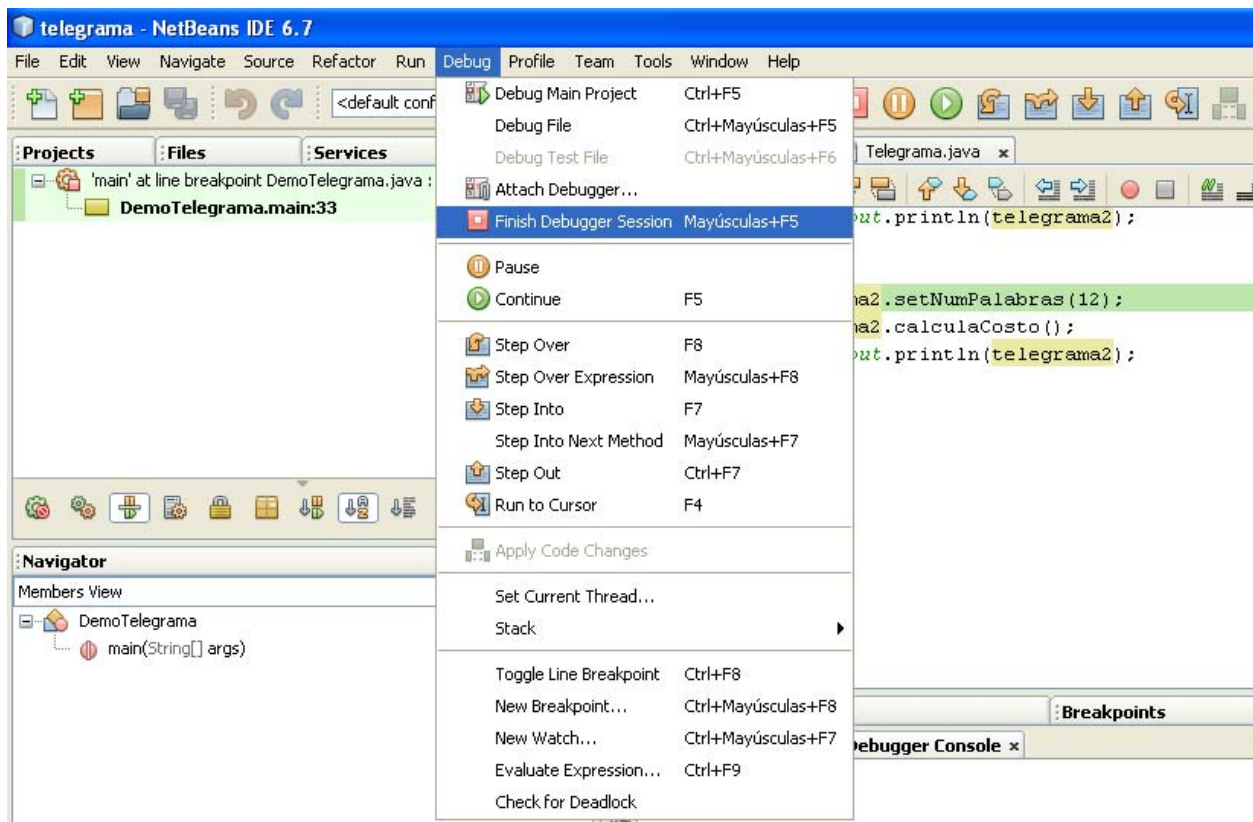


Figura 24

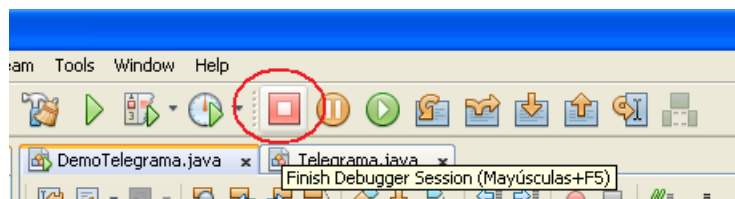


Figura 25

2. Al hacerlo, el programa terminará la sesión de depurado.