10.1. Ejercicio resuelto: Análisis y diseño orientado a objetos para calcular el volumen de una esfera

En esta sección se presenta el análisis y diseño de un algoritmo orientado a objetos para calcular el volumen de una esfera. Con este ejercicio resuelto y completo se pretende que el lector tenga una guía sobre la aplicación de la metodología para la solución de problemas a través de la computadora. A continuación se describen y redactan cada una de las secciones del análisis y se realiza el diseño orientado a objetos en Raptor.

10.1.1. Análisis

Recuerde que una buena práctica de un programador es realizar un análisis antes de iniciar el diseño e implementación de su aplicación. Aún en aplicaciones tan sencillas (como calcular el volumen de una esfera) es recomendable que el programador aplique la metodología para resolver problemas a través de la computadora, ya que esto le forjará el buen hábito de realizar las etapas de manera ordenada para obtener aplicaciones confiables.

10.1.1.1. Investigación preliminar

Para calcular el volumen de una esfera es necesario conocer la fórmula, para ello, el estudiante puede consultar en internet, libros, formularios, etc. o bien preguntarle a algún profesor de matemáticas. Al hacerlo, encuentra la ecuación correspondiente (Fig. 10.19).

$$V = \frac{4}{3}\pi r^3$$

Fig. 10.19. Fórmula para calcular el volumen de una esfera.

10.1.1.2. Definición del problema

En esta sección el analista debe redactar un listado de los pasos que va a realizar para solucionar este problema. Cabe mencionar que esta redacción es libre de estilo y puede hacerlo de manera práctica y sencilla. Para elaborar un algoritmo orientado a objetos que calcule el volumen de una esfera es necesario definir el problema redactando el listado de pasos a seguir (Fig. 10.20).

- Diseñar una clase para la esfera que tenga como atributo el radio y un método que calcule su volumen
- Declarar y crear un objeto de la clase Esfera
- Solicitar al usuario que capture el valor del radio e introducirlo al atributo del objeto
- Invocar al método que calcula y devuelve el valor del volumen de la esfera
- Mostrar el resultado

Fig. 10.20. Definición de un problema para calcular el volumen de una esfera.

10.1.1.3. Definición de los datos de entrada

En esta parte del análisis se deben identificar los datos que el usuario ingresará al sistema como datos de entrada. Es importante que estos datos sean plenamente identificados a través de una variable con su respectivo tipo de dato. Para calcular el volumen de una esfera es necesario que el usuario introduzca el valor del radio; para ello es necesario declarar una variable en la clase que sea capaz de almacenar este valor y se define a través de un atributo privado (Fig. 10.21).

Datos de entrada:

- Valor del radio de la esfera
  - Atributo: dbl`Radio` : real (double)
  - Mutator: `setRadio(double dblRadio)`

Fig. 10.21. Definición del dato de entrada para calcular el volumen de una esfera.

NOTA: Debido a que Raptor no contiene definiciones de propiedades; en su lugar se utiliza un atributo privado con su respectivo *mutator* (recuerde que un *mutator* es un método que recibe un valor como parámetro para introducirlo al atributo de un objeto).

10.1.1.4. Definición de información de salida

En esta sección se redacta la información de salida que arrojará el sistema, es decir, el valor que se mostrará como resultado. De manera semejante a los datos de entrada, puede redactarse de manera libre pero destacando la descripción del resultado mostrado, la variable, atributo, propiedad o método que lo contiene y su respectivo tipo de dato.
El algoritmo que calcula el volumen de una esfera arroja como resultado el valor calculado por el método `CalcularVolumen()` que devuelve un valor de tipo real (Fig. 10.22).

Información de salida:

- Valor del volumen de la esfera
  - Método: `CalcularVolumen()` : real (double)

Fig. 10.22. Información de salida para calcular el volumen de una esfera.

10.1.1.5. Variables auxiliares

Se conoce como variable auxiliar a aquel dato que no forma parte del conjunto de datos de entrada ni de salida, es decir, son variables de trabajo que se utilizan para manejar datos intermedios o cálculos internos de algún proceso.
En el ejercicio para calcular el volumen de una esfera es necesario definir variables auxiliares y, en este caso, simplemente se redacta el nombre de la clase y del objeto creado (Fig. 10.23).

Variables auxiliares:

- Clase: `Esfera`
- Objeto: `miEsfera`

Fig. 10.23. Variables auxiliares para calcular el volumen de una esfera.

10.1.2. Diseño

Después de realizar y documentar el análisis de la aplicación se procede a elaborar el diseño. En esta sección se describen los pasos a seguir para resolver el problema planteado.

A continuación se muestran las actividades para elaborar el algoritmo orientado a objetos que calcula el volumen de una esfera utilizando Raptor como herramienta de diseño.

10.1.2.1. Diseño descendente

Para preparar el diseño del algoritmo de nuestra aplicación, es necesario ejecutar Raptor y habilitar el modo orientado a objetos (Fig. 10.24).

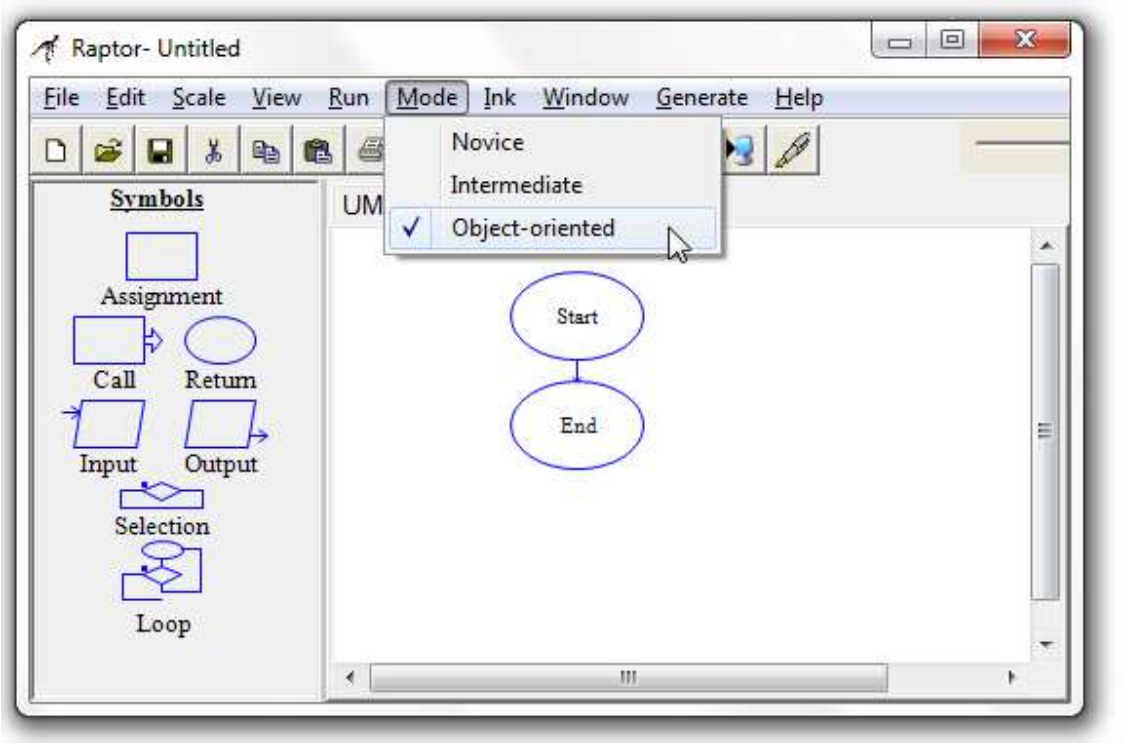

Fig. 10.24. Habilitar el modo orientado a objetos en Raptor.

Según lo plantado en la definición del problema de la fase de análisis, para calcular el volumen de una esfera es necesario diseñar una clase en Raptor con ciertos componentes, para ello, se habilita la pestaña UML y se oprime el botón que permite agregar una clase nueva (Fig. 10.25).

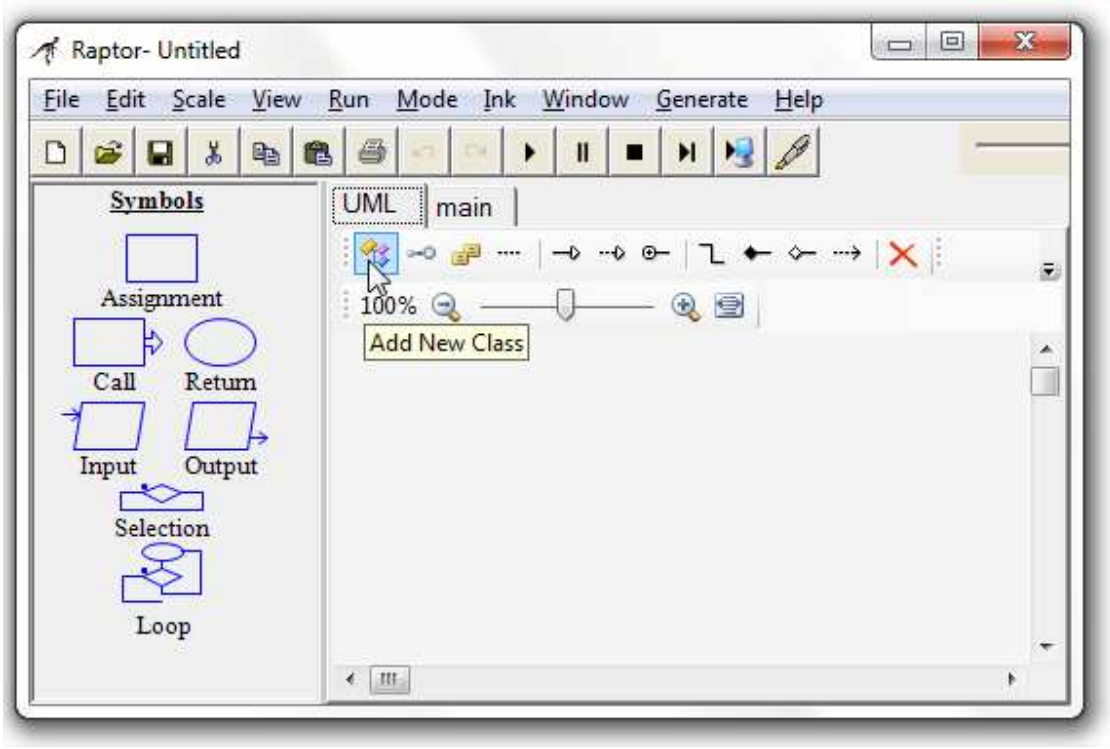

Fig. 10.25. Agregar una nueva clase en Raptor.

Después de agregar la clase se procede a definir su nombre (Fig. 10.26).

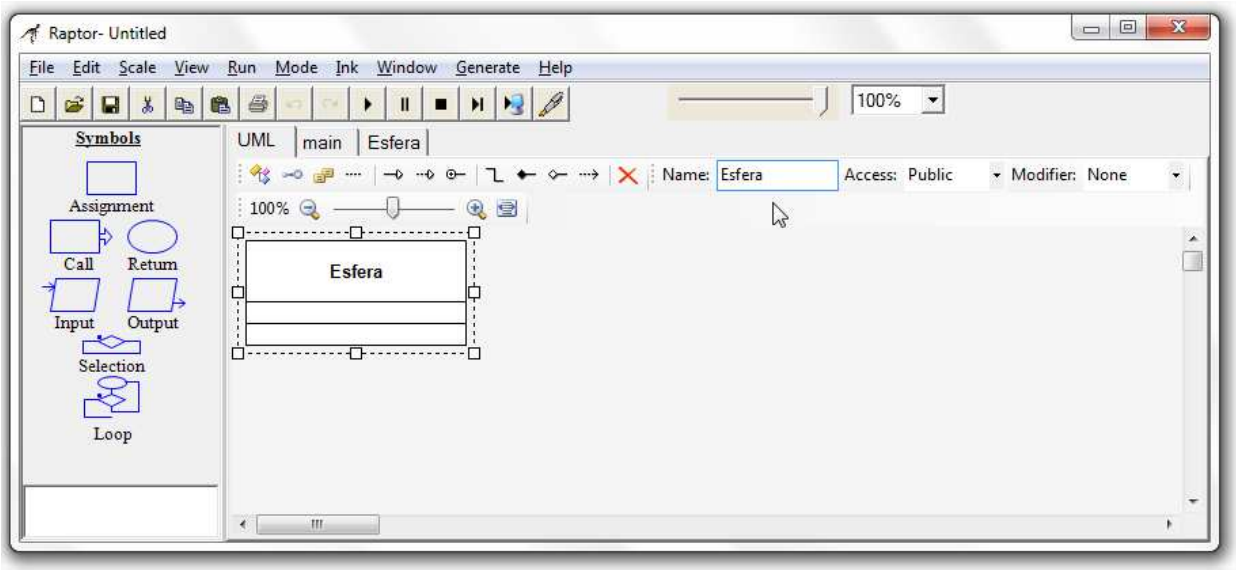

Fig. 10.26. Definir el nombre de la nueva clase en Raptor.

Una vez establecido el nombre de la clase se procede a definir sus componentes: un atributo privado para el radio de la esfera, su *mutator* y el método que calcula y devuelve el valor de su volumen. El atributo del radio de la clase en Raptor se puede agregar haciendo doble *click* en el diagrama de la clase en UML y oprimiendo el botón con la etiqueta "*New Field*" para posteriormente ingresar sus detalles: se define el nombre `dblRadio` con su tipo de dato real (`double`). Es importante destacar que este atributo tendrá un modificador de acceso privado para que no pueda ser accedido directamente desde fuera de la instancia de la clase y solamente se logre a través de su *mutator* (recuerde que en Raptor no se pueden definir propiedades) (Fig. 10.27).

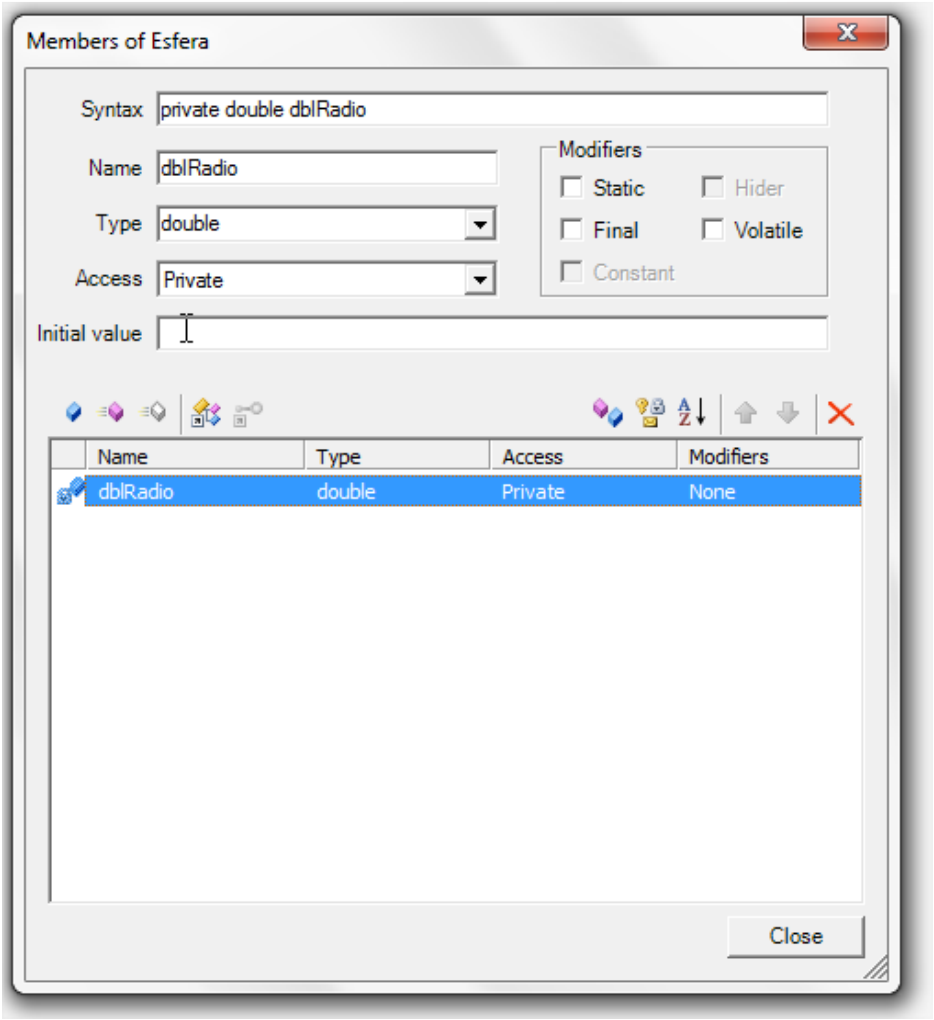

Fig. 10.27. Agregar un atributo a la clase.

En virtud de que se declara el atributo con un modificador de acceso privado, no se podría acceder directamente desde el exterior, para ello es necesario un mecanismo que funcione como intermediario entre el atributo privado y el exterior del objeto, de tal manera que permita introducirle valores. Como se ha mencionado con anterioridad, este mecanismo puede implementarse de dos formas diferentes:

1.- A través de una propiedad

2.- Por medio de un *mutator*

Sin embargo, Raptor no cuenta con la definición de propiedades, por lo que se considera la implementación de un método que reciba como parámetro el valor que se desea introducir al atributo privado del objeto conocido como *mutator*.

En este ejercicio se define un procedimiento identificado como `setRadio(double dblRadio)` que actúa como un *mutator*. Para ello se oprime el botón con la etiqueta "*New Method*" y se ingresan sus detalles (Fig. 10.28).

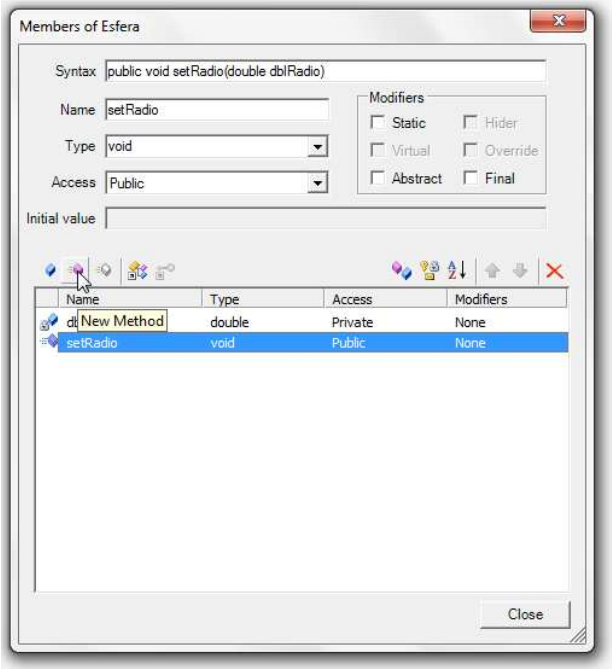

Fig. 10.28. Agregar el mutator.

Enseguida se procede a agregar el método que calcula y devuelve el valor del volumen de la esfera. Nótese que este método se trata de una función que no recibe parámetros, ya que el valor del radio de la esfera lo tomará de su atributo privado (previamente introducido), aplica la fórmula para calcular el volumen de la esfera y devuelve el resultado (por eso se declara de tipo `double`) (Fig.10.29).

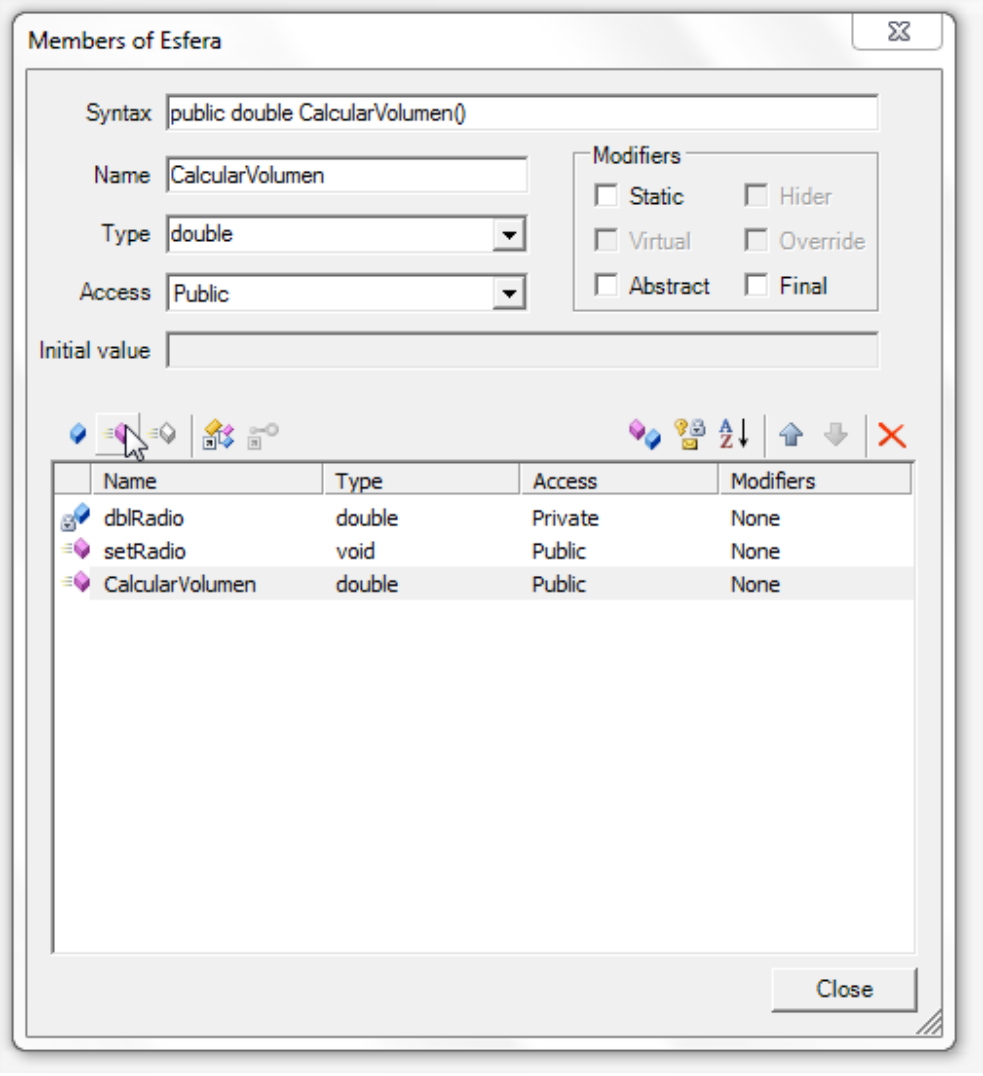

Fig. 10.29. Agregar el método que calcular el volumen

10.1.2.2. Refinamiento de la propuesta

Esta etapa del diseño contempla definir los detalles del algoritmo que solucionará la problemática planteada. Para lograrlo, se deben elaborar los diagramas de flujo de los métodos de la clase y del algoritmo principal.

Para diseñar el diagrama de flujo del *mutator* de la clase se activa la pestaña con la etiqueta "`Esfera`", se selecciona la pestaña "`setRadio`" y se ingresan sus detalles. El *mutator* es un procedimiento (no devuelve valor) que recibe como parámetro (`dblRadio`) el dato que desea ser introducido al atributo privado (`this.dblRadio`). Es muy importante destacar que el atributo privado y el parámetro pueden tener el mismo nombre, sin embargo esto no es un error, ya que la referencia `this` los distingue y diferencia; es decir, la variable con la palabra `this` hace referencia al atributo, mientras que la que no la tiene se refiere al parámetro recibido. Esto significa que `this.dblRadio` es el atributo privado y `dblRadio` es el parámetro recibido por el *mutator*, por lo tanto, se introduce el valor del parámetro recibido al atributo privado (Fig. 10.30).

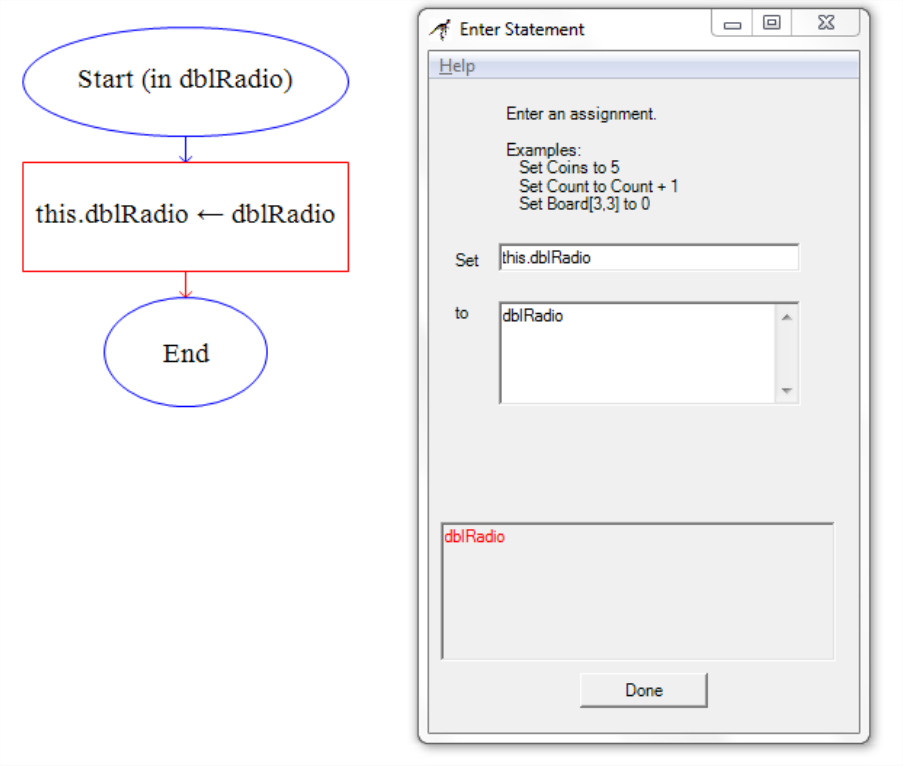

Fig. 10.30. Diagrama de flujo del mutator

De manera semejante se diseña el diagrama de flujo del método (función) que calcula el volumen de la esfera `Return` con la fórmula correspondiente. Este método utiliza la constante `Pi` con el valor de 3.1415927 (Fig. 10.31).

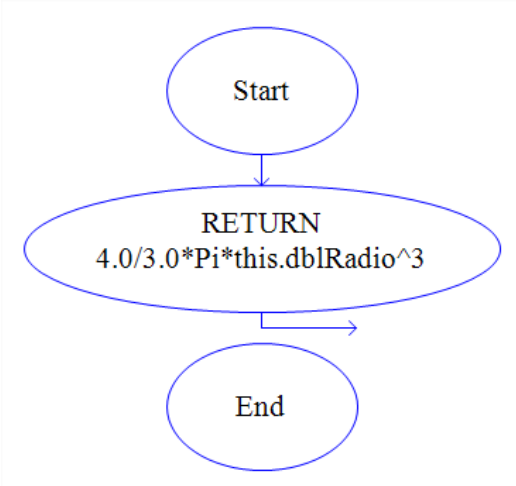

Fig. 10.31. Diagrama de flujo del método CalcularVolumen

Una vez definido el método de la clase que calcula y devuelve el valor del volumen de una esfera, a continuación se debe elaborar el diagrama de flujo del algoritmo principal. Para ello se debe activar la pestaña con la etiqueta "`main`".

El primer paso que debe ejecutar el algoritmo principal es limpiar la pantalla para después proceder a la creación de un objeto de la clase `Esfera`. Para ello se selecciona el símbolo con la etiqueta "*Assignment*" y se ingresa el nombre del objeto con la sentencia `new` para su creación (Fig. 10.32).

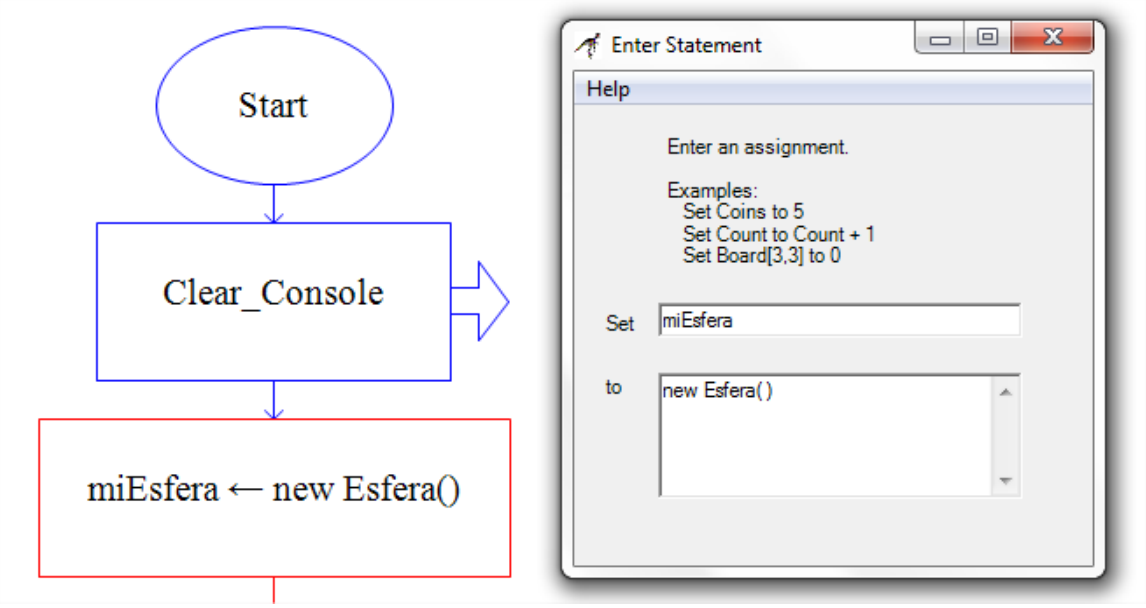

Fig. 10.32. Creación del objeto de una esfera

Una vez hecho esto se procede a definir e inicializar la variable `dblRadio` que servirá para recibir el valor ingresado por el usuario. Esto se logra a través de un símbolo con la etiqueta "`Assignment`". Es importante mencionar que se trata de una variable local e independiente al objeto (no confundir con el atributo privado del mismo nombre) (Fig. 10.33).

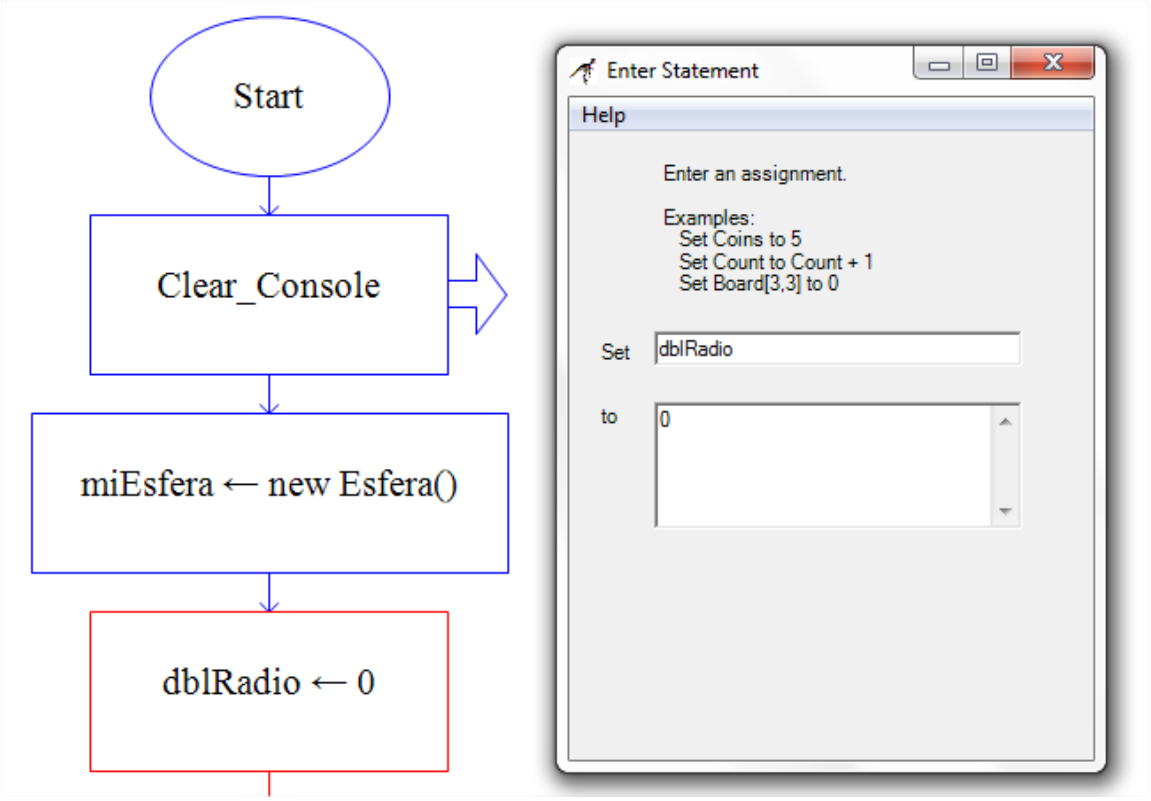

Fig. 10.33. Definición e inicialización de la variable de entrada

El siguiente paso es solicitar al usuario que ingrese el valor del radio de la esfera, almacenarlo temporalmente en la variable local e independiente (`dblRadio`) e introducirlo al atributo privado (`this.dblRadio`) del objeto recién creado (`miEsfera`) a través de su mutator (`setRadio`). Para ello se selecciona un símbolo con la etiqueta "*Input*" y se ingresan sus detalles (Fig. 10.34).

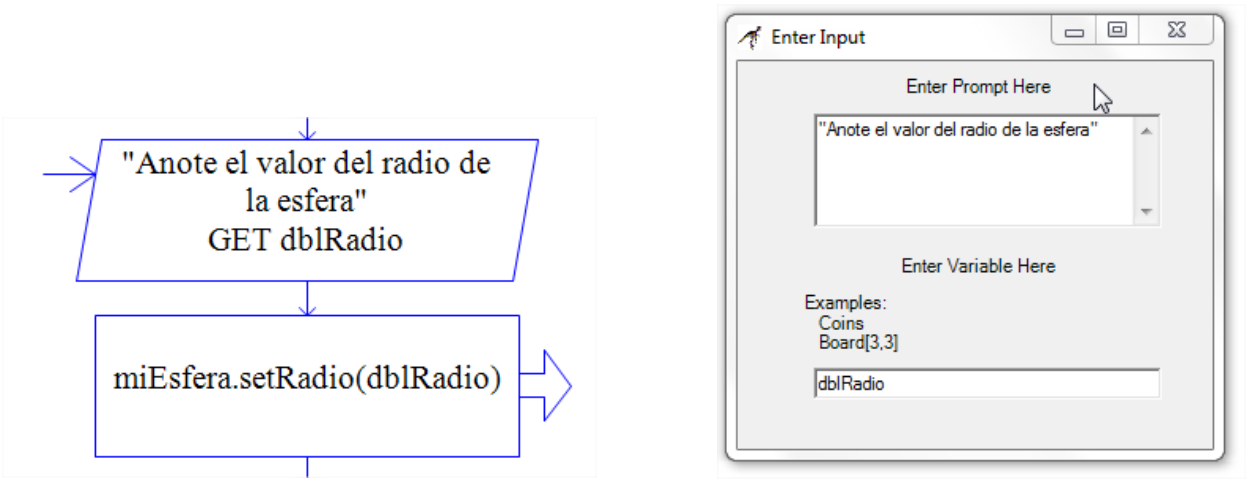

Fig. 10.34. Ingresar el radio de la esfera.

Una vez que se ha introducido el valor del radio al objeto, se procede a invocar el método que calcula su volumen y desplegar el resultado devuelto. Para ello se selecciona el símbolo con la etiqueta "*Output*" y se ingresan sus detalles (Fig.10.35).

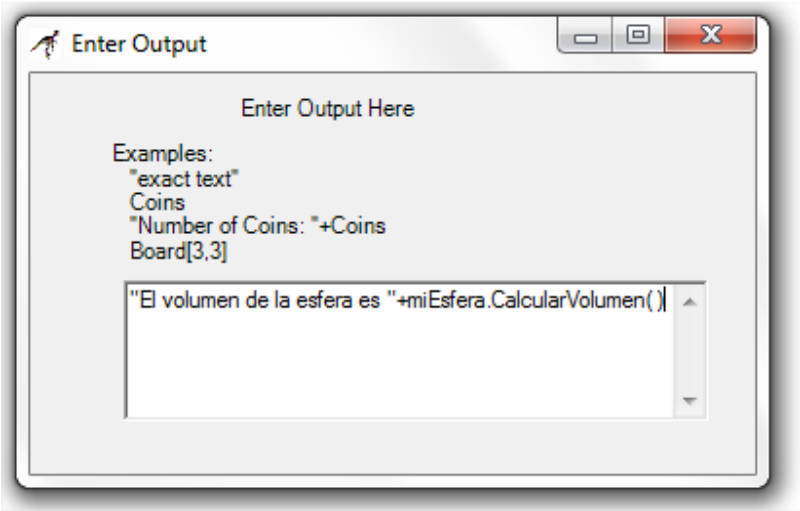

Fig. 10.35. Invocar el método y desplegar el resultado

De esta forma queda completamente definido el diagrama de flujo del algoritmo principal (Fig. 10.36).

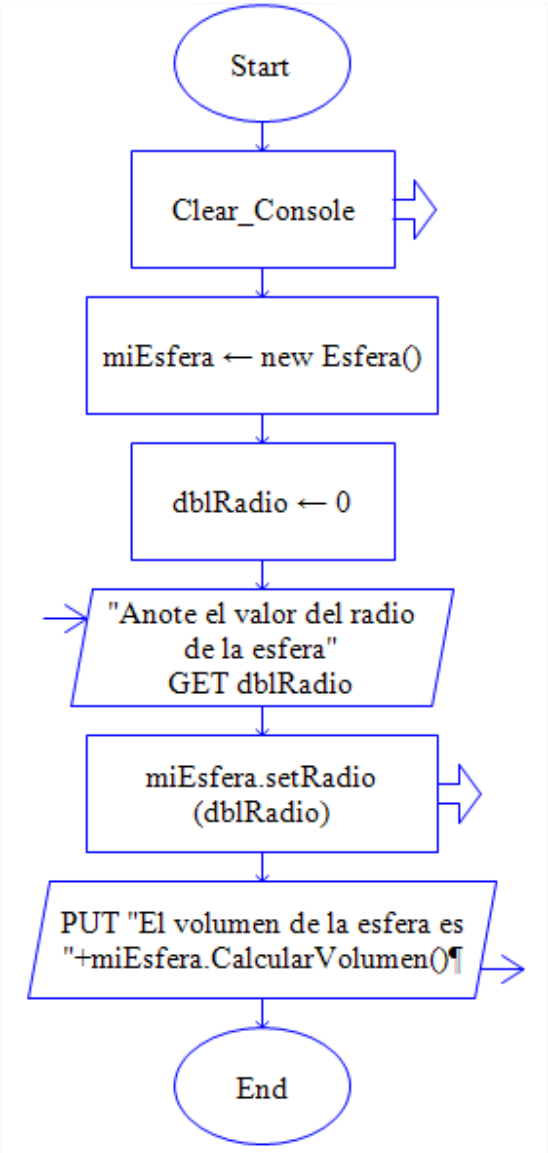

Fig. 10.36. Diagrama de flujo completo del algoritmo principal.

10.1.3. Ejecución y pruebas del algoritmo en Raptor

Después de concluir con el diseño de los diagramas de flujo, se procede a ejecutar el algoritmo, introducirle datos y asegurarse que los resultados arrojados sean los correctos. Para ello, se oprime el botón con la etiqueta "*Execute to Completion*" (Fig. 10.37).

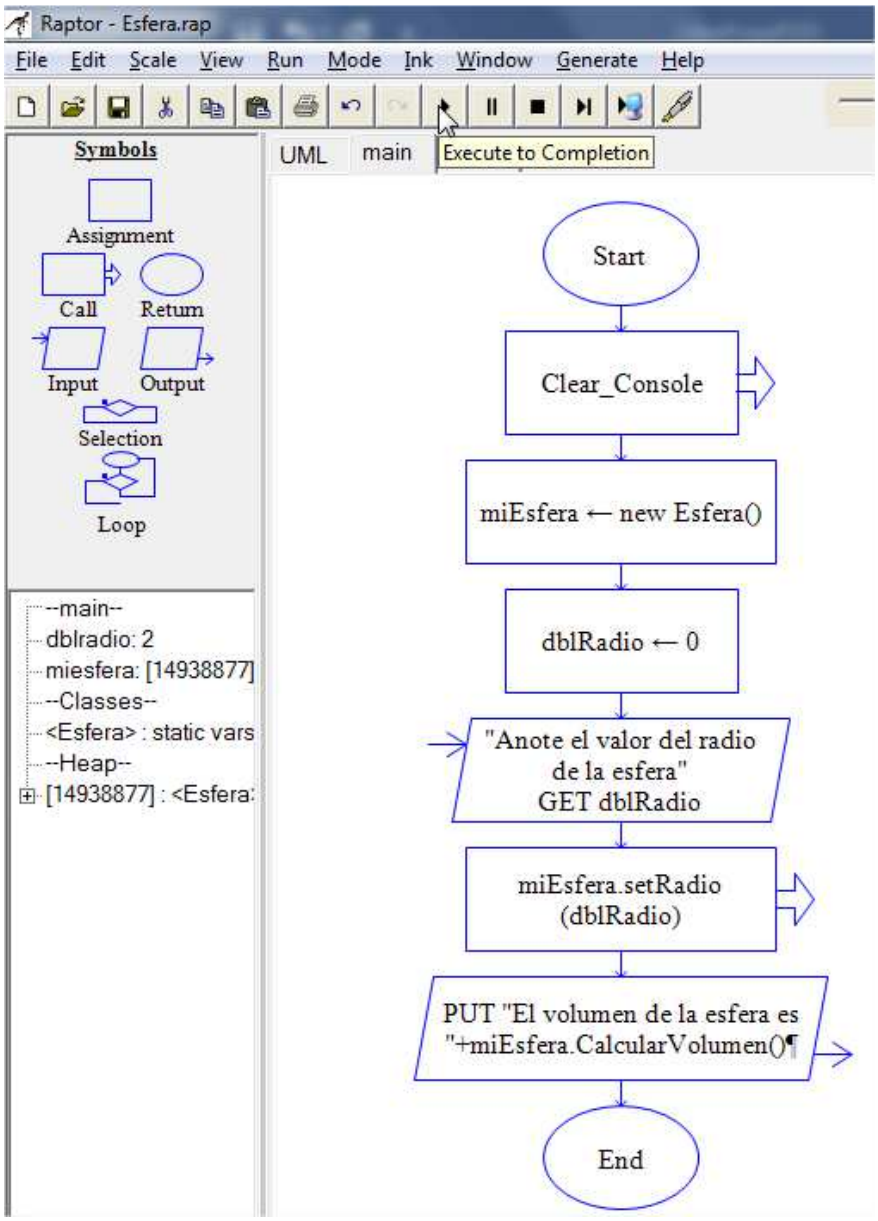

Fig. 10.37. Ejecución del algoritmo en Raptor.

Para asegurarse que el algoritmo es correcto se deben hacer “pruebas de escritorio”, es decir, ejecutarlo varias veces con diferentes conjuntos de datos de entrada para verificar los resultados.

Un algoritmo diseñado en Raptor se puede ejecutar paso a paso o completo. Al ponerlo en operación, se resalta y distingue el símbolo del diagrama que se está ejecutando en ese momento. Para verificar el algoritmo planteado, se procede a ejecutarlo, introducirle un valor y comprobar el resultado (Fig. 10.38).

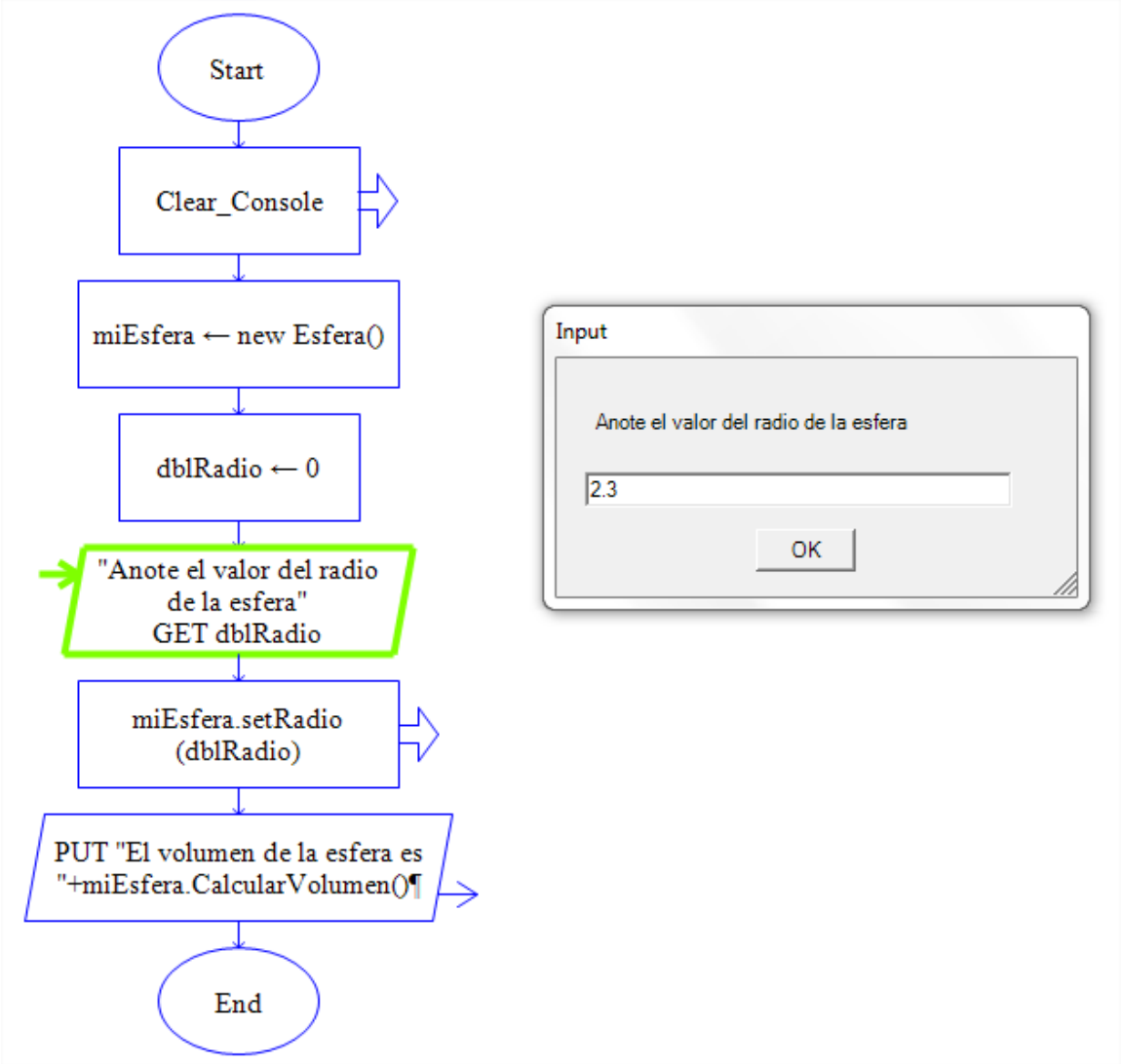

Fig. 10.38. Ingresar un dato

El usuario captura un dato que lo recibe una variable local, se introduce al objeto a través de su `mutator` y se despliega el resultado invocando a la función que calcula el volumen de la esfera (Fig. 10.39).

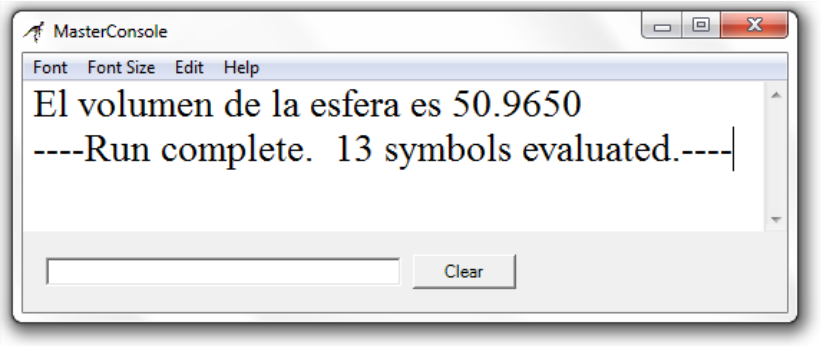

Fig. 10.39. Despliegue del resultado