Herramientas para el Mantenimiento del Software

By:
Miguel-Angel Sicilia

Herramientas para el Mantenimiento del Software

By:
Miguel-Angel Sicilia

Online:
< http://cnx.org/content/col10584/1.8/ >

C O N N E X I O N S

Rice University, Houston, Texas

Collection structure revised: January 14, 2009
PDF generated: February 5, 2011
For copyright and attribution information for the modules contained in this collection, see p. 38.

Table of Contents

Chapter 1

Reingeniería con Netbeans[1]

La ingeniería inversa se ha definido como el proceso de construir especificaciones de un mayor nivel de abstracción partiendo del código fuente de un sistema software o cualquier otro producto (se puede utilizar como punto de partida cualquier otro elemento de diseño, etc.).

A continuación se detalle cómo realizar Ingeniería Inversa a través de NetBeans, para ello primero hay que tener instalado NetBeans 5.5.

1.1 Ejemplo práctico con Netbeans

1.1.1 Instalar NetBeans 5.5

En primer lugar tenemos que tener instalado NetBeans 5.5 (http://www.netbeans.org[2]). De esta Web se puede descargar e instalar. Se puede instalar en diferentes idiomas.

A continuación debemos de comprobar si tenemos instalado o no el módulo UML Modelling[3].

Para ello, primero hay que asegurarse de que no está ya instalado. Esto puede hacerse en la opción de menú "Herramientas|Gestor de módulos...":

[1] This content is available online at <http://cnx.org/content/m17590/1.5/>.

[2] http://www.netbeans.org/

[3] UML Modeling: Reverse Engineering Java Applications. Netbeans. Disponible en: http://www.netbeans.org/kb/55/uml-re.html (<http://www.netbeans.org/kb/55/uml-re.html>).

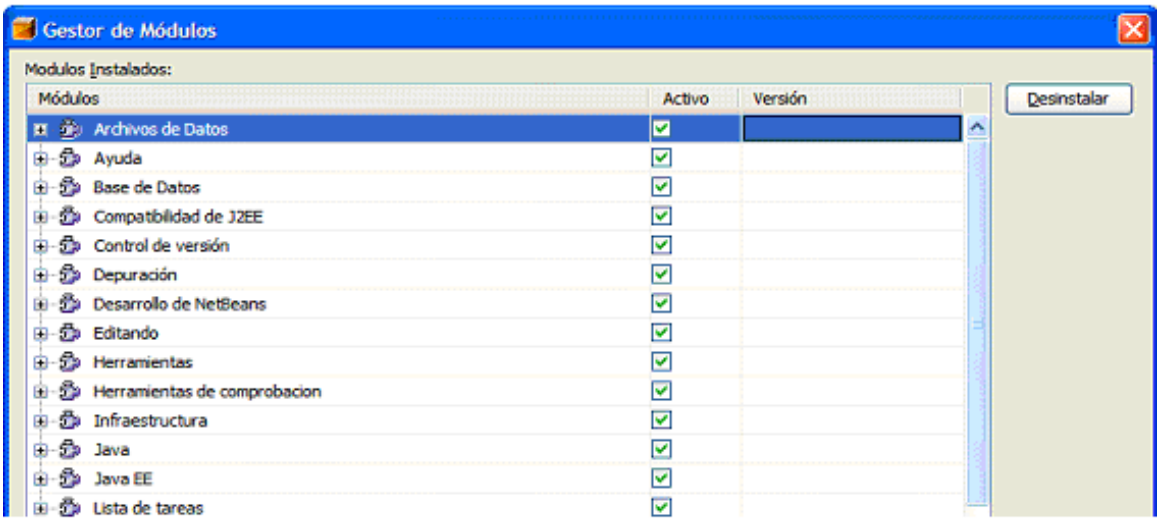

Figure 1.1

Figura 1. Módulos del Gestor de Módulos a través de la opción Herramientas

Una vez comprobado que el módulo de UMl no está instalado, utilizando el botón "Actualizar" y se podrán descargar actualizaciones disponibles.

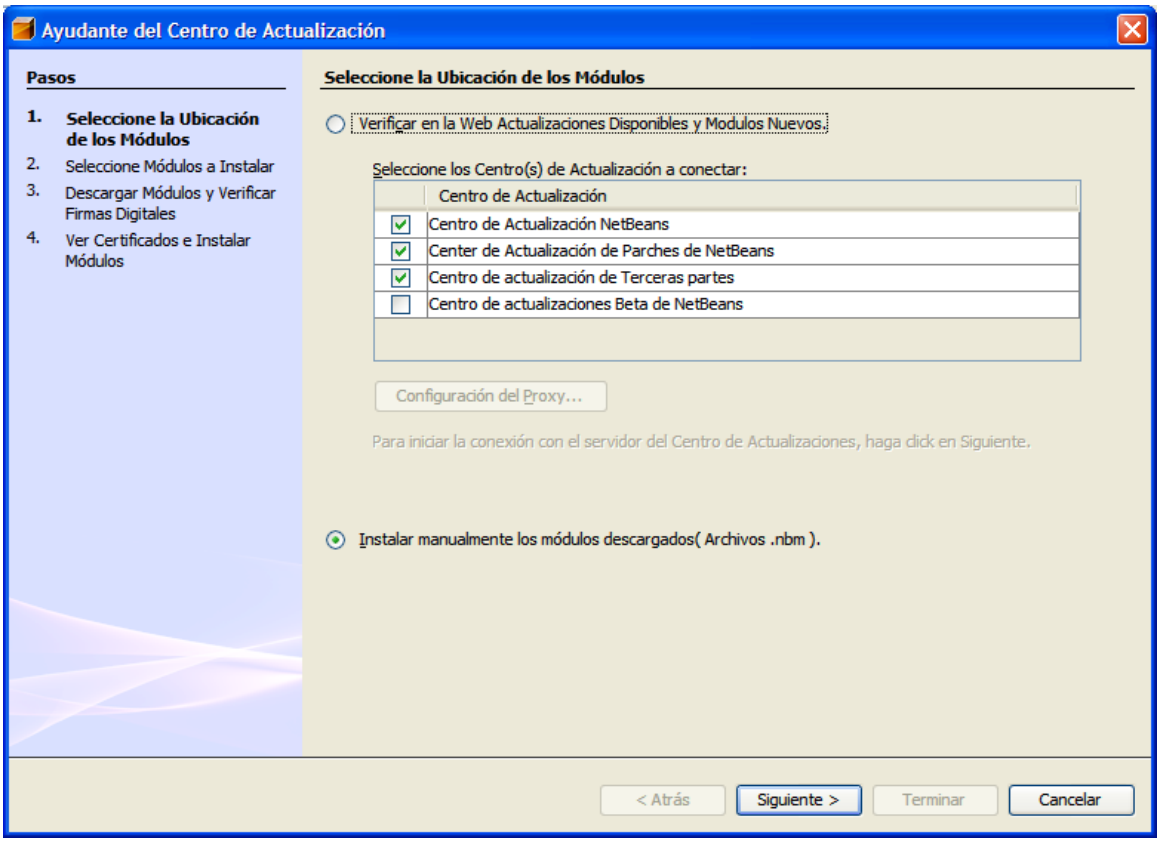

Figure 1.2

Figura 2. Instalar manualmente UML Modelling

Lo podemos encontrar para descargarlo en la Web e instalarlo manualmente en la siguiente dirección: http://www.linglom.com/downloads/uml-beta-nbms.zip[4]

Una vez descargado el fichero, lo descomprimimos y tendríamos que agregarlos en la siguiente pantalla:

[4] http://www.linglom.com/downloads/uml-beta-nbms.zip

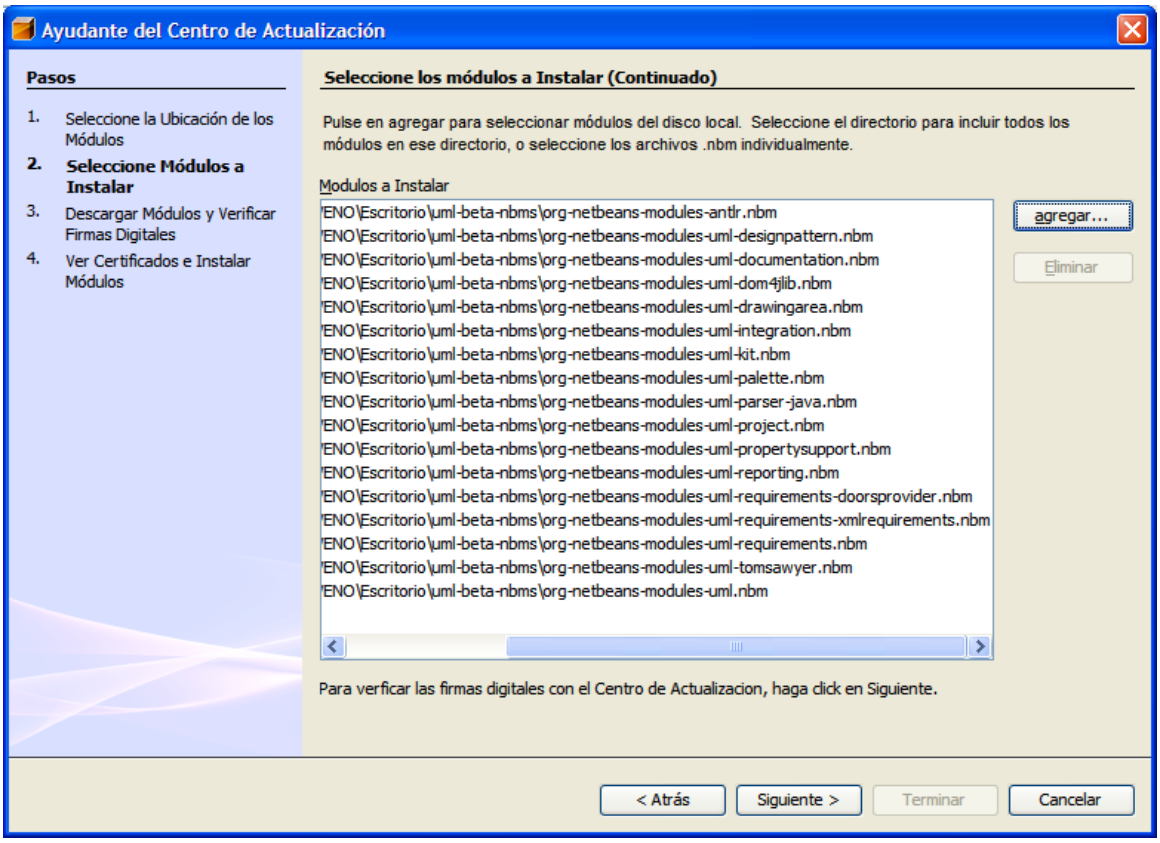

Figure 1.3

Figura 3. Visión de los módulos a instalar del fichero zip descomprimido.

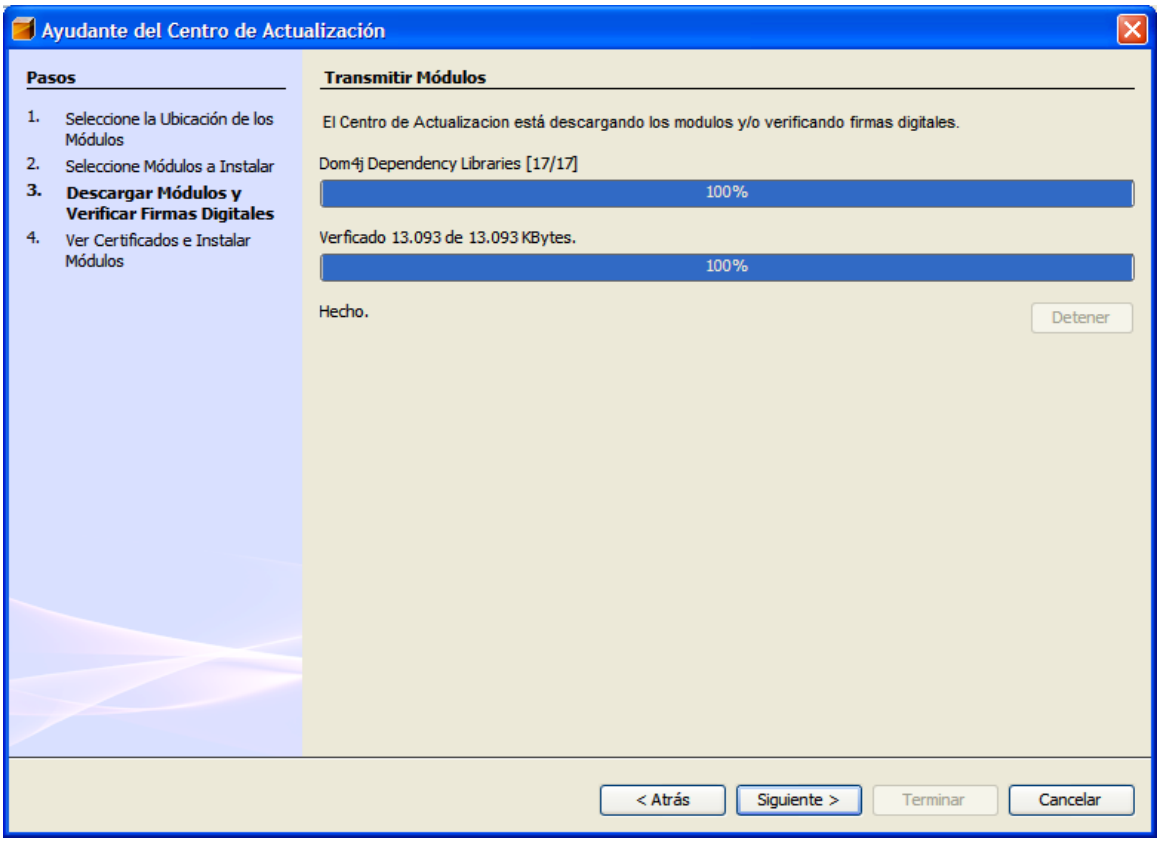

Figure 1.4

Figura 4. Instalación de los módulos y verificación de la firma digital.

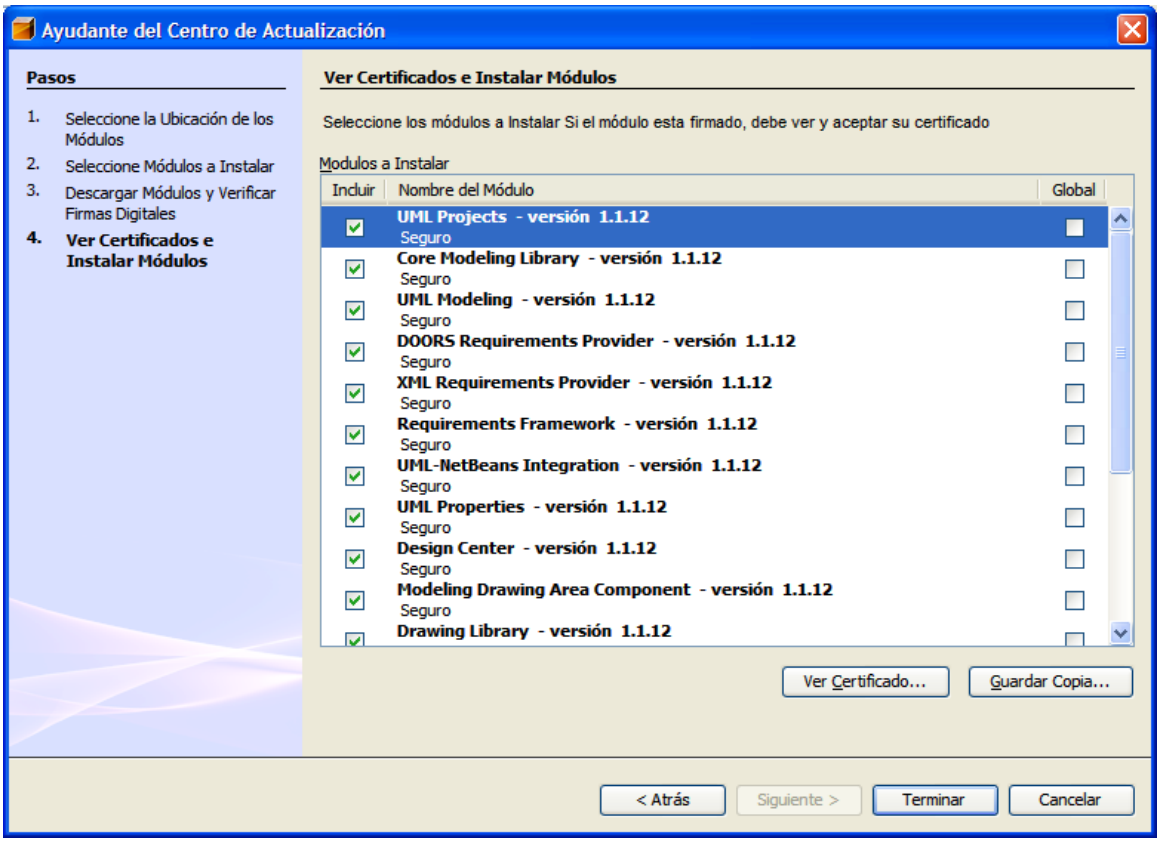

Figure 1.5

Figura 5. Comprobación de los módulos instalados

Una vez instalado UML Modelling, volvemos a "Herramientas-Gestor de módulos" y lo comprobamos que está bien instalado.

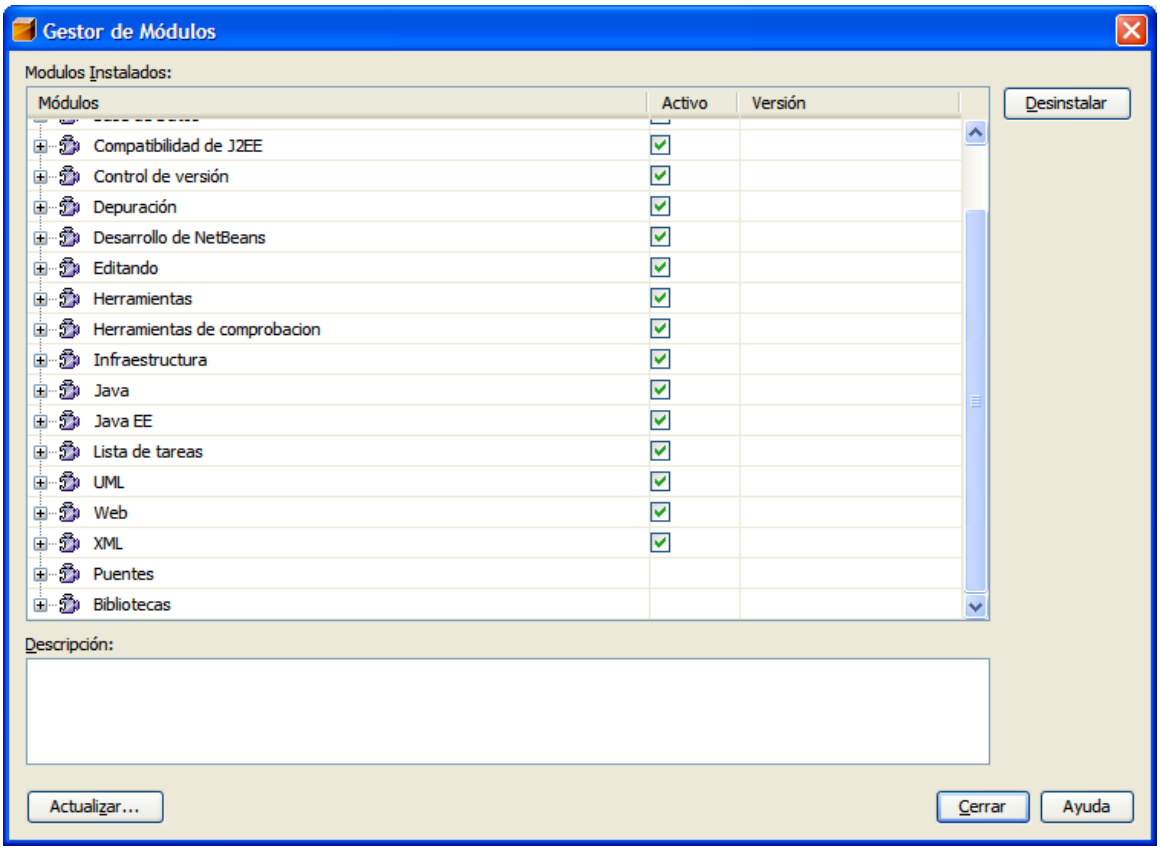

Figure 1.6

Figura 6. Comprobación de UML en el Gestor de módulos

1.1.2 Crear Diagrama de Uso con NetBeans 5.5

Creamos un nuevo proyecto, que denominamos UMLDemo. Para ello, vamos a: Archivo – Nuevo Proyecto:

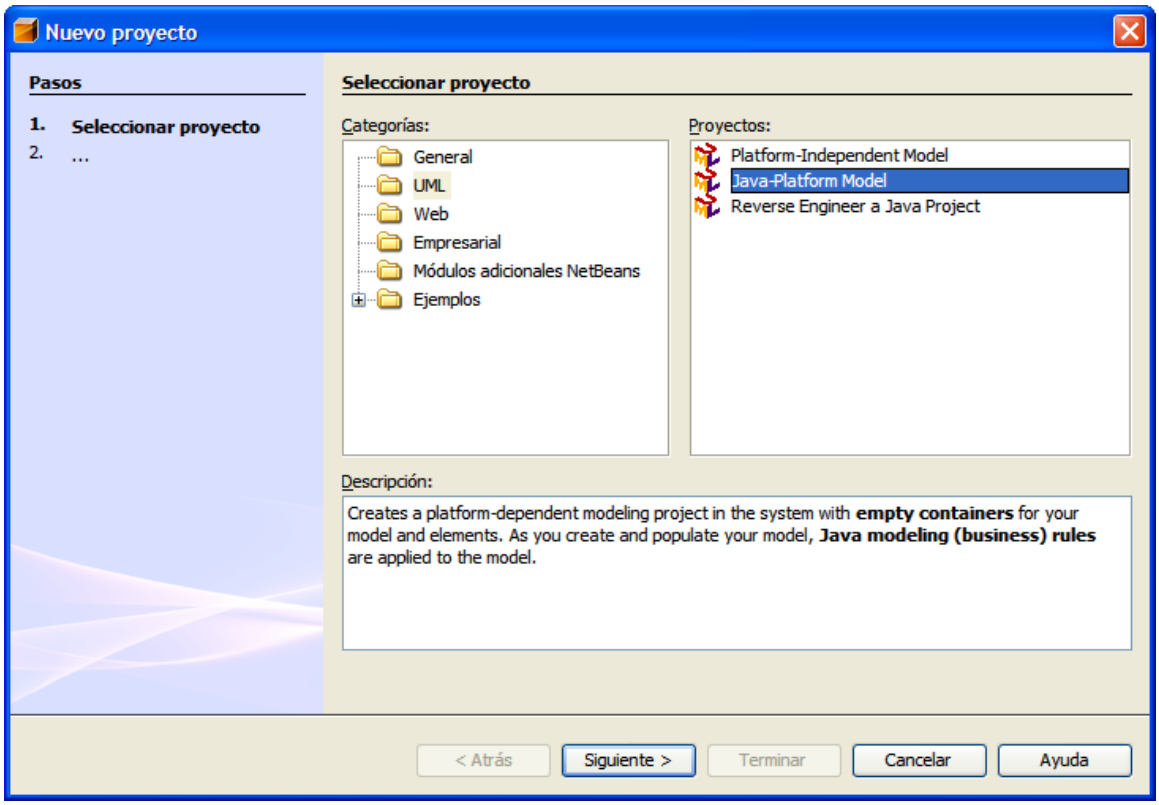

Figure 1.7

Figura 7. Realizamos un nuevo proyecto

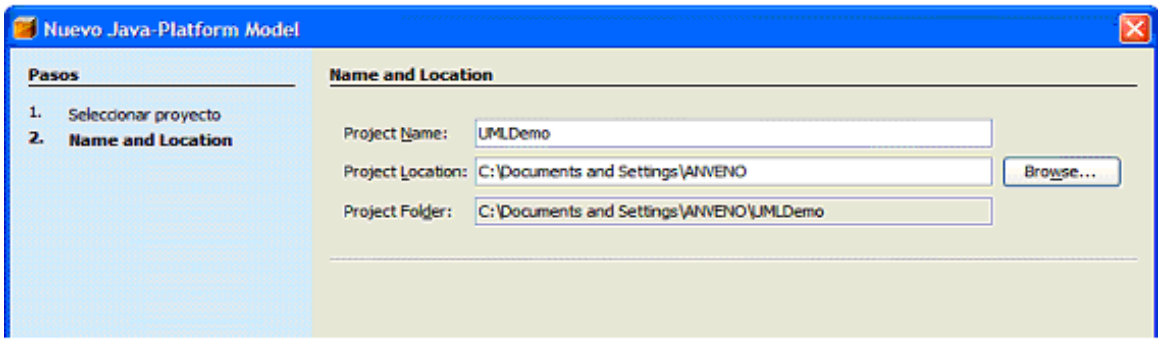

Figure 1.8

Figura 8. El proyecto se va a denominar UMLDemo

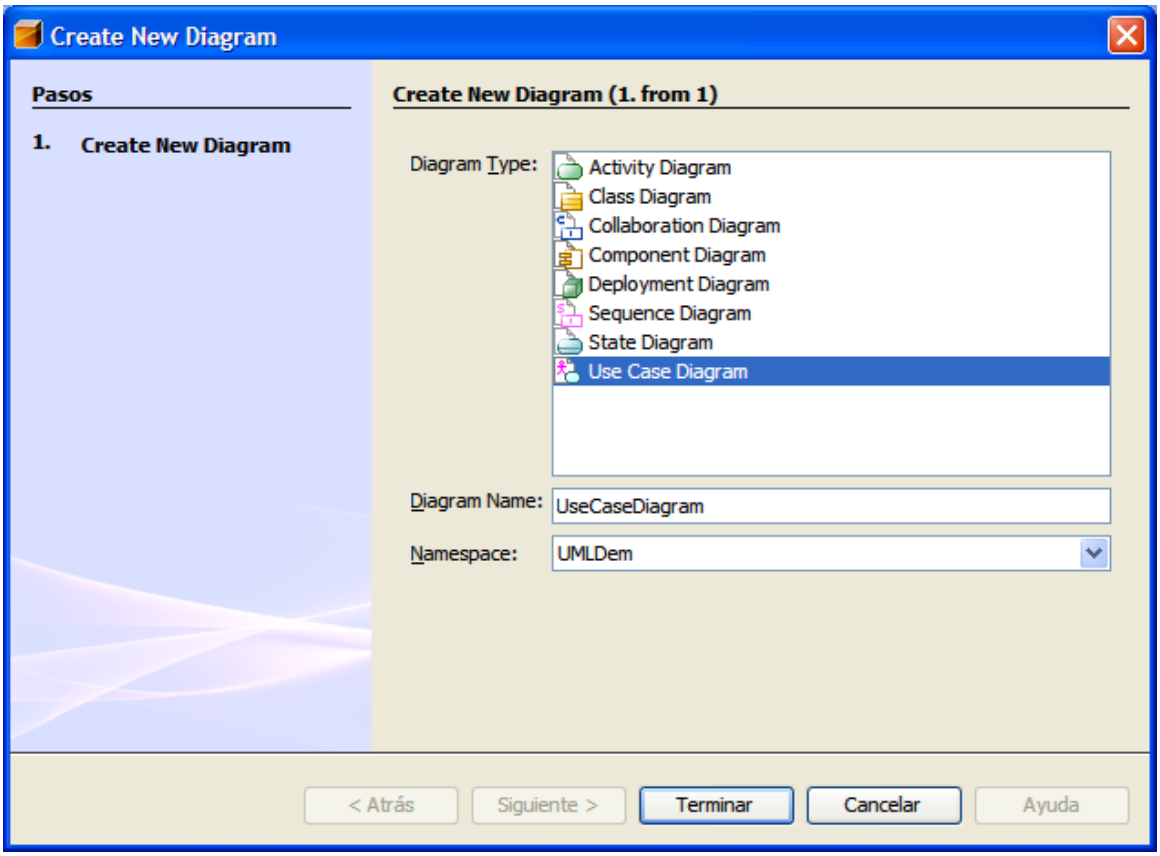

Figure 1.9

Figura 9. Seleccionamos el Diagrama de Caso de uso, cuyo nombre será UseCaseDiagram

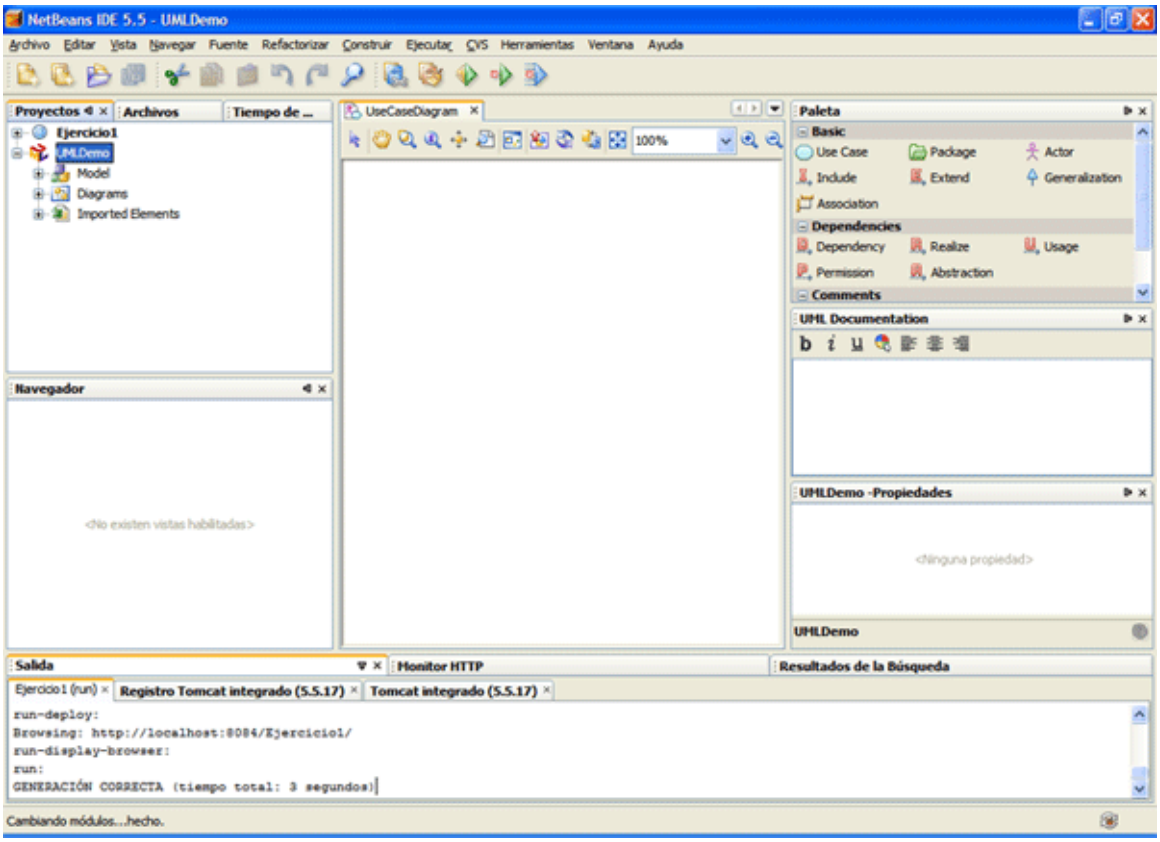

Figure 1.10

Figura 10. Pantalla Inicial para crear nuestro Diagrama de Caso de Uso
A continuación realizamos el siguiente diagrama de caso de uso:

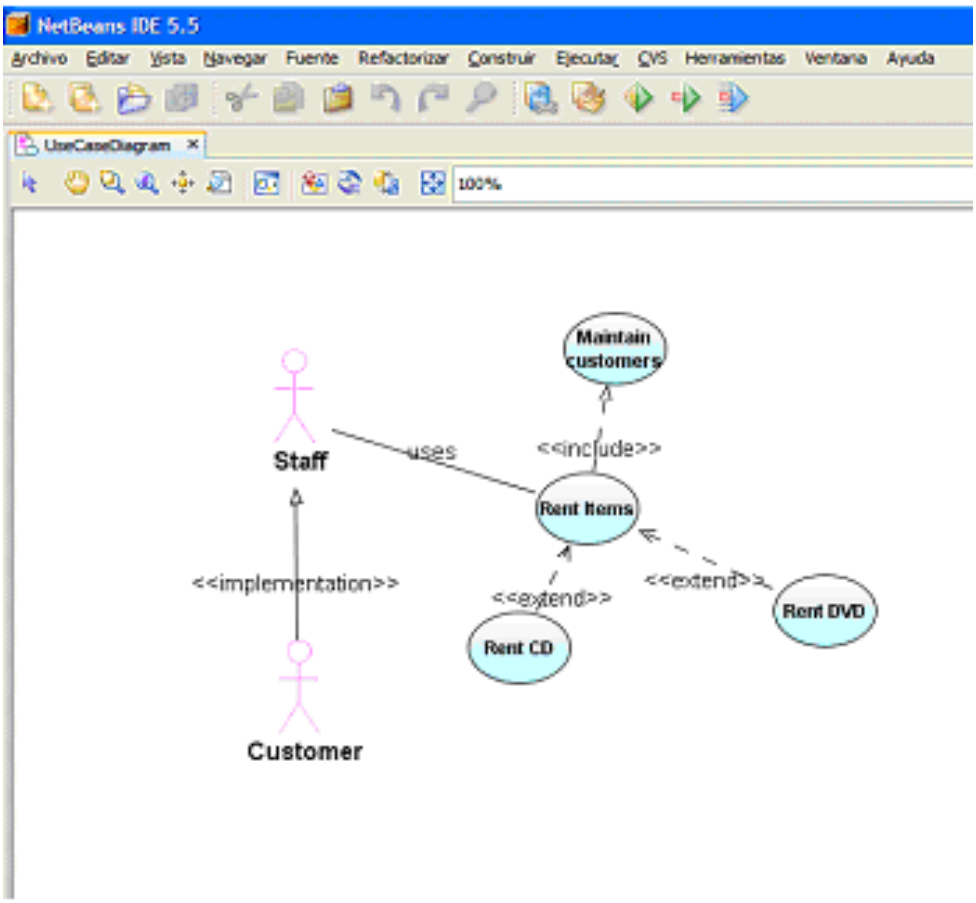

Figure 1.11

Figura 11. Ejemplo de un Diagrama de Caso de uso

1.1.3 Generar código con NetBeans 5.5

En nuestro ejemplo anterior (UMLDemo), vamos a realizar otro tipo de Diagrama, el Diagrama de clases.

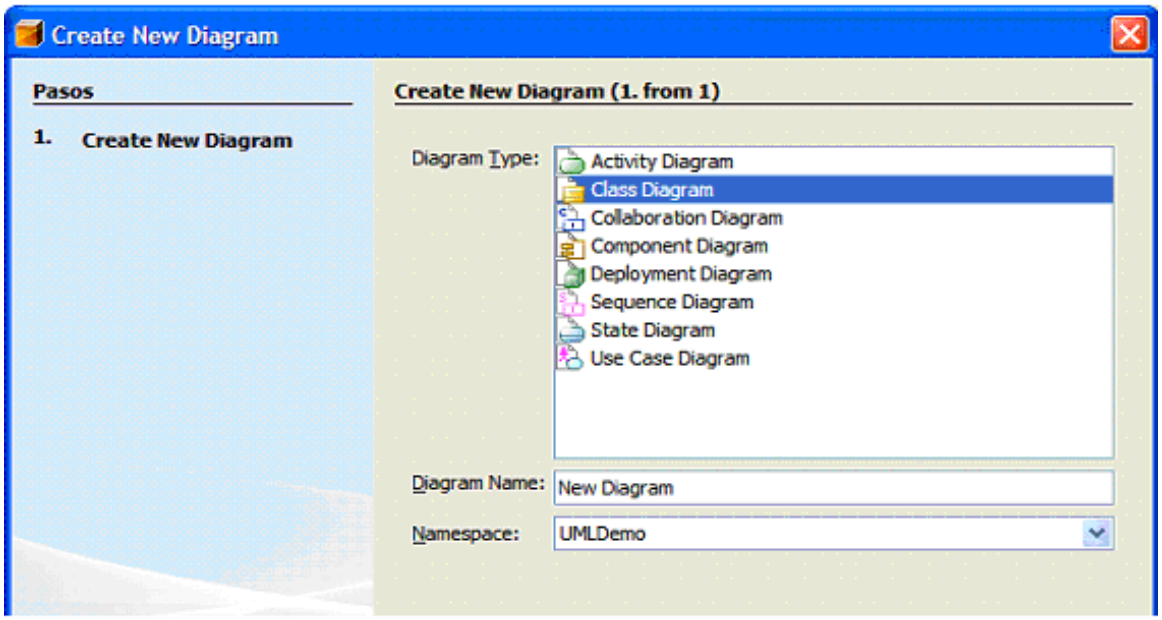

Figure 1.12

Figura 12. Crear Diagrama de clases
Y realizamos el siguiente diagrama, añadimos los atributos y operaciones:

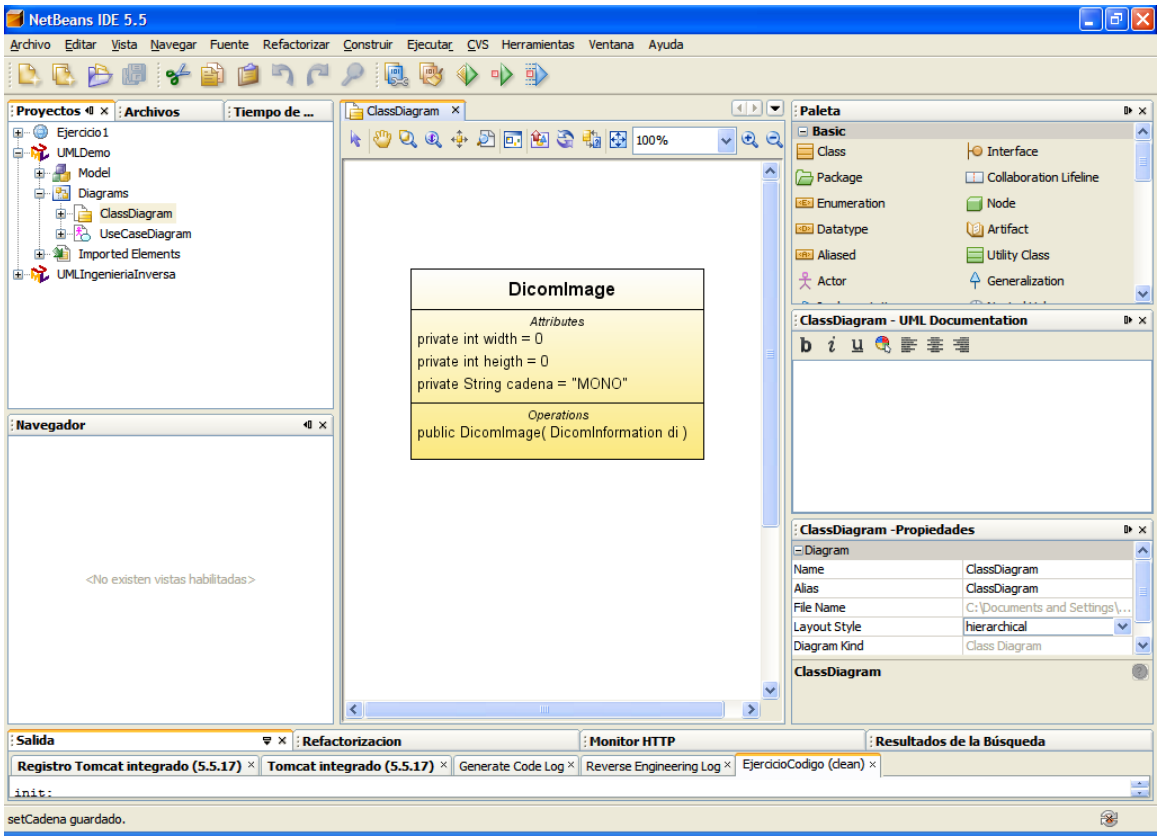

Figure 1.13

Figura 13. Ejemplo de Diagrama de clases

Creamos el proyecto, dónde vamos a generar el Código. A esto proyecto le vamos a llamar EjercicioCodigo:

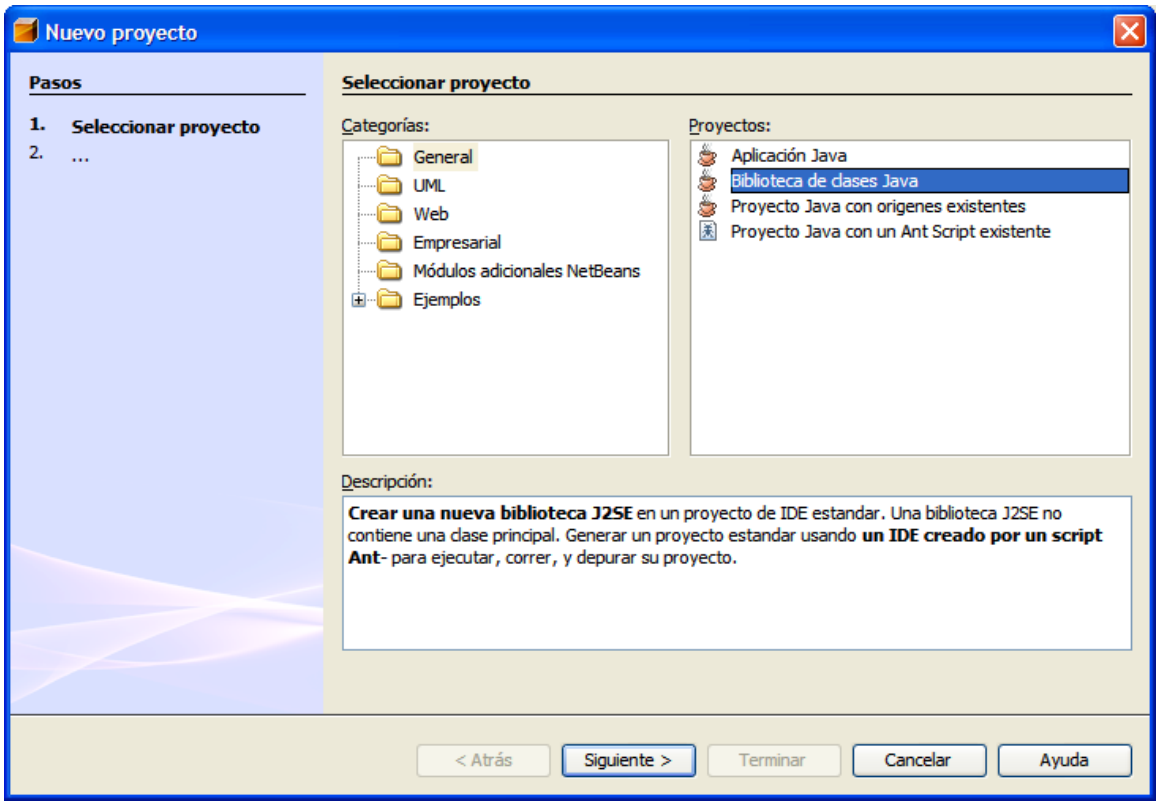

Figure 1.14

Figura 14. Crear proyecto Java

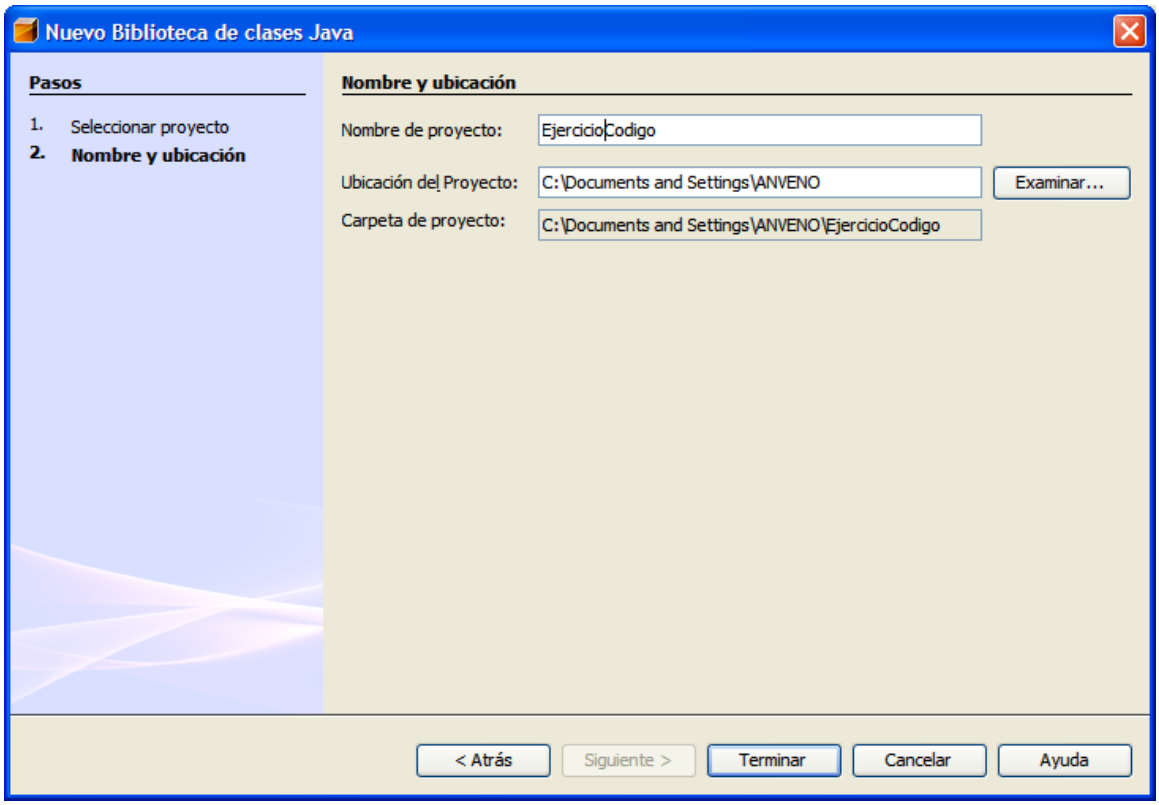

Figure 1.15

Figura 15. Nombre del Proyecto
A continuación, en UMLDemo, botón derecho, Generamos Código (Generate Code...)

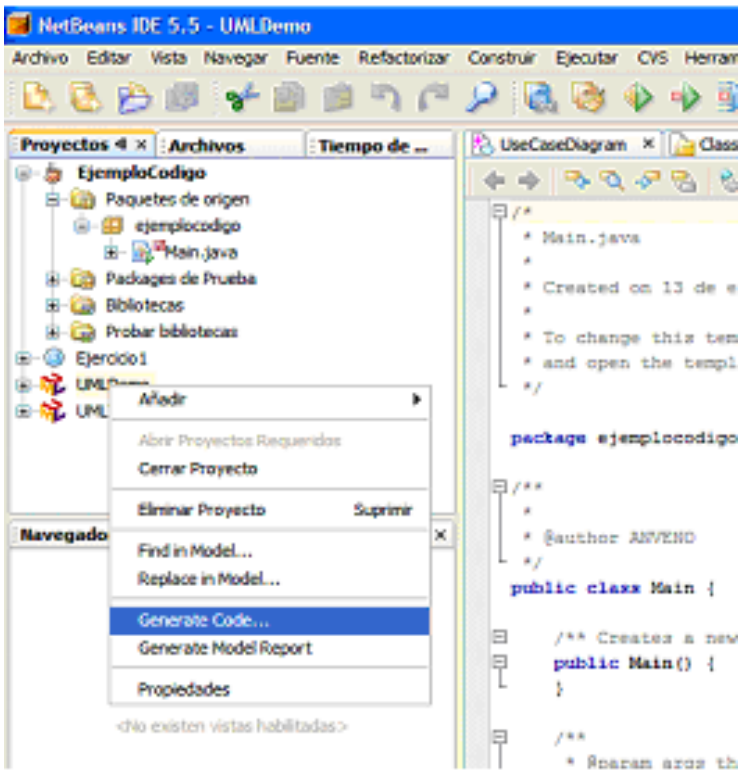

Figure 1.16

Figura 16. Nombre del Proyecto

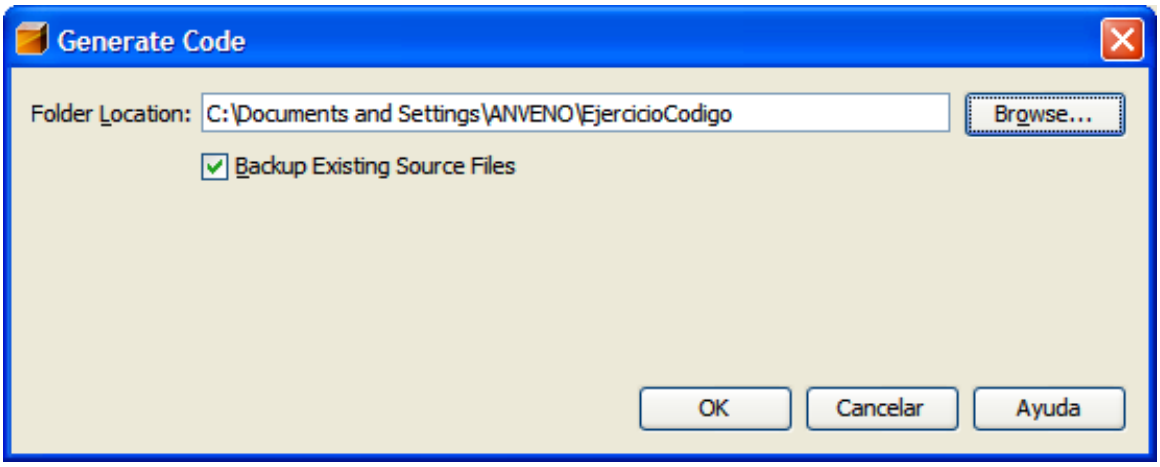

Figure 1.17

Figura 17. Ubicación del Código que vamos a generar
Y el código que se genera en EjercicioCodigo es el siguiente:

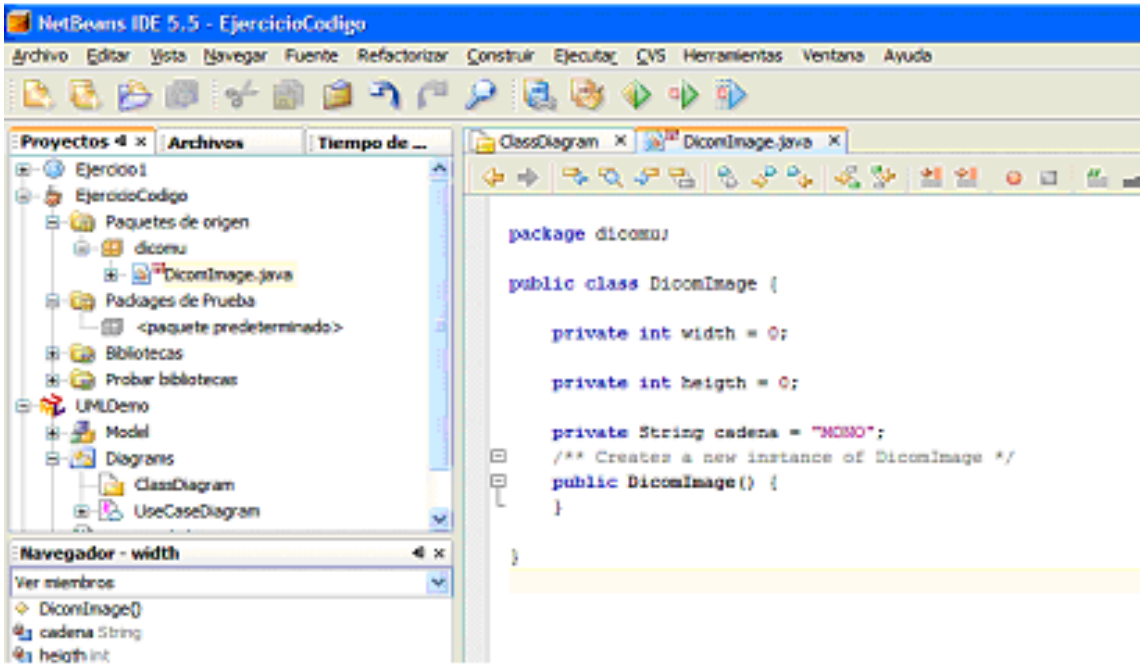

Figure 1.18

Figura 18. Código generado

1.1.4 Ingeniería Inversa con NetBeans 5.5

Vamos a realizar Ingeniería Inversa con NetBeans. Para ello creamos el siguiente proyecto:

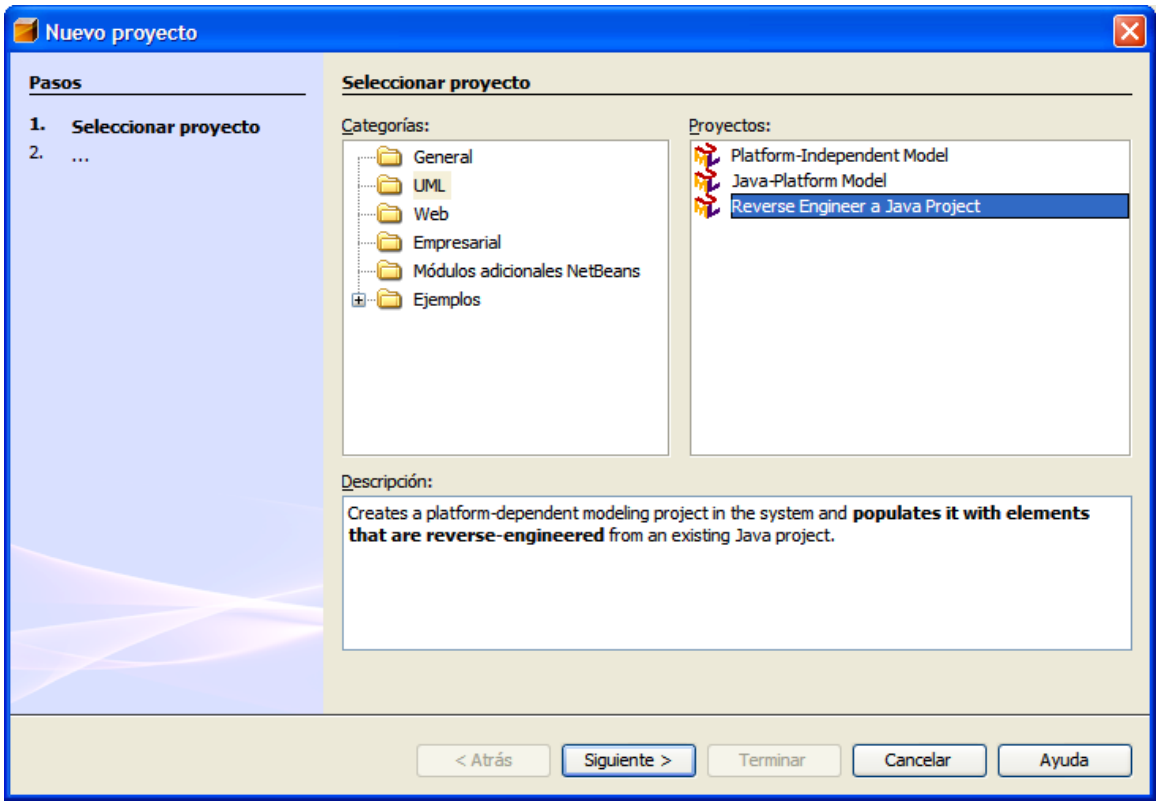

Figure 1.19

Figura 19. Creamos Proyecto para Ingeniería Inversa cuyo nombre será UMLIngenieriaInversa

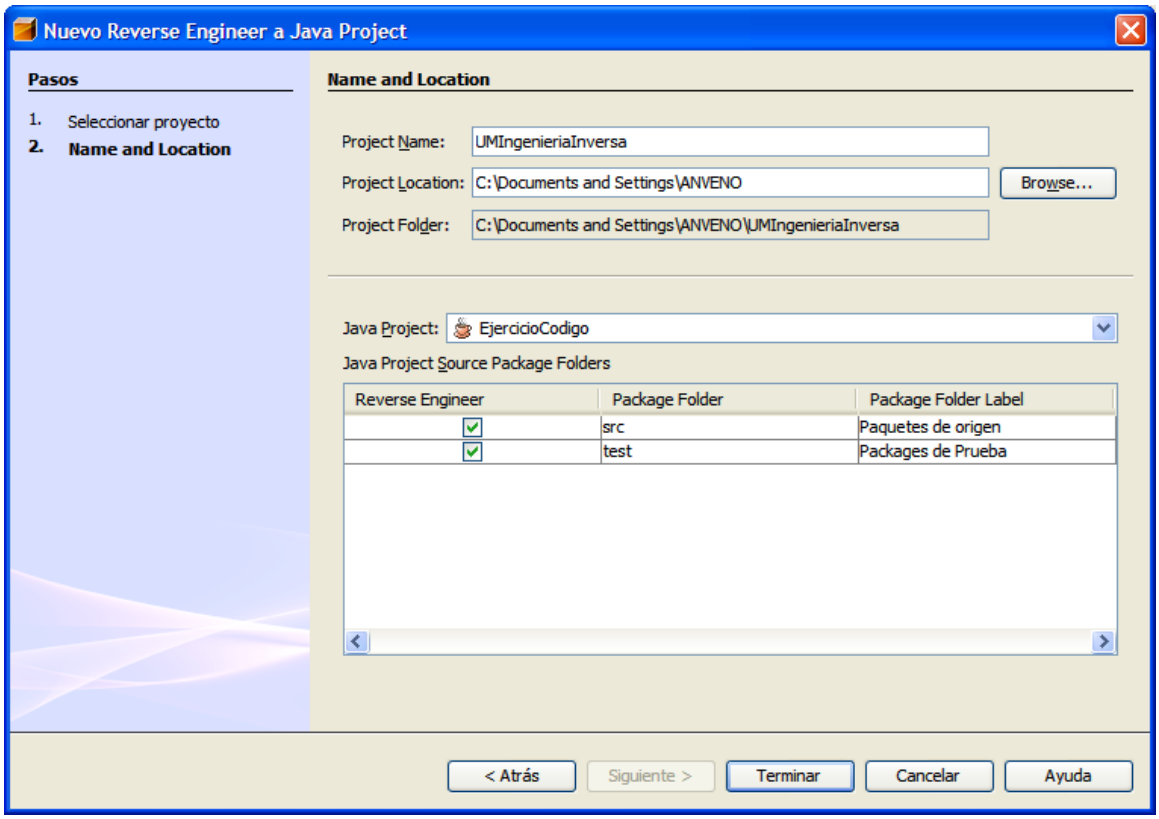

Figure 1.20

Figura 20. En Java Project ponemos nuestro Código Java

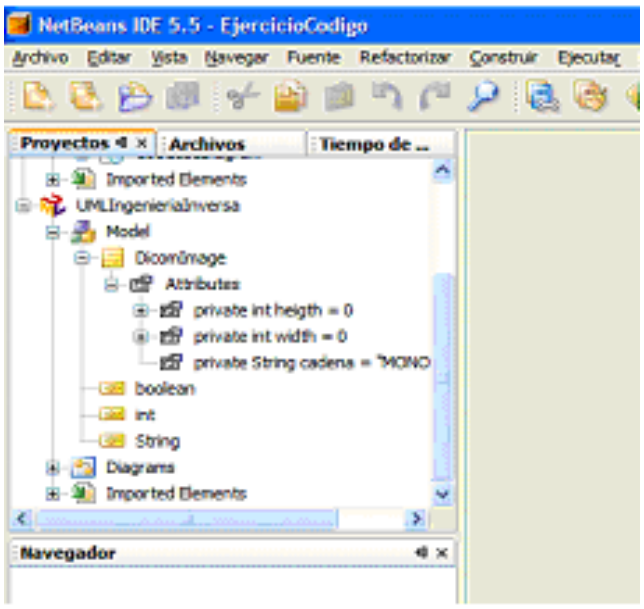

Figure 1.21

Figura 21. Estructura de nuestras carpetas
Seleccionamos nuestra carpeta DicomImage y botón derecho Create Diagram From Selected Elements

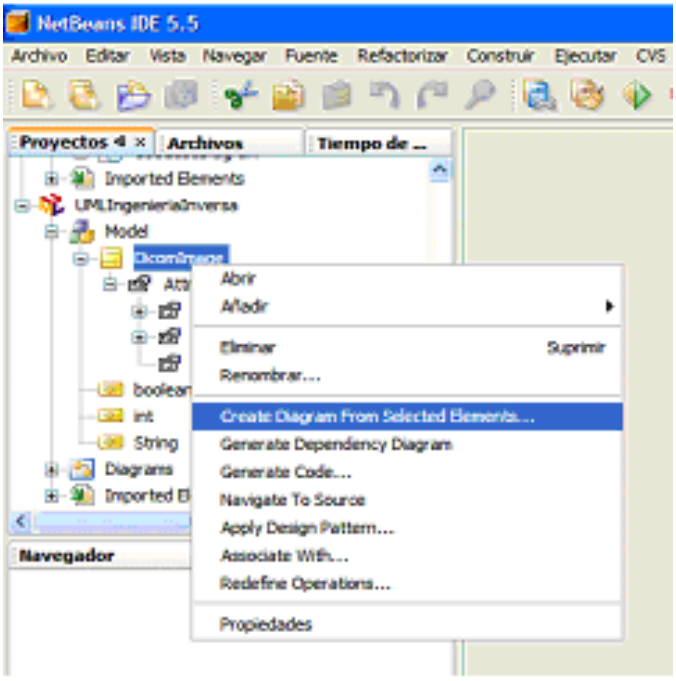

Figure 1.22

Figura 22. Seleccionamos la opción

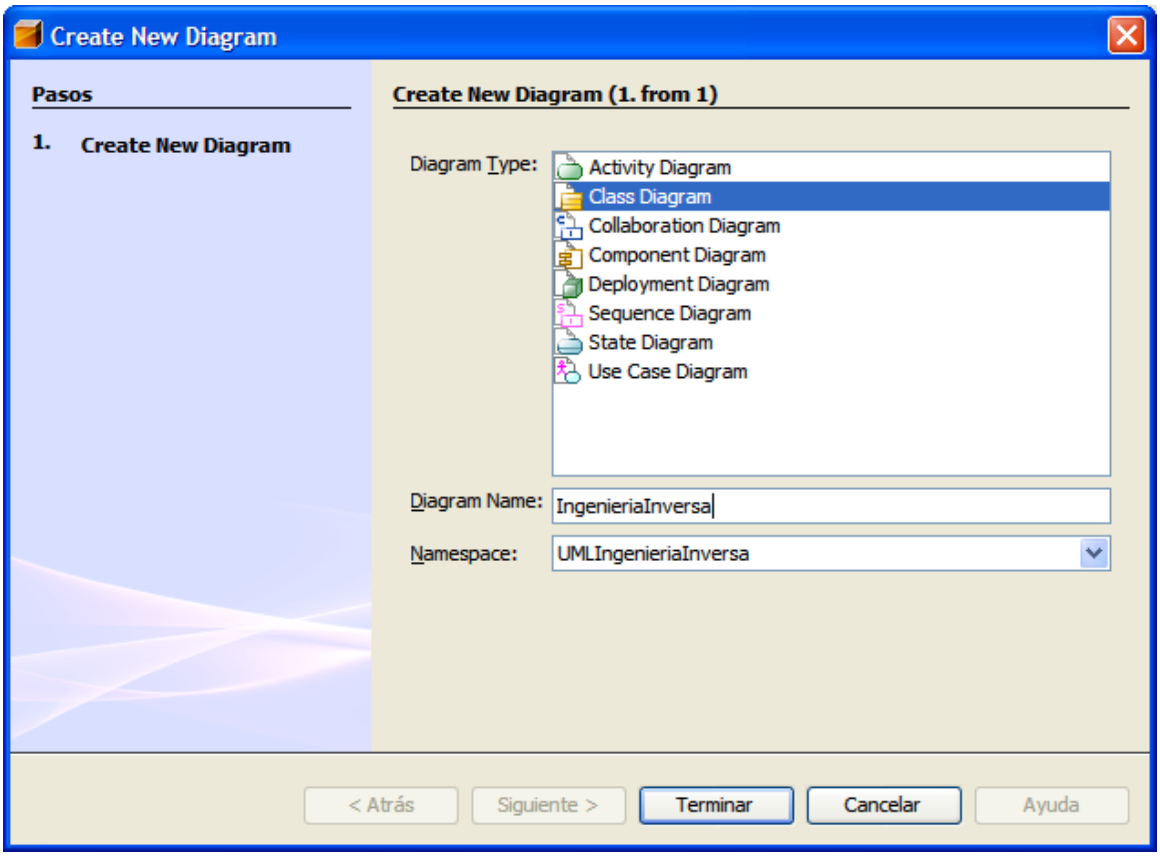

Figure 1.23

Figura 23. Elegimos el diagrama de Clases
Y obtenemos nuestro Diagrama de Clase.

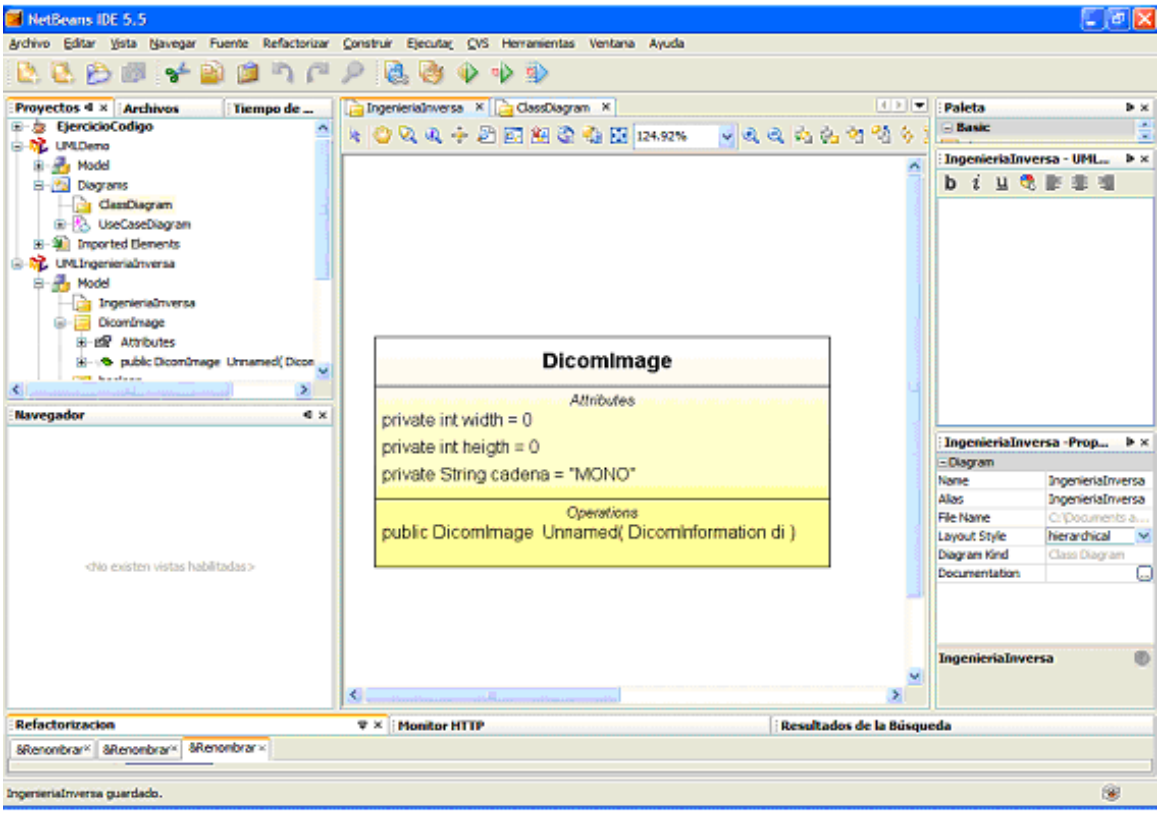

Figure 1.24

Figura 24. Realización de la Ingeniería Inversa

1.2 Ejemplo práctico de Reingeniería del Software

Un ejemplo sencillo para poder dar un panorama de lo que se trata reingeniería de software sería un sistema de un mini súper.

En este sistema se tienen:

- Bases de datos de los productos: en la cual viene toda la información de los productos en diferentes campos como precio de compra, precio de venta, descripción (nombre, marca, tipo, etc.), descuentos, inventario ,proveedor, clasificación, etc. Con 200 registros.
- Bases de datos de los trabajadores: aquí también existen diferentes campos por cada registro como salario, horario, descripción (nombre, dirección, teléfono, antigüedad, etc.), puesto, etc, con 20 registros.

Al paso del tiempo este sistema se ha ido modificando debido a que las demandas de la tienda fueron cambiando. Se modificaron elementos como: la creación de una base de datos especial para proveedores la cual tenía interacción con la de productos; se contrataron varias empresas de recursos humanos para la reclutación de su personal, por lo tanto se creó un registro de estas y se hizo una interacción entre estas

con los trabajadores; como había descuentos especiales en los productos para los trabajadores se hizo una interacción de la base de datos de trabajadores con la de productos. Aumentaron los trabajadores de 20 a 120, y los productos de 200 a 500.

Todas estas modificaciones se hicieron como se fue requiriendo, y por lo tanto solo se crearon parches en el sistema, lo que hizo que el sistema: se fuera alentando cada vez más, no tuviera una buena presentación, no fuera fácil su mantenimiento, etc.

Por lo tanto se tenía que modificar totalmente el sistema para una mejor manutención para lo cual utilizamos la reingeniería de software, ya que con esto no tendríamos que reconstruir el software totalmente sino que podíamos utilizarlo como base para la construcción de un software con reingeniería.

En primer lugar se decide traducir el código a una versión más actualizada del mismo lenguaje, esto con el fin de que nuestro programa sea más versátil, para esto se utilizo una herramienta para que la traducción se hiciera de manera automática. Utilizamos las herramientas de ingeniería inversa para poder extraer toda la información posible del software inicial para obtener el diseño del nuevo sistema, el cual tiene que tener las mismas funciones del anterior con la misma interfaz. Con el nuevo diseño y el código fuente, ya traducido a un lenguaje adecuado, se empezó a reestructurar el programa para que la rapidez en sus funciones fuera óptima, para esto se necesitaba que existiera una buena modulación y así se empezó a establecer módulos de los pequeños programas que se interrelacionaran, para conseguir tener varios módulos con fines similares, ya estando bien la modulación sería más fácil poder implantar una estructura adecuada. Finalmente se hizo la depuración de los datos, creando una base de datos general con apuntadores a datos, con una buena clasificación e interacción entre estos. Al relacionar los programas con nuestra base de datos generada tendremos como resultado un nuevo sistema creado con reingeniería el cual va a ser más rápido, más mantenible, actualizado, y con capacidad de crecimiento sin estropear la arquitectura original.

Con esto tendremos una optimización de costes y eficiencia.

Chapter 2

Refactoring con Netbeans[1]

Refactoring es el proceso de llevar a cabo cambios en su código sin afectar el comportamiento de la aplicación.

2.1 Caso práctico de Refactoring

Tengo una aplicación bancaria que ya fue correctamente implementada, esta aplicación usa extensivamente una clase llamada Transacción, la cual posee una operación cuya firma es:

public void asignarCtaCorriente(CuentaCorriente cc);

Esta operación es invocada una gran cantidad de veces por distintas operaciones en nuestra aplicación.

El programa es que se debe cambiar la firma de la operación para homogenizar el estándar de nomenclatura establecido por la empresa que desarrolló la aplicación. La nueva firma debería ser:

public void asignarCuentaCorriente(CuentaCorriente cc);

¿Cuál es la solución? Debemos iniciar el proceso de refactoring en nuestra aplicación para reflejar este cambio. Modificaremos la firma en la clase Transacción y luego modificar en todo el proyecto cada una de las invocaciones.

Si la aplicación es realmente grande, implicará un gran esfuerzo llevar a cabo este cambio manualmente. Probablemente se introduzcan errores en la aplicación.

2.2 Refactoring en NetBeans 5.5

A medida que se desarrolla una aplicación con NetBeans, de forma transparente para el programador se genera un repositorio de metadata del código que el programador genera. La metadata se podría definir brevemente como "datos que describen otros datos", por lo cual NetBeans puede contar con esta información para realizar cambios sobre los fuentes automáticamente. Gracias a este repositorio de metadata que NetBeans mantiene es posible automatizar muchas tareas de refactoring, por ejemplo, en el caso que describimos anteriormente, NetBeans se encarga de cambiar la firma de la operación de la clase Cuenta, buscar todas las invocaciones y modificarlas. Adicionalmente NetBeans provee un informe del impacto que tendrá el proceso de refactoring, para permitir al programador tomar decisiones respecto a este proceso. Luego de desplegado dicho informe se puede iniciar el proceso de refactoring de acuerdo a las decisiones tomadas por el programador.

2.3 Menú de Refactoring en NetBeans

Teniendo las siguientes clases de Java en nuestro proyecto NetBeans (proyecto llamado PruebaCirculo):

Circulo.java

[1] This content is available online at <http://cnx.org/content/m17586/1.3/>.

```
public class Circulo {

  private int x;

  private int y;

  private double radio;

   public Circulo()

  {

  }

  // constructor

  public Circulo( int valorX, int valorY, double valorRadio )

  {

     x = valorX;

     y = valorY;

     establecerRadio( valorRadio );

  }

  public void establecerX( int valorX )

  {

      x = valorX;

  }

  public int obtenerX()

  {

      return x;

  }

  public void establecerY( int valorY )
```

```
{
    y = valorY;
}
public int obtenerY()
{
    return y;
}
public void establecerRadio( double valorRadio )
{
    radio = ( valorRadio < 0.0 ? 0.0 : valorRadio );
}
public double obtenerRadio()
{
    return radio;
}
public double obtenerDiametro()
{
    return 2 * radio;
}
public double obtenerCircunferencia()
{
    return Math.PI * obtenerDiametro();
}
```

```
    public double obtenerArea()
    {
        return Math.PI * radio * radio;
    }
    public String toString()
    {
        return "Centro = [" + x + ", " + y + "]; Radio = " + radio;
    }
```

PruebaCirculo.java

```
import java.text.DecimalFormat;
import javax.swing.JOptionPane;
public class PruebaCirculo {
  public static void main( String[] args )
      Circulo circulo = new Circulo( 37, 43, 2.5 );
      String salida = "La coordenada X es " + circulo.obtenerX() +
        "\nLa coordenada Y es " + circulo.obtenerY() +
        "\nEl radio es " + circulo.obtenerRadio();
      circulo.establecerX( 35 );
      circulo.establecerY( 20 );
      circulo.establecerRadio( 4.25 );
      salida += "\n\nLa nueva ubicación y el radio del círculo son\n" +
        circulo.toString();
      DecimalFormat dosDigitos = new DecimalFormat( "0.00" );
      salida += "\nEl diámetro es " +
```

```
        dosDigitos.format( circulo.obtenerDiametro() );

      salida += "\nLa circunferencia es " +

        dosDigitos.format( circulo.obtenerCircunferencia() );

      salida += "\nEl área es " + dosDigitos.format( circulo.obtenerArea() );

      JOptionPane.showMessageDialog( null, salida );

      System.exit( 0 );

   } // fin de main

} // fin de la clase PruebaCirculo
```

La estructura de nuestras carpetas en el proyecto NetBeans será la siguiente:

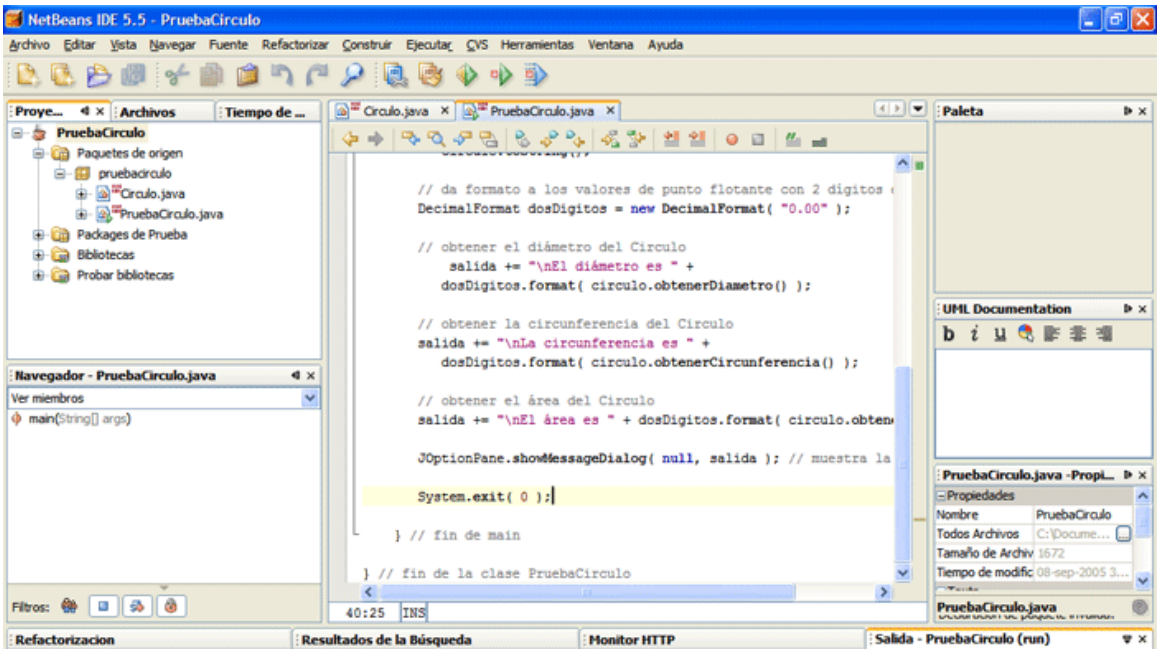

Figure 2.1

Figura 1. Visión del proyecto NetBeans con las clases java

Si nos situamos en cualquiera de nuestras clases y botón derecho, obtenemos el siguiente menú, donde podemos observar que tenemos la opción refactorizar:

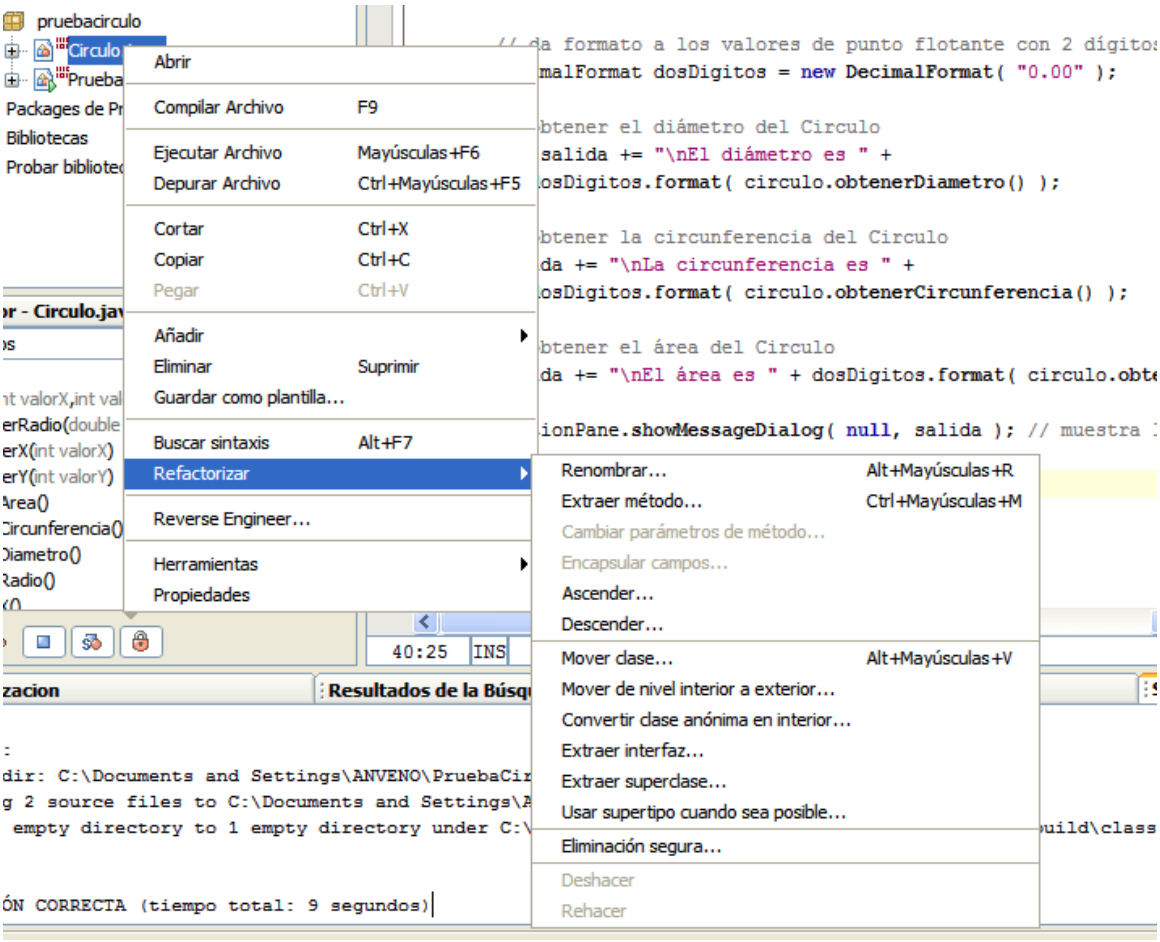

Figure 2.2

Figura 2. Menú de Refactorizar

A continuación, se encuentra una breve descripción de las diferentes opciones del menú:

- Renombrar → Cambia el nombre de una clase, variable o método. Adicionalmente modifica todo el código del proyecto para referenciar al nuevo nombre.

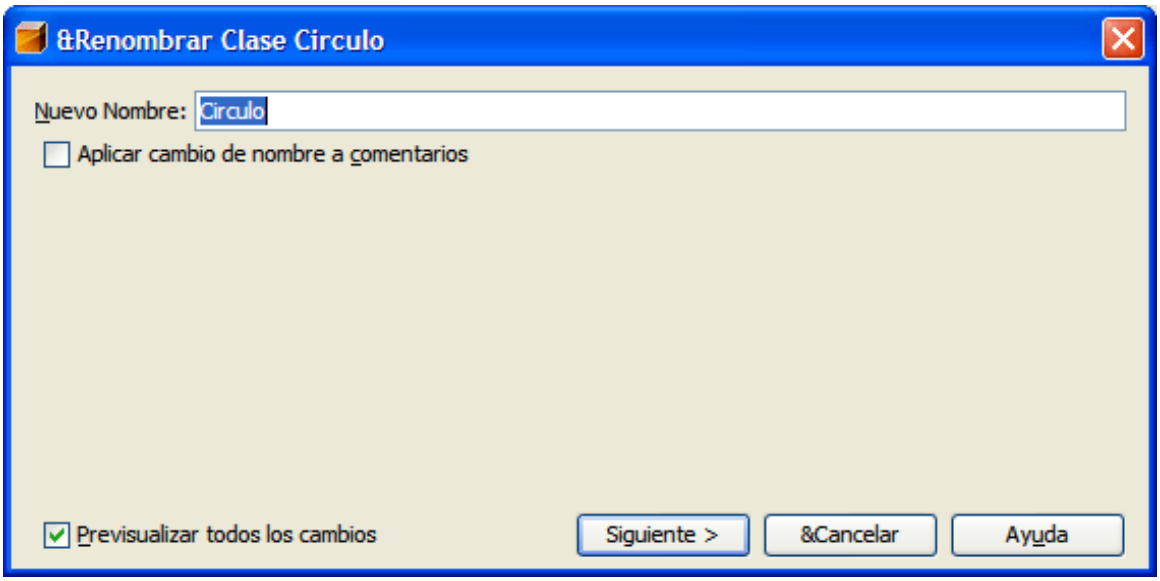

Figure 2.3

Figura 3. Opción renombrar

Cambiamos el nombre Circulo por Circulitos y observamos los cambios en el código que donde ponía Circulo ahora pone Circulitos:

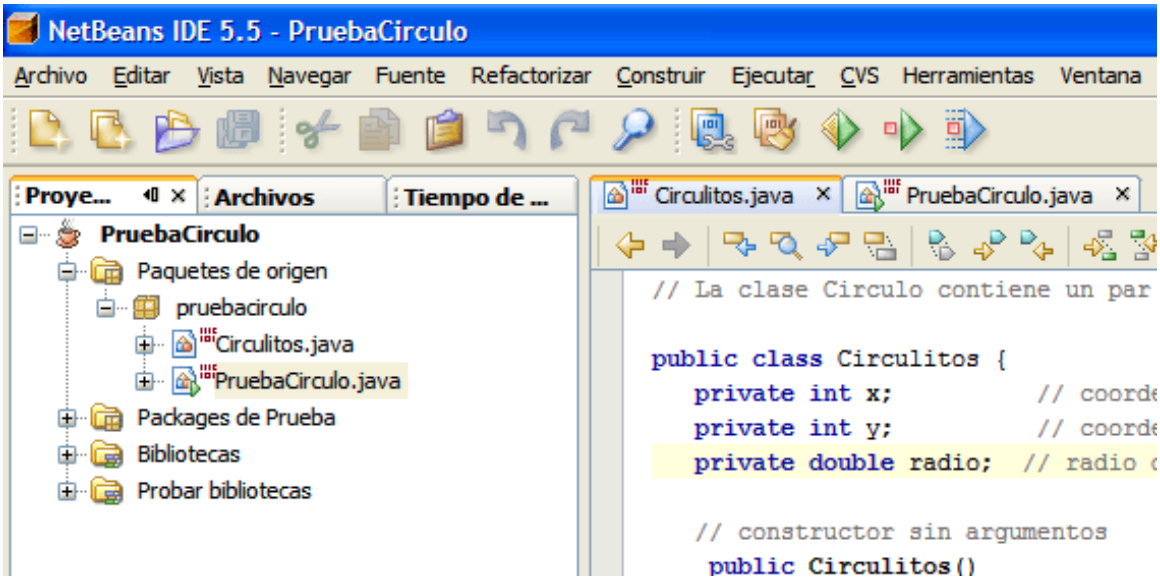

Figure 2.4

Figura 4. Cambios al aplicar Renombrar

- Cambiar parámetros de un método → Permite agregar, eliminar, modificar o cambiar el orden de los parámetros de un método, al igual que su modificador de acceso (private o public).
- Encapsular campos → Es muy común tener que acceder a los campos de una clase por medio de operaciones del tipo:

public set<nombre del campo>(...)
public get<nombre del campo>()

Es una tarea muy tediosa, por lo cual esta operación permite que el programador solo deba implementar los campos, delegando a NetBeans la tarea de "encapsularlos". También es posible que todo código del proyecto que accede directamente al campo, pase automáticamente a utilizar el "setter" o el "getter" determinado.

- Ascender métodos o campos → Permite subir un método o campo a otra clase de la cual hereda la clase que contiene al método o campo que deseamos subir.
- Descender clases anidadas, métodos o campos → Permite bajar una clase anidada, método o campo a otra clase la cual hereda de la clase que contiene a la clase anidada, método o campo que deseamos bajar.
- Mover una clase → Mueve una clase a otro package o dentro de otra clase. Adicionalmente modifica todo el código del proyecto para referenciar al nuevo lugar donde se movió la clase.

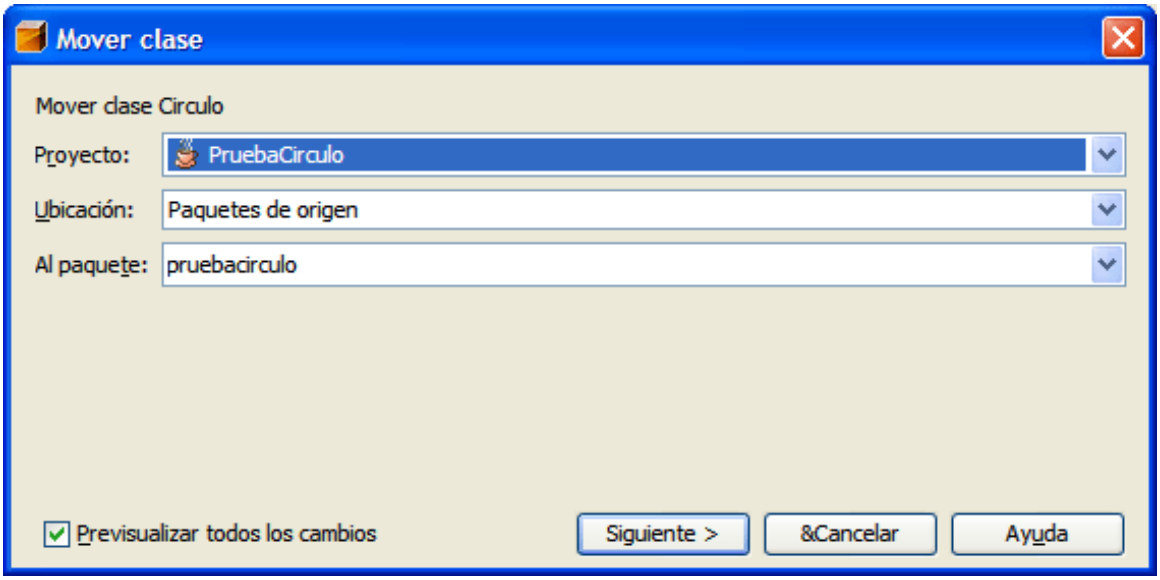

Figure 2.5

Figura 5. Opción Mover clase

- Convertir una clase anónima anidada a una clase anidada → Crea una nueva clase anidada, la cual tendrá un nombre y un constructor. La clase anónima anidada será sustituida por esta nueva clase anidada.

- Extraer una interfase → Permite seleccionar cuales métodos públicos no estáticos de una clase o interfase, irán a parar a una nueva interfase. La clase de la cual fue extraída la interfase implementará la nueva interfase creada. La interfase de la cual fue extraída la interfase extenderá la nueva interfase.

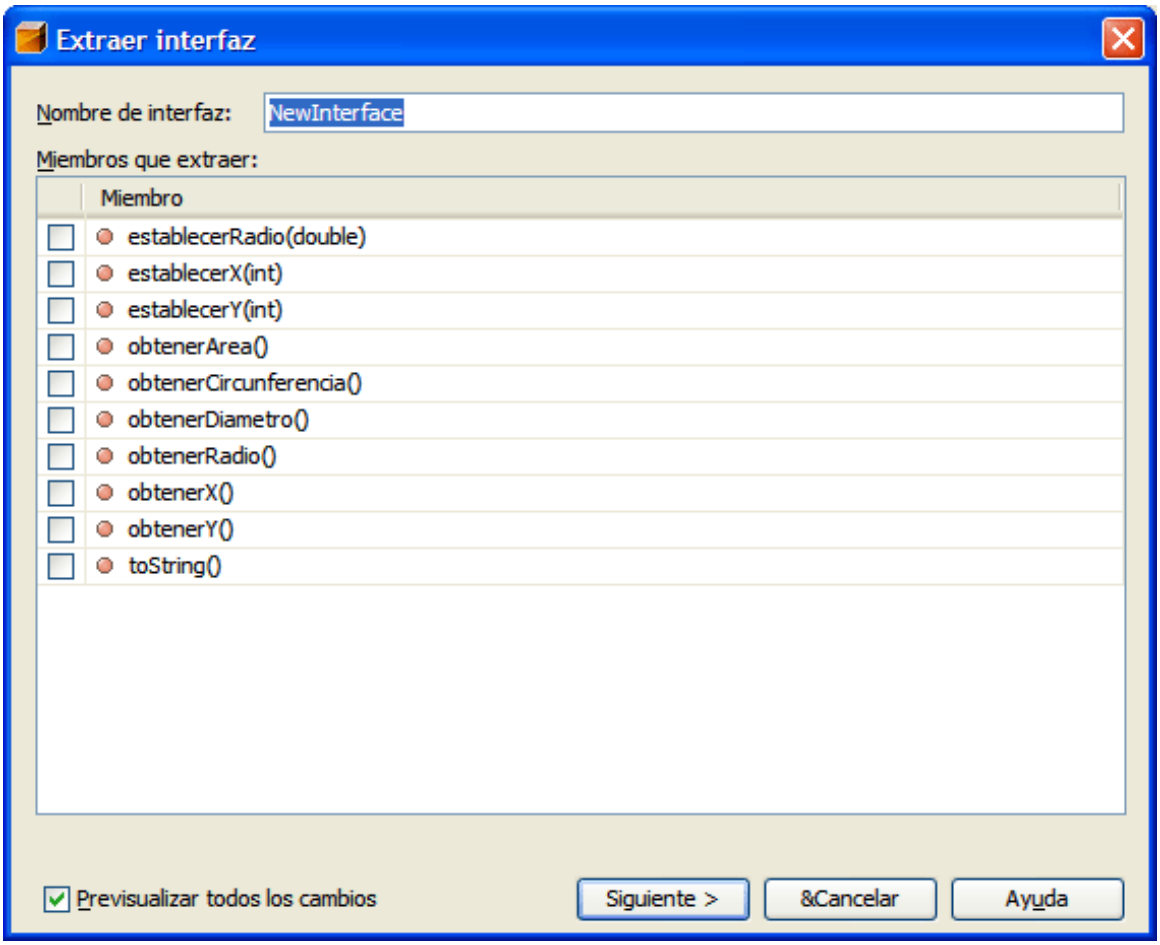

Figure 2.6

Figura 6. Opción Extraer interfaz

- Extraer una superclase → Despliega al programador los métodos y campos que se pueden mover a una superclase. El programador selecciona cuales desea mover y NetBeans creará una nueva clase abstracta que contendrá dichos campos y métodos, también hará que la clase refactorizada la extienda.

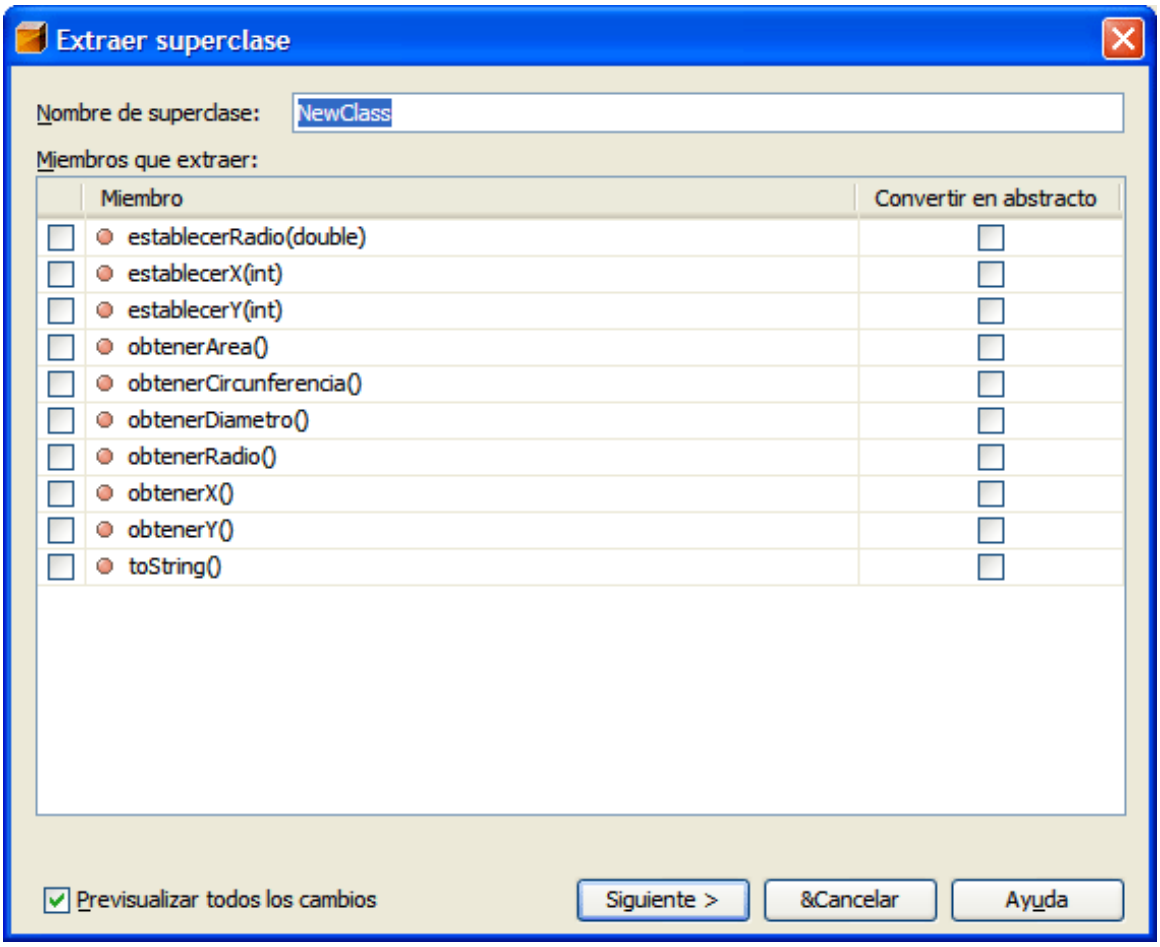

Figure 2.7

Figura 7. Opción Extraer superclase

- Usar supertipo cuando sea posible → Despliega al programador todas las clases que extiende la clase actual. El programador seleccionará una, y NetBeans buscará en todo el proyecto referencias a la clase que se quiere refactorizar, si encuentra referencias, determinará si es posible utilizar la superclase seleccionada.

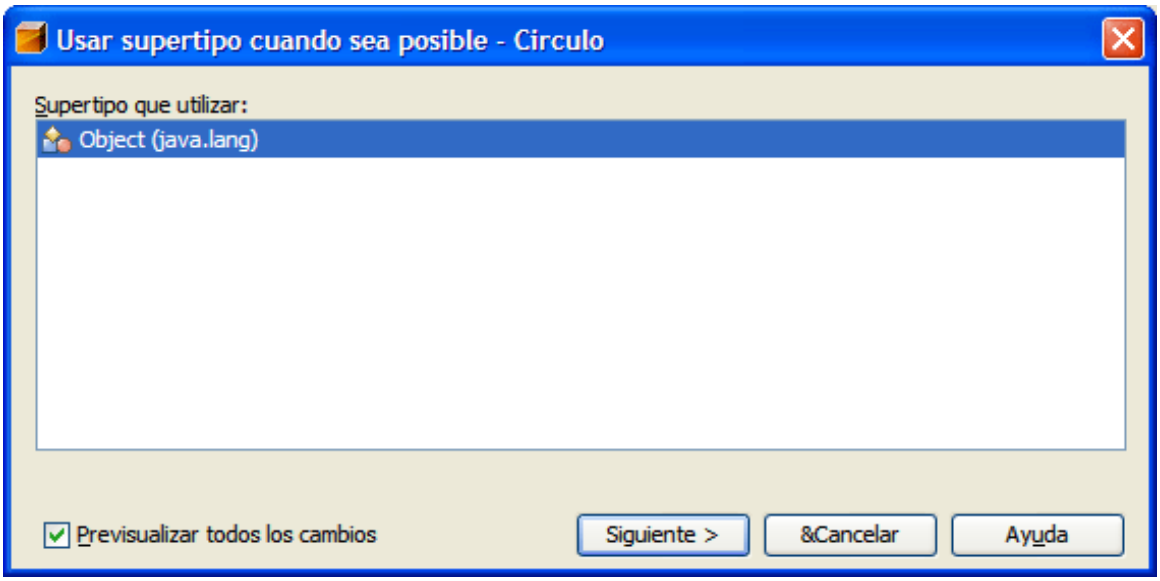

Figure 2.8

Figura 8. Opción Usar supertipo cuando sea posible

- Eliminación seguro → Cuando eliminamos un método o clase, debemos garantizarnos de que nadie más lo utilice en el proyecto. Esta operación verifica y notifica las referencias encontradas, proveyendo de mecanismos para que fácilmente el programador pueda eliminar una a una las referencias, para finalmente llevar a cabo la operación de borrado de modo seguro.

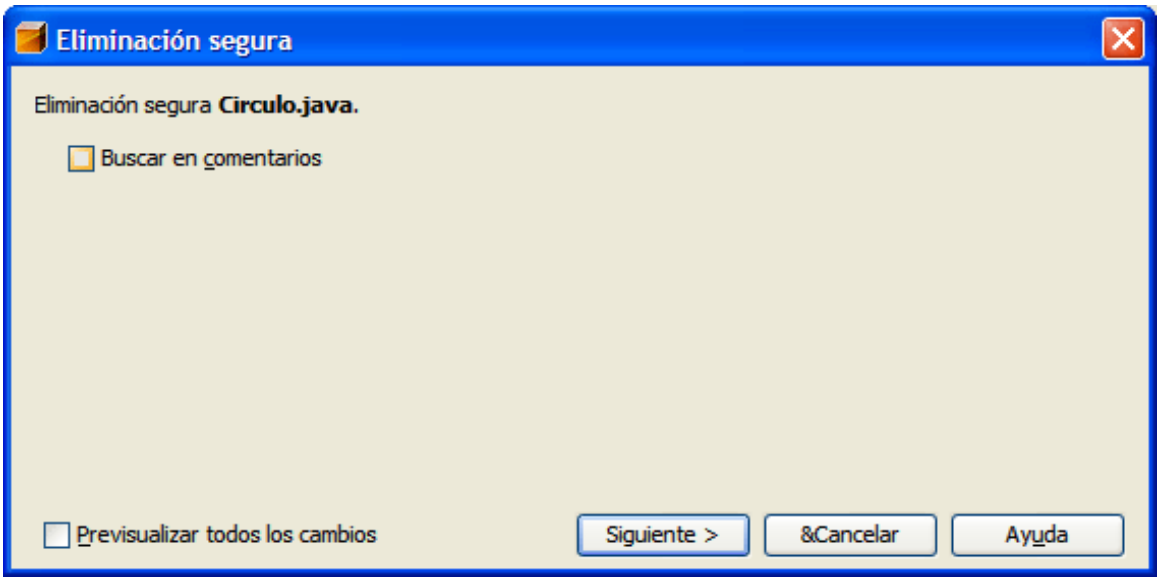

Figure 2.9

Figura 9. Opción Eliminación segura

- Introducir constantes, variables, campos o métodos → El programador selecciona un fragmento de código, y luego presiona las teclas <alt>+<Enter>. NetBeans desplegará varias opciones útiles, como encapsular ese fragmento en un método y referenciar al método, anidarse en while, if, etc.

Siempre que sea posible, utilizar las herramientas de refactoring que NetBeans provee, de este modo no solo será mucho más sencillo este tipo de procedimiento, sino que además ejecutaremos un proceso seguro mediante el cual no introduciremos errores humanos.

Index of Keywords and Terms

Keywords are listed by the section with that keyword (page numbers are in parentheses). Keywords do not necessarily appear in the text of the page. They are merely associated with that section. *Ex.* apples, § 1.1 (1) **Terms** are referenced by the page they appear on. *Ex.* apples, 1

Attributions

Collection: *Herramientas para el Mantenimiento del Software*
Edited by: Miguel-Angel Sicilia
URL: http://cnx.org/content/col10584/1.8/
License: http://creativecommons.org/licenses/by/2.0/

Module: "Reingeniería con Netbeans"
By: Miguel-Angel Sicilia
URL: http://cnx.org/content/m17590/1.5/
Pages: 1-24
Copyright: Miguel-Angel Sicilia, Verónica De la Morena
License: http://creativecommons.org/licenses/by/2.0/

Module: "Refactoring con Netbeans"
By: Miguel-Angel Sicilia
URL: http://cnx.org/content/m17586/1.3/
Pages: 25-36
Copyright: Miguel-Angel Sicilia, Verónica De la Morena
License: http://creativecommons.org/licenses/by/2.0/

Herramientas para el Mantenimiento del Software
Casos prácticos de Refactoring y Reingeniería con Netbeans.

About Connexions
Since 1999, Connexions has been pioneering a global system where anyone can create course materials and make them fully accessible and easily reusable free of charge. We are a Web-based authoring, teaching and learning environment open to anyone interested in education, including students, teachers, professors and lifelong learners. We connect ideas and facilitate educational communities.

Connexions's modular, interactive courses are in use worldwide by universities, community colleges, K-12 schools, distance learners, and lifelong learners. Connexions materials are in many languages, including English, Spanish, Chinese, Japanese, Italian, Vietnamese, French, Portuguese, and Thai. Connexions is part of an exciting new information distribution system that allows for **Print on Demand Books**. Connexions has partnered with innovative on-demand publisher QOOP to accelerate the delivery of printed course materials and textbooks into classrooms worldwide at lower prices than traditional academic publishers.