LABORATORIO

**CURSO** : ***Algorítmica II***

**TEMA** : *Polimorfismo y Sobrecarga – Aseguradora.*

**DURACIÓN ESTIMADA** : *02:40 horas.*

#### OBJETIVOS

El presente laboratorio tiene por objetivos:

* Utilizar el IDE de Netbeans
* Crear un proyecto para un caso de negocio.
* Construir y ejecutar una aplicación con múltiples clases.

#### RESUMEN

En esta práctica usted explorara un proyecto completo en Netbeans con múltiples clases y polimorfismo. Para ello desarrollaremos parcialmente el caso de negocio de BookStore, enfocándonos en la administración de empleados.

#### PROBLEMÁTICA

Se tiene una empresa aseguradora en la cual se maneja diferentes tipos de seguros (seguro de vivienda y médico). El seguro consta de un número de póliza, nombre del asegurado, valor de la prima que se paga, comisión de la aseguradora (0.015% del valor de la prima) y finalmente el valor de la cobertura de acuerdo a lo que se paga si se ejecuta el seguro. En el seguro médico se sigue la siguiente tabla para el cálculo de la cobertura:

|  |  |
| --- | --- |
| **Valor de la Prima** | **Cobertura** |
| 20 | 5000 |
| 40 | 10000 |
| 60 | 20000 |
| 80 | 40000 |
| 100 | 80000 |

El seguro médico toma en cuenta la edad a la cual inicio su seguro el cliente para definir el monto de la prima a abonar; con la siguiente regla que si el asegurado es menor de 40 años se aplica los montos antes mencionados; si el asegurado es mayor ó igual a 40 años y menor de 60 años, el valor de la prima es el doble de la prima normal para la misma cobertura; y si es mayor ó igual a 60 años, el valor de la prima es el triple de la prima normal para la misma cobertura.

En el seguro de vivienda se toma en cuenta la ubicación de la vivienda, el valor del predio en el autoevaluó y el monto de la prima será de 0.05% del valor del predio, si se desea asegurar el 100%. Si el monto de la prima es menor al 0.05% de porcentaje del valor del predio, entonces el seguro solo cubrirá el porcentaje al monto de la prima con respecto al valor del predio.

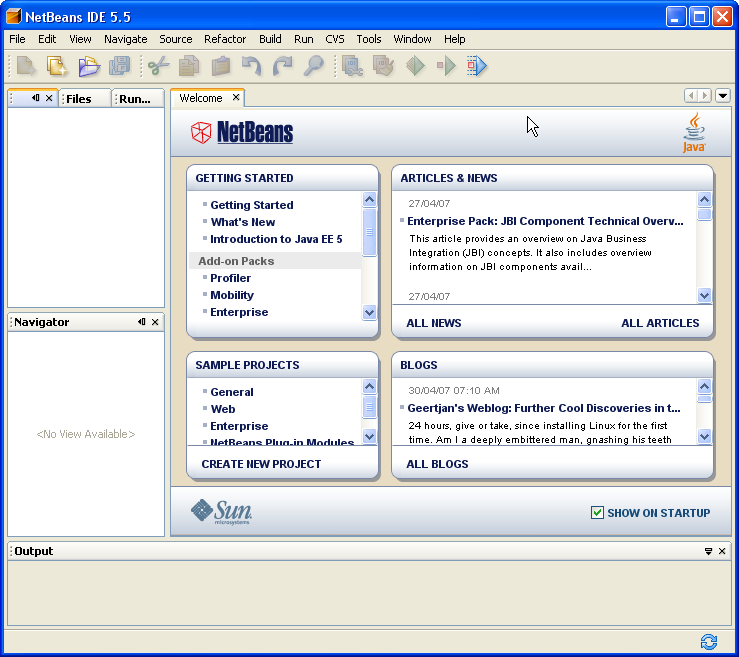
Realice una aplicación de una empresa aseguradora que realice las siguientes acciones:

* Registro de asegurados en la aseguradora.
* Cálculo del monto total de las primas del mes
* Cálculo el promedio de las edades de los seguros médicos
* Calculo el promedio del valor de los predios de los seguros de vivienda

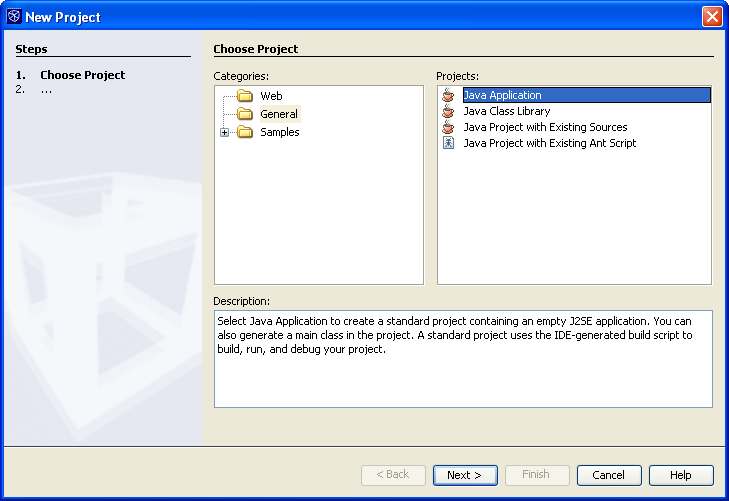
#### CONSTRUCCIÓN DE LA APLICACIÓN

1. Construimos nuestro proyecto **InsuranceApplication.**

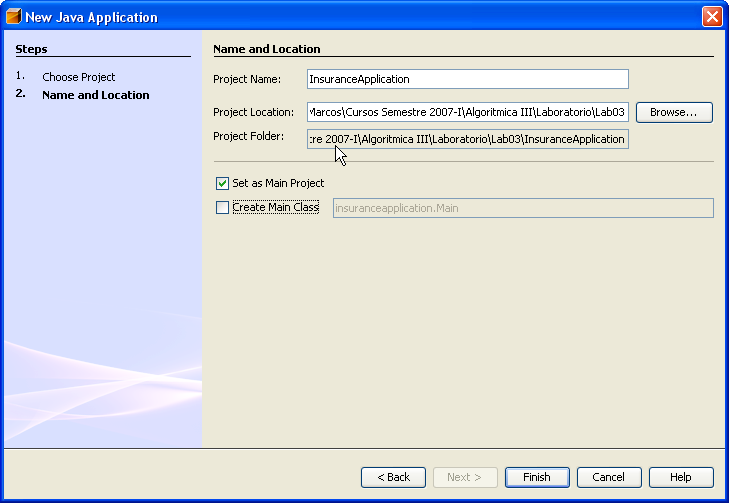
* Iniciamos cargando el NetBeans e inicialmente tendremos una pantalla como la siguiente:



* Procedemos a crear un nuevo proyecto y observamos que el asistente (Wizard) es cada vez más completo. Elegimos el menú **File** y seleccionamos la opción **New Project.** En la categoría de **General** y elegimos un proyecto de **Java Application**. Hacemos clic en el botón **Next**.

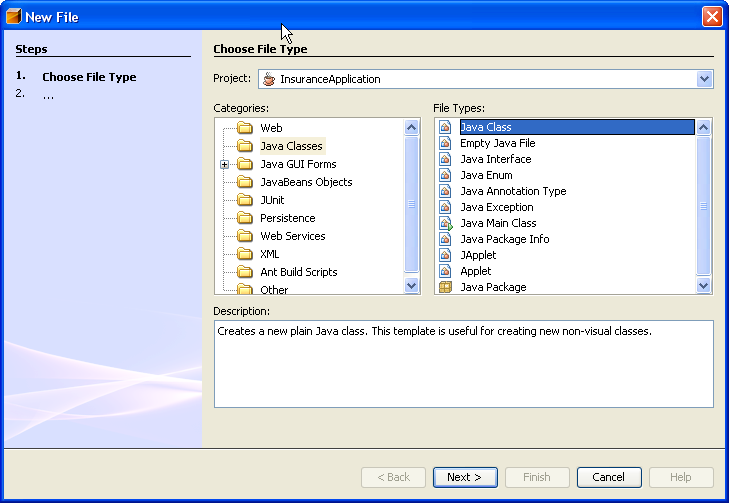


* Seleccionamos la ubicación del proyecto con el botón **Browse**. Luego colocamos el nombre **InsuranceApplication** al proyecto, y deseleccionamos la opción **Create Main Class**. Finalmente presionamos el botón de **Finish**.

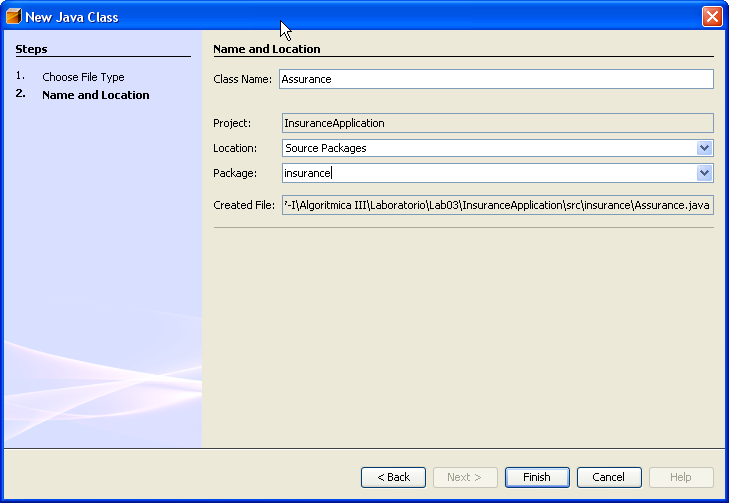


1. Adicionaremos la clase **Assurance** a nuestro proyecto.

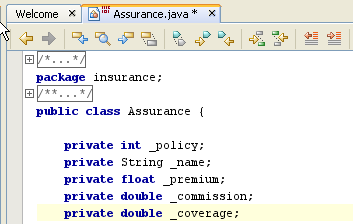
* En la ventana **Editing**. Primero seleccionamos la opción de menú **File | New File**, aparecerá la ventana de Plantillas (**New File**). En la subventana **Categories,** seleccionaremos el subconjunto **Java Classes**  y seleccionaremos **Java Class** y presionaremos el botón **Next.**



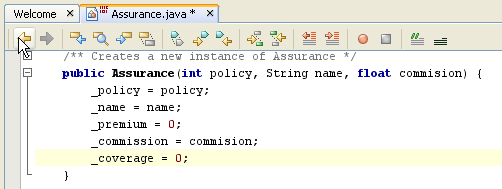
* En la ventana **Name and Location**, en el colocaremos el nombre de **Assurance** en la opción **Class Name** y el opción **Package** el nombre del paquete **Insurance** y hacemos clic en el botón **Finish.**



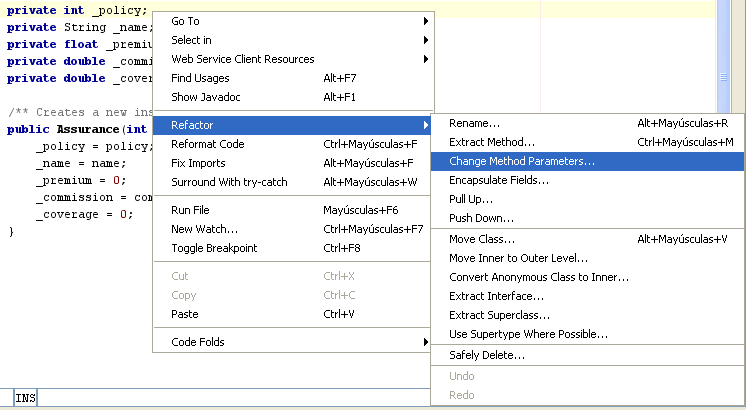
* A continuación incluiremos los siguientes atributos



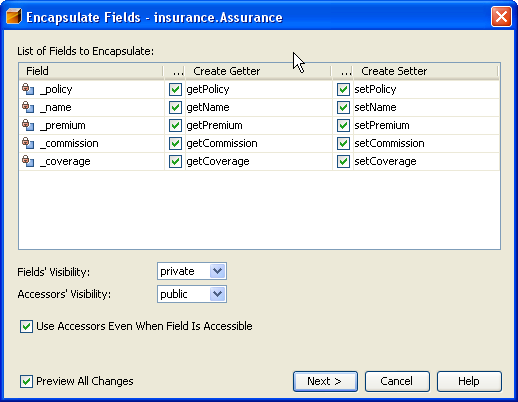
* Luego incluiremos el constructor que ingrese valores a los atributos.



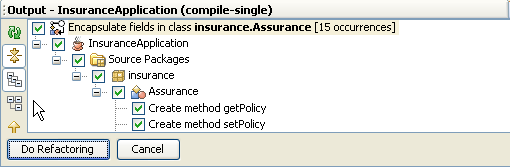
* Posteriormente incluiremos los métodos accesores, haciendo clic derecho en la clase y seleccionando la opción de menú **Refactor** y seleccionamos la opción **Encapsulate Fields**.



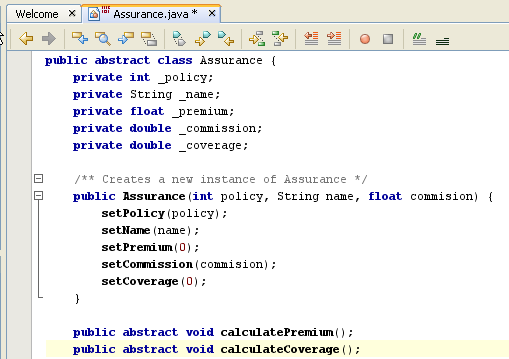
* Aparecerá la siguiente ventana en la cual se hace clic en **Next**.



* Para finalmente en la ventana de salida se realiza el clic en el botón **Do Recfactoring**

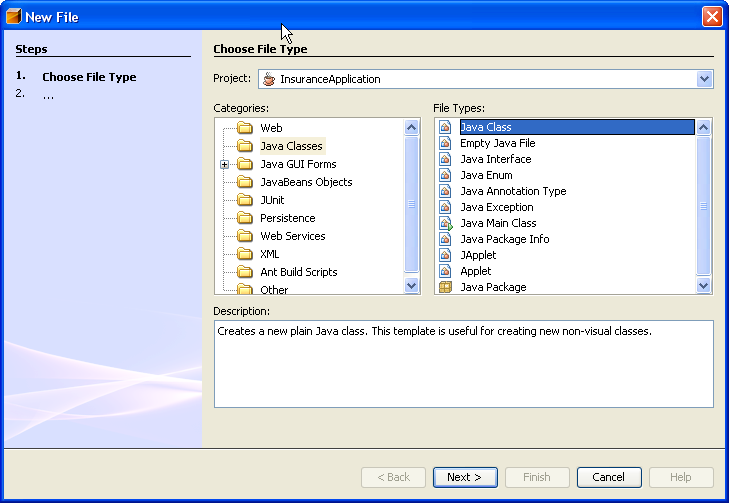
****

* Adiciona los métodos abstractos de calculo de Prima y de del monto de la cobertura. Así como, volver a la clase abstracta.

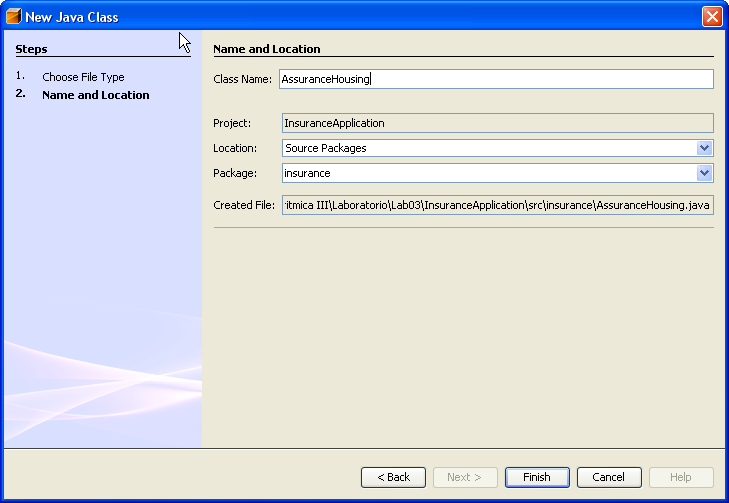


1. Adicionaremos la clase **AssuranceHousing** a nuestro proyecto.

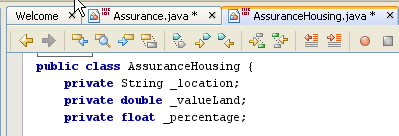
* En la ventana **Editing**. Primero seleccionamos la opción de menú **File | New File**, aparecerá la ventana de Plantillas (**New File**). En la subventana **Categories,** seleccionaremos el subconjunto **Java Classes**  y seleccionaremos **Java Class** y presionaremos el botón **Next.**



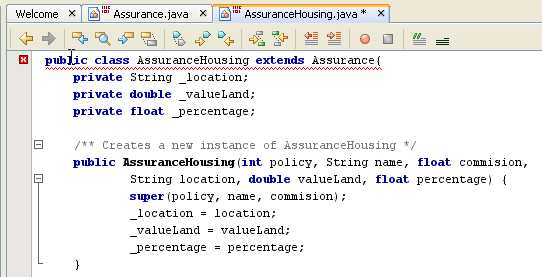
* En la ventana **Name and Location**, en el colocaremos el nombre de **AssuranceHousing** en la opción **Class Name** y el opción **Package** el nombre del paquete **Insurance** y hacemos clic en el botón **Finish.**



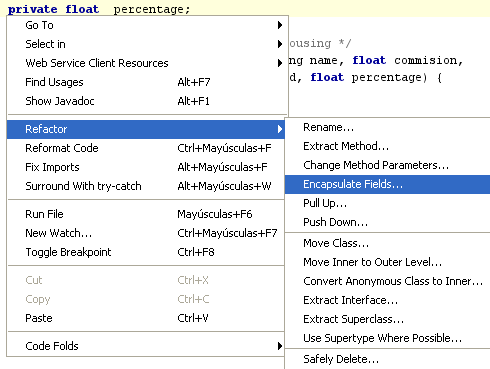
* A continuación incluiremos los siguientes atributos



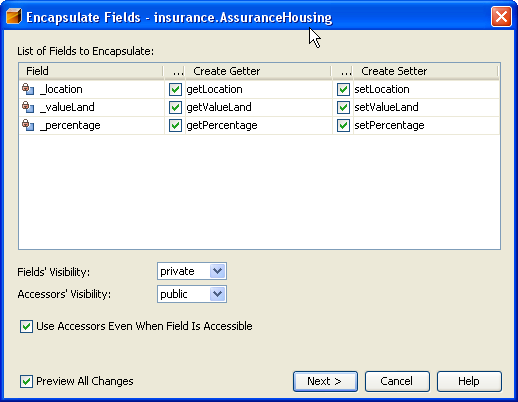
* Luego realizaremos la herencia simple desde la clase padre **Assurance**, adicionando a la declaración de la clase la herencia. A continuación incluiremos el constructor que ingrese parámetros a los atributos. El subrayado en rojo de la declaración de la clase es debido a que no se ha implementando todavía los métodos abstractos del padre.



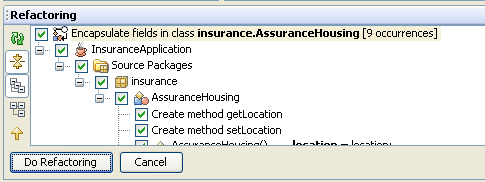
* Posteriormente incluiremos los métodos accesores, haciendo clic derecho en la clase y seleccionando la opción de menú **Refactor** y seleccionamos la opción **Encapsulate Fields**.



* Aparecerá la siguiente ventana en la cual se hace clic en **Next**.



* Para finalmente en la ventana de salida se realiza el clic en el botón **Do Recfactoring**

****

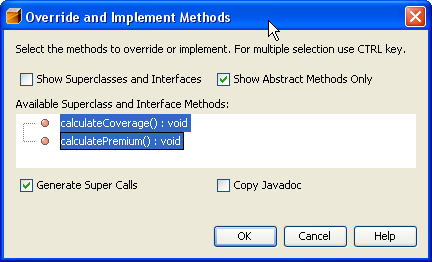
* Implementamos los métodos calculo de Prima y de del monto de la cobertura del padre; para locuaz seleccionamos en el menú **Source,**  la opción **Override Methods.** Aparecerá el siguiente menú y seleccionaremos la opción **Show AbstractMethod Only.** Luego seleccionamos los métodos con el mouse presionando la tecla **CTRL** y finalmente hacemos clic en el botón **OK**.

**public void calculatePremium() {**

**}**

**public void calculateCoverage() {**

**}**



* Procedemos a implementa el calculo de la Prima.

**public void calculatePremium() {**

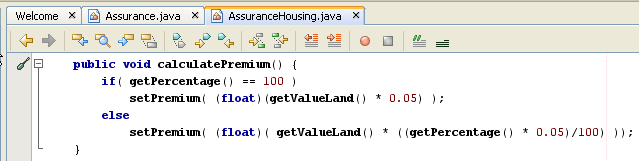
**if( getPercentage() == 100 )**

**setPremium( (float)(getValueLand() \* 0.05) );**

**else**

**setPremium( (float)( getValueLand() \* ((getPercentage() \* 0.05)/100) ))**

**}**



* Procedemos a implementa el calculo de la Cobertura.

**public void calculateCoverage() {**

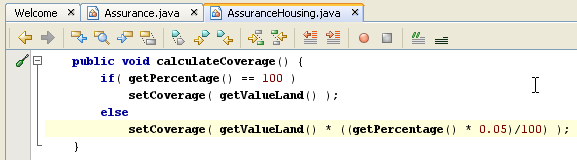
**if( getPercentage() == 100 )**

**setCoverage( getValueLand() );**

**else**

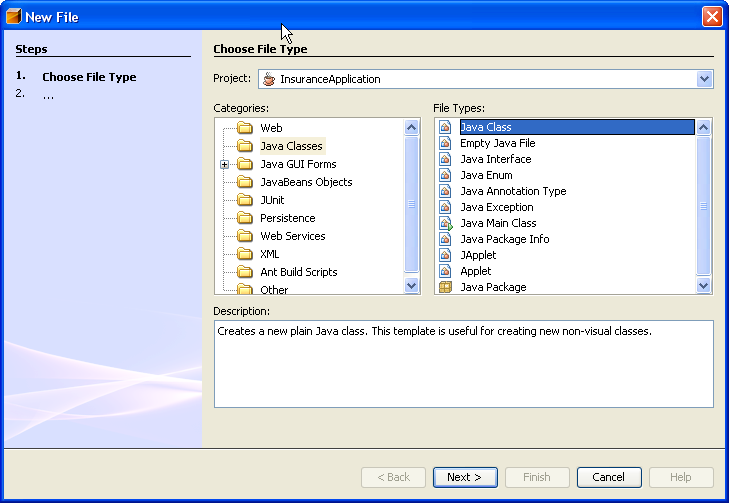
**setCoverage( getValueLand() \* (getPercentage()/100) );**

**}**

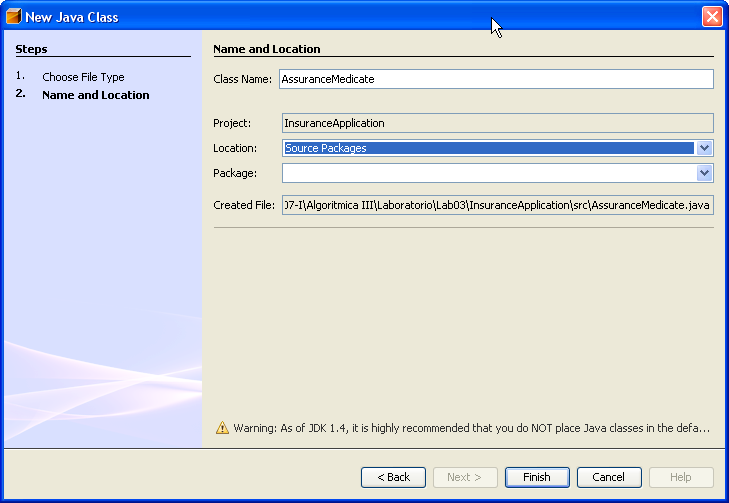


1. Adicionaremos la clase **AssuranceMedicate** a nuestro proyecto.

* En la ventana **Editing**. Primero seleccionamos la opción de menú **File | New File**, aparecerá la ventana de Plantillas (**New File**). En la subventana **Categories,** seleccionaremos el subconjunto **Java Classes**  y seleccionaremos **Java Class** y presionaremos el botón **Next.**

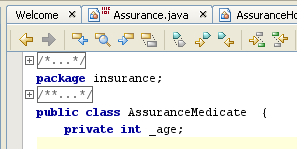


* En la ventana **Name and Location**, en el colocaremos el nombre de **AssuranceMedicate** en la opción **Class Name** y el opción **Package** el nombre del paquete **Insurance** y hacemos clic en el botón **Finish.**



* A continuación incluiremos los siguientes atributos

**private String \_age;**



* Luego realizaremos la herencia simple desde la clase padre **AssuranceMedicate**, adicionando a la declaración de la clase la herencia. A continuación incluiremos el constructor que ingrese parámetros a los atributos. El subrayado en rojo de la declaración de la clase es debido a que no se ha implementando todavía los métodos abstractos del padre.

**public class AssuranceMedicate extends Assurance {**

**private int \_age;**

**public AssuranceMedicate(int policy, String name, float commission,**

**float premium, int age) {**

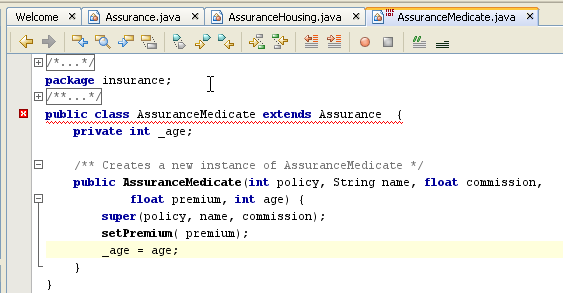
**super(policy, name, commission);**

**setPremium( premium);**

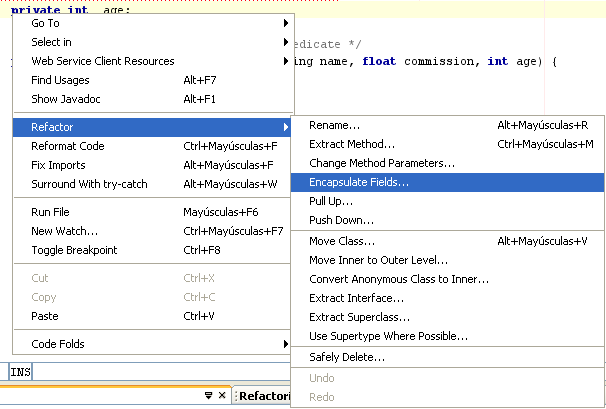
**\_age = age;**

**}**

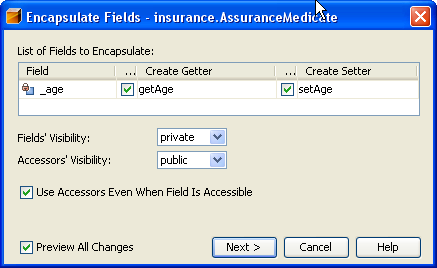
**}**



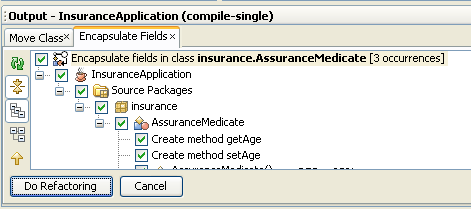
* Posteriormente incluiremos los métodos accesores, haciendo clic derecho en la clase y seleccionando la opción de menú **Refactor** y seleccionamos la opción **Encapsulate Fields**.



* Aparecerá la siguiente ventana en la cual se hace clic en **Next**.



* Para finalmente en la ventana de salida se realiza el clic en el botón **Do Recfactoring**

****

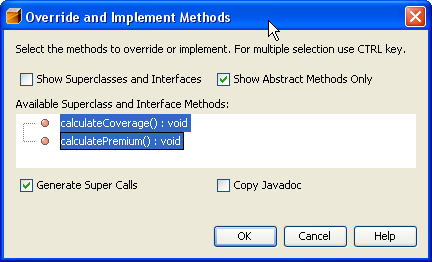
* Implementamos los métodos calculo de Prima y de del monto de la cobertura del padre; para locuaz seleccionamos en el menú **Source,**  la opción **Override Methods.** Aparecerá el siguiente menú y seleccionaremos la opción **Show AbstractMethod Only.** Luego seleccionamos los métodos con el mouse presionando la tecla **CTRL** y finalmente hacemos clic en el botón **OK**.

**public void calculatePremium() {**

**}**

**public void calculateCoverage() {**

**}**



* Procedemos a implementa el cálculo de la Prima.

**public void calculatePremium() {**

**if( \_age >= 60 )**

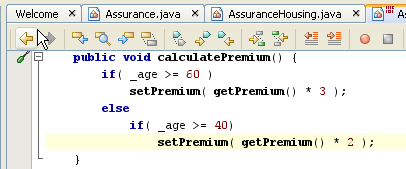
**setPremium( getPremium() \* 3 );**

**else**

**if( \_age >= 40)**

**setPremium( getPremium() \* 2 );**

**}**



* Procedemos a implementa el cálculo de la Cobertura.

**public void calculateCoverage() {**

**int premium = (int)getPremium();**

**if( \_age >= 60 )**

**premium = (int)(premium / 3);**

**else**

**if( \_age >= 40)**

**premium = (int)(premium / 2);**

**switch ( (int)premium ){**

**case 20: setCoverage( 5000 );**

**break;**

**case 40: setCoverage( 10000 );**

**break;**

**case 60: setCoverage( 20000 );**

**break;**

**case 80: setCoverage( 40000 );**

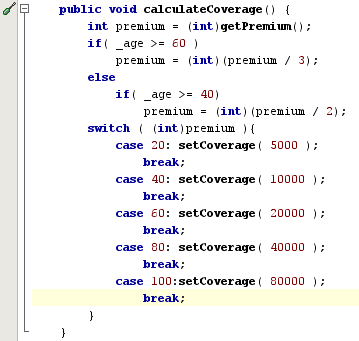
**break;**

**case 100:setCoverage( 80000 );**

**break;**

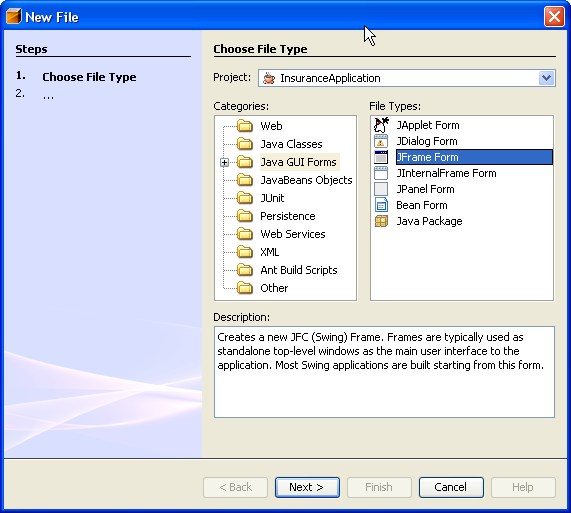
**}**

**}**

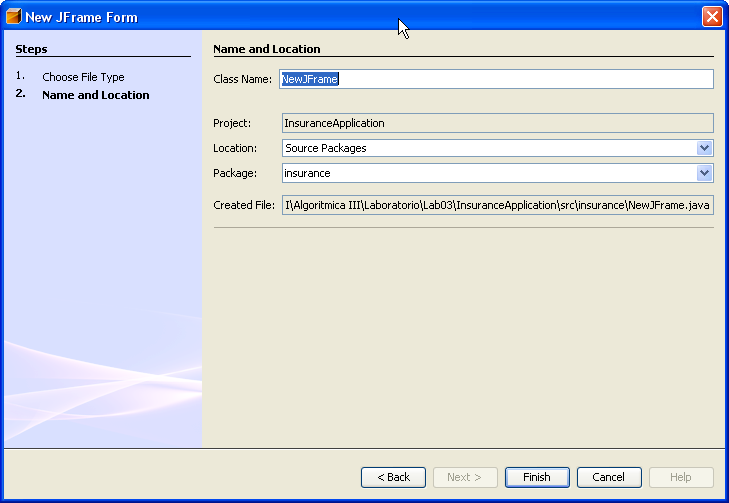


1. Interfaz Gráfica de usuario de la ventana de Principal

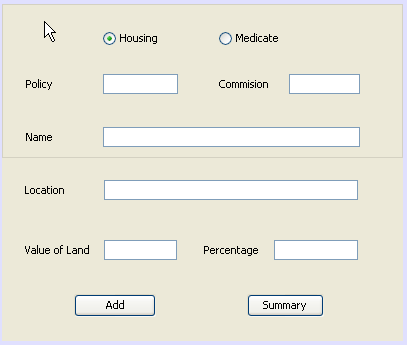
* Adicionaremos la ventana de **Insurance** a nuestro proyecto, para ello pasamos a la ventana GUI Editing. Primero seleccionamos la opción de menú **File | New File**, aparecerá la ventana de Plantillas (**New File**). En la subventana **Categories** seleccionaremos el conjunto **Java GUI Forms**  y seleccionaremos **JFrame Form** y presionaremos el botón **Next.**



* Aparecerá una ventana **New JFrame Form**, ahí colocaremos el nombre del archivo en **Class name** con el nombre de **InsuranceFrame** y presiona el botón **Finish.**



* En el entorno **GUI Editing**, hacer clic derecho sobre la ventana para seleccionar **setLayout** y seleccione la opción de **BorderLayoutLayout.** Coloque el nombre **Insurance Application** a la propiedad **title** enla ventana de propiedades. Adicione tres **JPanel** con la orientación norte, sur y centro; a continuación, redimensione los paneles del norte y del sur para que contengan una fila de componentes, con la opción **Design This Conainer.** Cambie los nombres de los paneles a **jPanelSouth**, **jPanelNorth** y **jPanelCenter**; haga clic derecho sobre el panel central para seleccione la opción **setLayout** y seleccione **cardLayout**. Realice clic derecho en el **jPanelCenter** de la ventana del **Inspector**, seleccione la opción **Add From Palette**, luego **Swing** yluego **JPanel,** para adicionar dos paneles a los que llamaremos: **jPanelHousing** y **jPanelMedicate.** Cambiamos su propiedad **Card Name** a **Housing** y **Medicate** respectivamente.



* Finalmente adicione los siguientes componentes al panel Norte:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Componente** | **Variable Name** | **text** | **selected** | **buttonGroup** |
| ButtonGroup | buttonGroupInsurance | --------- |  |  |
| JRadioButton | jRadioButtonHousing | Housing | true | buttonGroupInsurance |
| JRadioButton | jRadioButtonMedicate | Medicate | ----- | buttonGroupInsurance |
| JLabel | ------- | Policy |  |  |
| JLabel | ------- | Name |  |  |
| JLabel | ------- | Commision |  |  |
| JTextField | jTextFieldPolicy | ------- |  |  |
| JTextField | jTextFieldName | ------- |  |  |
| JTextField | jTextFieldCommision | 0.15 |  |  |

* Finalmente adicione los siguientes componentes al panel Sur:

|  |  |  |
| --- | --- | --- |
| **Componente** | **Variable Name** | **text** |
| JButton | jButtonAdd | Add |
| JButton | jButtonSummary | Summary |

* Finalmente adicione los siguientes componentes al panel Medico:

|  |  |  |  |
| --- | --- | --- | --- |
| **Componente** | **Variable Name** | **text** | **model** |
| JLabel | ------- | Premium |  |
| JLabel | ------- | Age |  |
| JTextField | jTextfieldAge | ------- |  |
| JComboBox | jComboBoxPremium | ------- | 20,40,60,80,100 |

* Finalmente adicione los siguientes componentes al panel Vivienda:

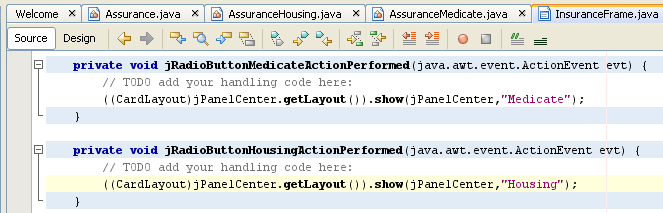
|  |  |  |
| --- | --- | --- |
| **Componente** | **Variable Name** | **text** |
| JLabel | ------- | Location |
| JLabel | ------- | Value of Land |
| JLabel | ------- | Percentage |
| JTextField | jTextfieldLocation | ------- |
| JTextField | jTextfieldValueLand | ------- |
| JTextField | jTextfieldPercentage | ------- |

* A continuación programaremos los eventos del los botones de radio con las siguientes instrucciones:

**import java.awt.CardLayout;**

**((CardLayout)jPanelCenter.getLayout()).show(jPanelCenter,"Housing");**

**((CardLayout)jPanelCenter.getLayout()).show(jPanelCenter,"Medicate");**

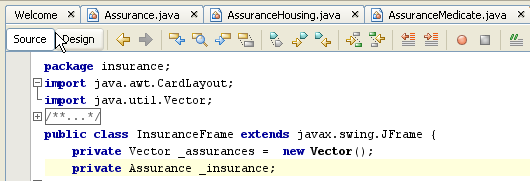


* Agregaremos un vector a la ventana para que contenga objetos de las clases hijas de **Insurance**

**import java.util.Vector;**

**private Vector \_assurances = new Vector();**

**private Assurance \_insurance;**



* Adicionaremos las siguientes instrucciones en el **actionPerformed** del botón **Add**

**int policy = Integer.parseInt( jTextFieldPolicy.getText() );**

**String name = jTextFieldName.getText();**

**float commision = Float.parseFloat( jTextFieldCommision.getText() );**

**if( jRadioButtonHousing.isSelected() ){**

**String location = jTextFieldLocation.getText();**

**double valueLand = Double.parseDouble( jTextFieldValueLand.getText() );**

**float percentage = Float.parseFloat( jTextFieldPercentage.getText() );**

**\_insurance = new AssuranceHousing(policy, name, commision,**

**location, valueLand, percentage );**

**}**

**else{**

**float premium = Float.parseFloat(**

**jComboBoxPremium.getSelectedItem().toString() );**

**int age = Integer.parseInt( jTextFieldAge.getText() );**

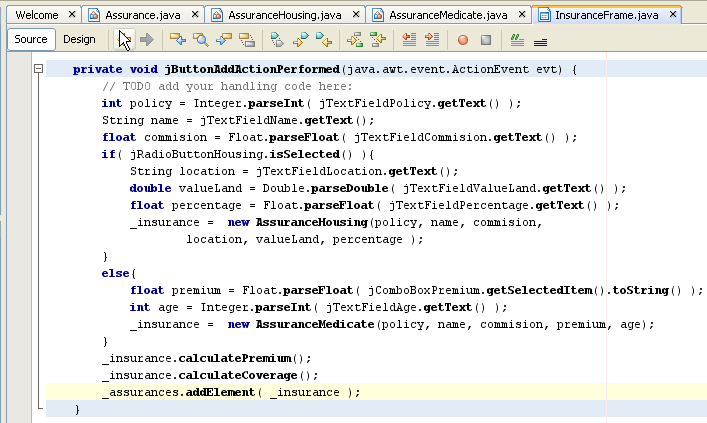
**\_insurance = new AssuranceMedicate(policy, name, commision, premium, age);**

**}**

**\_insurance.calculatePremium();**

**\_insurance.calculateCoverage();**

**\_assurances.addElement( \_insurance );**



* Adicionaremos las siguientes instrucciones en el **actionPerformed** del botón **Summary**

**double summary = 0, sumValueLand = 0, averageValueland = 0;**

**int sumAge = 0, countMedicate = 0, countHousing = 0, averageAge = 0;**

**for(Object insurance : \_assurances ) {**

**summary = summary + ((Assurance)insurance).getCoverage();**

**if( insurance instanceof AssuranceMedicate ){**

**countMedicate++;**

**sumAge = sumAge + ((AssuranceMedicate)insurance).getAge();**

**}**

**if( insurance instanceof AssuranceHousing){**

**countHousing++;**

**sumValueLand = sumValueLand +**

**((AssuranceHousing)insurance).getValueLand();**

**}**

**}**

**if (countMedicate != 0)**

**averageAge = (int)(sumAge/countMedicate);**

**if( countHousing != 0)**

**averageValueland = sumValueLand / countHousing;**

**JOptionPane.showMessageDialog(this,**

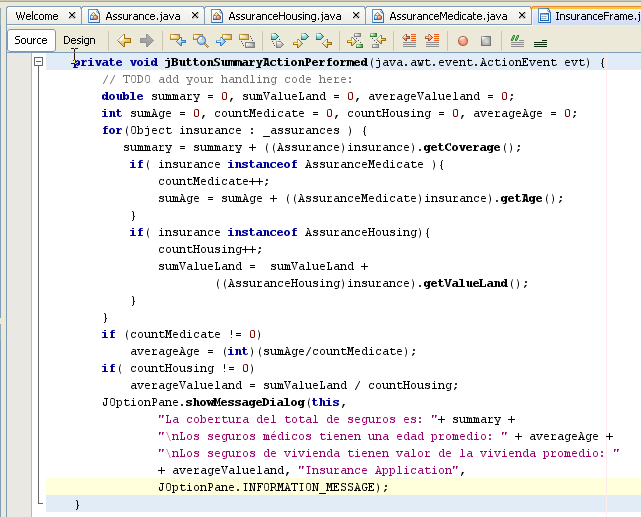
**"La cobertura del total de seguros es: "+ summary +**

**"\nLos seguros médicos tienen una edad promedio: " + averageAge +**

**"\nLos seguros de vivienda tienen valor de la vivienda promedio: "**

**+ averageValueland, "Insurance Application",**

**JOptionPane.INFORMATION\_MESSAGE);**



1. **Tarea:** Desarrollemos los siguientes puntos

* Adicionemos métodos para la modificación, eliminación y búsqueda de asegurado.
* Definiremos la forma de garantizar de no ingresar dos asegurados con la misma poliza.
* Adicionaremos métodos para registrar el pago de la prima periodicamente.