# Tutorial de BlueJ

*Versión en Español:*
Iván Alfonso Guarín V.
Coordinador Grupo GUIA
- Grupo de Investigación en Aprendizaje -
web: http://milano.uis.edu.co/guia/
email: gguia@milano.uis.edu.co

Versión 1.4

para BlueJ Versión 1.2.x

Michael Kölling
Mærsk Insitute
University of Southern Denmark

*Tabla de contenido*

# 1 Prólogo

## 1.1 Acerca de BlueJ

Este tutorial es una introducción al uso del entorno de programación BlueJ. BlueJ es un entorno de desarrollo para Java™ diseñado específicamente para la enseñanza en un curso introductorio. Fue diseñado e implementado por el equipo de BlueJ en la Universidad de Monash, Melbourne, Australia, y la Universidad de Southern Denmark, Odense.

Más información sobre BlueJ se encuentra disponible en http://www.bluej.org.

## 1.2 Alcance y Audiencia

Este tutorial está hecho para las personas que desean familiarizarse con las capacidades de este entorno de programación. Aquí no se explican las decisiones de diseño que subyacen en la construcción del entorno o la investigación en la cual se basa.

Este tutorial no tiene la intención de enseñar Java. A los principiantes en la programación en Java se les aconseja estudiar también con un texto introductorio o tomar un curso de Java.

Este no es un manual de referencia comprensivo del entorno. Muchos detalles se dejaron – se hace énfasis en una introducción breve y concisa, en vez de un cubrimiento completo de características.

Cada sección comienza con una oración inicial de resumen. Esto permite a los usuarios familiarizados con algunas partes decidir si ellos quieren leer o saltar cada parte en particular. La sección 10 repite únicamente las líneas resumen como una referencia rápida.

## 1.3 Copyright, licencia y redistribución

## 1.4 Realimentación

Comentarios, preguntas, correcciones, críticas y cualquier otra clase de realimentación concerniente al sistema BlueJ o a su tutorial son bienvenidas y acogidas activamente. Por favor escriba un correo electrónico a Michael Kölling ( mik@mip.sdu.dk ).

# 2 Instalación

BlueJ es distribuido en tres formatos diferentes: uno para sistemas Windows, uno para MacOS, y uno para los otros sistemas. La instalación es fácil y rápida.

*Prerrequisitos*

Usted debe tener J2SE v1.3 ( o. JDK 1.3 ) o posterior instalado en sus sistema para utilizar BlueJ. Si usted no tiene instalado el JDK usted puede descargarlo del sitio web de Sun en http://java.sun.com/j2se/. En MacOS X, una versión reciente del JDK está preinstalada – usted no necesita instalarla. Si usted encuentra una página de descargas que ofrece el "JRE"( Java Runtime Environment ) y el "SDK" ( Software Development Kit ), usted debe descargar el "SDK" – el JRE no es suficiente.

## 2.1 Instalación en Windows

El archivo de distribución para los sistemas Windows es llamado *bluejsetup-xxx.exe*, donde *xxx* es un número de versión. Por ejemplo, la distribución de Bluej versión 1.2.0 se llama *bluejsetup-120.exe*. Usted puede tener este archivo en disco, o puede descargarlo del sitio web de Bluej http://www.bluej.org. Ejecute este instalador.

El instalador le permite seleccionar un directorio para instalarlo. El también ofrece la opción de instalar un acceso directo en el menú de inicio y en el escritorio.

Luego de finalizada la instalación, usted encontrará el programa *bluej.exe* en el directorio de instalación de BlueJ.

La primera vez que usted lance BlueJ, él buscará el sistema Java ( JDK ). Si él encuentra disponible más de un sistema Java ( e.j. usted tiene JDK 1.3.1 y JDK 1.4 instalado ), una ventana de diálogo le dejará seleccionar uno de ellos para utilizarlo. Si él no encuentra un sistema Java, a usted se le solicitará que lo localice ( esto puede ocurrir cuando un sistema JDK ha sido instalado, pero las correspondientes entradas de registro han sido removidas ).

El instalador de BlueJ también instala un programa llamado *vmselect.exe*. Usando este programa, usted puede cambiar posteriormente cuál versión de Java utiliza BlueJ. Ejecute *vmselect* para ejecutar BlueJ con una versión diferente de Java.

La selección del JDK es almacenado para cada versión de BlueJ. Si usted tiene diferentes versiones instaladas de BlueJ, usted puede usar una versión de BlueJ con el JDK 1.3.1 y otra versión de BlueJ con JDK 1.4. Cambiar la versión de Java para BlueJ hará que se cambie para todas las instalaciones de BlueJ de la misma versión para el mismo usuario.

## 2.2 Instalación en Macintosh

Por favor note que BlueJ se ejecuta solamente en MacOS X. El archivo de distribución para MacOS es llamado *BlueJ-xxx.sit*, donde *xxx* es un número de versión. Por ejemplo, la distribución de Bluej versión 1.2.0 se llama *BlueJ-120.sit*. Usted puede tener este archivo en disco, o puede descargarlo del sitio web de Bluej http://www.bluej.org.

Este archivo puede ser descomprimido por el expansor *StuffIt*. Muchos navegadores descomprimirán este archivo por usted. De otro modo, haciendo doble clic se descomprimirá.

Luego de descomprimir, usted tendrá una carpeta llamada BlueJ-*xxx*. Mueva esta carpeta dentro de su carpeta de Aplicaciones ( o donde quiera que a usted le guste guardarlo ). No son necesarias posteriores instalaciones.

## 2.3 Instalación en Linux/Unix y otros sistemas

El archivo de distribución para MacOS es llamado *BlueJ-xxx.sit*, donde *xxx* es un número de versión. Por ejemplo, la distribución de Bluej versión 1.2.0 se llama *BlueJ-120.sit*. Usted puede tener este archivo en disco, o puede descargarlo del sitio web de Bluej http://www.bluej.org.

El archivo de distribución general es un archivo jar ejecutable. Se llama *bluej-xxx.jar*, donde *xxx* es el número de versión. Por ejemplo, la distribución de Bluej versión 1.2.0 se llama *bluej-120.jar*. Usted puede tener este archivo en disco, o puede descargarlo del sitio web de Bluej http://www.bluej.org.

Ejecute el instalador a través del siguiente comando. NOTA: Para este ejemplo, el archivo utilizado es el de la distribución *bluej-120.jar* – usted necesita utilizar el nombre del archivo que vaya a utilizar ( con el número de versión correcto ).

```
< jdk-path>/bin/java -jar bluej-120.jar
```

`< jdk-path>` es el directorio, donde el JDK fue instalado.

Una ventana se lanza, permitiéndole escoger el directorio de instalación de Bluej y la versión del JDK utilizada para ejecutar BlueJ. Importante: El camino para BlueJ ( esto es, cualquiera de los directorios padre ) no deben contener espacios en blanco.

Haga clic en *Install*. Luego de finalizar, BlueJ deberá estar instalado.

## 2.4 Problemas de Instalación

Si usted tiene algún problema, verifique la sección *Frequently Asked Questions* ( FAQ ) en el sitio web de BlueJ ( http://www.bluej.org/help/faq.html ) y lea la sección *How To Ask For Help* ( http://www.bluej.org/help/ask-help.html ).

# 3 Comenzando – edit / compile / execute

## 3.1 Lanzando BlueJ

En Windows y MacOS, está instalado un programa llamado *BlueJ*. Ejecútelo.

En sistemas Unix el instalador guarda un script llamado *bluej* en el directorio de instalación. Desde una interfaz GUI, haga doble clic en el archivo. Desde una línea de comandos usted puede iniciar BlueJ con o sin un proyecto como argumento:

```
$ bluej
o
$ bluej examples/people
```

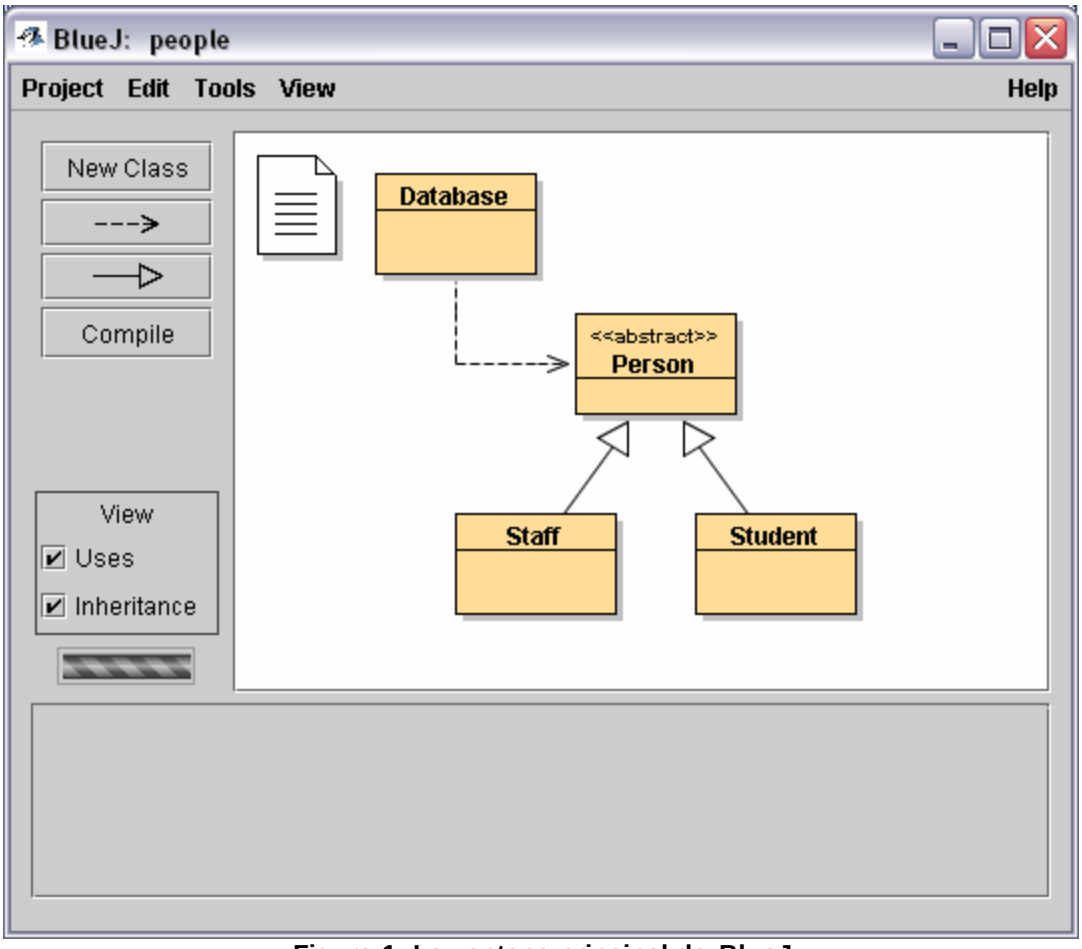


**Figura 1: La ventana principal de BlueJ**

## 3.2 Abriendo un proyecto

*Resumen:* Para abrir un proyecto, seleccione *Open* del menú *Project.*

Los proyectos en BlueJ, como los paquetes estándar en Java, son directorios conteniendo los archivos incluidos en el proyecto.

Luego de iniciar BlueJ, use el comando de menú Project – Open... para seleccionar y abrir un proyecto.

Algunos proyectos de ejemplo están incluidos en la distribución estándar de BlueJ en el directorio *examples*.

Para esta sección del tutorial, abra el proyecto *people*, el cual está incluido en este directorio. Usted puede encontrar el directorio *examples* en el directorio instalación de BlueJ. Luego de abrir el proyecto, usted debería ver algo similar a la ventana mostrada en la Figura 1. La ventana puede no lucir exactamente como en su sistema, pero las diferencias deberían ser menores.

## 3.3 Creando objetos

*Resumen:* Para crear un objeto, seleccione un constructor del menú emergente de la clase.

Una de las características fundamentales de BlueJ es que usted no puede ejecutar únicamente una aplicación directamente, sino que también debe interactuar directamente con objetos simples pertenecientes a cualquier clase y, ejecutar sus métodos públicos. Una ejecución en BlueJ normalmente se realiza creando un objeto y luego invocando uno de los métodos del objeto. Esto es muy útil durante el desarrollo de una aplicación – usted puede probar clases individualmente tan pronto como hayan sido escritas. No hay necesidad de escribir primero la aplicación.

> Nota al pie*: Los métodos estáticos pueden ser ejecutados directamente sin crear primero un objeto. Uno de los métodos estáticos puede ser "main", así podemos hacer lo mismo que normalmente pasa en las aplicaciones – iniciar una aplicación sólo ejecutando un método estático main. Regresaremos a esto luego. Primero, haremos algunas otras cosas, cosas más interesantes las cuales normalmente no pueden ser realizadas en los entornos Java.*

Los cuadros que se ven en la parte central de la ventana principal ( etiquetados *Database*, *Person*, *Staff* y *Student* ) son iconos representando las clases involucradas en esta aplicación. Usted puede obtener un menú con operaciones, aplicable a una clase haciendo clic en el icono de la clase con el botón derecho del ratón (Macintosh: ctrl.-clic[1]) ( Figura 2 ). Las operaciones mostradas son operaciones *new* con cada uno de los

[1] Donde se mencione clic derecho en este tutorial, los usuarios Macintosh deberían leerlo como *ctrl.-clic*.

constructores definidos para esta clase ( primero ) seguidas por algunas operaciones proporcionadas por el entorno.

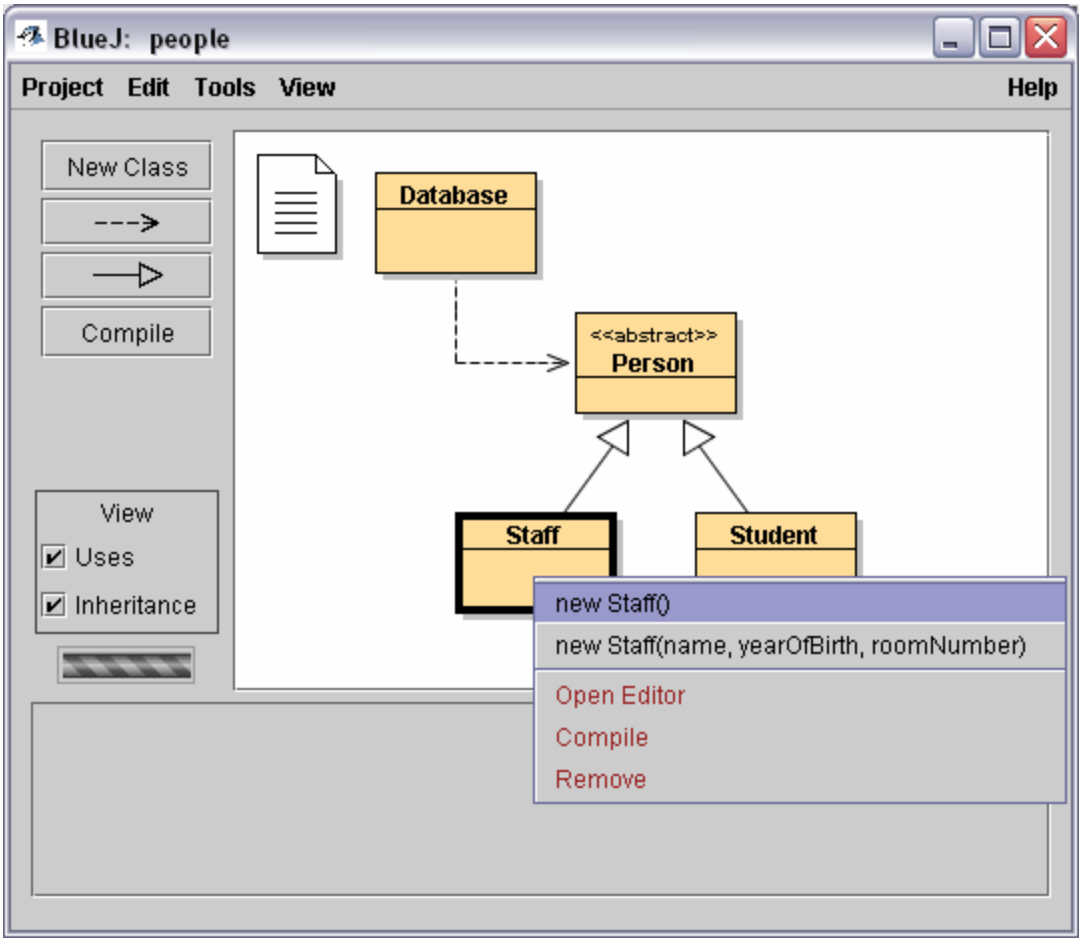


**Figura 2: Operaciones de Clase ( menú emergente )**

Queremos crear un objeto *Staff*, así usted debería hacer clic derecho sobre el icono de *Staff* ( lo cual hace que emerja el menú mostrado en la Figura 2 ). El menú muestra dos constructores para crear un objeto *Staff*, uno con parámetros y otro sin ellos. Primero, seleccione el constructor sin parámetros. Aparece la ventana de diálogo mostrada en la Figura 3.

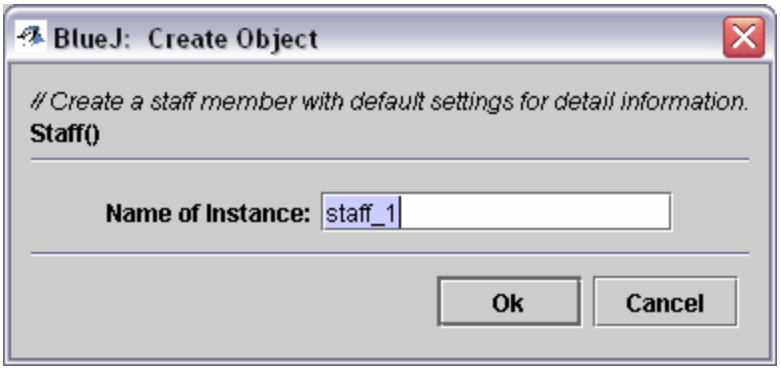


**Figura 3: Creación de Objeto sin parámetros**

Esta ventana de diálogo pregunta por un nombre para el objeto a ser creado. Al mismo tiempo, se sugiere un nombre por defecto ( *staff_1* ). Este nombre por defecto es lo suficientemente bueno por ahora, así que sólo haga clic en *OK*. Un objeto *Staff* será creado.

Desde que el objeto ha sido creado, él se coloca en el área de objetos ( Figura 4 ). Esto es todo lo necesario para crear un objeto: seleccione un constructor del menú de la clase, ejecútelo y usted tendrá el objeto en el área de objetos.

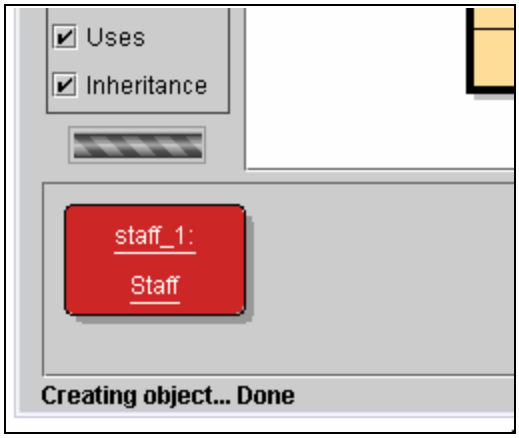


**Figura 4: Un Objeto sobre el área de Objetos**

Usted puede haber notado que la clase *Person* está etiquetada como <<abstract>> ( ésta es una clase abstracta ). Usted notará ( si lo trata de probar ) que no puede crear objetos desde clases abstractas ( como la especificación del lenguaje Java lo define ).

## 3.4 Ejecución

*Resumen:* Para ejecutar un método, selecciónelo del menú emergente del objeto.

Ahora que ha creado un objeto, usted puede ejecutar sus operaciones públicas. ( Java llama a las operaciones *métodos*. ) Haga clic con el botón derecho del ratón sobre el objeto y emerge un menú con las operaciones del objeto ( Figura 5 ). El menú muestra los métodos disponibles para este objeto y dos operaciones especiales proporcionadas por el entorno ( *Inspect* y *Remove* ). Discutiremos esto luego. Primero, concentrémonos en los métodos.

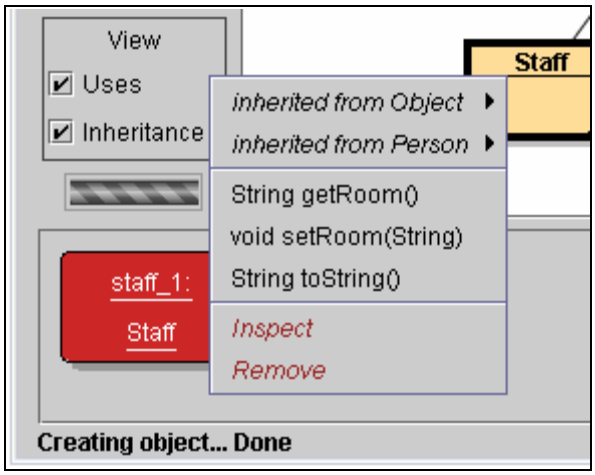


**Figura 5: El menú de Objeto**

Usted ve que existen los métodos *getRoom* y *setRoom* los cuales configuran y retornan el número de la habitación para este miembro del staff. Haga un llamado a *getRoom*. Simplemente selecciónelo de su menú de objeto y éste se ejecutará. Una ventana de diálogo aparece mostrando el resultado del llamado ( Figura 6 ). En este caso el nombre dice "( unknown room )" porque no especificamos ninguna habitación para esta persona.

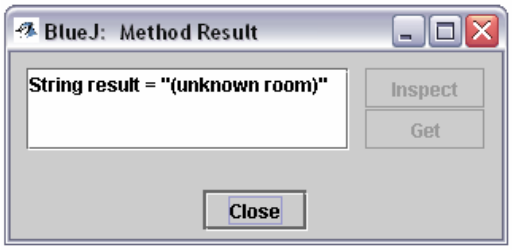


**Figura 6: Ventana mostrando el resultado de una función**

Los métodos heredados de una superclase están disponibles a través de un submenú. En la parte superior del menú emergente de objeto existen dos submenús, uno para los métodos heredados de *Object* y otro para los de *Person* ( Figura 5 ). Usted puede llamar a los métodos de *Person* ( tales como *getName* ) seleccionándolos desde el submenú. Pruébelo. Usted notará que la respuesta es igualmente vaga: se responde "( unknown name )", debido a que no hemos dado un nombre a la persona.

Ahora tratemos de especificar un número de habitación. Esto se mostrará como un llamado que tiene parámetros. ( Los llamados a *getRoom* y *getName* tienen valores de retorno, pero no parámetros ).

Llame la función *setRoom* seleccionándola del menú. Una ventana de diálogo aparece solicitándole que digite un parámetro ( Figura 7 ).

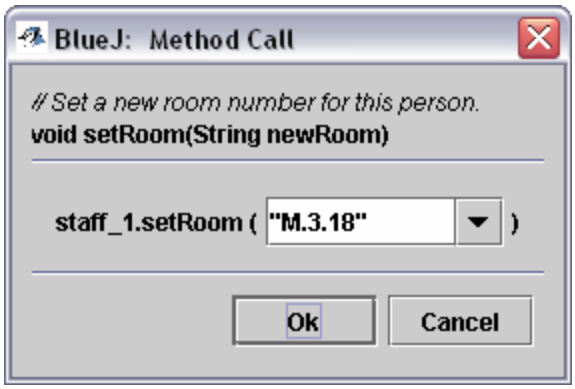


**Figura 7: Ventana de Diálogo del Llamado a una función con parámetros**

En la parte superior de esta ventana de diálogo se muestra la interfaz del método a ser llamado ( incluyendo el comentario y la signatura ). Debajo de esto, existe un campo de entrada de texto donde usted puede digitar el parámetro. La signatura al comienzo nos indica se que espera un parámetro de tipo String. Digite la nueva habitación, como una cadena de caracteres ( incluyendo las comillas ) en el campo de texto y haga clic en *OK*.

Esto es todo – debido a que este método no retorna un parámetro, no hay ventana resaltada. Llame de nuevo al método *getRoom* para verificar que la habitación realmente ha sido cambiada.

Practique la creación de objetos y el llamado de métodos por un rato. Trate llamando al constructor con argumentos y llame algunos otros métodos hasta que se familiarice con estas operaciones.

## 3.5 Editando una clase

*Resumen:* Para editar el archivo fuente de una clase, haga doble clic sobre su icono de clase.

Hasta ahora hemos tratado solamente son una interfaz de objetos. Ahora es tiempo de profundizar. Usted puede ver la implementación de una clase seleccionando *Open Editor* desde las operaciones de la clase. ( Recuerde: haga clic derecho sobre el icono de la clase y se muestran las operaciones de la clase. ) Hacer doble clic sobre la clase es un atajo para la misma función. El editor no se describe en mucho detalle dentro de este tutorial, pero no es muy importante para su uso. Los detalles del editor serán descritos luego. Por ahora, abra la implementación de la clase *Staff*. Encuentre la implementación del método *getRoom*. El retorna, como su nombre lo sugiere, el número de la habitación del miembro del staff. Cambiemos el método agregando el prefijo "`room`" al resultado de la función ( así que el método retorna, "room M.3.18" en lugar de sólo "M.3.18" ).

Podemos hacer esto cambiando la línea

```
    return room;
```

por

```
    return "room " + room;
```

BlueJ soporta instrucciones Java por completo y sin modificar, así que no hay nada especial acerca de cómo implemente sus clases.

## 3.6 Compilación

*Resumen:* Para compilar una clase, haga clic en el botón Compile en el editor. Para compilar un proyecto, haga clic en el botón Compile en la ventana de proyecto.

Luego de insertar el texto ( antes de hacer cualquier otra cosa ), verifique la vista del proyecto ( ventana principal ). Usted notará que ha cambiado el icono de clase para la clase *Staff*: él ahora está rayado. La apariencia rayada marca las clases que no han sido compiladas desde el último cambio. Regrese al editor.

> Nota al lado*: Usted debe preguntarse por qué los iconos de clase no fueron rayados cuando usted abrió por primera vez este proyecto. Esto es porque las clases en el proyecto* People *fueron distribuidas estando compiladas. A menudo los proyectos en BlueJ son distribuidos sin compilar, así que espere ver más clases sin compilar al abrir un por primera vez un proyecto.*

En la barra de herramientas en la parte superior del editor existen algunos botones con funciones utilizadas frecuentemente. Uno de ellos es *Compile*. Esta función le permite compilar esta clase directamente desde el editor. Ahora haga clic en el botón *Compile*. Si usted no tuvo ninguna equivocación, un mensaje debería aparecer en el área de información en la parte inferior del editor, notificándolo que la clase ha sido compilada. Si usted cometió una equivocación que se basa en un error de sintaxis, la línea de error es marcada y un mensaje de error se muestra en el área de información. ( En el caso de que su compilación funcione correctamente la primera vez, trate de introducir un error de sintaxis ahora – tal como un punto y coma faltante – y compile de nuevo, observe qué aparece )

Luego de que ha compilado exitosamente la clase, cierre el editor.

> Nota al lado*: No hay necesidad de guardar explícitamente el código fuente de la clase. El código fuente es guardado automáticamente donde sea apropiado ( e.j. cuando el editor se cierra o antes de compilar una clase ). Usted puede guardar explícitamente una clase si usted lo desea ( hay una función en el menú del editor de clase ), pero esto sólo se necesita realmente si su sistema es inestable y frecuentemente se daña, y usted se encuentra preocupado sobre la posible pérdida de su trabajo.*

La barra de herramientas de la ventana de proyecto también tiene un botón *Compile*. Esta operación compila el proyecto completo. ( De hecho, ella determina cuáles clases necesitan ser recompiladas y luego recompila aquellas clases en el orden correcto. ) Trate de hacer esto cambiando dos o más clases ( de tal forma que dos o más clases aparecen rayadas en el diagrama de clases ) y luego haga clic en el botón *Compile*. Si

se detecta un error en una de las clases compiladas, el editor se abrirá y la localización del error y el mensaje serán mostrados.

Usted puede notar que el área de objetos está vacía de nuevo. Los objetos son removidos cada vez que la implementación se cambia.

## 3.7 Ayuda con errores del compilador

*Resumen:* Para obtener ayuda sobre un mensaje de error del compilador, haga clic en el símbolo de interrogación cercano al mensaje de error.

Muy frecuentemente, los estudiantes principiantes tienen dificultad en entender los mensajes de error del compilador. Nosotros tratamos de proporcionar alguna ayuda.

Abra el editor de nuevo, introduzca un error en el código fuente, y compile. Un mensaje de error debería mostrarse en el área de información del editor. En la parte derecha del área de información aparece un botón con el símbolo de interrogación ( ? ) en el cual usted puede hacer clic para obtener más información acerca de este tipo de error ( Figura 8 ).

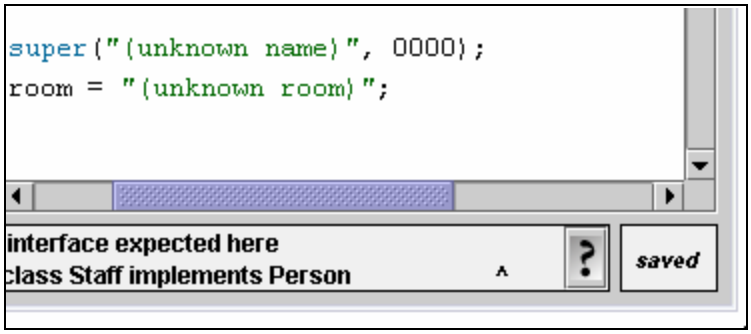


**Figura 8: Un error del compilador y el botón de Ayuda ( *Help* )**

En este momento, los textos de ayuda no están disponibles para todos los mensajes de error. Algunos textos de ayuda aún se deben escribir. Pero es difícil hacerlo – muchos errores ya se han explicado. El resto serán escritos e incluidos en una futura versión de BlueJ.

# 4 Haciendo un poco más...

En esta sección cubriremos algunas otras cosas que usted puede hacer en el entorno. Cosas que no son esenciales, pero utilizadas muy comúnmente.

## 4.1 Inspección

*Resumen:* Un objeto puede ser pasado como un parámetro a un llamado de un método haciendo clic sobre el icono del objeto.

Cuando usted ejecuta métodos de un objeto, usted puede haber notado la operación *Inspect,* la cual está disponible sobre los objetos además de los métodos definidos para los usuarios ( Figura 5 ). Esta operación permite verificar el estado de las variables de instancia ( "campos" o "fields" ) de objetos. Trate de crear un objeto con algunos valores definidos por el usuario ( e.j. un objeto *Staff* que tiene parámetros en el constructor ). Luego seleccione *Inspect* del menú de objeto. Una ventana de diálogo aparece mostrando los campos de objeto, sus tipos y sus valores ( Figura 9 ).

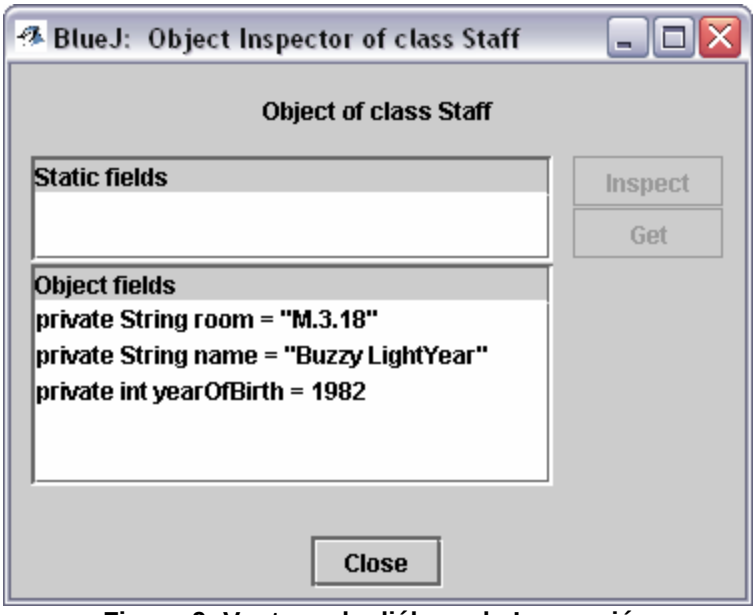


**Figura 9: Ventana de diálogo de Inspección**

La inspección es útil para verificar rápidamente que una operación mutadora ( una operación que cambia el estado de un objeto ) fue ejecutada correctamente. Así, la inspección es una herramienta simple de traceo.

En el ejemplo con la clase *Staff*, todos los campos son tipos de datos simples ( cualesquiera que no sean objetos o cadenas de caracteres ). El valor de estos tipos puede ser mostrado directamente. Usted puede ver inmediatamente si el constructor ha realizados las asignaciones correctas.

En casos más complejos, los valores de los campos pueden ser referencias a objetos definidos por el usuario. Para un ejemplo de tal caso usaremos otro proyecto. Abra el proyecto *people2*, el cual se incluye también en la distribución estándar de BlueJ. La ventana del escritorio de *people2* se muestra en la Figura 10. Como usted puede ver, este segundo ejemplo tiene una clase *Address* además de las clases vistas previamente. Uno de los campos en la clase *Person* es del tipo *Address,* definido por el usuario.

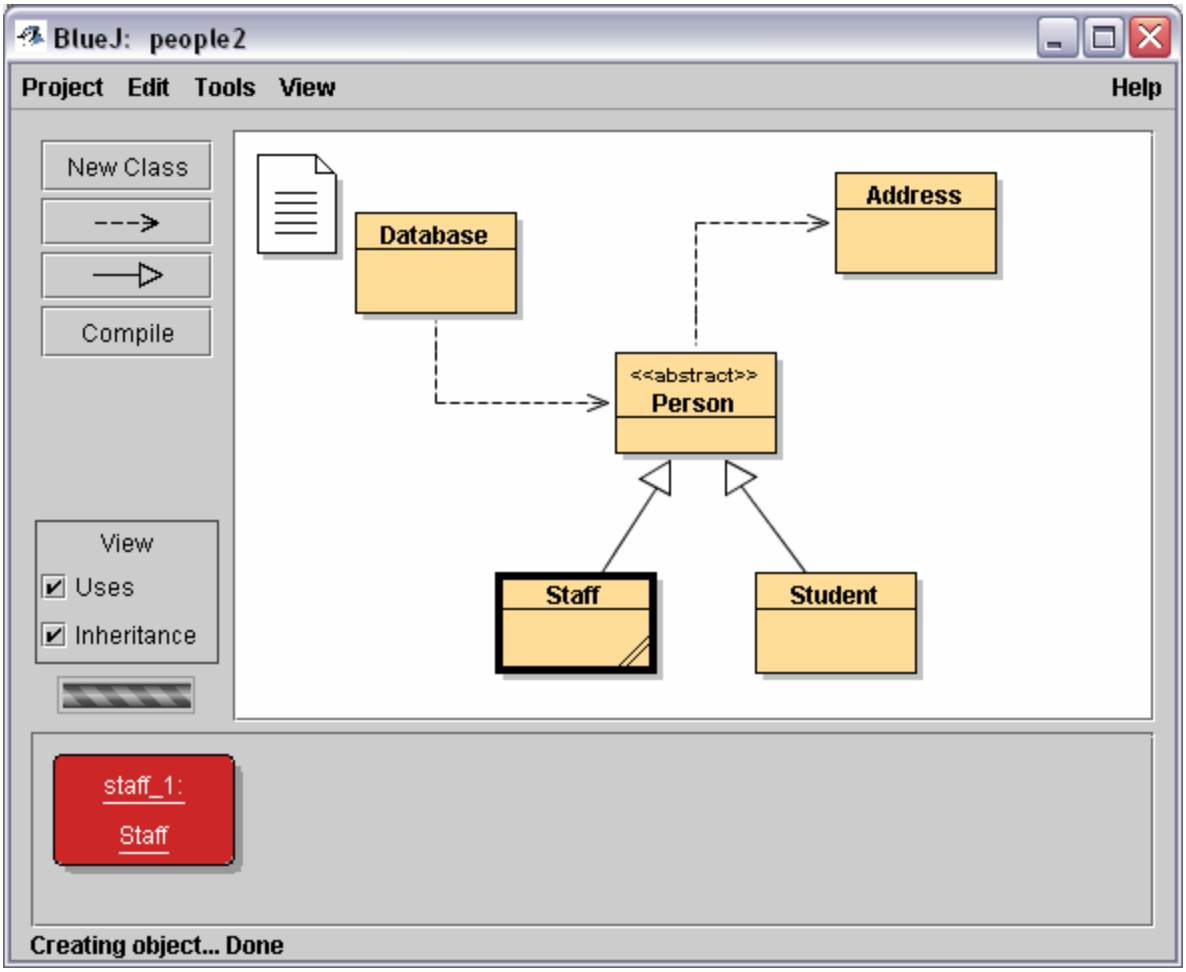


**Figura 10: La Ventana del proyecto *people2***

Para lo siguiente que queremos probar – inspección con campos de objeto – cree un objeto *Staff* y luego llame el método *setAddress* de este objeto ( usted lo encontrará en el submenú de los métodos heredados de *Person* ). Digite una dirección. Internamente, el código de *Staff* crea un objeto de clase *Address* y lo almacena en su campo *address*.

Ahora, inspeccione el objeto *Staff*. La ventana de diálogo de la inspección resultante se muestra en la Figura 11. Los campos dentro del objeto *Staff* ahora incluyen *address*. Como usted puede ver, su valor se muestra como <*object reference*> – debido a que este objeto es complejo, definido por el usuario, su valor no puede ser mostrado directamente en esta lista. Para examinar luego, seleccione el campo *address* en la lista y haga clic en el botón *Inspect* en la ventana de diálogo. ( Usted también puede hacer

doble clic en el campo *address*. ) Otra ventana de inspección se desplegará, mostrando los detalles del objeto *Address* ( Figura 12 ).

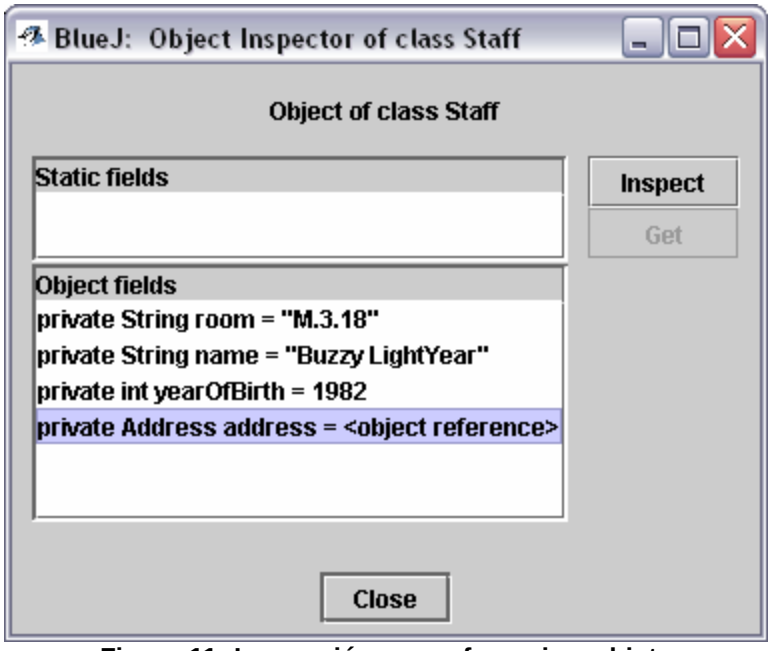


**Figura 11: Inspección con referencia a objeto**

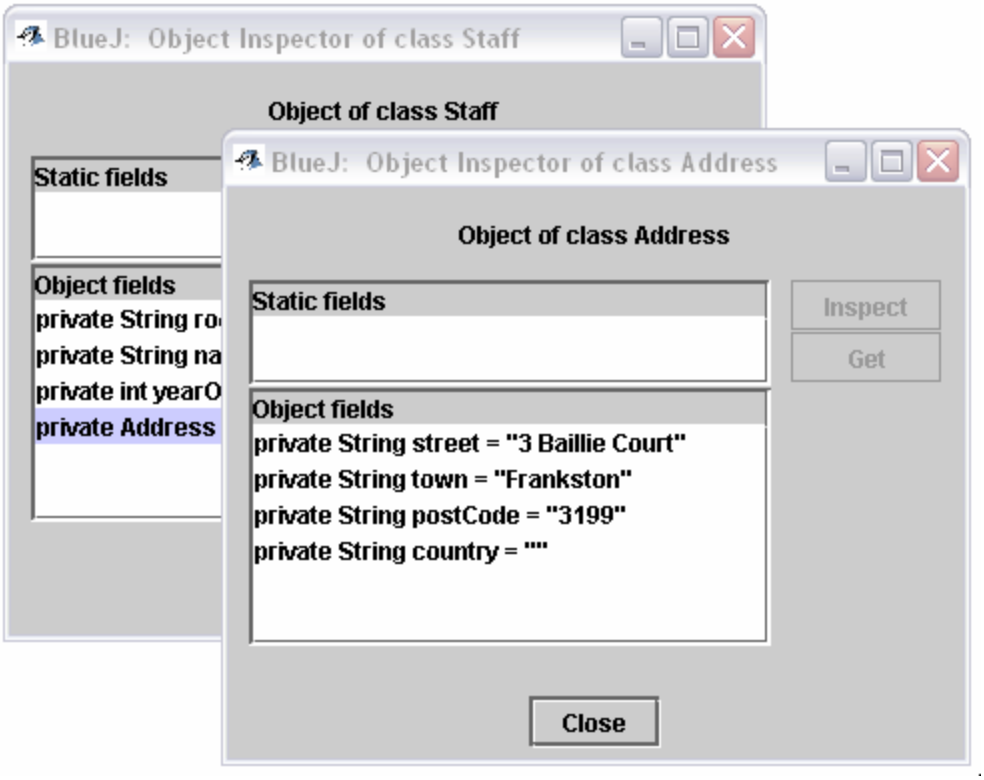


**Figura 12: Inspección de objeto interno**

Si el campo seleccionado es público, entonces en vez de hacer clic en *Inspect*, usted también podría seleccionar el campo *address* y hacer clic en el botón *Get*. Esta operación coloca el objeto seleccionado en el área de objeto. Allí usted puede examinarlo más a fondo haciendo llamados a sus métodos.

## 4.2 Pasando objetos como parámetros

*Resumen:* La inspección de un objeto permite un simple traceo, verificando el estado interno de ese objeto.

Los objetos pueden ser pasados a los métodos de otros objetos. Probemos con un ejemplo. Cree un objeto de la clase *Database*. ( Usted notará que la clase *Database* tiene sólo un constructor, el cual no tiene parámetros, así la construcción de un objeto es fácil y rápida. ) El objeto *Database* tiene la habilidad de almacenar una lista de personas. El tiene operaciones para agregar objetos de tipo *Person* y mostrar todas las personas almacenadas actualmente. ( Llamarlo actualmente *Database* es un poco exagerado! )

Si usted no tiene ya un objeto *Staff* o *Student* en el área de objeto, cree uno de estos objetos también. Para lo siguiente, usted debe tener al mismo tiempo un objeto *Database* y un objeto *Staff* o *Student* en el área de objetos.

Ahora haga el llamado al método *addPerson* del objeto *Database*. La signatura le indica que se espera un parámetro de tipo *Person*. ( Recuerde: la clase *Person* es abstracta, entonces no hay objetos directamente instanciados del tipo *Person*. Sino que, debido a la herencia, los objetos *Student* y *Staff* se pueden sustituir por objetos *Person*. Entonces es permitido un *Student* o *Staff* donde se espera un objeto *Person*. ) Para pasar el objeto, el cual usted tiene en el área de objetos, como parámetro al llamado que usted está haciendo, usted puede digitar su nombre en el campo del parámetro o, como un atajo, haga clic en el objeto. Esto agrega el nombre dentro de la ventana de diálogo del método. Haga clic en el botón *OK* y luego el llamado se realiza. Debido a que no hay valor retornado para este método, no vemos el resultado inmediatamente. Usted puede llamar el método *listAll* sobre el objeto *Database* para verificar que la operación realmente se produjo. La operación *listAll* escribe la operación de la persona en la salida estándar. Usted notará que una ventana de terminal de texto se abre automáticamente para mostrar el texto.

Haga de nuevo esto con más de una persona almacenada en “database”.

# 5 Creando un nuevo proyecto

Este capítulo lo conduce por un tour rápido sobre la configuración de un nuevo proyecto.

## 5.1 Creando el directorio del proyecto

*Resumen:* Para crear un proyecto, seleccionar *New...* del menú *Project*.

Para crear un nuevo proyecto, seleccione Project – New... del menú. Una ventana de diálogo de selección de archivo se abre para dejarle especificar el nombre y la localización del nuevo proyecto. Trate de hacer esto ahora. Usted puede escoger cualquier nombre para su proyecto. Luego de hacer clic en el botón *OK*, un directorio será creado con el nombre que usted especificó, y la ventana principal muestra el nuevo proyecto, vacío.

## 5.2 Creando clases

*Resumen:* Para crear una clase, haga clic el botón *New Class* y especifique el nombre de la clase.

Ahora usted puede cambiar sus clases haciendo clic en el botón *New Class* ubicado en la barra de herramientas del proyecto. Se le solicitará un nombre para la clase –este nombre tiene que ser un identificador Java válido.

Usted también puede escoger entre cuatro tipos de clases: abstract, interface, applet o "standard". Esta selección determina cuál esqueleto de código se crea inicialmente para su clase. Usted puede cambiar el tipo de la clase luego, editando el código fuente ( por ejemplo, agregando la palabra clave "abstract" en el código ).

Después de crear una clase, ella es representada por un icono en el diagrama. Si ella no es una clase estándar, el tipo ( interface, abstract, o applet ) se indica en el icono de la clase. Cuando usted abre el editor para una nueva clase, usted notará que se crea un esqueleto por defecto para una clase – esto debería facilitar el comienzo del trabajo. El código por defecto es correcto sintácticamente. Ella puede ser compilada ( pero no hace mucho ). Trate de crear algunas pocas clases y compilarlas.

## 5.3 Creando dependencias

*Resumen:* Para crear una flecha, haga clic en el botón de la flecha y arrastre la flecha en el diagrama, o sólo escriba el código fuente en el editor.

El diagrama de clases muestra dependencias entre clases en la forma de flechas. Las relaciones de herencia ( "extends" o "implements" ) se muestran como flechas con cabeza triangular; las relaciones de uso, "uses", se muestran como flechas simples.

Usted puede agregar dependencias ya sea gráficamente ( directamente en el diagrama ) o textualmente en el código fuente. Si usted agrega una flecha gráficamente, se actualiza el código fuente automáticamente; si usted agrega una dependencia en el código, el diagrama se actualiza.

Para agregar una flecha gráficamente, haga clic en el botón de la flecha apropiada ( flecha con cabeza triangular para “extends” o “implements”, flecha simple para “uses” ) y arrastre la flecha de una clase a otra.

Agregando una flecha de herencia se inserta una definición “extends” o “implements” en el código fuente de la clase ( dependiendo si el objetivo es una clase o una interfaz ).

Agregando una flecha “uses” no cambia inmediatamente el código ( a menos que el objetivo sea una clase de otro paquete. En tal caso se genera una instrucción “import”, pero no hemos visto eso aún en los ejemplos ). Teniendo una flecha “uses” en el diagrama que apunte a una clase que actualmente es usada en el código, generará luego un mensaje de alerta avisando que la relación “uses” hacia una clase fue declarada pero la clase nunca fue utilizada.

Agregar las flechas en el texto es fácil: sólo digite el código como usted lo hace normalmente. Tan pronto como la clase es guardada, el diagrama se actualiza. ( y recuerde: cerrando el editor automáticamente se guarda el código. )

## 5.4 Removiendo elementos

*Resumen:* Para remover una clase, seleccione la función *Remove* de su menú emergente. Para remover una flecha, seleccione *Remove* del menú *Edit* y haga clic en la flecha.

Para remover una clase del diagrama, seleccione la clase y luego seleccione *Remove Class* del menú *Edit*. Usted también puede seleccionar *Remove* del menú emergente de la clase. Para remover una flecha, seleccione *Remove Arrow* del menú y luego seleccione la flecha que desee remover.

# 6 Traceando

Esta sección introduce los aspectos más importantes de la funcionalidad de traceo en BlueJ. En charlas con profesores de computación, nosotros hemos escuchado muy a menudo el comentario que el uso de un traceador en primer año de enseñanza sería recomendable, pero no hay tiempo suficiente para introducirlo. Los estudiantes trabajan con el editor, compilador y la ejecución; no existe más tiempo para introducir otra herramienta complicada.

Esta es la razón por la que hemos decidido hacer el traceador tan simple como sea posible. La meta es tener un traceador que usted pueda explicar en 15 minutos, y que los estudiantes sólo lo usen luego o en posteriores instrucciones. Observemos si se tuvo éxito.

Primero que todo, hemos reducido la funcionalidad de los traceadores tradicionales a tres tareas:
- configurar puntos de parada ( breakpoints )
- avanzar paso a paso a través del código
- inspeccionar variables

Por lo tanto, cada una de estas tres tareas es muy simple. Ahora probaremos cada una de ellas.

Para comenzar, abra el proyecto *debugdemo*, el cual está incluido en el directorio *examples* en la distribución de Bluej. Este proyecto contiene unas pocas clases para el único propósito de demostrar la funcionalidad del traceador – de otra manera este proyecto no tendría mucho sentido.

## 6.1 Configurando puntos de parada

*Resumen:* Para configurar un punto de parada ( breakpoint ), haga clic en el área del punto de parada a la izquierda del texto del editor.

Configurar un punto de parada le permite interrumpir la ejecución en cierto punto en el código. Cuando la ejecución se interrumpe, usted puede investigar el estado de sus objetos. Esto a menudo le ayuda a entender qué está pasando en su código.

En el editor, a la izquierda del texto, está el área de puntos de parada ( Figura 13 ). Usted puede configurar un punto de parada haciendo clic dentro de esta área. Un pequeño signo *stop* aparece para marcar el punto de parada. Pruebe esto ahora. Abra la clase *Demo*, encuentre el método *loop*, y configure un punto de parada en algún lugar dentro del ciclo *for*. El signo *stop* debería aparecer en su editor.

```
public int loop(int count)
{
        int sum = 17;

    for (int i=0; i<count; i++) {
        sum = sum + i;
        sum = sum - 2;
    }
    return sum;
}
```

**Figura 13: Punto de parada**

Cuando una línea de código es alcanzada y tiene un punto de parada asociado, entonces la ejecución es interrumpida. Pruébelo ahora.

Cree un objeto de la clase *Demo* y llame al método *loop* con un parámetro, por ejemplo, 10. Tan pronto como el punto de parada se alcanza, la ventana del editor emerge, mostrando la línea de código actual, y una ventana del traceador emerge. Se verá algo como la Figura 14.

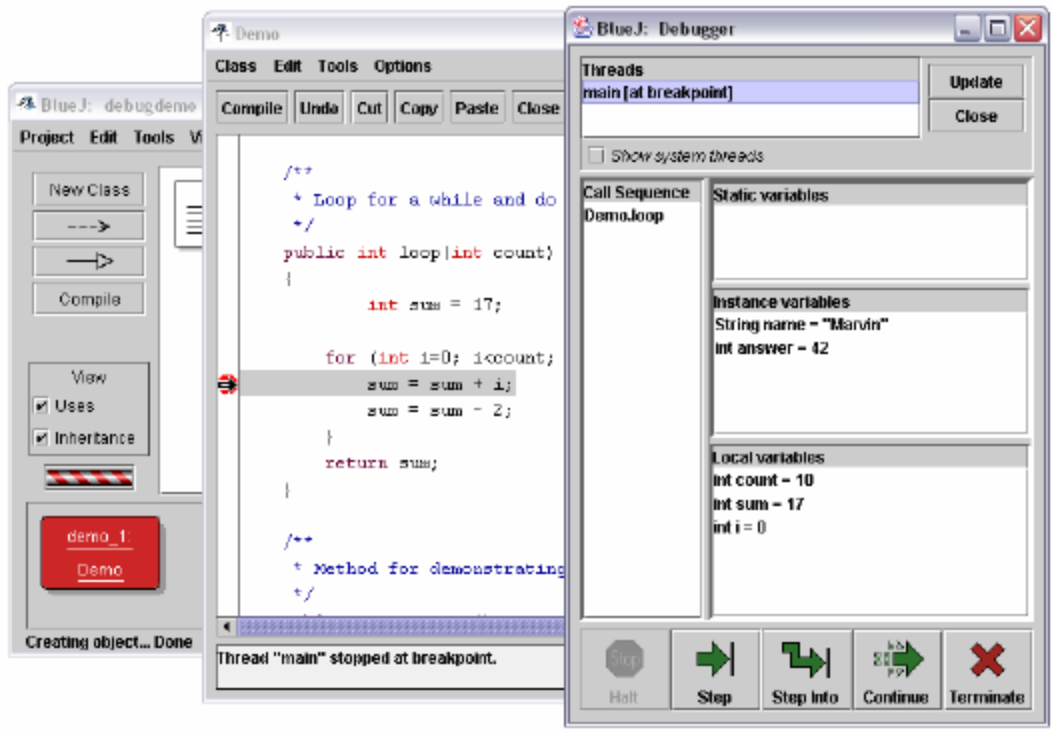


**Figura 14: Ventana del Traceador**

La línea marcada en el editor indica la línea que será ejecutada luego. ( La ejecución se detiene *antes* que esta línea sea ejecutada. )

## 6.2 Paso a paso a través del código

*Resumen:* Para ir paso a paso a través del código, use los botones *Step* y *Step Into* del traceador.

Ahora que hemos detenido la ejecución ( la cual nos convence que el método realmente se puede ejecutar y que este punto en el código se alcanza ), podemos avanzar con pasos simples a través del código y ver cómo progresa la ejecución. Para hacer esto, haga clic repetidamente en el botón *Step* dentro de la ventana del traceador. Usted debería ver la línea de código cambiando dentro del editor ( la marca de línea se mueve con la línea que se ejecuta ). Cada vez que usted hace clic en el botón *Step*, una línea simple de código se ejecuta y la ejecución se detiene de nuevo. Note también que los valores de las variables mostradas en la ventana del traceador cambian ( por ejemplo el valor de *sum*. ) Así usted puede ejecutar paso a paso y observar qué pasa. Cuando se canse de esto, usted puede hacer clic en el punto de parada de nuevo para removerlo, y luego clic en el botón *Continue* dentro de la ventana del traceador para reiniciar la ejecución y continuar normalmente.

Probemos de nuevo con otro método. Configure un punto de parada en la clase *Demo*, método *carTest( )*, en la línea que dice

```
places = myCar.seats(  );
```

Llame al método. Cuando se alcanza el punto de parada, usted está a punto de ejecutar una línea que contiene un llamado a un método, el método *seats( )* en la clase *Car*. Haciendo clic en *Step* se podría pasar sobre la línea completa. Probemos en este momento *Step Into*. Si usted pasa dentro de un llamado al método, entonces entra en el método y ejecuta las líneas pertenecientes a ese método línea por línea ( no como un simple paso ). En este caso, usted se encuentra dentro del método *seats( )* en la clase *Car*. Usted ahora puede pasar felizmente a través de este método hasta que alcance el final y retorne al método que lo llamó. Note cómo el traceador muestra los cambios.

*Step* y *Step Into* se comportan igual si la línea actual no contiene llamados a métodos.

## 6.3 Inspeccionando variables

*Resumen:* Inspeccionar variables es fácil – ellas son mostradas automáticamente en el traceador.

Cuando usted tracea su código es importante estar habilitado para inspeccionar el estado de sus objetos ( variables locales y variables de instancia ).

Hacer esto es trivial – la mayoría de cosas ya las ha visto. Usted no necesita comandos especiales para inspeccionar variables, variables estáticas, variables de instancia del objeto actual y variables locales del método actual, siempre todas son automáticamente mostradas y actualizadas.

Usted puede seleccionar métodos en la secuencia de llamado para ver variables de otros métodos y objetos activos actualmente. Pruebe, por ejemplo, un punto de parada en el método *carTest( ),* de nuevo. En el lado izquierdo de la ventana del traceador, usted ve la secuencia de llamado. El actualmente muestra

```
Car.seats
Demo.carTest
```

Esto indica que `Car.seats` fue llamada por `Demo.carTest`. Usted puede seleccionar `Demo.carTest` en esta lista para inspeccionar el código fuente y los valores actuales de las variables en este método.

Si usted pasa por la línea que contiene la instrucción `new Car( ... )`, usted puede observar que el valor de la variable local `myCar` se muestra como <*object reference*>. Todos los valores de tipo objeto ( excepto para los String ) se muestran de esta manera. Usted puede inspeccionar esta variable haciendo doble clic sobre ella. Haciendo esto abrirá una ventana de inspección de objeto identica a la descrita anteriormente ( sección 4.1 ). No existe diferencia real entre inspeccionar objetos aquí e inspeccionar objetos en el área de objetos.

## 6.4 Detener y terminar

*Resumen:* Detener y terminar ( *Halt – Terminate* ) pueden ser utilizados para detener una ejecución temporal o permanentemente.

Algunas veces un programa está ejecutándose por mucho tiempo, y usted debe adivinar si todo está funcionando bien. Quizás existe un ciclo infinito, quizás este proceso toma mucho tiempo. Bien, podemos verificar. Llame el método *longloop( )* de la clase *Demo*. Esta ejecución demora un momento.

Ahora queremos saber qué es lo que pasa. Muestre la ventana del traceador, si aún no se encuentra en la pantalla. ( De otro modo, haga clic en el símbolo cambiante que indica que la máquina se está ejecutando, este es un atajo para mostrar el traceador. )

Ahora haga clic en el botón *Halt*. La ejecución se interrumpe tal como si tuviéramos un punto de parada. Usted ahora puede pasar algunas líneas, observe las variables, y vea que todo está bien – esto necesita sólo un poco más de tiempo para completarse. Usted puede escoger sólo algunas veces *Continue* y *Halt* para ver qué tan rápido está contando. Si usted no desea seguir ( por ejemplo, usted ha descubierto que se encuentra realmente en un ciclo infinito ) usted puede sólo presionar *Terminate* para terminar la ejecución total. *Terminate* no debería ser utilizado muy frecuentemente – usted puede dejar objetos escritos perfectamente en un estado consistente para terminar la máquina, así que es aconsejable usarlo sólo como un mecanismo de emergencia.

# 7 Creando aplicaciones de escritorio

*Resumen:* Para crear una aplicación de escritorio, use Project - Export...

BlueJ puede crear archivos jar ejecutables. Los archivos jar ejecutables pueden ser ejecutados en algunos sistemas hacienda doble clic en el archivo ( por ejemplo en Windows y MacOS X ), o utilizando el comando `java -jar <file-name>.jar` ( Unix o DOS prompt ).

Probaremos esto con el proyecto de ejemplo *hello*. Abralo ( está en el directorio *examples* ). Asegúrese de que el proyecto está compilado. Seleccione la función *Export...* del menú *Project*.

Una ventana de diálogo se abre para dejarle especificar el formato de almacenamiento ( Figura 15 ). Escoja "jar file" para crear un archivos jar ejecutable. Para crear el archivo jar ejecutable, usted también tiene que especificar una clase principal “main class”. Esta clase debe tener definido un *método main* válido ( con la signatura `public static void main( String[] args )` ).

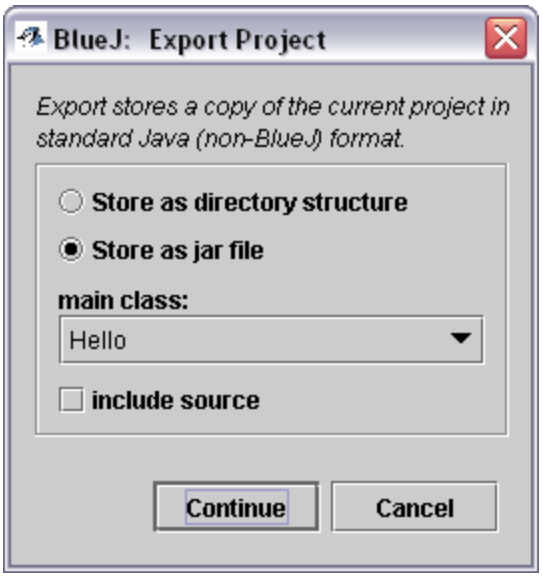


**Figura 15: La ventana de diálogo para exportar ( "Export" )**

En nuestro ejemplo, la selección de la clase principal es fácil: sólo existe una clase. Seleccione *Hello* del menú emergente. Si usted tiene otros proyectos, seleccione la clase que tiene el método "main" que usted desea ejecutar.

Usualmente, usted no debería incluir los archivos fuente en archivos ejecutables. Pero usted puede, si también desea distribuir sus archivos fuente.

Haga clic en *Continue*. Luego, usted verá una ventana de diálogo para selección de archivos que le permite especificar un nombre para el archivo jar a ser creado. Digite *hello* y haga clic en *OK*. La creación del archivo ejecutable jar se ha completado.

Usted puede hacer doble clic en el archivo jar sólo si la aplicación usa una interfaz GUI. Nuestro ejemplo usa texto de E/S, así que debemos iniciarla desde una terminal de texto. Trate de ejecutar el archivo jar ahora.

Abra una ventana de terminal o ventana de DOS. Luego vaya al directorio donde usted guardó su archivo jar ( usted debería ver un archivo *hello.jar* ). Asumiendo que Java está instalado correctamente en su sistema, entonces usted debería estar en capacidad de digitar

```
java -jar hello.jar
```

para ejecutar el archivo.

# 8 Creando applets

## 8.1 Ejecutando un applet

*Resumen:* Para ejecutar un applet, seleccione Run Applet del menú emergente del applet.

BlueJ permite crear y ejecutar applets al igual que las aplicaciones. Hemos incluido un applet en el directorio *examples* en la distribución de Bluej. Primero, queremos tratar de ejecutar un applet. Abra el proyecto *appletdemo* del directorio *examples*.

Usted verá que este proyecto tiene sólo una clase; ella se llama *CaseConverter*. El icono de clase está marcado ( con la etiqueta <<applet>> ) como un applet. Luego de compilarla, seleccione el comando *Run Applet* del menú emergente de la clase.

Una ventana de diálogo emerge para permitirle hacer algunas selecciones ( Figura 16 ).

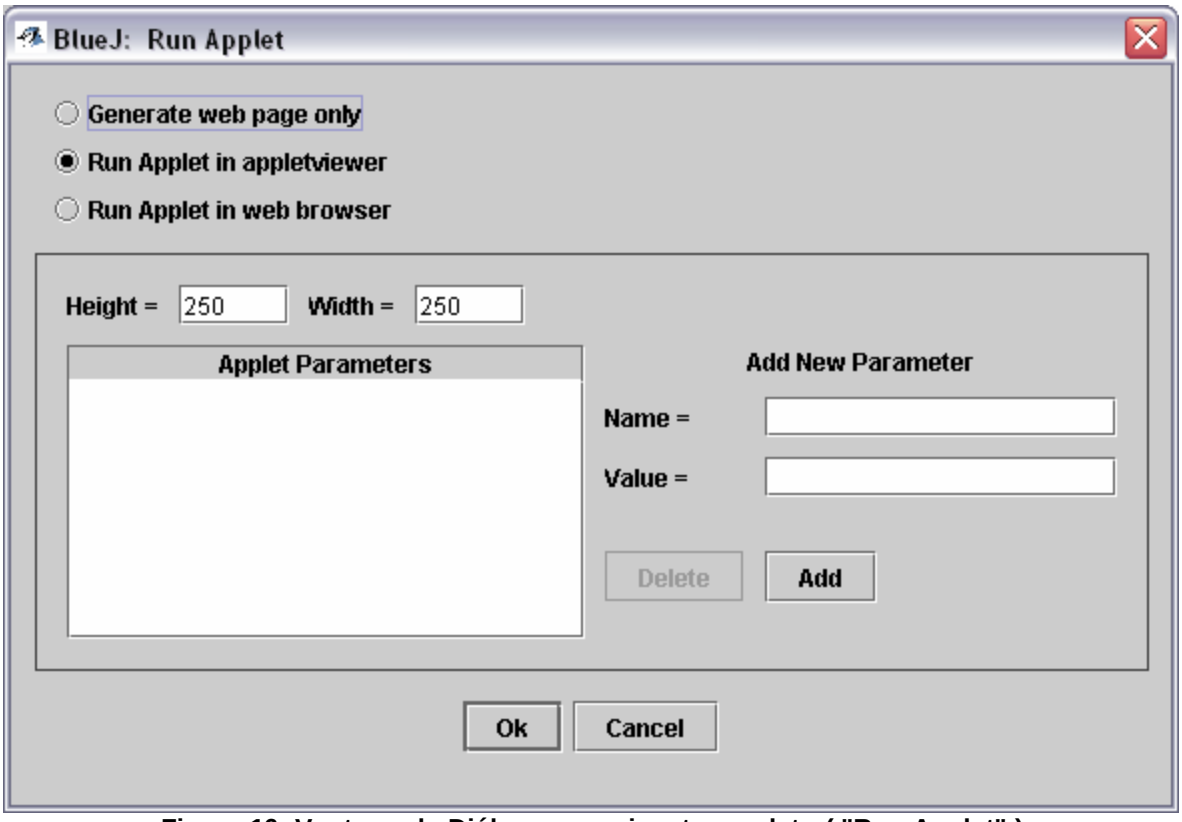


**Figura 16: Ventana de Diálogo para ejecutar applets ( "Run Applet" )**

Usted ve que tiene la oportunidad de escoger la ejecución del applet en un navegador ( browser ) o en appletviewer ( o sólo generar la página web sin ejecutarla ). Deje la configuración que viene por defecto y haga clic en *OK*. Luego de unos pocos segundos,

un visor de applets emergerá mostrando el applet de conversión de mayúsculas y minúsculas “case converter”.

El visor de applets appletviewer está instalado junto con el JDK, así se garantiza que sea la misma versión de su compilador Java. Este visor generalmente causa menos problemas que los navegadores. Su navegador web puede ejecutar una versión diferente de Java y, dependiendo de cuál versión use el navegador, puede causar problemas. Con la mayoría de los navegadores actuales debería funcionar bien, se cree.

En sistemas Microsoft Windows y MacOS, BlueJ usa su navegador por defecto. En sistemas Unix, el navegador es definido en la configuración de BlueJ.

## 8.2 Creando un applet

*Resumen:* Para crear un applet, haga clic en el botón *New Class* y seleccione *Applet* como el tipo de clase.

Luego de haber visto cómo ejecutar un applet, queremos crear uno propio.

Cree una nueva clase con *Applet* como el tipo de clase ( usted puede seleccionar el tipo en la ventana de diálogo de *New Class* ). Compile, luego ejecute el applet. Eso es! Eso no estuvo tan mal, no es así?

Los Applets ( como otras clases ) son generados con un esqueleto de clase por defecto que contiene cierto código válido. Para los applets, este código muestra un simple applet con dos líneas de texto. Usted ahora puede abrir el editor y editar el applet para insertar su propio código.

Usted verá que todos los métodos comunes del applet están allí, cada uno con una explicación como comentario explicando su propósito. Todo el código de ejemplo está en el método *paint*.

## 8.3 Probando el applet

En algunas situaciones puede ser útil crear un objeto applet en el área de objetos ( como para las clases normales ). Usted puede hacer esto – el constructor se muestra en el menú emergente del applet. Desde el área de objeto usted no puede ejecutar el applet por completo, pero usted puede llamar algunos métodos. Esto puede ser útil para probar métodos simples que usted haya escrito como parte de la implementación de su applet.

# 9 Otras Operaciones

## 9.1 Abriendo paquetes que no son de BlueJ en BlueJ

*Resumen:* Paquetes no BlueJ pueden ser abiertos con el comando *Project*: *Open Non BlueJ….*

BlueJ le permite abrir paquetes existentes que fueron creados fuera de BlueJ. Para hacer esto, seleccione Project – Open Non BlueJ... del menú. Seleccione el directorio que contiene los archivos fuente de Java, luego haga clic en el botón Open in BlueJ. El sistema le pedirá la confirmación acerca de si usted quiere abrir este directorio.

## 9.2 Agregando clases existentes a su proyecto

*Resumen:* Las clases pueden ser copiadas en un proyecto desde afuera utilizando el comando *Add Class from File... .*

A menudo, usted quiere usar clases que usted tenían anteriormente e incluirlas en sus proyectos de BlueJ.

Por ejemplo, un profesor puede dar el código de una clase en Java a sus estudiantes, para ser utilizada en un proyecto. Usted puede incorporar fácilmente una clase existente en su proyecto seleccionando Edit – Add Class from File... del menú. Esto le permitirá seleccionar un archivo fuente Java ( con un nombre terminado en *.java* ) para ser importado.

Cuando la clase es importada en el proyecto, una copia es tomada y almacenada en el directorio actual del proyecto. El efecto es exactamente el mismo a si usted hubiera creado esta clase y hubiese escrito todo su código fuente.

Una alternativa es agregar el código fuente de la nueva clase al directorio del proyecto desde fuera de BlueJ. La próxima vez que usted abra este proyecto, la clase será incluida en el diagrama del proyecto.

## 9.3 Llamando a *main* y a otros métodos estáticos

*Resumen:* Los métodos estáticos ( *static* ) pueden ser llamados desde el menú emergente de la clase.

Abra el proyecto *hello* del directorio *examples*. La única clase existente en el proyecto ( clase *Hello* ) define un método *main* estándar.

Haga clic derecho sobre la clase, y usted verá que el menú de la clase incluye no solamente el constructor de la clase, sino también el método estático *main*. Usted puede ahora llamar directamente el método *main* desde este menú ( sin crear primero un objeto, como se esperaría para un método estático ).

Todos los métodos estáticos pueden ser llamados como el descrito anteriormente. El método estándar *main* espera un arreglo de String como argumento. Usted puede pasar un arreglo de String, usando la sintaxis estándar de Java para arreglos de constantes. Por ejemplo, usted podría pasar

```
{"one", "two", "three"}
```

( incluyendo las llaves ) al método. Pruébelo!

> Nota al lado: *En Java estándar, los arreglos de constantes no pueden ser utilizados como argumentos actuales para los llamados a métodos. Ellos sólo pueden ser usados como inicializadotes. En BlueJ, para habilitar llamados interactivos de los métodos estándar main, permitimos el paso de arreglos de constantes como parámetros.*

## 9.4 Generando la documentación

*Resumen:* Para generar la documentación para un proyecto, seleccione *Project Documentation* del menú *Tools*.

Usted puede generar la documentación para su proyecto e.j. el formato estándar de *javadoc* desde BlueJ. Para hacer esto seleccione en el menú Tools - Project Documentation. Esta función generará la documentación para todas las clases en un proyecto desde el código fuente de las clases y abre un navegador web para mostrarlo.

Usted también puede ver y generar la documentación para una clase directamente desde el editor de BlueJ. Para hacer esto, abra el editor y use el menú emergente en la barra de herramientas del editor. Cambie la selección de *Implementation* a *Interface*. Esto mostrará el estilo de documentación *javadoc* ( la interfaz de la clase ) en el editor.

## 9.5 Trabajando con librerías

*Resumen:* La API estándar de clases de Java puede ser visualizada seleccionando *Help - Java Standard Libraries*.

Frecuentemente, cuando usted escribe un programa en Java, usted tiene que referirse a las librerías estándar de Java. Usted puede abrir un navegador web mostrando la documentación de la API del JDK seleccionando Help - Java Standard Classes del menú ( si usted está en línea en internet ).

La documentación del JDK también puede ser instalada y usada localmente ( fuera de línea ). Los detalles son explicados en la sección de ayudas del sitio web de BlueJ.

## 9.6 Creando objetos desde librerías de clases

*Resumen:* Para crear objetos desde la librería de clases, use *Tools – Use Library Class*.

BlueJ también ofrece una función para crear objetos desde clases que no son parte de su proyecto, sino definidas en una librería. Usted puede, por ejemplo, crear objetos de la clase String o ArrayList. Esto puede ser muy útil para una experimentación rápida con estos objetos de librerías.

Usted puede crear un objeto de librería seleccionando Tools – Use Library Class... del menú. Una ventana de diálogo emerge solicitando que sea digitado un nombre de clase completamente válido, tal como *java.lang.String*. ( Note que usted debe digitar el nombre completamente válido, esto es, el nombre incluyendo los nombres de los paquetes que contienen la clase. )

El campo de entrada tiene asociado un menú emergente mostrando las clases usadas recientemente. Después que un nombre de clase ha sido entrado, presionando *Enter* se mostrarán todos los constructores y métodos estáticos de esta clase en una lista en la ventana de diálogo. Cualquiera de estos constructores o métodos estáticos ahora pueden ser invocados seleccionándolos desde esta lista.

La invocación procede como cualquier otro constructor o llamado a un método.

# 10 Sólo los resúmenes

*Comenzando*

1. Para abrir un proyecto, seleccione *Open* del menú *Project*.
2. Para crear un objeto, seleccione un constructor del menú emergente de la clase.
3. Para ejecutar un método, selecciónelo del menú emergente del objeto.
4. Para editar el archivo fuente de una clase, haga doble clic sobre su icono de clase.
5. Para compilar una clase, haga clic en el botón *Compile* en el editor. Para compilar un proyecto, haga clic en el botón *Compile* en la ventana de proyecto.
6. Para obtener ayuda sobre un mensaje de error del compilador, haga clic en el símbolo de interrogación cercano al mensaje de error.

*Haciendo un poco más...*

7. Un objeto puede ser pasado como un parámetro a un llamado de un método haciendo clic sobre el icono del objeto.
8. La inspección de un objeto permite un simple traceo, verificando el estado interno de ese objeto.

*Creando un Nuevo proyecto*

9. Para crear un proyecto, seleccionar *New...* del menú *Project*.
10. Para crear una clase, haga clic el botón *New Class* y especifique el nombre de la clase.
11. Para crear una flecha, haga clic en el botón de la flecha y arrastre la flecha en el diagrama, o sólo escriba el código fuente en el editor.
12. Para remover una clase, seleccione la función *Remove* de su menú emergente.
13. Para remover una flecha, seleccione *Remove* del menú *Edit* y haga clic en la flecha.

*Traceando*

14. Para configurar un punto de parada ( breakpoint ), haga clic en el área del punto de parada a la izquierda del texto del editor.
15. Para ir paso a paso a través del código, use los botones *Step* y *Step Into* del traceador.
16. Inspeccionar variables es fácil – ellas son mostradas automáticamente en el traceador.
17. Detener y terminar ( Halt – Terminate ) pueden ser utilizados para detener una ejecución temporal o permanentemente.

*Creando aplicaciones de escritorio*

*18.* Para crear una aplicación de escritorio, use *Project - Export...*

*Creando applets*

19. Para ejecutar un applet, seleccione *Run Applet* del menú emergente del applet.
20. Para crear un applet, haga clic en el botón *New Class* y seleccione *Applet* como el tipo de clase.

*Otras operaciones*

21. Paquetes no BlueJ pueden ser abiertos con el comando *Project: Open Non BlueJ….*
22. Las clases pueden ser copiadas en un proyecto desde afuera utilizando el comando *Add Class from File...* .
23. Los métodos estáticos ( static ) pueden ser llamados desde el menú emergente de la clase.
24. Para generar la documentación para un proyecto, seleccione *Project Documentation* del menú *Tools*.
25. La API estándar de clases de Java puede ser visualizada seleccionando *Help - Java Standard Libraries*.
26. Para crear objetos desde la librería de clases, use *Tools – Use Library Class*.