**UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA**
**FACULTAD DE INGENIERÍA (UNIDAD MEXICALI)**
**DOCUMENTO DEL SISTEMA DE CALIDAD**

# Formato para prácticas de laboratorio

| CARRERA | PLAN DE ESTUDIO | CLAVE ASIGNATURA | NOMBRE DE LA ASIGNATURA |
|---|---|---|---|
| IC | 2003-1 | 2531 | Programación Orientada a Objetos |

| PRÁCTICA No. | LABORATORIO DE | | DURACIÓN (HORA) |
|---|---|---|---|
| 5 | NOMBRE DE LA PRÁCTICA | Ingeniero en Computación y Licenciado en Sistemas Computacionales<br>Entorno de Desarrollo Integrado *Eclipse* | 2 |

## 1 INTRODUCCIÓN

Los entornos de desarrollo integrados, o IDEs por sus siglas en inglés, son herramientas útiles para los desarrolladores de software ya que facilitan las tareas del ciclo de desarrollo. Esto es, sin un IDE, el desarrollador debe cargar primero un editor de texto para escribir su código fuente, después emplear un compilador para crear el código que se ejecutará y posteriormente ejecutar el código generado. Cuando se emplea un IDE, todos los pasos se pueden realizar desde un mismo programa. Adicionalmente, los IDEs pueden facilitar el proceso de depuración de código. Algunos IDEs modernos tambien proveen la posibilidad de ayudar al desarrollador en la escritura del código.

La herramienta que se estudiará en esta practica tiene como nombre Eclipse y es una herramienta que se emplea por desarrolladores profesionales. Esta herramienta no solo puede emplearse para el desarrollo de programas en Java sino también en C y otros lenguajes, incluso puede extenderse por medio de plugins para soportar practicamente cualquier tipo de desarrollo.

## 2 OBJETIVO (COMPETENCIA)

El alumno empleará el entorno de desarrollo integrado Eclipse para desarrollar programas en Java.

| Formuló | Revisó | Aprobó | Autorizó |
|---|---|---|---|
| Cecilia Curlango Rosas | MC. Gloria Etelbina Chavez Valenzuela y LSC Monica Lam Mora | | MC. MIGUEL ANGEL MARTINEZ ROMERO |
| Maestro | Coordinador de la Carrera | Gestión de la Calidad | Director de la Facultad |

## 3 FUNDAMENTO

*Eclipse* es una herramienta de desarrollo integrado (IDE) de gratuita de código abierto desarrollada por la Eclipse Foundation. Al estar escrito en Java, Eclipse es indenpendiente de plataforma. Puede descargarse la versión más reciente de este IDE de http://www.eclipse.org donde además se encuentran manuales y articulos sobre esta herramienta.

Para ejecutar este IDE en las máquinas del laboratorio, basta con escribir *eclipse* en la línea de mandos. Ésto provocará que se inicie la ejecución de Eclipse. Cuando se ejecuta por primera vez, aparece una caja de dialogo como la que se ve en la Figura 1: Selección de espacio de trabajo. Aquí debemos especificar el nombre del directorio en el que almacenaremos todos los proyectos y programas que estaremos desarrollando. Además para evitar que se haga esta pregunta cada vez que iniciemos Eclipse, se puede seleccionar la opción indicada.


*Figura 1: Selección de espacio de trabajo.*

Una vez que hemos ingresado a Eclipse, se ve el espacio principal de trabajo que inicialmente se encuentra vacio. A continuación se describirá el proceso de crear una aplicación sencilla utilizando Eclipse.

### Planteamiento del Problema y Resolución

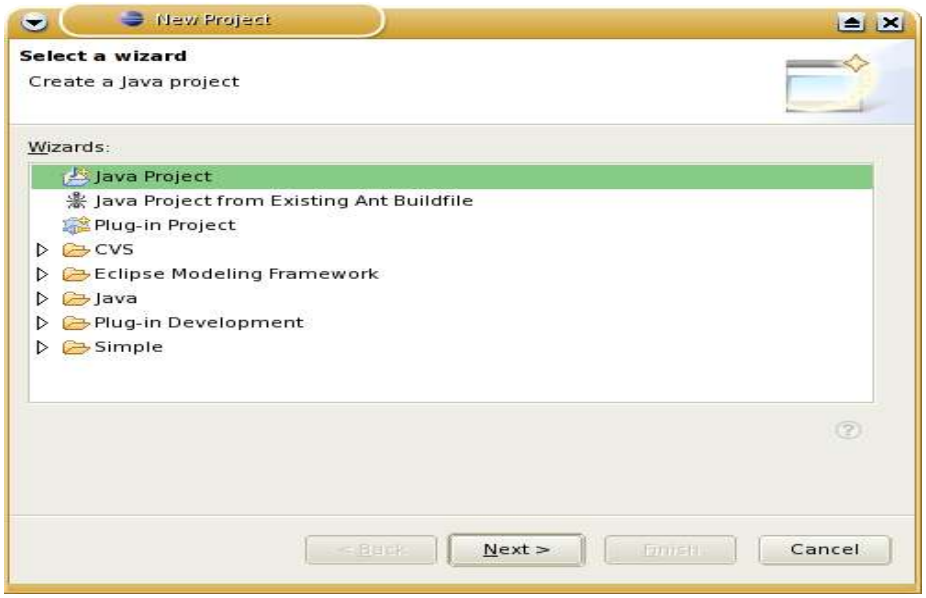
La aplicación ejemplo consistirá en desarrollar un sistema para trabajar con rectas en el plano cartesiano. Para ello, se tendrá una clase Punto, una clase Recta y una clase para probar el funcionamiento de las anteriores. La Figura 2: Diagrama de Clases muestra el diagrama UML de las clases que compondrán el sistema.

## 3 FUNDAMENTO

| Punto |
| --- |
| +x: int<br>+y: int |
| +getX(): int<br>+getY(): int<br>+setX(nuevaX:int)<br>+setY(nuevaY:int)<br>+Punto(valorX:int,valorY:int)<br>+Punto() |

| Recta |
| --- |
| +puntoA: Punto<br>+puntoB: Punto |
| +getPuntoA(): Punto<br>+getPuntoB(): Punto<br>+setPuntoA(nuevoPunto:Punto)<br>+setPuntoB(nuevoPunto:Punto)<br>+Recta(A:Punto,B:Punto)<br>+Punto()<br>+calcularLongitud(): double |

Para implementarlo utilizando Eclipse, debemos primero crear un proyecto llamado en este caso Cartesiano. Para ello seleccionamos del menu principal *File-New-Project* y aparecerá una caja de dialogo como la de la Figura 3: Selección de Tipo de Proyecto. Aquí seleccionaremos simplemente *Java Project* y el boton *Next*.




A continuación aparecerá otra caja de dialogo en la que escribiremos el nombre del proyecto que es *Cartesiano* de modo que se verá como en la Figura 4: Datos del Proyecto. Una vez escrito el nombre del proyecto presionaremos el botón *Finish* y con esto estaremos listos para agregar clases a nuestro proyecto.

## 3 FUNDAMENTO

Ahora se agregarán clases al proyecto. Del menu principal, seleccionar *File-New-Class*, esto nos presentará una


caja de dialogo como la de la Figura 5: Datos de la clase Punto aqui escribiremos el nombre de la clase que será *Punto* y también nos aseguraremos que no se cree ni el método *main()* ni los constructores de la superclase. Una vez especificada la forma que tendrá la clase presionamos *Finish* y con esto se generará el código en Java según lo que especificamos.

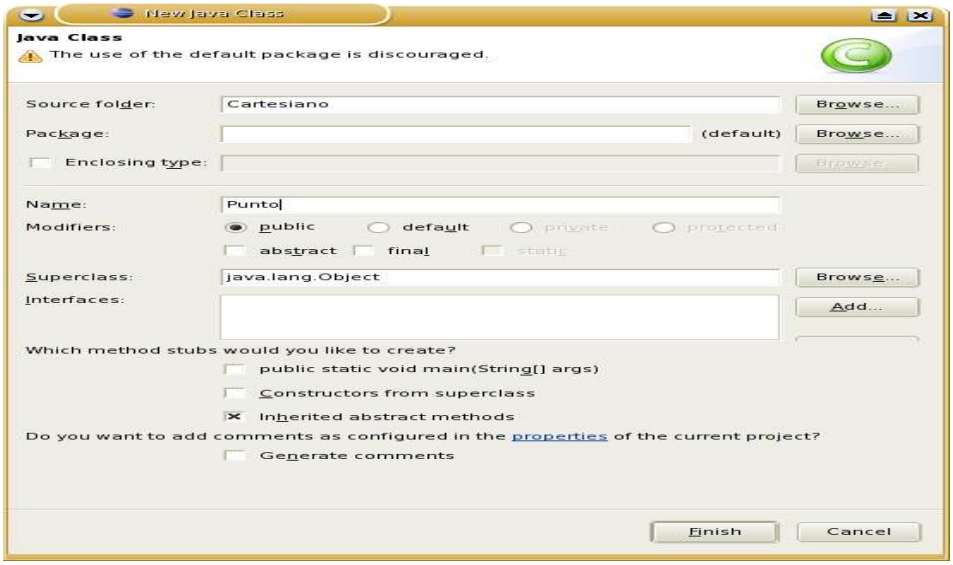
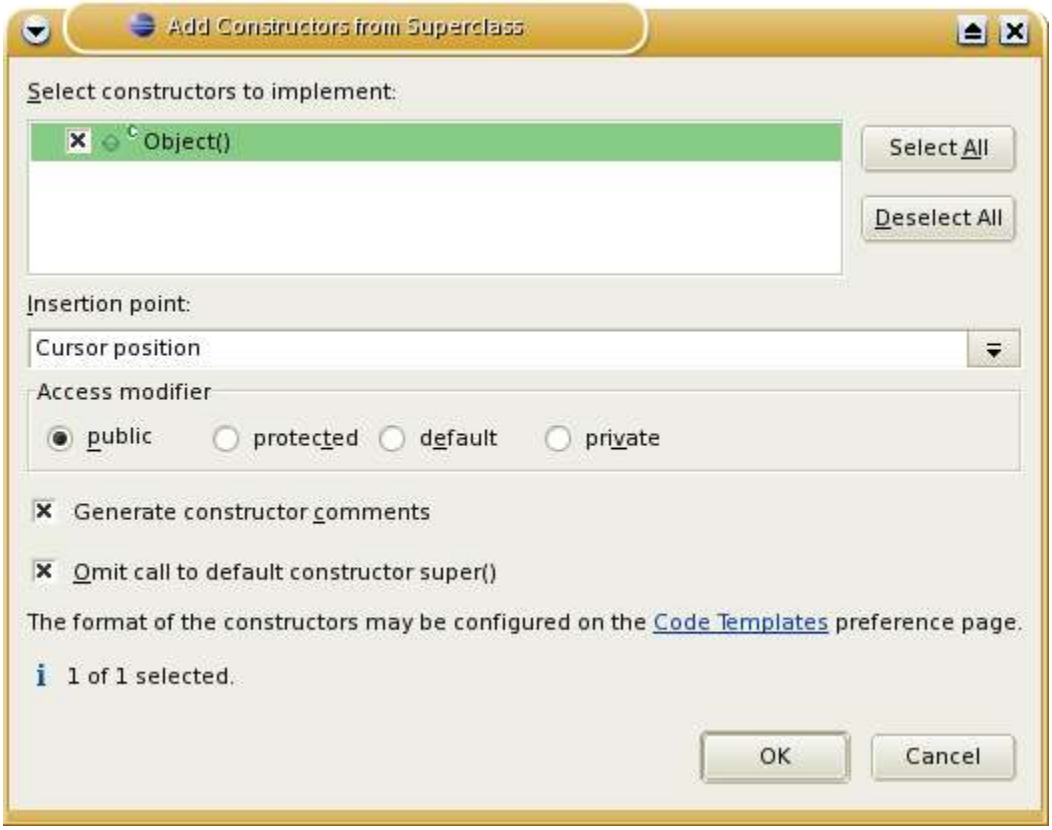



*Figura 5: Datos de la clase Punto*

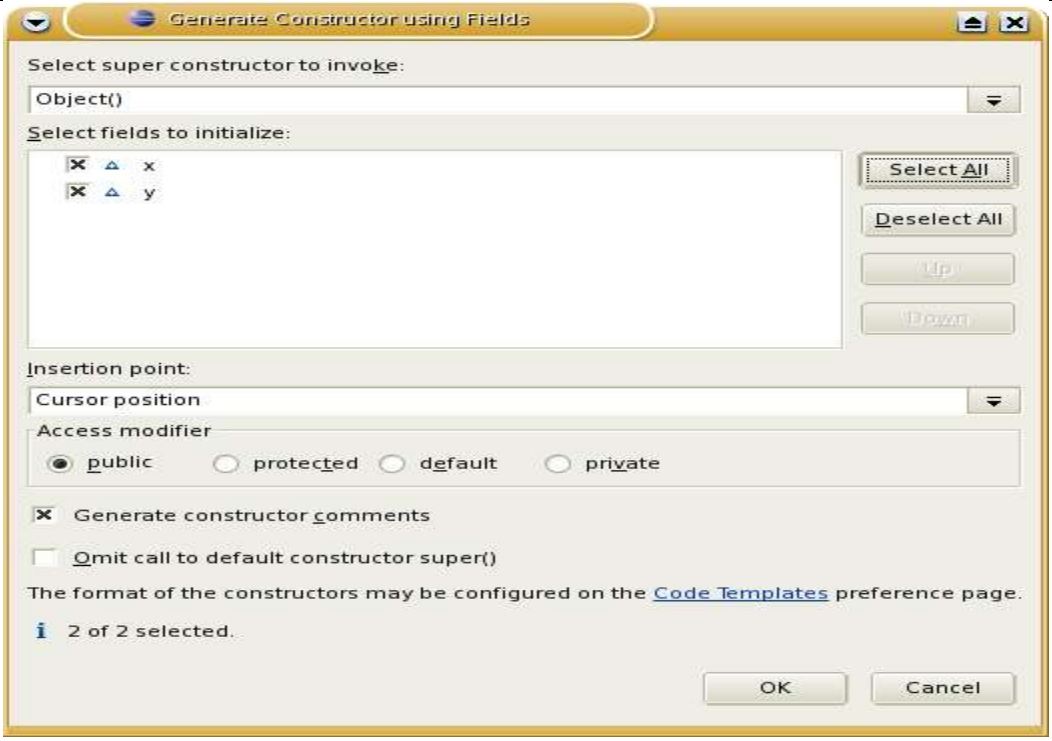
## 3 FUNDAMENTO

Como siguiente paso escribiremos los atributos de la clase, como lo hariamos en cualquier editor de texto. A continuación agregaremos los métodos constructores. El primero que escribiremos será el constructor default, esto es el que no tiene ningún parámetro. Para esto vamos a aprovechar una de las bondades de Eclipse que es la generación de código, seleccione del menu principal *Source-Add constructor from super class*. Con esto aparece la caja de dialogo de la Figura 6: Agregar constructor de la super clase. Para este caso seleccionaremos las opciones para generar comentarios y omitir la llamada al constructor *super()*.




El resultado de esta operación es que se genera el esqueleto de un constructor de nuestra clase, evitandonos el tener que escribir. Ahora nos toca escribir el código que deberá ejecutarse cuando se invoque este constructor. Este código consistirá simplemente de inicializar nuestros atributos *x, y* en 0. Así mismo escribiremos un comentario que describa lo que sucederá al invocar este método.

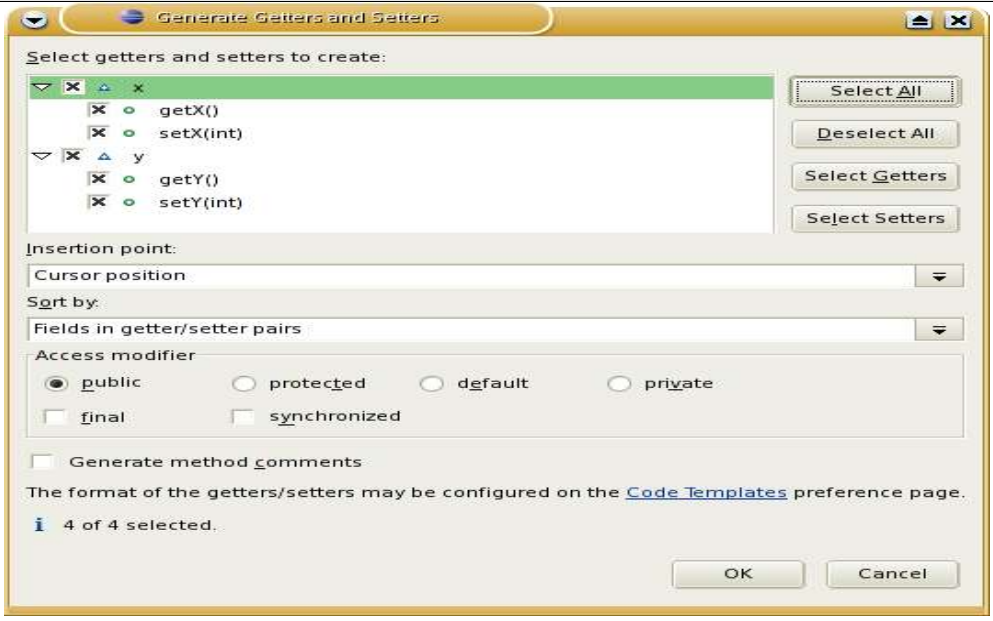
## 3 FUNDAMENTO




En seguida emplearemos una técnica similar para crear el esqueleto del segundo constructor. En este caso seleccionamos del menu principal *Source-Generate constructor using fields*. La caja de dialogo que aparecerá será similar a la Figura 7: Constructor con parámetros y aquí presionamos el boton *Select All* para que se seleccionen todos los campos que queremos que formen parte de los parámetros del constructor, que en este caso son todos los atributos que definimos para nuestra clase. Así mismo nos aseguramos que la opción de generar comentarios este seleccionada y la de invocar al constructor *super()* no lo esté. Al presionar el boton *OK* se genera el método y en este caso vemos que se agregó también codigo para asignar los parámetros del método a los atributos de la clase. Solo nos resta escribir el comentario correspondiente.

El siguiente paso agregaremos los métodos accesores, que son aquellos que tendremos disponibles para controlar el acceso a nuestros atributos. Estos métodos son de gran importancia ya que no solo nos permiten conocer/establecer el valor de los atributos sino que nos brinda la oportunidad de incluir código de validación o de realizar algún procesamiento adicional antes de asignar un valor a un atributo o antes entregarlo a quien invoque este método. Eclipse nos facilita la generación de éstos métodos y para hacerlo, solo basta con seleccionar del menu principal *Source-Generate getters and setters*. Al hacer esto, aparece la ventana de dialogo que aparece en

## 3 FUNDAMENTO




la Figura 8: Especificación de accesores. Si hacemos click en las flechita que se encuentra enseguida de los atributos, se expande el listado y nos muestra los dos métodos que podemos generar. Con el botón *Select All* seleccionamos todos los métodos mostrados y al dar *OK* se genera el código. Cabe mencionar que también se tiene una opción para generar un comentario para cada método. Nuestra tarea como desarrolladores se facilita y podemos concentrarnos en tan solo agregar código adicional por ejemplo para validar valores que recibimos en los parámetros antes de asignarlos a los atributos.

Con este último paso terminamos de crear la clase *Punto* auxiliandonos de las bondades de Eclipse. Como resultado de los pasos anteriores, tendremos una ventana que se verá aproximadamente como el de la Figura 9: Código de la clase Punto.

## 3 FUNDAMENTO

```
*Punto.java

public class Punto {
    int x;
    int y;
    /**
     * Por default todo punto es (0,0)
     */
    public Punto() {
        // TODO Auto-generated constructor stub
        x = 0;
        y = 0;
    }
    /**
     * @param x coordenada en X
     * @param y coordenada en Y
     */
    public Punto(int x, int y) {
        super();
        // TODO Auto-generated constructor stub
        this.x = x;
        this.y = y;
    }
    public int getX() {
        return x;
    }
    public void setX(int x) {
        this.x = x;
    }
    public int getY() {
        return y;
    }
    public void setY(int y) {
        this.y = y;
    }

}
```

*Figura 9: Código de la clase Punto*

Para ejecutar un programa, seleccionamos del menu *Run-Run* y aparecerá la caja de dialogo como la de la Figura 10: Ejecución de un programa y aquí deberemos asegurarnos que el nombre del proyecto y de la clase que contiene el método *main()* esta especificado. Si no lo esta, podremos utilizar los botones *Browse* para buscarlos.


*Figura 10: Ejecución de un programa*

## 3 FUNDAMENTO

Al presionar el botón *Run*, veremos la salida en la parte inferior de Eclipse en la pestaña *Console* y se verá como en la Figura 11 Ejecución de CrearLíneas.


*Figura 11: Ejecución de CrearLíneas*

Para enviar para enviar parámetros al programa que se va a ejecutar como se haría desde la línea de mandos, seleccionamos la pestaña A*rguments* y escribimos los valores separados por espacio como lo haríamos en la línea de mandos. Esto se puede ver en la Figura 12 Paso de argumentos.

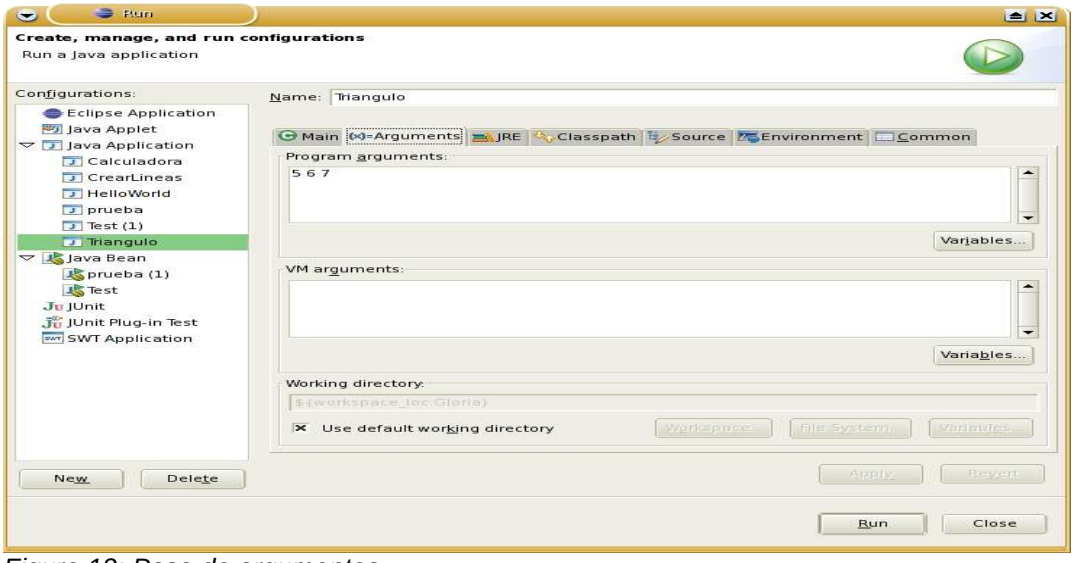



*Figura 12: Paso de argumentos*

## 3 FUNDAMENTO

Cuando se trata de un Applet, cambia un poco la forma de enviar parametros ya que estos reciben valores por medio de variables que se envian desde el archivo html. Eclipse tambien provee una forma de enviar valores a un Applet. Como se puede ve en la Figura 13 Parametros de un Applet, al dar Run-run, aparecerá la caja de dialogo acostumbrada pero en este caso tendrá una pestaña *Parameters* y será aquí donde se darán de alta.

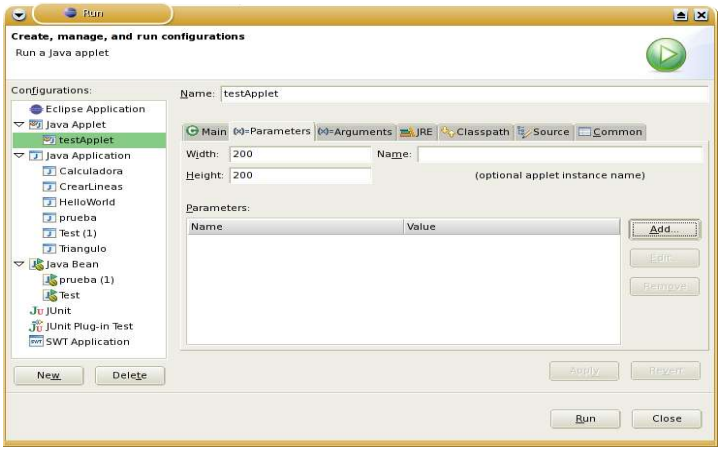



*Figura 13: Parametros de un Applet*

Para enviar escribir los parametros del Applet presionamos el botón *Add* y aparecerá una caja de dialogo como la de la Figura 14 Variables de parámetro. y aquí escribimos el nombre de la variable y su valor.

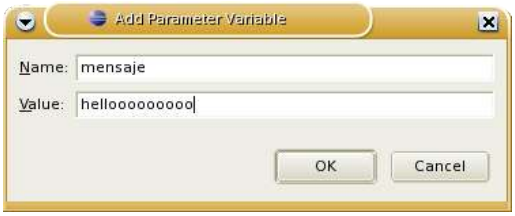



*Figura 14: Variables de parámetro.*

## 3 FUNDAMENTO

### Depuración de Programas

Una de las tareas muy común que se realiza durante la etapa de desarrollo es la de depuración de programas. Esto es, a pesar de nuestros mejores esfuerzos los programas no trabajan como deben hacerlo y debemos examinarlos durante su ejecución para ver que esta provocando nuestro error. Nuevamente, Eclipse es una herramienta poderosa que nos permitirá hacer esto de una manera fácil. Para depurar un programa, debemos seleccionar del menu principal *Window-Open Perspective-Debug*.

Esto cambiará las ventanas que se mostraban en la perspectiva anterior. (Para regresar a la perspectiva anterior la opción es *Window-Open Perspectiva-Java.*) En esta perspectiva de depuración podremos ejecutar nuestro programa una línea a la vez y esta manera podremos seguirle los pasos a nuestro programa. Este tipo de operación es de gran utilidad cuando se está aprendiendo Java por que nos permite ver exactamente cuando es que se estan ejecutando los métodos de cada clase. La Figura 15: Controles para ejecución por pasos muestra los botones que se emplean para la ejecución por pasos de los programas.

*Figura 15: Controles para ejecución por pasos*

| 4 PROCEDIMIENTO (DESCRIPCIÓN) | |
|---|---|
| **A EQUIPO NECESARIO** | **MATERIAL DE APOYO** |
| Computadora con el sistema operativo Linux, Java y Eclipse | Practica impresa |
| **B DESARROLLO DE LA PRÁCTICA** | |

## 4 PROCEDIMIENTO (DESCRIPCIÓN)

1. Seguir los pasos descritos en la sección de Fundamento para crear el proyecto *Cartesiano* y la clase *Punto*.
2. Utilizando las funciones para generación de código de Eclipse crear la clase *Línea*.
3. Crear una clase *CrearLineas* en la que se cree una *Recta* apartir de dos objetos *Punto* y se calcule y muestre la medida de esta recta. La ejecución deberá arrojar una salida como la de la Figura 12: Ejecución de CrearLineas.
4. Cambiar a la perspectiva de depuración y ejecutar el programa del inciso anterior por pasos, observando los valores que van tomando las variables.
5. Modificar los métodos de la clase *Punto* para que solo se acepten parámetros con valores positivos.
6. Cree una clase *PruebaPunto* que demuestre que las modificaciones hechas en el inciso anterior funcionan.
7. Crear una clase *Cuadrado*. Incluya un constructor que defina un cuadrado apartir de 4 objetos tipo *Punto*, un constructor que lo defina apartir de 4 objetos tipo *Recta*, y otro que cree un cuadrado con dimensiones predefinidas.
8. Cree una clase *PruebaCuadrado* que demuestre el funcionamiento de la clase *Cuadrado*.
9. Probar la clase *PruebaCuadrado* con valores escritos desde la línea de mandos.
10. Modificar la clase *PruebaCuadrado* trabaje cuando recibe valores por la línea de mandos y cuando no.

| C | CÁLCULOS Y REPORTE |
|---|---|
| | |

## 5 RESULTADOS Y CONCLUSIONES

El alumno debe obtener los resultados presentados en la practica para los programas de ejemplo, asi como explicar claramente el funcionamiento de todos los programas de la practica.

## 6 ANEXOS