



## UNIVERSIDAD AUTÓNOMA DE CAMPECHE

**NOMBRE DEL PROFESOR:** Ing. Héctor Manuel Quej Cosgaya**NOMBRE DE LA PRÁCTICA:** Polimorfismo**PRÁCTICA NÚM.** [ 6 ]

<b>LABORATORIO:</b>	Centro de Ingeniería Computacional
<b>MATERIA:</b>	Lenguaje de Programación II
<b>UNIDAD:</b>	Subcompetencia II
<b>TIEMPO:</b>	2 horas

**OBJETIVO:**

Demostrar las ventajas y posibilidades que proporciona el mecanismo del Polimorfismo a los programas que sigan el paradigma Orientado a Objetos.

**MARCO TEÓRICO:**

El Polimorfismo es el mecanismo de la POO por el cual pueden realizarse conexiones entre variables de superclases con objetos de clases descendientes. Este principio permite realizar una generalización, olvidándose de los detalles en concreto de un grupo de objetos o clases y buscar en ellos un punto en común, un ancestro el cual compartan. Gracias a esto pueden diseñarse métodos que funcionen con una gran cantidad de objetos, al mismo tiempo que cada uno mantiene su integridad y funcionamiento propios. Este mecanismo, que se basa en los otros tres, es capaz de proporcionar a los programas con una gran flexibilidad y poder, y esta práctica busca demostrarlo haciendo uso de un pequeño y sencillo panel de dibujo.

**LISTA DE MATERIALES:**

- Java SDK
- Entorno de Desarrollo Integrado NetBeans 7.0 o superior
- El proyecto de NetBeans 'Polimorfismo' comprimido dentro del archivo 'Polimorfismo.rar'

**EQUIPO DE LABORATORIO:**

- Computadora Personal

**DESARROLLO DE LA PRÁCTICA:**

1. **Descomprime** el proyecto de NetBeans "Polimorfismo", ubicado dentro del archivo "Polimorfismo.rar" que se te proporcionó junto con esta práctica.
2. **Abre** el IDE NetBeans. Normalmente hay un ícono de acceso directo en el escritorio para él.



UNIVERSIDAD AUTÓNOMA DE CAMPECHE

3. Haz clic en el botón “Abrir proyecto...” de la barra de tareas principal de NetBeans.

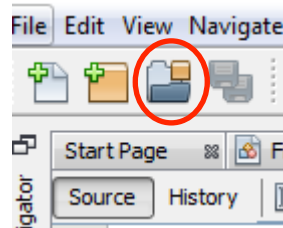


Fig. 1 – El botón “Abrir proyecto”

4. Navega en el selector de proyectos hasta el proyecto que descomprimiste en el paso 1. Selecciona “Abrir” cuando lo encuentres.
5. Observa la pestaña “Proyectos” del lado izquierdo. Si no la ves, presiona CTRL + 1.

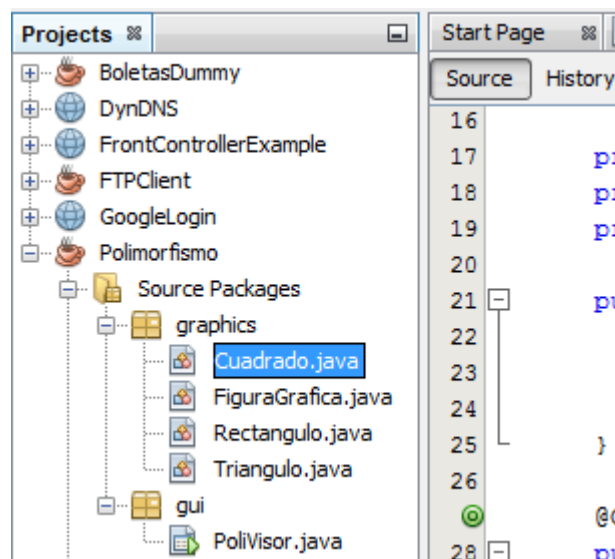


Fig. 2 – La pestaña “Proyectos” mostrando el proyecto “Polimorfismo”

6. Abre la clase “PoliVisor”, haciendo doble clic sobre ella en el panel Proyectos.



UNIVERSIDAD AUTÓNOMA DE CAMPECHE

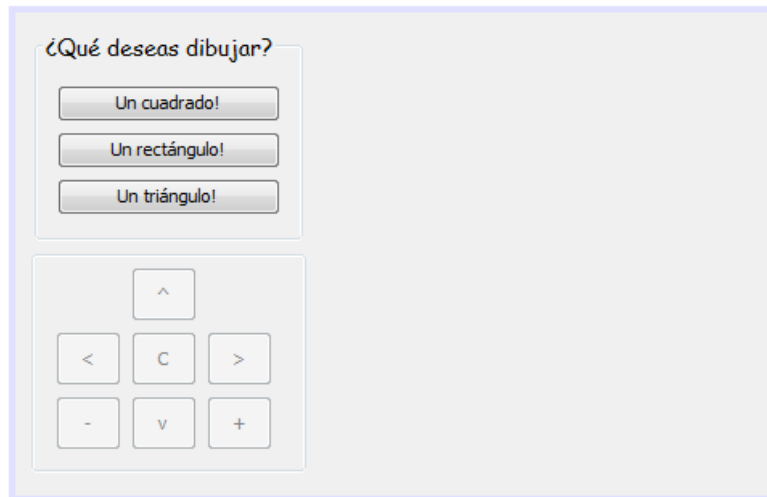


Fig. 3 – La interfaz del PoliVisor

En el estado en el que se encuentra, la clase PoliVisor únicamente puede dibujar cuadrados. Veámosla en acción:

7. **Ejecuta** la clase PoliVisor, haciendo clic derecho sobre ella en el panel Proyectos y seleccionando “Ejecutar archivo”. O por otro lado, con la clase activa, presiona SHIFT + F6.
8. **Haz clic** en el botón “Un cuadrado!” de PoliVisor. Observa que del lado derecho aparece dicha figura.
9. **Juega** un rato con el cuadrado. Utiliza los botones con las flechas para moverlo, el botón central para cambiar su color, y los botones “-” y “+” para cambiar su tamaño. Cuando termines, **cierra** la ventana PoliVisor.

¿Por qué los otros dos botones no hacen nada? ¡Fácil, porque esa es tu misión! Si no, la práctica no tendría sentido. Continuemos.

10. **Abre** la clase “FiguraGrafica.java”, haciendo doble clic sobre ella en el panel Proyectos.

La clase FiguraGrafica se supone que es la base sobre la cual se construyeron las demás figuras. Sin embargo nota que su diseño es bastante pobre, y que los métodos que la conforman son demasiado complejos. No te preocupes, por ahora te echaremos una mano. ¡Más adelante, irás solo!

11. **Declara** a la clase FiguraGrafica como **abstracta**.
12. **Declara** como **abstractos** a todos los métodos de FiguraGrafica. ¡No te olvides de borrar sus cuerpos también!



## UNIVERSIDAD AUTÓNOMA DE CAMPECHE

Ahora, 2 de las otras 3 clases marcan errores. ¡Tranquilo, nadie dijo que ser programador sería fácil! Ponte cómodo, que esto apenas comienza. Comencemos por el triángulo, que es el más difícil.

13. **Abre** la clase “Triangulo.java”, haciendo doble clic sobre ella en el panel Proyectos.
14. **Haz clic** sobre el foquito con una señal de advertencia que aparece junto a la declaración de la clase Triangulo.
15. **Selecciona** “Implementar todos los métodos abstractos” del menú emergente que aparece.

```
5  * @author Héctor Quej Cosgaya
6  * @author Jose Aguilar Canepa
7  *
8  * ¡Esta clase necesita tu ayuda!
9  */
10 public class Triangulo extends FiguraGrafica {
11     Implement all abstract methods
12     Make class Triangulo abstract
13 }
```

Fig. 4 – No te acostumbres, será solo por esta práctica.

16. **Observa** que la definición de **todos** los métodos abstractos de FiguraGrafica se agregaron a Triangulo. Esto es sólo un atajo de lo que normalmente tú tendrías que haber escrito.

Para dibujar un triángulo, necesitamos tres puntos. Implementaremos esos puntos como un conjunto de coordenadas.

17. **Agrega** las siguientes variables a la clase Triangulo

```
private int[] x = {50, 15, 85};
private int[] y = {15, 65, 65};
```

18. **Sustituye** el contenido de los métodos generados por el IDE con los que aparecen debajo:

```
public void dibujar(Graphics g) {
    g.drawPolygon(x, y, 3);
}
```

```
public void encojer(Graphics g) {
    x[1] += 5; y[1] -= 5;
    x[2] -= 5; y[2] -= 5;
    this.dibujar(g);
}
```



## UNIVERSIDAD AUTÓNOMA DE CAMPECHE

```
public void agrandar(Graphics g) {
    x[1] -= 5; y[1] += 5;
    x[2] += 5; y[2] += 5;
    this.dibujar(g);
}

public void cambiarColor(Graphics g, Color c) {
    g.setColor(c);
    this.dibujar(g);
}

public void mover(Graphics g, String direccion) {
    switch(direccion) {
        case "arriba" : y[0]-=5; y[1]-=5; y[2]-=5; break;
        case "abajo" : y[0]+=5; y[1]+=5; y[2]+=5; break;
        case "izquierda" : x[0]-=5; x[1]-=5; x[2]-=5; break;
        case "derecha" : x[0]-=5; x[1]+=5; x[2]+=5; break;
    }
    this.dibujar(g);
}
```

Eso es todo lo que nuestra clase Triangulo necesita para funcionar. ¿No nos crees?

19. De regreso a la clase PoliVisor, **haz clic** en la vista "Diseño", si es que no está seleccionada ya.
20. **Haz doble clic** sobre el botón "Un triángulo!" que aparece en la vista diseño. Regresarás a la vista Fuente y tendrás un nuevo método.
21. **Copia** este fragmento de código dentro del nuevo método que creaste (botonTrianguloActionPerformed)

```
figura = new Triangulo();
figura.dibujar(getPanelGraphics());
enableControls();
```

22. **Ejecuta** la clase PoliVisor, justo como lo hiciste en el paso 8. Si te aparece una advertencia, selecciona "Ejecutar de todas maneras".
23. En la ventana PoliVisor, **haz clic** en el botón "Un triángulo!". ¿Qué pasa?

¡Un triángulo salvaje apareció en el panel de la derecha! Sin haber modificado NADA de la clase PoliVisor (salvo la creación del nuevo objeto), ¡la nueva figura se acopla a la perfección! Este es solo una pequeña muestra del poder del **polimorfismo**.

24. **Juega** un rato con tu nuevo Triángulo. Alterna entre Cuadrado y Triángulo. Observa que el



**UNIVERSIDAD AUTÓNOMA DE CAMPECHE**

programa funciona bien con ambas. Cuando termines, **cierra** la ventana PoliVisor.

25. **Repite** los pasos 14 – 25, pero esta vez, con la clase Rectangulo. Puedes guiarte en las dos clases que ya funcionan si necesitan ayuda. ¡Sin embargo, procura aplicar un poquito de lógica!

¿Te avientas a dibujar ahora un círculo? ¿Un hexágono? ¿Una estrella? ¡Sólo tienes que preocuparte por la lógica interna de cada figura, y el PoliVisor hará el resto!

**Fin de la Práctica**

**RETROALIMENTACIÓN:**

- Investiga cuál es el método de la clase Graphics utilizado para dibujar círculos. Con este conocimiento, implementa la figura Círculo en el PoliVisor.
- Explica de qué manera se utiliza la ligadura dinámica en la clase PoliVisor.
- ¿Podría sustituirse la clase abstracta “FiguraGrafica” por una interfaz? ¿Qué cambios habrían de hacerse? ¿Cuál crees que sea la mejor estrategia?

**RECOMENDACIONES ADICIONALES:**

- Lee el capítulo 13 del Dean (Herencia y polimorfismo)

**BIBLIOGRAFÍA:**

- Dean, J. S., & Dean, R. H. (2009). Introducción a la programación con Java. México: Mc Graw Hill.
- The Java Tutorials: Polymorphism:  
<http://docs.oracle.com/javase/tutorial/java/land/polymorphism.html>
- Apuntes del profesor.