




Principios de la Tecnología de Objetos

La POO frente a la Programación Tradicional

Copyright

-  Copyright (c) 2004
José M. Ordax
-  Este documento puede ser distribuido solo bajo los
términos y condiciones de la Licencia de
Documentación de javaHispano v1.0 o posterior.
-  La última versión se encuentra en
<http://www.javahispano.org/licencias/>

Programación tradicional

- Se desarrolla a partir de procedimientos y datos.
- Los datos se estructuran con el fin de que puedan ser procesados por un conjunto de procedimientos diferentes.
- Ambos, estructuras de datos y procedimientos, están sujetos a cambios.

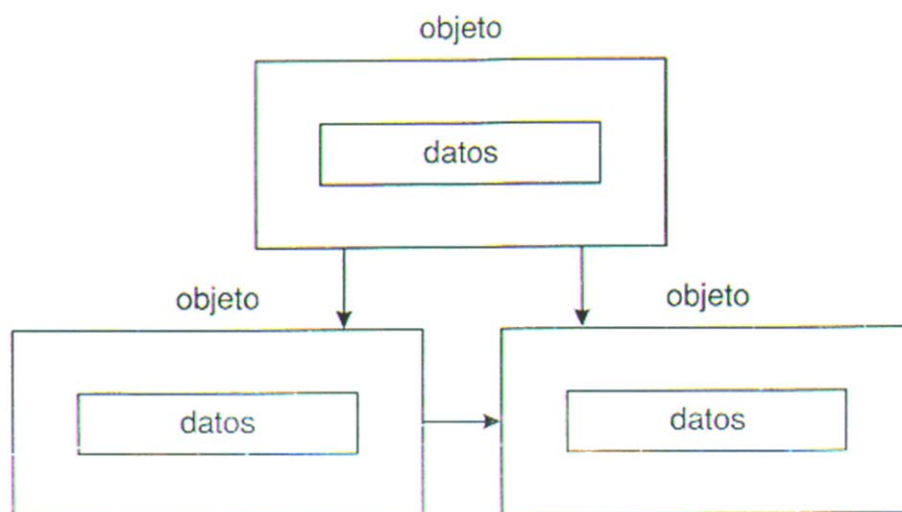
Programación tradicional



Programación O.O.

- Un programa es una colección de una sola entidad básica: el objeto.
- Este combina los datos con los procedimientos que actúan sobre ellos.
- Durante la ejecución, los objetos se envían mensajes entre si, para ejecutar las acciones requeridas.
- La organización jerárquica de los objetos en clases, permite que datos y métodos sean heredados.

Programación O.O.



Ejemplo

- Había una vez en un departamento de informática dos programadores expertos:
 - Bill Gates: programador procedural.
 - Sam Palmisano: programador de orientación a objetos.
- Un día, su director de proyecto les dio las especificaciones de un problema y prometió una subida de sueldo al programador que terminase antes el programa.

Nota: Los nombres utilizados en esta historia son ficticios, cualquier coincidencia con la vida real es casualidad.







Ejemplo

- Las especificaciones del problema eran las siguientes:
 - Necesitamos tener en una interfaz gráfica (GUI): un cuadrado, un círculo y un triángulo.
 - Cuando el usuario pulse sobre una de las figuras, está rotará 360° en el sentido de las agujas del reloj.
 - Y además, sonará una música específica (*.wav) a cada tipo de figura.









Nota: Los nombres utilizados en esta historia son ficticios, cualquier coincidencia con la vida real es casualidad.

Ejemplo

-  Bill se metió en su despacho y se puso a pensar:
 -  ¿Cuáles son las cosas que tiene que hacer este programa?
 -  ¿Qué procedimientos necesito?
-  Y el mismo se contestó:
 -  rotar
 -  sonar

Nota: Los nombres utilizados en esta historia son ficticios, cualquier coincidencia con la vida real es casualidad.

Ejemplo

-  Mientras tanto, Sam se fue a la cafetería pensando:
 -  ¿Cuáles son las cosas que hay en este problema?
 -  ¿Quiénes son los participantes?
-  El primer pensamiento fue:
 -  Las figuras: cuadrado, círculo y triángulo.
 -  Evidentemente había otros participantes como el sonido, el evento de ratón, etc... pero Sam contaba con librerías que implementaban estos componentes.

Nota: Los nombres utilizados en esta historia son ficticios, cualquier coincidencia con la vida real es casualidad.

Ejemplo



El programa de Bill se hizo muy rápido y tenía esta pinta:

```
rotar(numFigura)
{
    // Dependiendo de numFigura, rotar 360º correctamente.
}

sonar(numFigura)
{
    // Dependiendo de numFigura, buscar el archivo *.wav
    // correcto y hacerlo sonar.
}
```

Nota: Los nombres utilizados en esta historia son ficticios, cualquier coincidencia con la vida real es casualidad.

Ejemplo



Sam tardó algo mas porque escribió tres clases, una para cada figura:

Cuadrado		
	Circulo	
		Triangulo
rotar() { // Rotar 360º. }	rotar() { // Rotar 360º. }	rotar() { // Rotar 360º. }
sonar() { // Hacer sonar s	sonar() { // Hacer sonar s	sonar() { // Hacer sonar su archivo *.wav

Nota: Los nombres utilizados en esta historia son ficticios, cualquier coincidencia con la vida real es casualidad.

Ejemplo

- Bill pensaba que ya había ganado.
- Pero como siempre ocurre en todos los proyectos, se encontraron con un cambio en las especificaciones de última hora.
- Junto con las tres figuras ya comentadas, había una nueva llamada ameba que cuando el usuario pulsara sobre ella, también debía rotar 360° y hacer sonar un su música (*.aif).
- Bill se mostró muy molesto, mientras que Sam mantuvo la serenidad.




Nota: Los nombres utilizados en esta historia son ficticios, cualquier coincidencia con la vida real es casualidad.

Ejemplo

- El procedimiento rotar del programa de Bill seguía funcionando dado que dependiendo de la figura calculaba su rectángulo, punto central y rotaba.
- Sin embargo, el procedimiento sonar..... Estaba mas complicado. ¿Cómo se manejan los archivos *.aif que son distintos de los *.wav?
Bill tuvo que modificar el procedimiento.

Nota: Los nombres utilizados en esta historia son ficticios, cualquier coincidencia con la vida real es casualidad.

Ejemplo


 El programa de Bill quedó así:

```
rotar(numFigura)
{
    // Dependiendo de numFigura, rotar 360º correctamente.
}

sonar(numFigura)
{
    if(es una ameba)
        // Buscar el archivo *.aif y hacerlo sonar.
    else
        // Dependiendo de numFigura, buscar el archivo *.wav
        // correcto y hacerlo sonar.
}
```

Nota: Los nombres utilizados en esta historia son ficticios, cualquier coincidencia con la vida real es casualidad.

Ejemplo

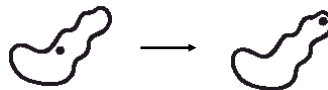
 Mientras tanto, Sam no tuvo que tocar nada del código ya desarrollado y probado. Simplemente añadió una clase mas:

Ameba
<pre>rotar() { // Rotar 360º. } sonar() { // Hacer sonar su archivo *.aif }</pre>

Nota: Los nombres utilizados en esta historia son ficticios, cualquier coincidencia con la vida real es casualidad.

Ejemplo

- Esta vez ya habían tardado el mismo tiempo. No sabían quien podría haber ganado:
- Pero cuando el director de proyecto echó un vistazo a los programas, se dio cuenta de que estaban mal.
- El último cambio en las especificaciones no había sido del todo claro:
- La ameba debía rotar girando sobre el punto de mas a la derecha y no sobre el punto central como habían pensado Bill y Sam.




Nota: Los nombres utilizados en esta historia son ficticios, cualquier coincidencia con la vida real es casualidad.

Ejemplo

- Bill se puso a pensar:
- Hummm... podría añadir un nuevo if-then-else en el procedimiento rotar.
- Y añadir *hardcoded* las coordenadas del punto de rotación.
- Pero algo le decía que estaba complicando el diseño y mantenimiento posterior de la solución.
- Además, ¿quién le decía que no volverían a cambiar las especificaciones?

Nota: Los nombres utilizados en esta historia son ficticios, cualquier coincidencia con la vida real es casualidad.


Ejemplo

 El programa de Bill quedó así:

```
rotar(numFigura, x, y)
{
    // Si numFigura no es del tipo ameba.
    // Calcular el punto central basado en un rectángulo,
    // y rotar 360°.
    // Si era una ameba.
    // Usar x e y como el punto de rotación
    // y rotar 360°.
}
```

Nota: Los nombres utilizados en esta historia son ficticios, cualquier coincidencia con la vida real es casualidad.

Ejemplo

 Mientras tanto, Sam simplemente retocó la clase Ameba. No tuvo que modificar nada del código ya probado en las clases Triangulo, Círculo y demás.

Ameba
<i>x</i> <i>y</i>
<i>rotar()</i> { <i>// Rotar 360° sobre el punto (x,y)</i> }
<i>sonar()</i> { <i>// Hacer sonar su archivo *.aif</i> }

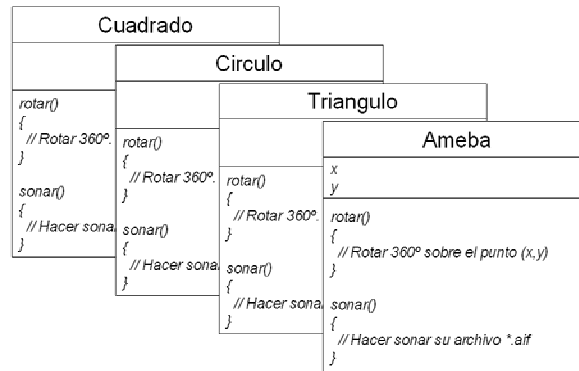
Nota: Los nombres utilizados en esta historia son ficticios, cualquier coincidencia con la vida real es casualidad.

Ejemplo



Cuando Bill vio el diseño de Sam, comenzó a buscarle pegas:

Lo primero que le encontró es que tenía el código repetido muchas veces.



Nota: Los nombres utilizados en esta historia son ficticios, cualquier coincidencia con la vida real es casualidad.

Ejemplo



Pero entonces Sam le enseñó a Bill el último diseño. El que había visto Bill estaba obsoleto.

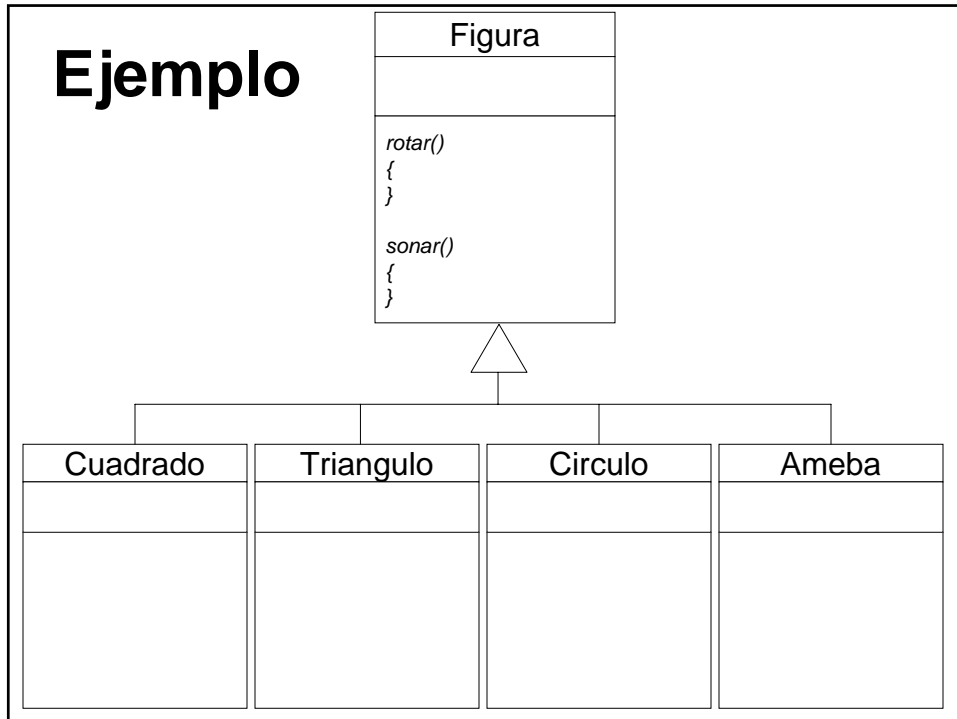
Sam, había incluido el uso de la herencia.

Se había fijado en que todas las clases tenían un comportamiento común: rotar y sonar.





Por tanto desarrolló una clase nueva llamada Figura que encapsulaba este comportamiento común y de la que heredaban el resto.

Nota: Los nombres utilizados en esta historia son ficticios, cualquier coincidencia con la vida real es casualidad.

Ejemplo

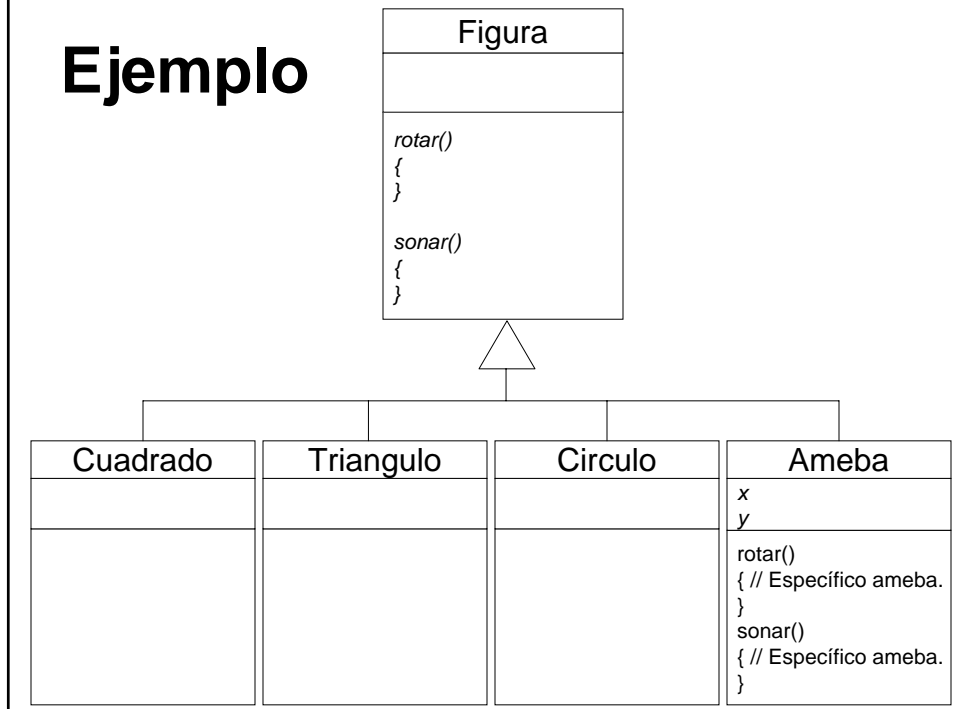


Ejemplo

-  Bill, que seguía picado siguió buscando pegas:
-  El comportamiento de la ameba era distinto al del resto de figuras.
-  Luego la solución de Sam estaba mal porque la clase ameba heredaba el comportamiento de Figura que era genérico.
-  Entonces Sam le comentó a Bill la especialización y la sobrescritura de comportamiento, añadiéndola al diseño.

Nota: Los nombres utilizados en esta historia son ficticios, cualquier coincidencia con la vida real es casualidad.

Ejemplo



Ejemplo



Bill, siguió preguntando:



¿Y cómo le dices a la ameba que haga algo?



¿No tienes que llamar a un procedimiento y decirle que tipo de figura tiene que rotar?



Sam sonrió:



Esa es la esencia de la Orientación a Objetos. Cuando un Tu programa necesita rotar un triángulo, directamente le Dice al triángulo que rote. El programa no sabe ni tiene que saber como se rotan triángulos.

Nota: Los nombres utilizados en esta historia son ficticios, cualquier coincidencia con la vida real es casualidad.

Ejemplo

- Bill se convenció y se apuntó a un curso de Orientación a Objetos y Java.
- ¿Y quién ganó al final?
- Ganó una tal Mónica del departamento de pruebas.
- Aunque nadie entendió por qué.

Nota: Los nombres utilizados en esta historia son ficticios, cualquier coincidencia con la vida real es casualidad.

Bibliografía

- Object-Oriented Analysis and Design
Grady Booch.
Addison-Wesley.
- The Essence of Object-Oriented Programming with Java™ and UML
Bruce E. Wampler
Addison-Wesley.
- Design patterns
Erich Gamma, Richard Helm, Ralph Johnson, John M. Vlissides
Addison-Wesley.

