

Hola, te saluda Ubaldo Acosta. Bienvenida o bienvenido nuevamente. Espero que estés listo para comenzar con esta lección.

Vamos a estudiar el tema de polimorfismo en Java.

¿Estás listo? Ok, ¡Vamos!

© Todos los Derechos Reservados Global Mentoring



Polimorfismo es la habilidad de ejecutar métodos sintácticamente iguales en tipos distintos. Vamos a revisar este concepto de la programación orientada a objetos con el siguiente ejemplo.

Un objeto (creado en la memoria heap) solo tiene una forma y un tipo y esto nunca cambia durante toda la vida del objeto creado. Sin embargo una variable de un tipo, puede referenciar a objetos de diferentes tipos, siempre y cuando haya una relación entre estos tipos, como puede ser una relación de herencia (Clase Padre o Clase Hija).

Una variable de tipo de una clase padre puede almacenar referencias de clases hijas o del mismo tipo de la clase padre, y mandar a llamar los métodos que tiene en común utilizando polimorfismo, es decir, ejecuta los métodos de la clase hija, con la variable de tipo de la clase padre. Esto es precisamente el concepto de polimorfismo.

La importancia del polimorfismo es que podemos generalizar un método que reciba distintos tipos en la jerarquía de clases definida (clases padre e hijas), por ejemplo un método que imprima los detalles de cada clase, sin importar si es un objeto es de tipo padre o de tipo hijo, y todo esto, con una variable de tipo padre.

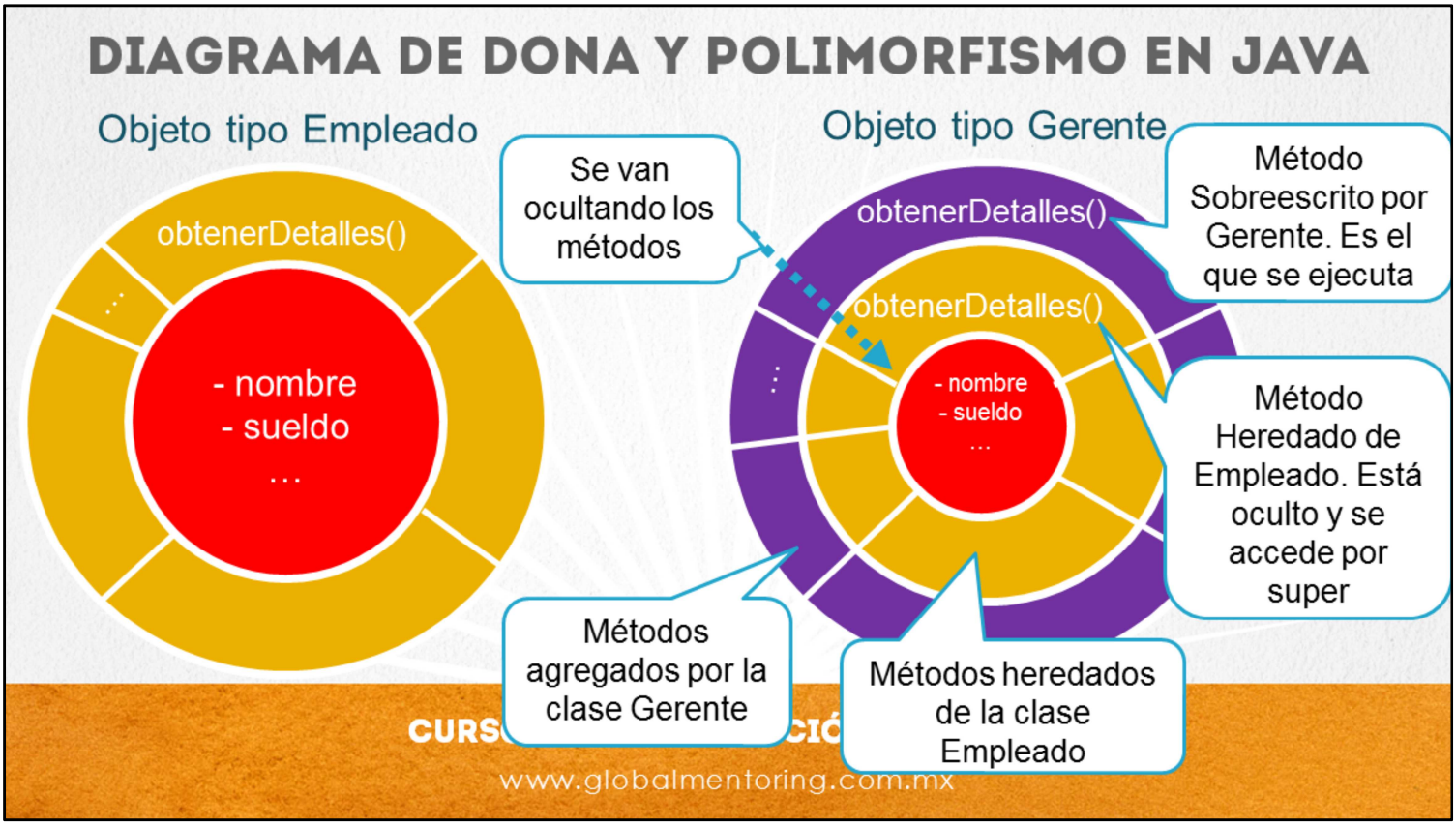
En el ejemplo mostrado, observamos que en el código se crea una variable de tipo Empleado (tipo padre), llamada emp. Esta variable se almacena en la memoria stack, ya que es una variable local. Y a la variable emp, se le asigna una referencia de un objeto de tipo padre, es decir de tipo Empleado, con el atributo de nombre Juan. Y cuando se manda a llamar el método obtenerDetalles con la variable emp, el método que se ejecuta es el de la clase Empleado.

Hasta aquí no hay nada nuevo, ya que estamos llamando un método del mismo tipo de la clase Padre. Sin embargo posteriormente con la misma variable de tipo padre, le asignamos una referencia de tipo hijo, es decir la clase Gerente, con el atributo de nombre Karla, y posteriormente llamamos al método obtenerDetalles. Aquí es donde puede surgir la duda, ¿Cuál es el método que se manda a llamar?, el de la clase Empleado debido a que la variable emp es de tipo Empleado, o el del método Gerente, ya que la referencia a la que apunta la variable emp es de un objeto de tipo Gerente?

La respuesta a esta pregunta tiene que ver precisamente con el concepto de polimorfismo, el cual dice que el método que se ejecute será del objeto cuya referencia esté apuntando en tiempo de ejecución, por lo tanto podemos concluir que en este último caso, el método que se ejecuta es el método obtenerDetalles de la clase Gerente, ya que es el tipo al cual hace referencia la variable emp. Así que sin importar que la variable emp sea de tipo empleado, el método que finalmente se ejecutará será del tipo al cual haga referencia.

La variable emp puede almacenera una referencia de tipo Gerente debido a que la clase Gerente extiende de la clase Empleado, de otra forma no sería posible almacenar esta referencia, por ejemplo, la variable emp no puede almacenar una referencia de un tipo String, ya que la clase String no hace extends de la clase Empleado, y por tanto no tienen nada en común, por ello hablar de polimorfismo, esta también tratar obligatoriamente del tema de herencia y de algún tipo de relación entre las clases que ejecutarán el polimorfismo.

© Todos los Derechos Reservados Global Mentoring



Podemos resumir el concepto de polimorfismo con el siguiente diagrama de dona.

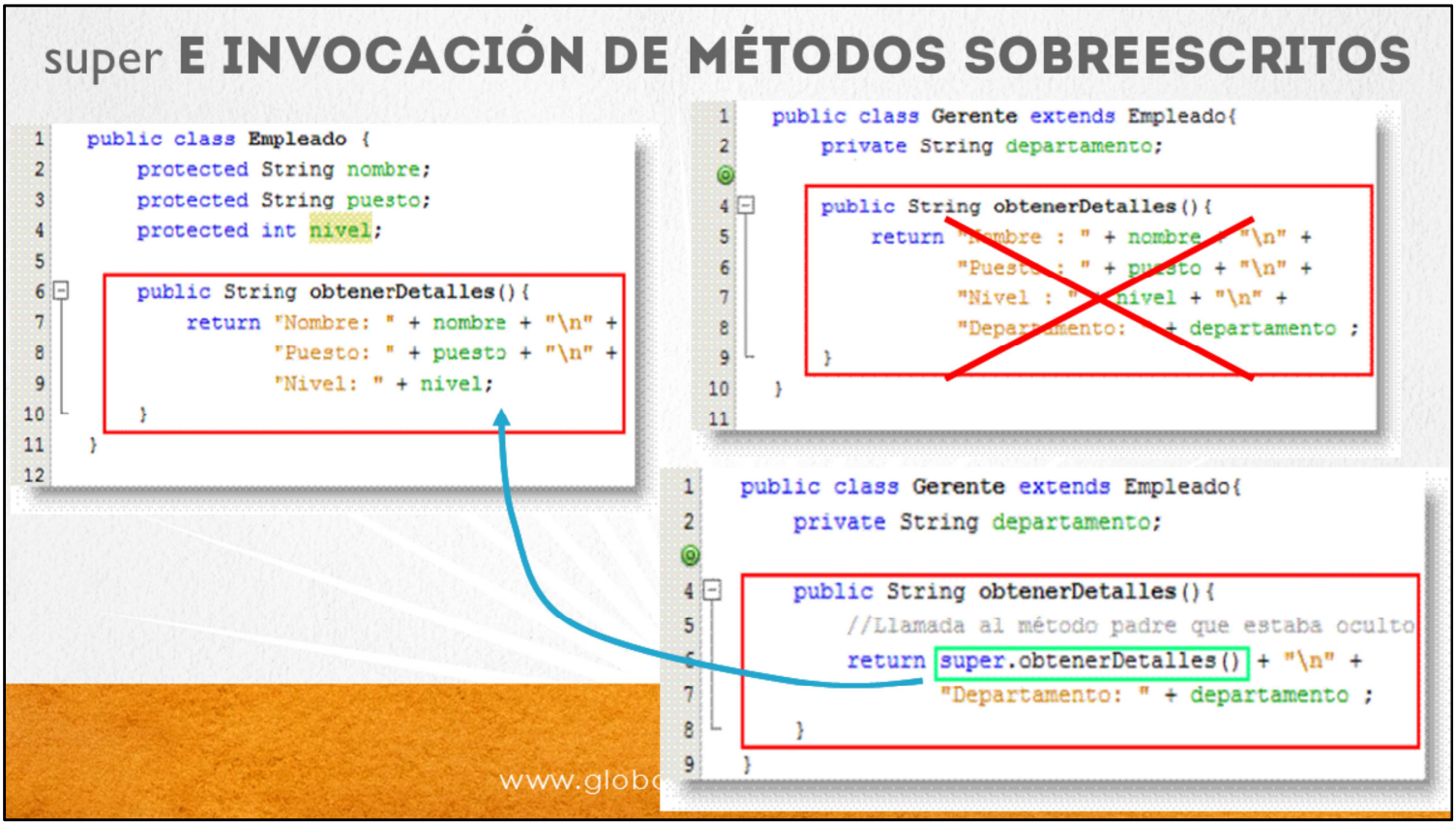
Observamos dos casos, un objeto de tipo Empleado y otro objeto de tipo Gerente, y observando este diagrama podremos fácilmente saber qué método es el que se está ejecutando en el objeto y la razón por la cual se ejecuta el método que indiquemos.

En el caso del tipo Empleado, una variable que ejecute el método obtenerDetalles() no tendrá ninguna duda que se ejecutará el método definido en esta clase, ya que es la clase padre.

Pero en el caso del tipo Gerente, como hemos visto, puede existir la duda de cual método se ejecuta, si el de la clase hija o de la clase padre. Sin embargo, como podemos observar el diagrama de dona del objeto de tipo Gerente, los métodos más externos son los definidos en la clase Gerente (color morado), y los más internos son los métodos heredados de la clase Empleado (color amarillo). Por lo que podemos decir que el método que se ejecutará es el método más externo, y el que quedará oculto será el método heredado, o lo que es lo mismo, el método más interno. Si quisiéramos acceder al método oculto podemos usar la palabra super seguida del método oculto. A continuación veremos un ejemplo de esto.

Hemos agregado algunas notas sobre el diagrama de dona para que podamos observar el orden en que se van accediendo y ocultando los métodos. Además los métodos que no quedan ocultos, es decir, que no se han sobreescrito, se acceden directamente como si se hubieran definido directamente en la clase Gerente, ya que como sabemos ese es el concepto de herencia.

© Todos los Derechos Reservados Global Mentoring



Una subclase puede invocar a un método de la clase padre por medio de la palabra "super“. La palabra "super", es como el operador "this" pero hace referencia a la clase padre.

La palabra "super" puede referenciar atributos y métodos de la clase padre. El método que se invoca por medio de super, no necesariamente debe estar declarado en la clase padre, sino podría estar declarado en una clase de más arriba en la jerarquía de clases.

Como podemos observar en la figura, una de las ventajas del uso de super es la reutilización de código. Así que en lugar de duplicar el código de la clase Empleado, podemos aprovechar lo que ya realiza este método y sólo agregar al método sobreescrito por la clase Gerente lo que necesitamos para completar este método, es decir, agregar el atributo departamento. De esta manera podemos reutilizar el código de la clase Padre si aplica.

Vamos a ver a continuación un ejercicio para aplicar todos estos conceptos.

© Todos los Derechos Reservados Global Mentoring

EJERCICIOS CURSO FUNDAMENTOS DE JAVA

- **ABRIR LOS ARCHIVOS DE EJERCICIOS EN PDF.**
- **EJERCICIO**: Ejercicio de Polimorfismo en Java.

CURSO DE PROGRAMACIÓN CON JAVA
www.globalmentoring.com.mx

© Todos los Derechos Reservados Global Mentoring



En Global Mentoring promovemos la Pasión por la Tecnología Java. Te invitamos a visitar nuestro sitio Web donde encontrarás cursos Java Online desde Niveles Básicos, Intermedios y Avanzados, y así te conviertas en un experto programador Java.

Además agregamos nuevos cursos para que continúes con tu preparación como programador Java profesional. A continuación te presentamos nuestro listado de cursos:

- Lógica de Programación
- Fundamentos de Java
- Programación con Java
- Java con JDBC
- HTML, CSS y JavaScript
- Servlets y JSP's
- Struts Framework
- Hibernate Framework
- Spring Framework
- JavaServer Faces
- Java EE (EJB, JPA y Web Services)
- JBoss Administration
- Android con Java
- HTML5 y CSS3

Datos de Contacto:

Sitio Web: www.globalmentoring.com.mx
Email: informes@globalmentoring.com.mx

© Todos los Derechos Reservados Global Mentoring