APRENDERAPROGRAMAR.COM

RESUMEN DE VENTAJAS DE LA HERENCIA EN JAVA. AVANZAR COMO PROGRAMADORES: SWING, GESTIÓN DE ERRORES Y MÁS ALLÁ. (CU00698B)

Sección: Cursos
Categoría: Curso “Aprender programación Java desde cero”

Fecha revisión: 2029

Resumen: Entrega nº98 curso Aprender programación Java desde cero.

Autor: Alex Rodríguez

RESUMEN DE HERENCIA EN JAVA

A lo largo del tutorial hemos intentado reflejar que la filosofía de la programación orientada a objetos, encarnada en la herencia, polimorfismo, encapsulamiento y ocultación de la información, etc. tiene una serie de ventajas que la hacen ser quizás la metodología de programación más utilizada hoy en día.

Resumimos como ventajas de la herencia:

1. **Evitar duplicidad y favorecer la reutilización de código** (las subclases utilizan el código de superclases).

2. **Facilitar el mantenimiento** de aplicaciones. Podemos cambiar las clases que usamos fácilmente.

3. **Facilitar la extensión** de las aplicaciones. Podemos crear nuevas clases a partir de otras existentes.

Con lo que hemos visto podemos decir que la herencia tiene dos implicaciones importantes: la primera, la reutilización de código (herencia de código) y la segunda el permitir el polimorfismo (herencia del tipo). La herencia admite tres variantes:

a) **Herencia a partir de clases concretas** (*extends* o herencia simple): heredamos el tipo y la implementación. Opcionalmente en la clase que extiende agregaremos nuevos métodos o sobreescribiremos otros ya existentes.

b) **Herencia a partir de interfaces** (*implements* o forma de herencia múltiple): heredamos el tipo sin implementación. Para poder instanciar, todos los métodos deben ser sobreescritos.

c) **Herencia a partir de clases abstractas** (*extends* sobre una clase abstracta, variante de herencia simple): heredamos el tipo y posiblemente un fragmento de implementación. Para poder instanciar, aquellos métodos abstractos han de ser sobreescritos, y la subclase pasaría a ser concreta. Si no se implementan todos los métodos abstractos, la subclase sigue siendo abstracta.

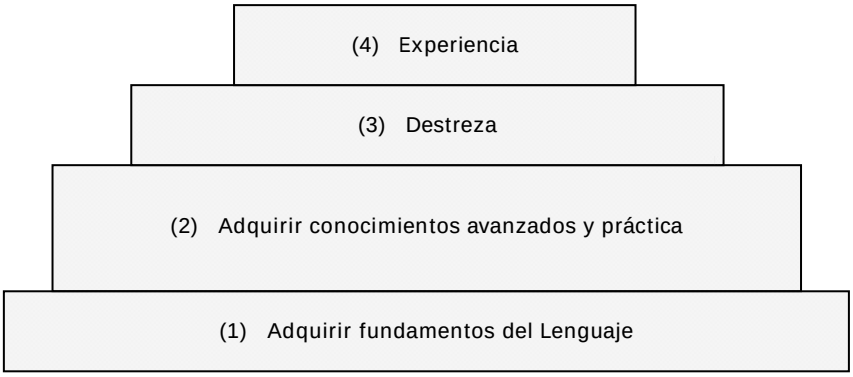
Observamos que en Java todos los tipos de herencia implican que se hereda el tipo. No en todos los lenguajes de programación ocurre igual.

Respecto a la herencia múltiple, Java no admite cualquier forma de herencia múltiple. Permite una forma de herencia múltiple de interfaces con la palabra clave *implements*, pero sólo herencia simple de clases, con la palabra clave *extends*.

PROGRESAR COMO PROGRAMADORES JAVA: SWING, GESTIÓN DE ERRORES Y MÁS ALLÁ. ¿QUÉ HEMOS APRENDIDO Y QUÉ NO HEMOS APRENDIDO CON ESTE TUTORIAL?

Hemos llegado al final del recorrido por Java en este tutorial que hemos denominado “Aprender programación Java desde cero”. ¿Qué hemos aprendido?

Desde nuestro punto de vista, los fundamentos para continuar progresando como programadores Java. Una de las mejores formas de abordar un área de conocimiento desconocida consiste en adquirir los fundamentos o principios fundamentales (cosa que hemos tratado de hacer mediante explicaciones y esquemas) y practicar (que en nuestro caso se traduciría por escribir código y hacer múltiples pruebas con el ordenador). Con la lectura y práctica de las secciones anteriores estamos en disposición de continuar progresando como programadores Java con unas bases sólidas. Esquemáticamente podríamos verlo así:



Nos hemos centrado en adquirir los fundamentos que nos permitan programar teniendo un entendimiento adecuado de lo que hacemos. Sin embargo, el recorrido para dominar Java, al igual que casi cualquier lenguaje, es un recorrido largo que puede requerir varios años.

Hemos hablado de la concepción del lenguaje, de la filosofía de la programación orientada a objetos, así como de algunas clases, interfaces y conceptos de diseño. Algunas clases importantes como LinkedList, HashMap, Vector, HashSet, Collections, y otras no las hemos abordado en profundidad o ni siquiera las hemos citado. No hemos discutido con amplitud el **diseño de clases** y programas (cohesión, acoplamiento, etc.) o su **prueba y depuración**. Tampoco hemos abordado las **clases internas**, de las que podemos decir brevemente que son clases definidas dentro de otra clase y cuya utilidad es definir un tipo que va a ser utilizado exclusivamente por la clase envolvente.

Tampoco hemos hablado de **gestión de errores** ni de las interfaces gráficas de usuario y de las importantes **bibliotecas del API de Java awt y swing**, o de la gestión de eventos, que constituyen partes muy importantes de la programación Java. No hemos abordado algunos patrones de diseño de interés como el patrón Singleton.

La extensión y contenidos de este curso básico sobre Java han sido motivo de discusión y los objetivos que planteamos el equipo editorial de aprenderaprogramar.com fueron los siguientes:

a) Generar contenidos originales, didácticos y prácticos.

b) Generar un curso breve, con contenidos claramente definidos y abordables en el marco de un curso o asignatura de duración limitada.

c) Definir los fundamentos que permitieran la progresión natural de los alumnos mediante otros cursos o el entrenamiento individual.

Java es un lenguaje de relativa complejidad conceptual y de gran extensión, lo que hace ciertamente difícil su didáctica. Los tutoriales o cursos existentes muchas veces resultan demasiado áridos o extensos para los alumnos y esto se traduce en altas tasas de abandono. Si has llegado a esta parte final de este curso, confiamos en que sea porque te ha resultado didáctico, entretenido y de extensión adecuada. Si ha sido así, estamos seguros de que los fundamentos adquiridos serán sólidos y un valor añadido a tus capacidades en el área de la programación.

Próxima entrega: CU00699B

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente:

http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=68&Itemid=188