APRENDERAPROGRAMAR.COM

JERARQUÍAS DE HERENCIA EN JAVA. CONCEPTO DE SUPERCLASES Y SUBCLASES. EL API JAVA. EJEMPLOS. (CU00685B)

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

Fecha revisión: 2029

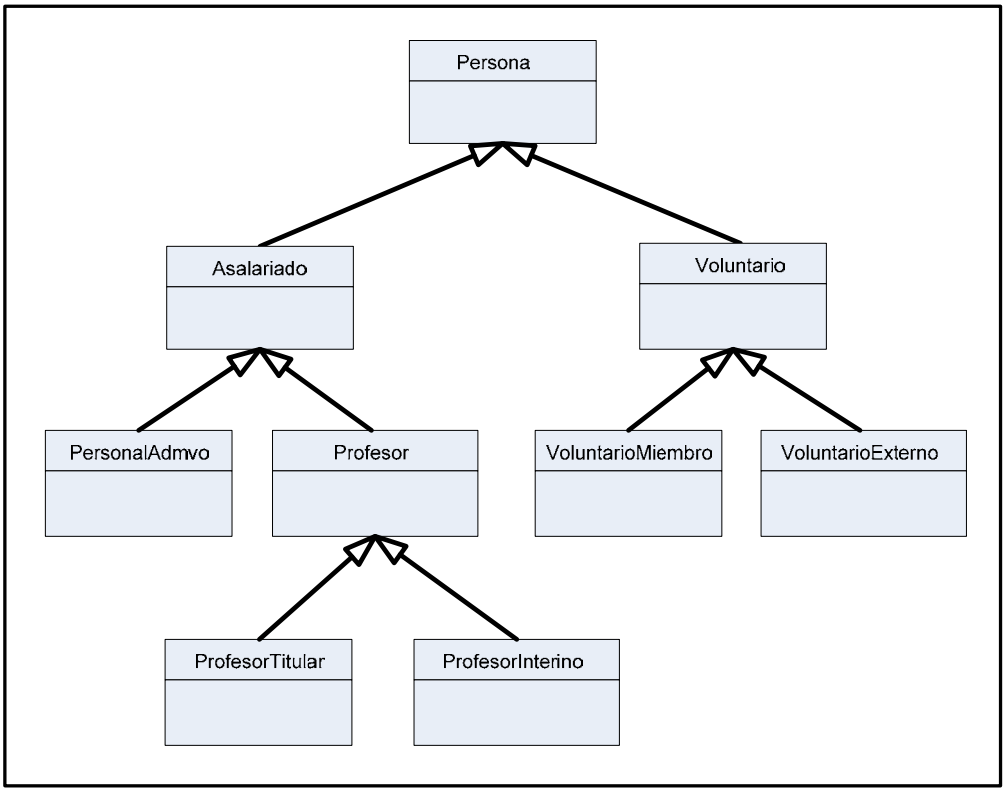
Resumen: Entrega nº85 curso Aprender programación Java desde cero.

Autor: Alex Rodríguez

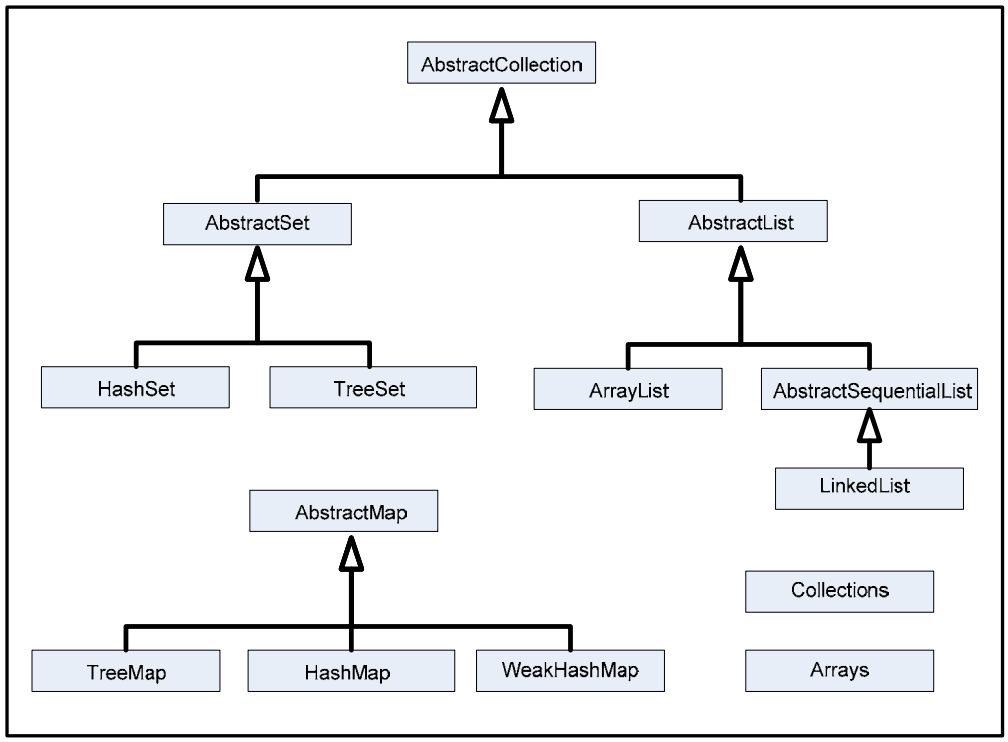
JERARQUÍAS DE HERENCIA EN JAVA. SUPERCLASES Y SUBCLASES.

En Java muchas subclases pueden heredar de una misma superclase, y a su vez una subclase puede convertirse en superclase de otra. Así las cosas, podemos hablar de una jerarquía de herencia. La jerarquía es el esquema organizativo de las clases con relación de herencia entre sí.

En un proyecto nosotros definimos nuestra propia jerarquía de herencia.



Las clases del API de Java igualmente tienen establecida una jerarquía de herencia, en este caso definida por el equipo de desarrollo del lenguaje de la multinacional Oracle.



La herencia es una técnica de abstracción que nos permite agrupar objetos con características comunes. Todas las aplicaciones Java usan herencia, aunque no lo hagan de forma explícita. Esto es así porque cuando creamos una clase sin ninguna referencia a una superclase, el propio compilador inserta en la clase creada una relación de herencia directa con la la clase java.lang.Object. Por este motivo todos los objetos, sean de la clase que sean, pueden invocar métodos de la clase java.lang.Object como equals() y toString(), aunque no siempre se van a obtener buenos resultados así.

Si quisiéramos podríamos escribir para todas las clases *public class NombreDeLaClase extends Object*, aunque como es algo implícito a Java normalmente no lo escribiremos por ser redundante. En los diagramas representativos de la jerarquía de herencia ocurre lo mismo: Object siempre está en la cabecera del diagrama, aunque normalmente no se represente ya que se da por sobreentendido. Recordar que en Java los tipos primitivos no son objetos: no son instancias de clase, y por tanto no heredan de la superclase Object.

Los campos privados de una superclase no son accesibles por la subclase directamente. Decimos que la subclase no tiene derechos de acceso sobre los campos privados de la superclase. Para acceder a ellos tendrá que hacer uso de métodos de acceso o modificación. Una subclase puede invocar a cualquier método público de su superclase como si fuese propio. Lo veremos con ejemplos.

Próxima entrega: CU00686B

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