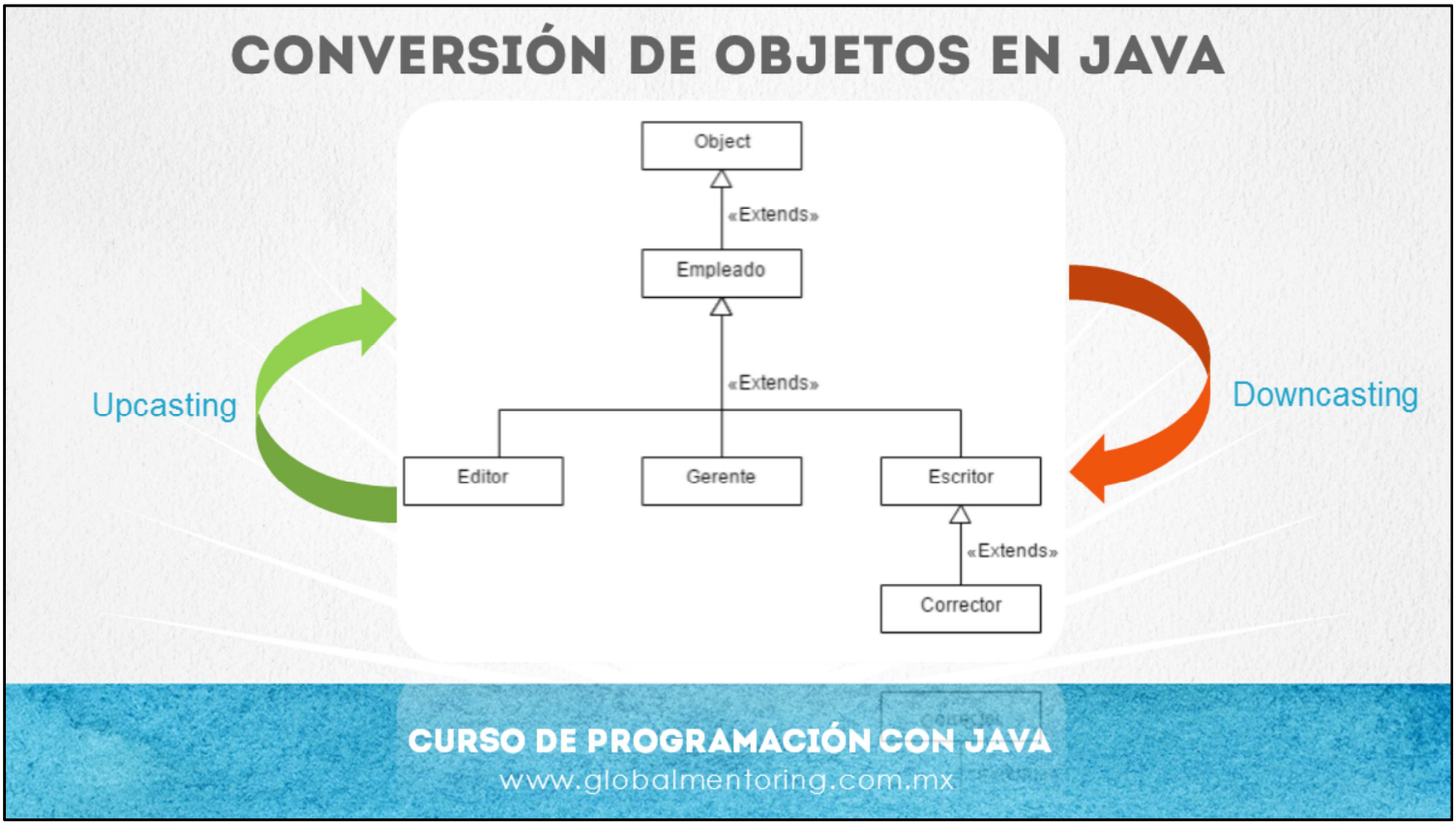
Hola, te saluda Ubaldo Acosta. Bienvenidos nuevamente a este curso de Programación con Java.

En esta lección vamos a revisar el tema de forEach en Java.

Comencemos de inmediato.



Sabemos que un objeto creado en la memoria heap, siempre va a ser del mismo tipo, por lo que su tipo nunca cambiará.

Sin embargo una variable de cierto tipo puede apuntar distintos tipos, siempre y cuando haya una relación entre ellos. En la figura podemos ver que una variable de tipo Empleado, puede almacenar referencias de objetos de tipo Empleado, o cualquier subclase, como puede ser Gerente, Escritor, Corrector, etc.

Hay ocasiones en las que es necesario hacer una conversión del tipo de la variable que apunta a un objeto, con el objetivo de ejecutar cierto código o generalizar nuestro código y así soportar múltiples operaciones para múltiples tipos de datos desde una misma función generalizada. Esto nos permite reutilizar código, entre varios beneficios más de la programación orientada a objetos y en general de las buenas prácticas de programación.

Si hacemos una conversión de un tipo más específico (inferior en la jerarquía de clases) a un tipo superior, se conoce como upcasting, es decir, conversión hacia arriba, y si por el contrario convertimos de un tipo superior a un tipo inferior se conoce como downcasting.

Este tema se utilizará en Java en muchísimos casos, por ello es importante tenerlo muy claro. El upcasting se realiza de manera automática, y no hay necesidad de escribir algún código de manera explícita, debido a que si es un tipo superior, en automático se realiza la conversión, esto es muy similar a los tipos primitivos en Java y la conversión entre ellos.

Por otro lado, la conversión hacia abajo o downcasting no se realiza de manera automática, y es necesario especificar cual es el tipo al que queremos convertir, y de hecho en caso de realizar una conversión errónea se lanzará una excepción de tipo ClassCastException. Más adelante veremos el tema de excepciones.

Para realizar un downcasting solo debemos especificar entre paréntesis el tipo al cual deseamos convertir nuestro tipo de dato, y con ello tendremos un tipo de dato que por consiguiente ya incluye todas las características del tipo más específico recién convertido. Veamos un ejemplo de esto.

CONVERSIÓN DE OBJETOS EN JAVA

Ejemplo de conversión de objetos en Java:

```
1  public class EjemploConversionObjetos {
2
3      public static void main(String[] args) {
4          //Asignamos una referencia de un objeto de menor jerarquia
5          Empleado empleado = new Escritor("Roy", 15000, TipoEscritura.CLASICO);
6
7          empleado.getTipoDeEscrituraEnTexto();//Esto no es posible, no accesible
8
9          //Convierte objeto  - Downcasting
10         Escritor escritor = (Escritor) empleado;
11
12         //Finalmente podemos acceder a los metodos de la clase Escritor
13         resultado = escritor.getTipoDeEscrituraEnTexto();
14
15         resultado = ((Escritor)empleado).getTipoDeEscrituraEnTexto();
16
17         System.out.println("resultado tipoEscritura:" + resultado);
18     }
19 }
```

Podemos observar en el código un ejemplo de conversión, este ejemplo lo vamos a desarrollar a detalle, sin embargo vamos a mencionar varias cuestiones para que veamos como funciona la conversión de objetos.

El objetivo de convertir objetos es poder acceder a los métodos de cada objeto según necesitemos. Por ejemplo en la línea 5, estamos creando un objeto de tipo Escritor, recordemos que este objeto nunca cambia de tipo, siempre será el mismo. Esta referencia es asignada a una variable de tipo Empleado, es decir, a una clase de mayor Jerarquía según nuestro diagrama de clases de la lámina anterior. Esto se conoce como upcasting, por lo que no requiere ninguna sintaxis especial para hacer una conversión hacia arriba en la jerarquía de clases. Lo único que sucede es que ya no es posible acceder a los métodos creados en la clase Escritor, sino que solo están disponibles los métodos de la clase Empleado. Por ello, si quisiéramos mandar a llamar algún método de la clase Escritor directamente, no sería posible ya que la variable que estamos usando es de un tipo superior, y no puede acceder directamente a los métodos de la clase de menor jerarquía.

Es aquí donde entra el tema de conversión de objetos, ya que podemos acceder a los métodos de la clase Escritor, sin embargo necesitamos convertir la variable de tipo Empleado. En la línea 10 podemos observar esta conversión, que se conoce como downcasting, la sintaxis es muy simple, solo es poner entre paréntesis el tipo de dato al cual queremos convertir, y posteriormente la variable a convertir, pero debe haber una relación en la jerarquía de clases de los tipos de datos que deseamos convertir. Esto es similar a la conversión de tipos primitivos, pero con las ventajas del manejo de objetos.

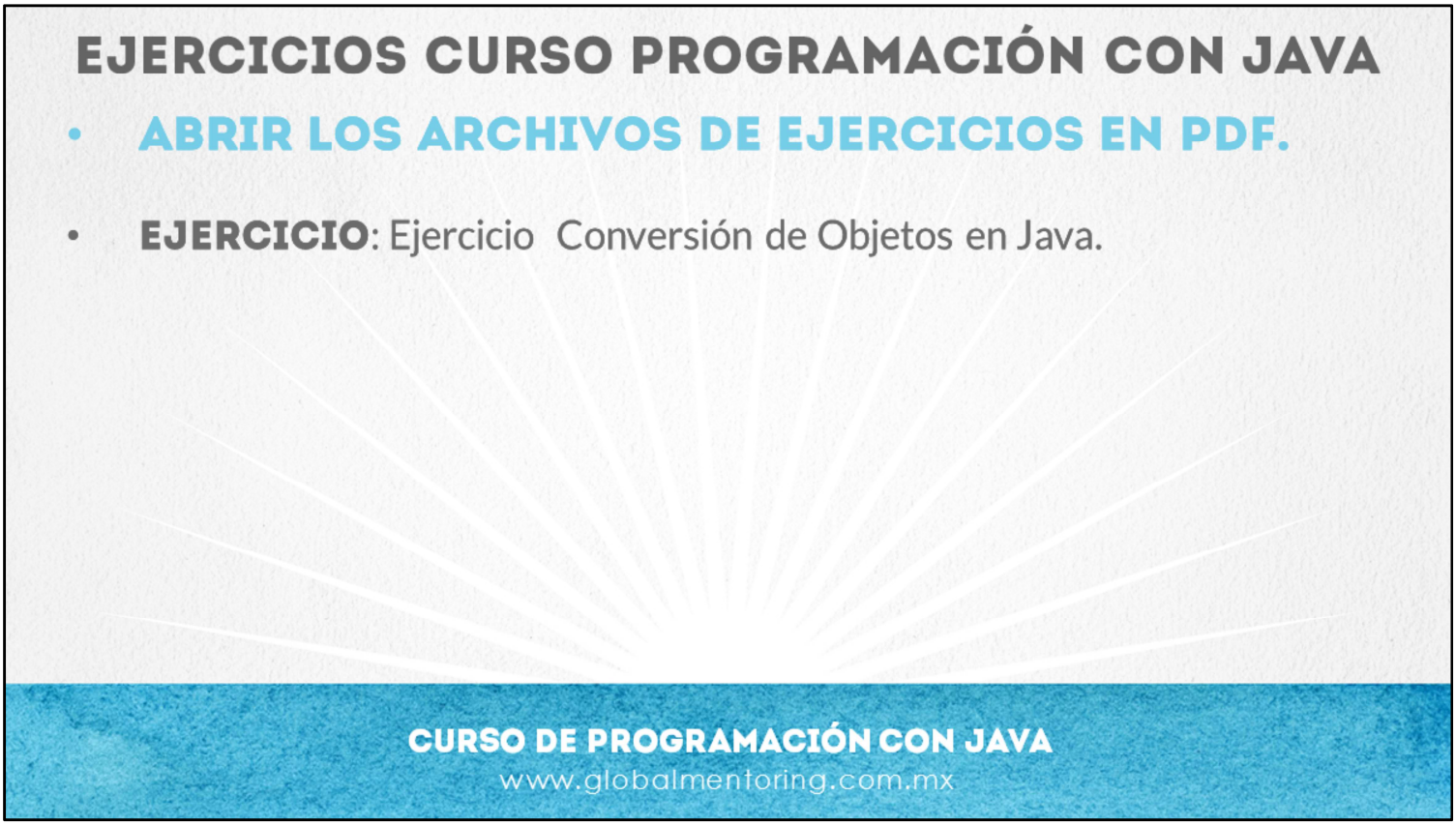
Otra forma de hacer esto, es ahorrarnos la variable, y convertir en la misma línea de código, esto lo podemos ver en la línea 15. Este tipo de conversión es muy común, por lo que debemos entender que no hay necesidad de crear una variable y realizar la asignación para que podamos ejecutar la conversión de objetos, ciertamente es una sintaxis más compleja, pero una vez que nos acostumbremos será fácil detectar lo que sucede.

Básicamente la conversión en línea, comienza por la misma sintaxis que es escribir entre paréntesis el tipo deseado y posteriormente la variable a convertir, lo siguiente es envolver entre paréntesis estos dos elementos, y con ello prácticamente tendremos acceso como si fuera la variable ya convertida al tipo deseado, con acceso a los métodos y atributos de la clase de menor jerarquía.

Cabe mencionar que si es un upcasting, en lugar de downcasting, no hay necesidad de especificar el tipo de dato a convertir, ya que el compilador detectará que la clase es de mayor jerarquía y se hará una conversión en automático.

Una vez que hemos hecho la conversión hacia abajo (línea 10), podemos utilizar cualquiera de los métodos del tipo de menor jerarquía pero con más detalle, por ejemplo supongamos que la clase escritor ha definido un método llamado getTipoDeEscrituraEnTexto(), el cual nos indicará si el escritor escribe a mano o a computadora. Así podemos ya acceder a este método, el cual no es accesible desde la clase empleado, pero si desde la clase escritor.

De esta manera haciendo una conversión de datos es posible acceder a los métodos y/o atributos de las clases de menor jerarquía y así recuperar la funcionalidad que al momento de utilizar tipos más genéricos o de mayor jerarquía se pierde. Veamos un ejemplo de conversión de objetos.



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.

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