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

Vamos a estudiar el tema argumentos variables en Java.

¿Estás listo? ¡Vamos!

PARAMETROS VARIABLES

Parámetros variables en Java: varargs

```
public class EjemploVarArgs {

    public static void main(String[] args) {
        //Imprimimos varios números
        imprimirNumeros(15,20,3,61,75,18,10);
    }

    public static void imprimirNumeros(int... numeros){
        //Recorremos cada elemento
        int elemento;
        for(int i=0; i < numeros.length; i++){
            elemento = numeros[i];
            System.out.println("Elemento: " + elemento);
        }
    }
}
```

Argumento usando varargs

Ya hemos visto el tema de parámetros, pero ¿Qué pasa cuando queremos pasar varios parámetros del mismo tipo a una función?. Una forma sería pasarlos como un arreglo, sin embargo, como sabemos para ello necesitamos conocer el numero de elementos que tendrá dicho arreglo.

Una forma de lograr enviar parámetros de una manera sencilla e indirectamente crear este arreglo de parámetros, es utilizando el concepto de varargs, que básicamente son argumentos variables.

Existen varias reglas para usar varargs, una de ellas es que los elementos que se envían, al convertirse al final en un arreglo, deben ser del mismo tipo.

Además, para indicar que se trata de un argumento variable se debe agregar tres puntos (…) después del tipo de datos que va a recibir el argumento.

Como se observa en la lámina hemos creado una función llamad imprimirNumeros, la cual recibe un argumento llamado números, y para definir que es un tipo de argumentos variables hemos agregado los tres puntos después del tipo de dato.

Posteriormente desde el método main, lo que hemos hecho es mandar a llamar este método indicando un número de parámetros de tipo entero, pero sin preocuparnos por el número de elementos que estamos enviando al método imprimirNumeros. Esta es precisamente la característica de varargs, la cual crear un arreglo con el largo de elementos al recibir todos los elementos enviados al método.

Una vez que el método imprimirNumeros recibe todos los elementos, crear un arreglo del tipo indicado, y ya podemos recorrer cada uno de los elementos tal como sabemos que debemos recorrer un arreglo de una dimensión.

FOREACH

Parámetros variables y uso de forEach

```
public class EjemploVarArgs {

    public static void main(String[] args) {

        imprimirNumerosForEach(15,20,3,61,75,18,10);

    }

    public static void imprimirNumerosForEach(int... numeros){

        //Usamos un forEach en lugar de un for normal
        for(int numero : numeros){
            System.out.println("El numero es:" + numero );
        }
    }
}
```

Argumento usando varargs

En la lámina anterior hemos utilizado un ciclo for para iterar cada uno de los elementos del arreglo vararg, sin embargo existe una notación simplificada del ciclo for llamada forEach, la cual podemos utilizar para iterar arreglos o colecciones, las cuales estudiaremos más adelante.

Sin embargo queremos mostrarles como simplificar la iteración de los elementos de un arreglo utilizando el concepto de forEach. Básicamente lo que hacemos es declarar un ciclo for, pero a diferencia de un ciclo for normal, el cual tiene 3 elementos a declarar, un ciclo forEach, solo tiene dos elementos a declarar, que es la variable que va a recibir cada uno de los elementos del arreglo, y el arreglo que se va a iterar.

Java internamente se encargará de hacer la iteración respectiva y dentro del bloque forEach lo que usaremos es la variable que contendrá cada uno de los elementos del arreglo por cada iteración.

La única desventaja es que si necesitamos conocer el índice del elemento que se está iterando de este ciclo for, esta sintaxis no nos servirá, ya que en ningún momento se ha declarado un contador que nos ayude a saber cual es el elemento que se está iterando, ya que el ciclo forEach únicamente nos regresa el elemento iterado, pero no el índice que se está iterando.

Ese sería el único inconveniente a tener en cuenta, por lo demás vemos claramente cómo la sintaxis es más limpia y menos propensa a errores, ya que no necesitamos declarar varios elementos para iterar un arreglo, por lo que es más simple y más seguro utilizar esta notación para recorrer los elementos de un arreglo.

VARIOS ARGUMENTOS

Envio de varios argumentos:

```
public class EjemploVarArgs {

    public static void main(String[] args) {

        variosParametros("Juan", true, 15,20,14);

    }


    public static void variosParametros(String nombre, boolean valido, int... numeros){
        System.out.println("Nombre: " + nombre);

        System.out.println("Valido: " + valido);

        //Usamos un forEach
        for(int numero : numeros){
            System.out.println("El numero es:" + numero );
        }
    }

}
```

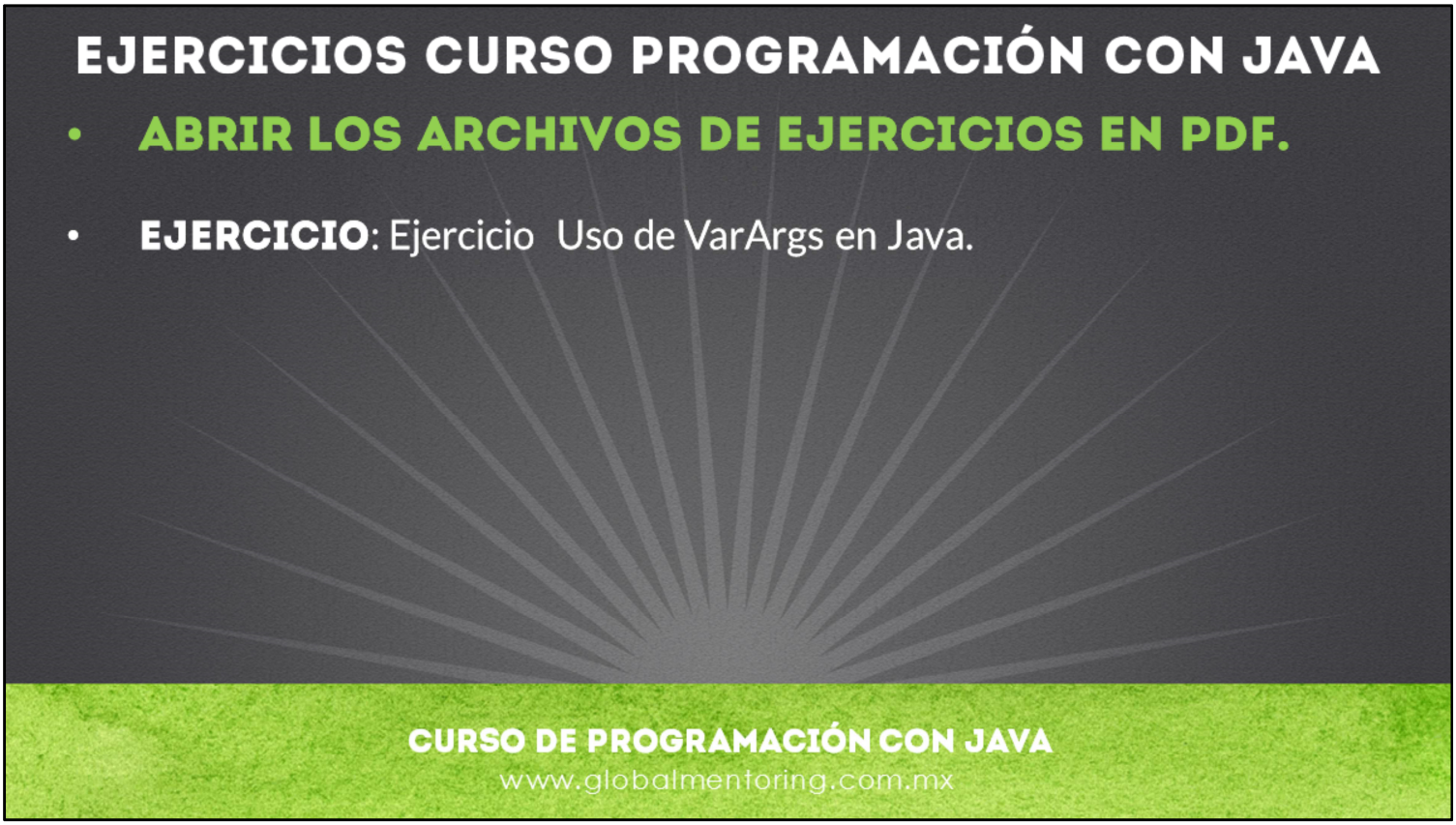
Varargs debe ser el último elemento en los parámetros declarados

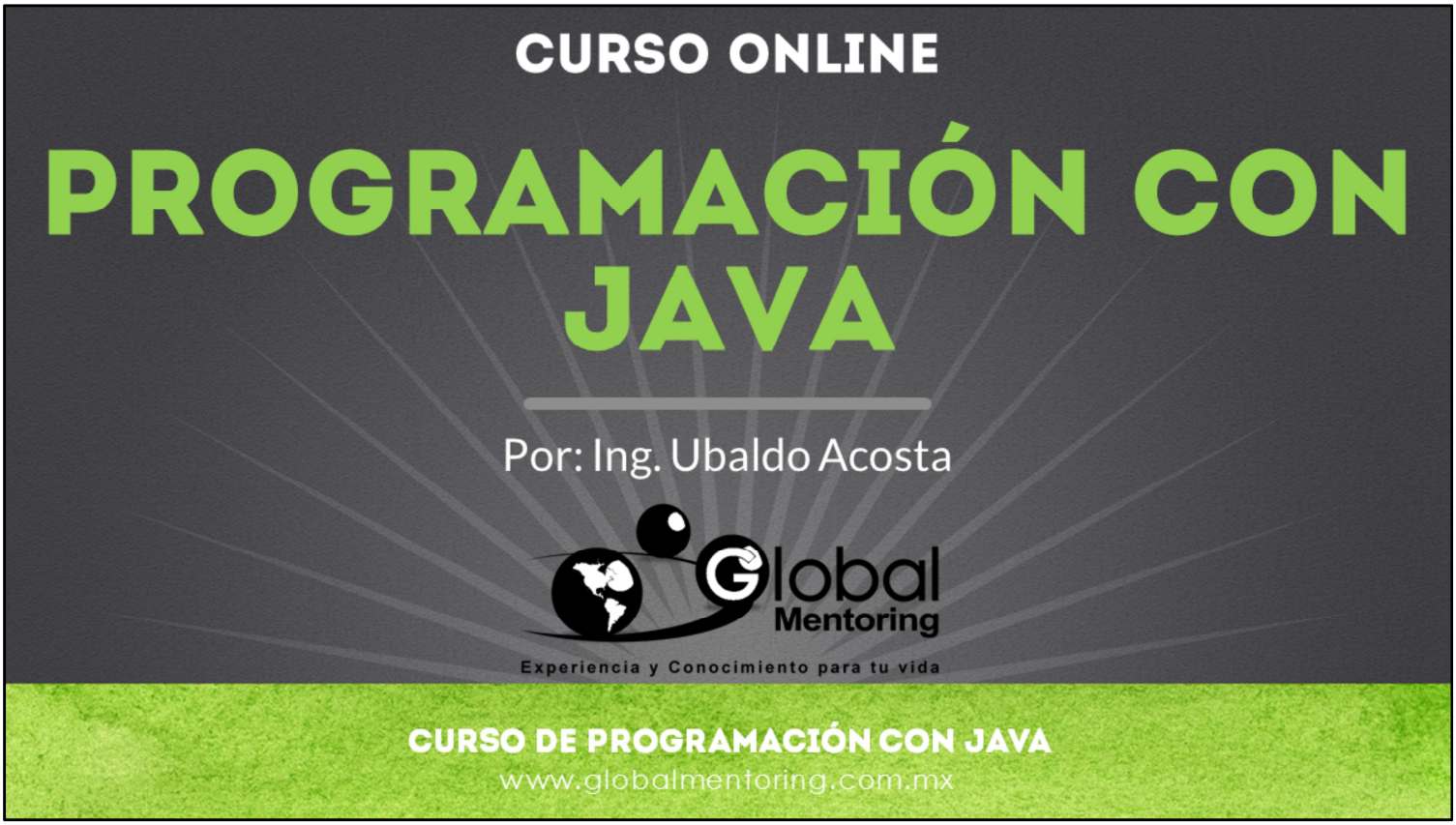
En esta lámina básicamente explicaremos un regla más del uso de varargs, la cual indica que si vamos a utilizar el concepto de varargs, es necesario que sea el último de los argumentos declarados, de esta manera al manda a llamar la función, no habrá confusión para el compilador, por que en cuanto detecte que comienza el tipo de datos del tipo vararg definido, es que comenzará a crear el arreglo de elementos enviados a la función.

En pocas palabras el compilador de Java detecta el número de elementos enviados y una vez que ha terminado de asignar los elementos que no son vararg, revisará si el último argumento declarado es de tipo vararg, y así es como todos los valores restantes que sean del mismo tipo que el argumento vararg definido, los asignará a este elemento vargarg y como sabemos finalmente los convierte en un arreglo.

Esta regla es muy importante para el uso de vararg, ya que en ocasiones lo queremos combinar con varios argumentos, y es donde debemos respetar esta regla.

A continuación vamos a crear un ejercicio para practicar cada uno de estos ejemplos.





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