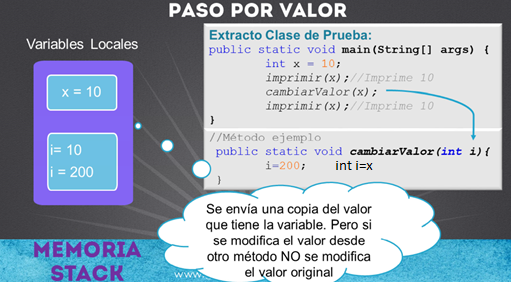
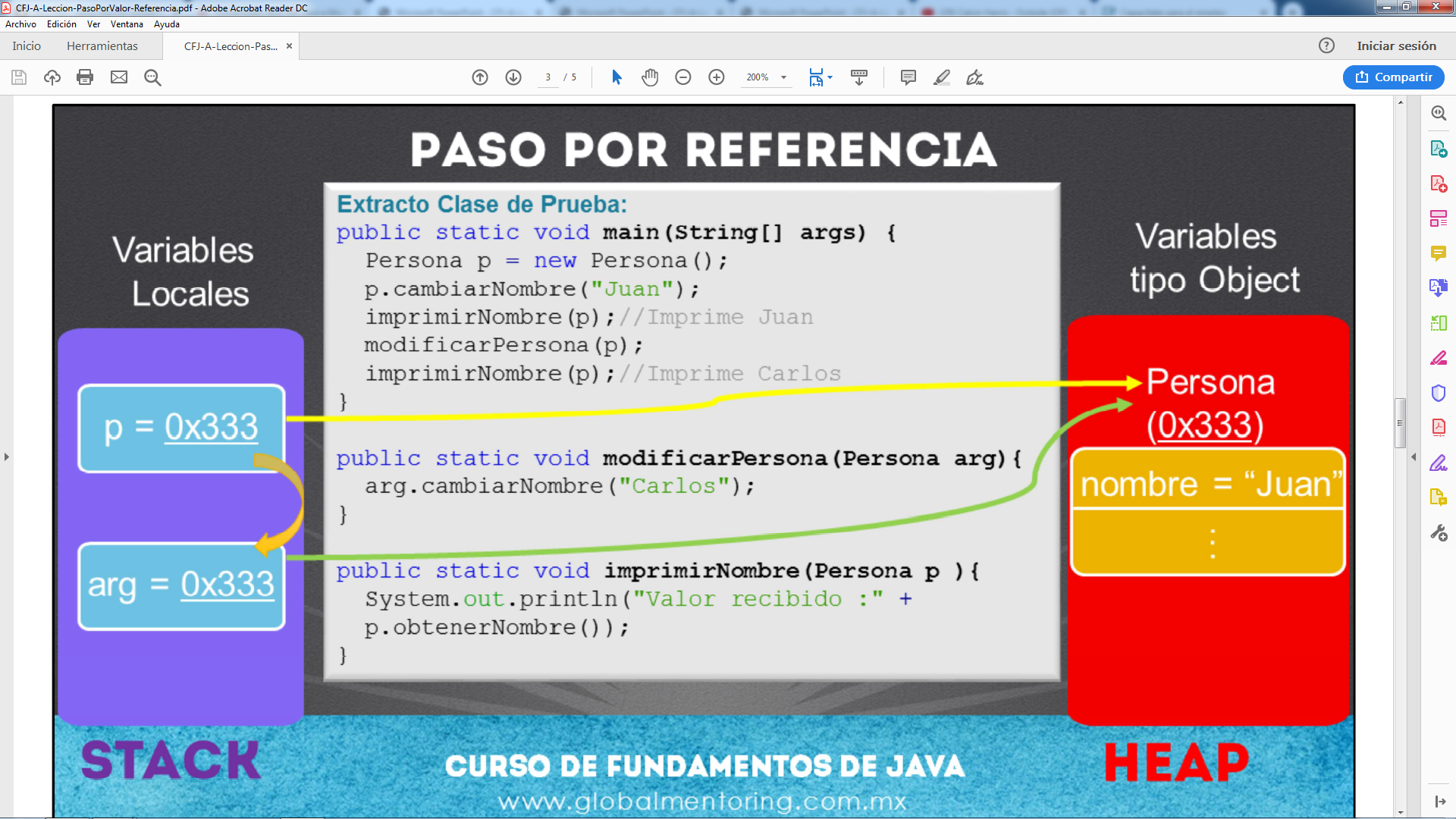
**SECCIÓN 14**

**PASO POR VALOR Y PASO POR REFERENCIA**

****

Por ejemplo en el código mostrado podemos observar un ejemplo de paso por valor, también conocido como valores de tipo primitivo. Lo que observamos en el código es que al crear una variable, en este caso x, y asignarle un valor, por ejemplo 10, esta variable al ser modificada en otro método su valor original no se ve modificado, ya que solamente paso una copia de su valor original.

La manera de cambiar el valor de la variable x sería cambiándolo dentro del mismo método. Sin embargo en este ejemplo el método llamado cambiarValor intenta realizar un cambio a la variable x, pero como está fuera del alcance de este método acceder a esta variable, únicamente puede cambiar el valor de su variable local llamada i. Y al cambiar el valor de esta variable y regresar al método que hizo la llamada original



Al utilizar un objeto como parámetro y enviarlo a un argumento de un método lo que estamos haciendo es enviando la referencia del objeto que se está deseando utilizar, y en lugar de utilizar una copia del valor del objeto, lo que realmente estamos haciendo es apuntar al mismo objeto, con la finalidad de modificarlo directamente, sin necesidad de hacer una copia.

En el código podemos observar la variable p de tipo Persona, esta clase la crearemos en la sección de ejercicios, pero básicamente tiene un atributo llamado nombre de tipo String, y lo que podemos observar es que se modifica el valor del atributo nombre, esto debido a que la variable local llamada arg apunta al mismo objeto creado en la memoria Heap de tipo Persona, y por lo tanto al modificar el atributo del mismo objeto, entonces se modifica el objeto creado originalmente, y podemos acceder a este cambio incluso desde otros métodos.

De esta manera podemos observar como en Java cuando trabajamos con objetos, realmente lo que estamos proporcionando es el valor de la referencia de memoria, en este caso el valor de 0x333, y con esta referencia accedemos directamente al objeto, y así podemos modificarlo. Finalmente estas modificaciones las podremos acceder incluso fuera del método donde fue creado originalmente el objeto, o desde el método donde se creó dicho objeto.