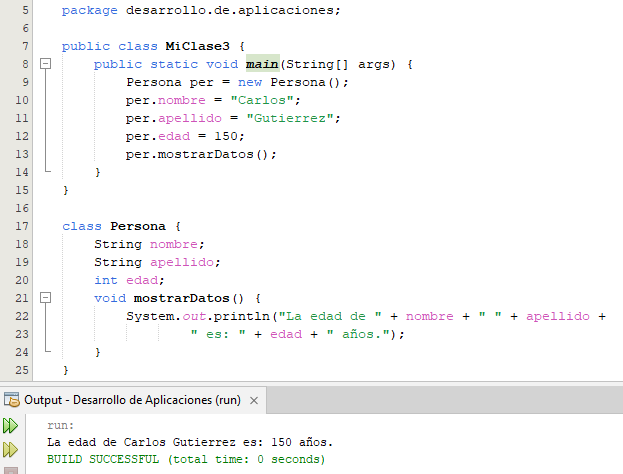
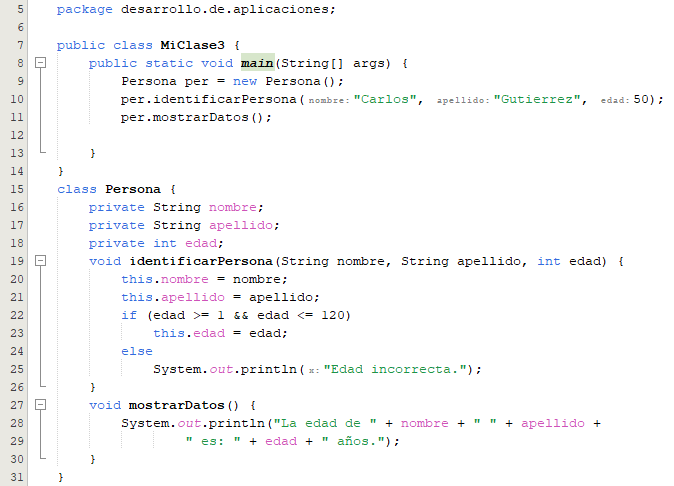
**Desarrollo Avanzado de Aplicaciones I**

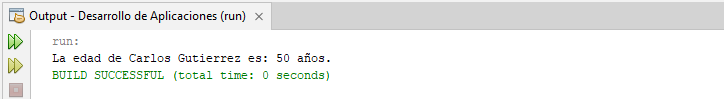
**Ejercicio 1**. Desarrollar una aplicación que use una clase “Persona” que tiene como atributos: nombre, apellido y edad sin encapsulamiento y con encapsulamiento. Se debe validar que la edad esté entre 1 y 120 años.

Clase Persona sin Encapsulamiento

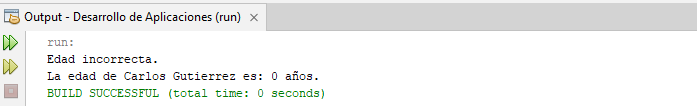


Clase Persona con Encapsulamiento



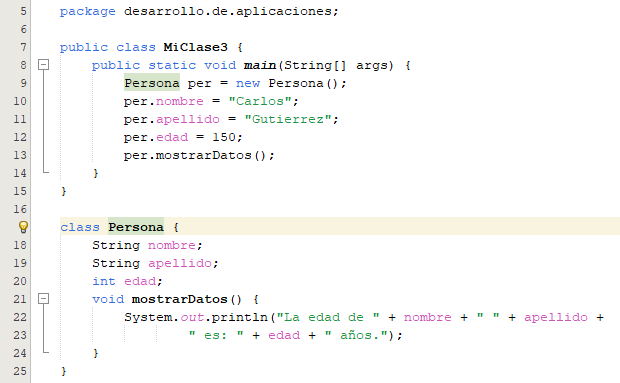


Y cambiando el parámetro edad a 150 muestra el siguiente resultado:

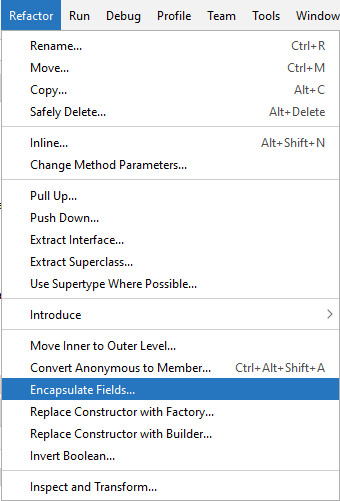


**Ejercicio 2**. Utilizar la opción “Refactor\Encapsulate Fields…” para generar los getters y setters de la clase “Persona” automáticamente.

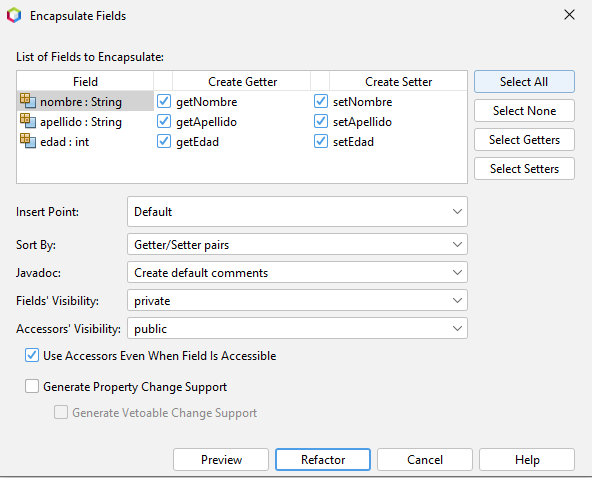
1. Posicionar el cursor en la clase “Persona”.



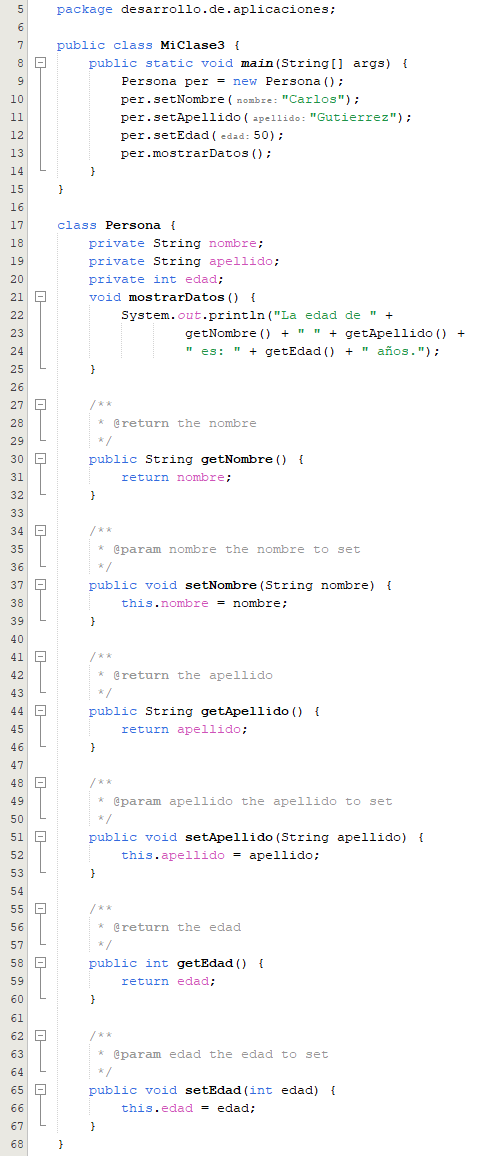
1. Hacer click en el menú “Refactor” y luego en “Encapsulate Fields…”.

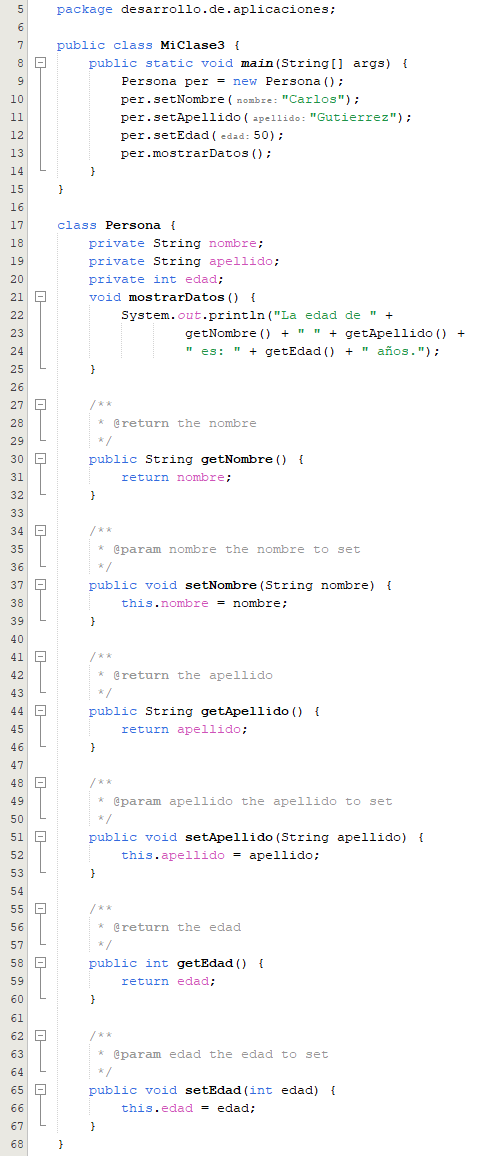


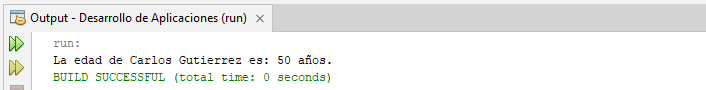
1. Hacer click en el botón “Select All” y luego hacer click en “Refactor”.

****

1. El resultado es el siguiente:

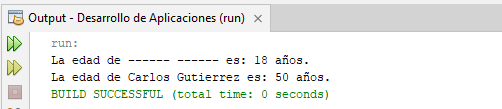
****

****

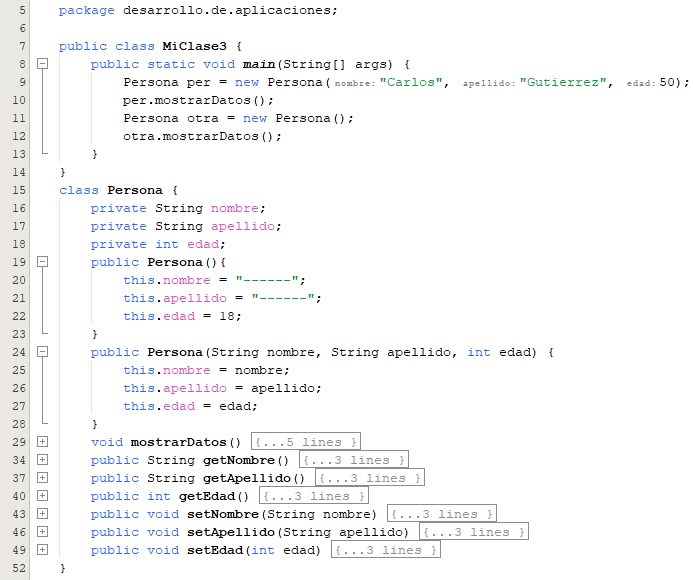


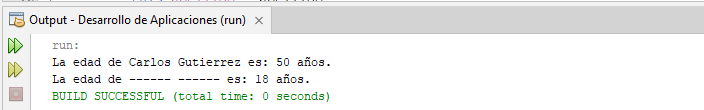
**Ejercicio 3**. Modificar el ejercicio anterior para que la clase “Persona” tenga un constructor sin parámetros que le asigne a los atributos los siguientes valores: nombre = “------”, apellido = “--------” y edad = 18.



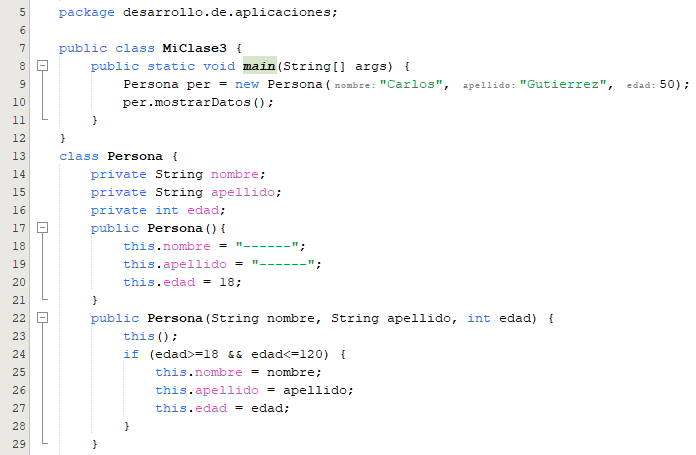


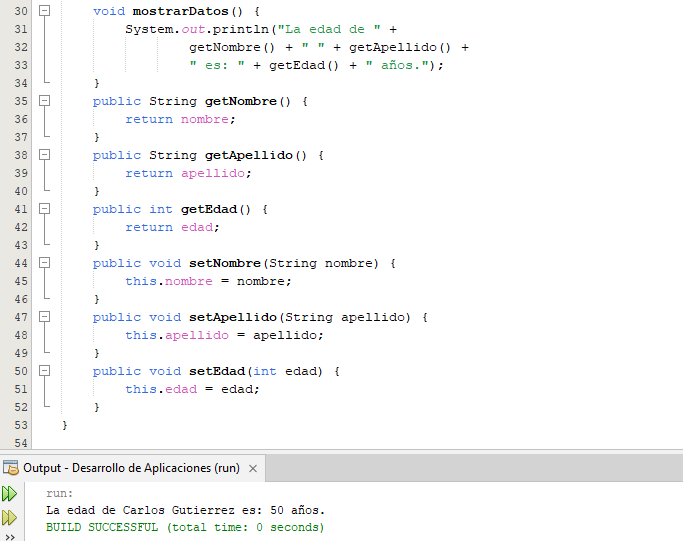
**Ejercicio 4**. Modificar el ejercicio anterior para que la clase “Persona” tenga un constructor con parámetros.



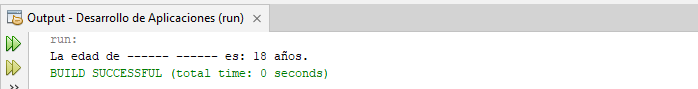


**Ejercicio 5**. Modificar el ejercicio anterior para que la clase “Persona” agregue una validación para el atributo edad y si no cumple la validación llame a otro constructor usando la referencia “this”.





Y cambiando el parámetro edad a 150 muestra el siguiente resultado:



**Ejercicio 6**. Desarrollar una aplicación que tenga una clase “Fecha” y que tenga como atributos dia, mes y año, los tres con tipo de dato int y modificador de acceso privado. La clase debe tener sus métodos getters y setters para acceder a los atributos, un constructor sin parámetros donde se le asigne como valores: dia = 1, mes = 1, año = 2000 y un constructor con parámetros donde reciba valores para asignarle a los tres atributos una fecha ingresada por teclado. La clase debe tener un método mostrarFecha() que devuelva un valor tipo String con el siguiente resultado “dia/mes/año”. La aplicación debe tener un método main que al ejecutarse debe mostrar lo siguiente:

Fecha Inicial: 1/1/2000

Fecha Final: 24/2/2023

**Ejercicio 7**. Modificar el ejercicio anterior para que tenga un método donde se valide si los valores que tienen dia, mes y año son una fecha correcta. Este método debe ser llamado en el constructor con parámetros. Si la fecha es incorrecta, debe asignarle una fecha correcta cualquiera.

**Ejercicio 8**. Desarrollar una aplicación que tenga una clase “Lampara” y que tenga como atributos “color” que es un String e “ilumina” que es boolean que indica si está encendida (true) o apagada (false). Una vez creado el objeto no se puede modificar el color, solo se puede prender o apagar la lampara. Crear lo métodos necesarios (getters, setters, constructor, etc.) para que se cumplan las indicaciones dadas. Mostrar la funcionalidad a través de un método main, en donde se cree un objeto de la clase y se pueda visualizar el estado del objeto cuando se prenda y apague la lampara.

**Ejercicio 9**. Definir una clase “Taxi” que debe ser pública y debe estar en su archivo “Taxi.java”. La clase debe tener los siguientes atributos:

* placa (String, que solo permita 6 caracteres alfanumericos y no puede ser modificado luego de ser inicializado.).
* conductor (String, para nombre y apellido).
* permiso (boolean, que representa si su permiso está activo).
* servicios (int, que almacena la cantidad de servicios que realiza el taxi).
* ganancia (double, que almacena la suma de lo que cobra en cada servicio).

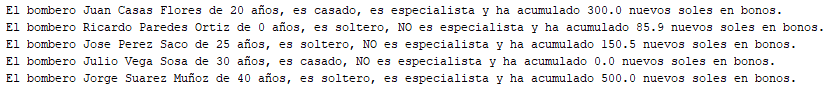
La clase debe tener los siguientes métodos:

* Un constructor con parámetros donde se inicialicen sus atributos.
* Los getters y setters necesarios.
* Un método que active e inactive su permiso.
* Un método que acumule la cantidad de servicios que va realizando el taxi. El taxi no puede realizar servicios cuando su permiso está inactivo. Este método debe recibir como parámetro el precio del servicio y se debe acumular en su atributo respectivo.
* Un método que muestre los datos del taxi.

Se debe crear un método main en una clase de otro paquete que utilice la clase “Taxi”.

**Ejercicio 10**. Definir una clase pública “Bombero” considerando los siguientes atributos de clase: nombre (String), apellidos (String), edad (int, solo se permiten bomberos mayores de edad y hasta los 60 años), casado (boolean), especialista (boolean), totalBono (double, acumula los bonos recibidos) y BONO (double, constante que representa el monto estándar que recibe cada bombero por atender una emergencia). La clase debe tener los siguientes métodos:

* Un constructor que reciba parámetros para la inicialización solo de los atributos nombre y apellidos.
* Un constructor que reciba parámetros para la inicialización solo de los atributos nombre, apellidos y edad. Si la edad es incorrecta debe utilizar el constructor anterior para asignar solo nombre y apellidos y a la edad asignarle 18 (Usar referencia this en la primera línea).
* Un constructor que reciba parámetros para la inicialización de todos los atributos excepto totalBono. Debe usar la referencia this para llamar al constructor anterior para validar la edad.
* Un constructor que reciba parámetros para la inicialización de todos los atributos. Debe usar la referencia this para llamar al constructor anterior para validar la edad.
* Los métodos para poder establecer y obtener los valores de los atributos.
* Un método para asignarle un bono al bombero por haber atendido una emergencia. El bono se duplica si el bombero es casado o especilista y se triplica si es casado y especialista a la vez.
* Un método para mostrar los datos del bombero, siguiendo el siguiente patrón:



Se debe crear un método main en una clase de otro paquete que utilice la clase “Bombero”.