

ciclo: [DAM]

MÓDULO DE [PROGRAMACIÓN]

[Tarea Nº 04]

Alumno:

[Juan Carlos Filter Martín]

[15456141A]

**Contenido**

[1. Documentos que se adjuntan a este informe. 3](#__RefHeading___Toc83_3824614597)

[2. RA04\_e) Se han desarrollado programas que instancie y utilicen objetos de las clases creadas anteriormente. 3](#__RefHeading___Toc85_3824614597)

[Crear proyecto con nombre y apellidos. 3](#__RefHeading___Toc87_3824614597)

[Dentro del proyecto crear un paquete donde se creen las clases que se piden. 4](#__RefHeading___Toc89_3824614597)

[Clase Persona 5](#__RefHeading___Toc91_3824614597)

[3. RA04\_f) Se han utilizado mecanismos para controlar la visibilidad de las clases y de sus miembros. 6](#__RefHeading___Toc93_3824614597)

[Crear atributos privados: nombre y edad. 6](#__RefHeading___Toc95_3824614597)

[4. RA04\_d) Se han creado constructores. 6](#__RefHeading___Toc97_3824614597)

[Crear dos constructores (con parámetros y sin parámetros) 6](#__RefHeading___Toc99_3824614597)

[Crear getter y setter para los atributos de la clase. 7](#__RefHeading___Toc101_3824614597)

[5. RA04\_g) Se han definido y utilizado clases heredadas. 7](#__RefHeading___Toc103_3824614597)

[Crear una clase Camarero que herede de Persona. 7](#__RefHeading___Toc105_3824614597)

[6. RA04\_h) Se han creado y utilizado métodos estáticos. 8](#__RefHeading___Toc107_3824614597)

[Crear un método estático en la clase camarero que muestre un contador. 8](#__RefHeading___Toc109_3824614597)

[7. RA04\_i) Se han creado y utilizado conjuntos y librerías de clases. 10](#__RefHeading___Toc111_3824614597)

[La función main será la que usemos para probar que todo funciona correctamente. 10](#__RefHeading___Toc113_3824614597)

[Puesto que las clases están en un paquete distinto al del proyecto, incorporar ese paquete a la aplicación. 10](#__RefHeading___Toc468_4194412131)

[Crear un objeto que nos permita crear una persona. 11](#__RefHeading___Toc470_4194412131)

[Crear varios objetos que nos permita crear varios camareros. 12](#__RefHeading___Toc472_4194412131)

[A través del método estático mostrar el número de camareros creado. 13](#__RefHeading___Toc474_4194412131)

[Indica en el informe, qué sentencia usarías para crear una interface. 14](#__RefHeading___Toc476_4194412131)

[Ejecutar la aplicación con estos valores 16](#__RefHeading___Toc478_4194412131)

# Documentos que se adjuntan a este informe.

A continuación se detallan los documentos que componen la presente entrega de la tarea:

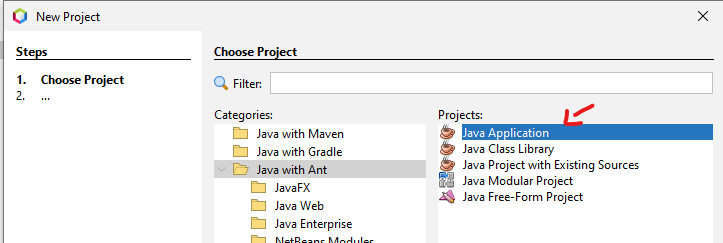
1. Informe de elaboración de la tarea.
2. Proyecto NetBeans

# RA04\_e) Se han desarrollado programas que instancie y utilicen objetos de las clases creadas anteriormente.

## Crear proyecto con nombre y apellidos.

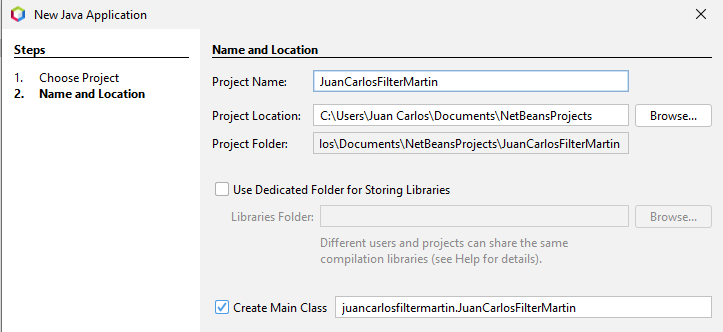
Para ello vamos a New project

En mi caso voy a elegir Ant y java application



Ahora en esta ventana pondremos el nombre de proyecto:

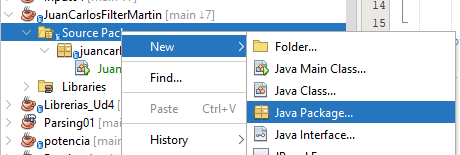
**JuanCarlosFilterMartin**



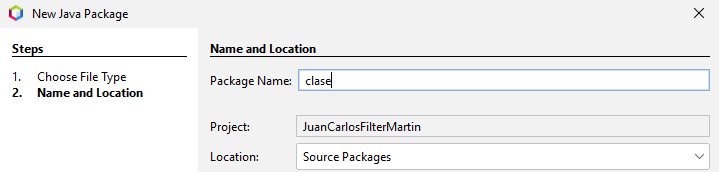
## Dentro del proyecto crear un paquete donde se creen las clases que se piden.

Con botón derecho donde se encuentra la carpeta de los paquetes

**New > Java Package..**



Ahora introducimos el nombre del paquete **clase**

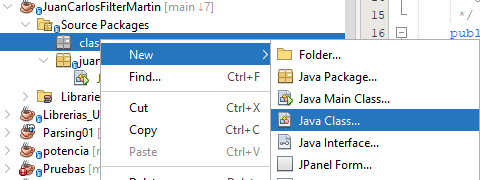


Ya tendríamos el paquete **Clases** creado

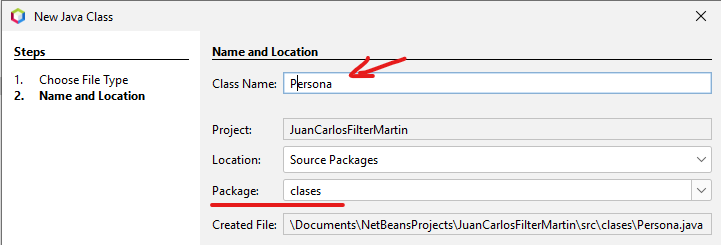


### Clase Persona

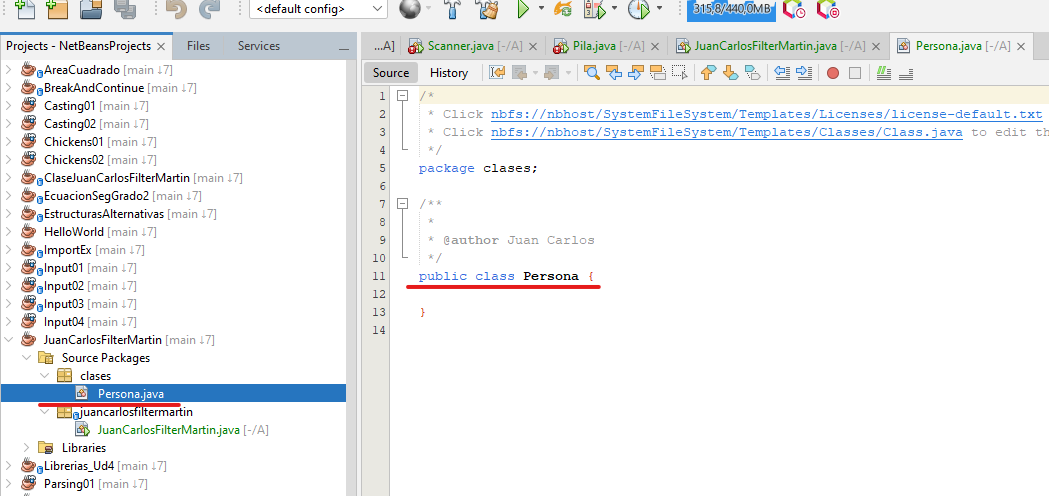
Para crear la clase **Persona** vamos al paquete **clases > botón derecho > java Class..**  (*Para que podamos crear una clase)*



Introducimos el nombre de la clase **Persona** y como podemos ver también indica donde se va a crear en el paquete **clases**



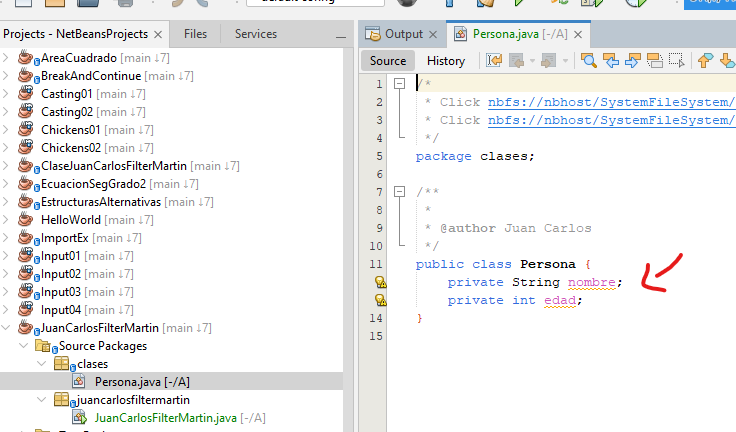
Con esto ya tendríamos la clase **Persona** creada



# RA04\_f) Se han utilizado mecanismos para controlar la visibilidad de las clases y de sus miembros.

## Crear atributos privados: nombre y edad.

Para ellos tendremos que poner **private** delante del tipo de dato de la variable



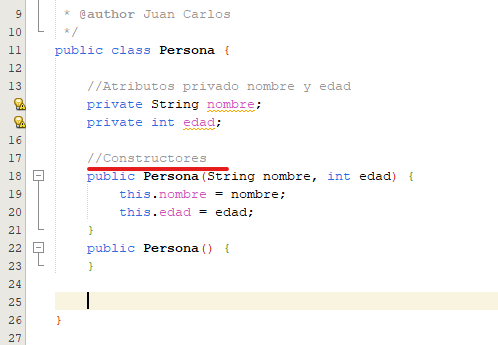
*(Al ser una clase que va a heredar estos atributos no estaría mal ponerlos protected, de no ser así la clase camarero que se cre más adelante extends persona para poder acceder a estos solo podrá mediante los métodos get y set public)*

# RA04\_d) Se han creado constructores.

## Crear dos constructores (con parámetros y sin parámetros)

Para crear los constructores tenemos que poner **public** seguido del nombre de la clase.

(Para un constructor con parámetros hay que indicar el tipo de dato seguido del nombre y dentro del constructor decir que la variable de la clase = a la variable del parámetro del constructor)

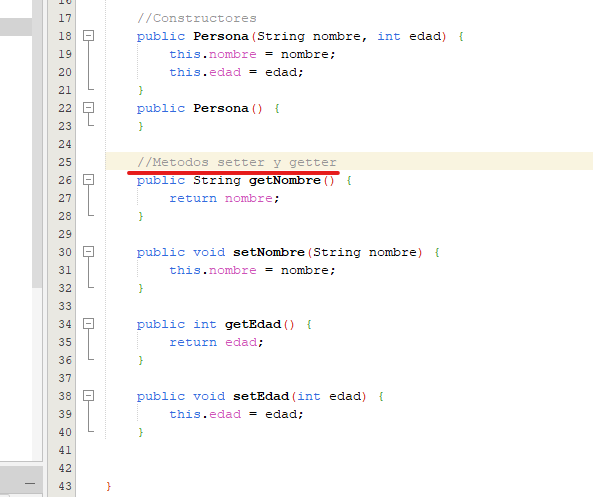


## Crear getter y setter para los atributos de la clase.

**→ A***tributos privados y solo accedidos a través de los getter y setter.*

Para ello lo indicamos mediante public :

* seguido del tipo de dato → get
* Seguido de la palabra reservada void (parámetros a introducir) → set

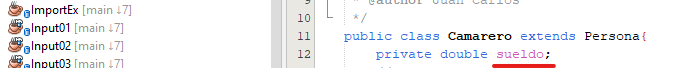


# RA04\_g) Se han definido y utilizado clases heredadas.

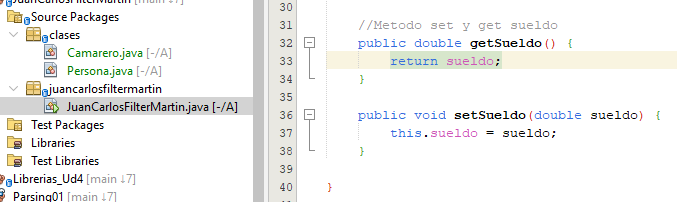
## Crear una clase Camarero que herede de Persona.

*→ Con un atributo llamado sueldo, donde introduciremos el sueldo del camarero.*

Primero creamos el atributo privado y de **tipo double** (porque es un sueldo y puede contener decimales)

**

Y para poder modificar este sueldo o acceder a el para que lo muestre creamos métodos set y get de sueldo

**

# RA04\_h) Se han creado y utilizado métodos estáticos.

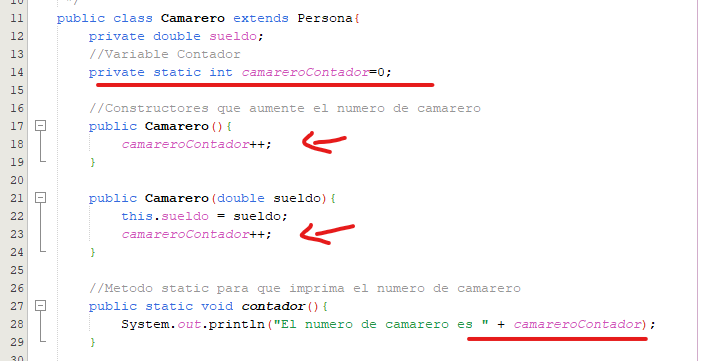
## Crear un método estático en la clase camarero que muestre un contador.

*→ Se incrementará cada vez que creemos un camarero nuevo.*

Lo primero es crear una variable **camareroContador** inicializada a 0 de tipo **privado y static**

Ahora crearemos los constructores con y sin parámetros y dentro de ellos agregar un incremento a esa variable de 1 (**camareroContador++;**). Para que cuando llamemos a alguno constructor al crear ese objeto la variable aumente.

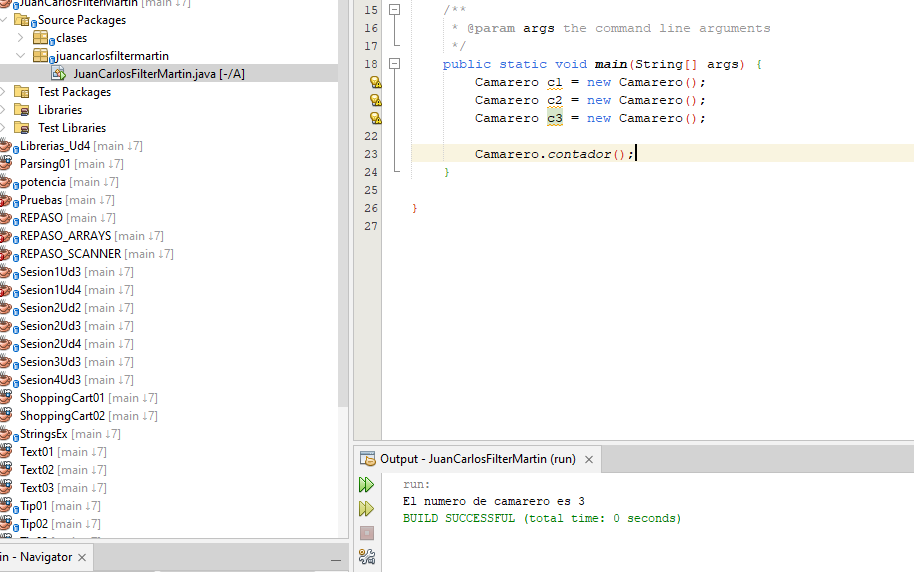
Por ultimo mediante el **método static** contador **imprimirá** por pantalla el resultado de la variable **camareroContador**



Si vamos a la clase main **JuanCarlosFilterMartin**  y creamos varios objetos de la clase Camarero podemos ver como al introducir :

**Camarero.contador** *(Al ser un metodo estatico se llama directamente de la clase)*

Podemos ver el resultado que es 3



# RA04\_i) Se han creado y utilizado conjuntos y librerías de clases.

## La función main será la que usemos para probar que todo funciona correctamente.

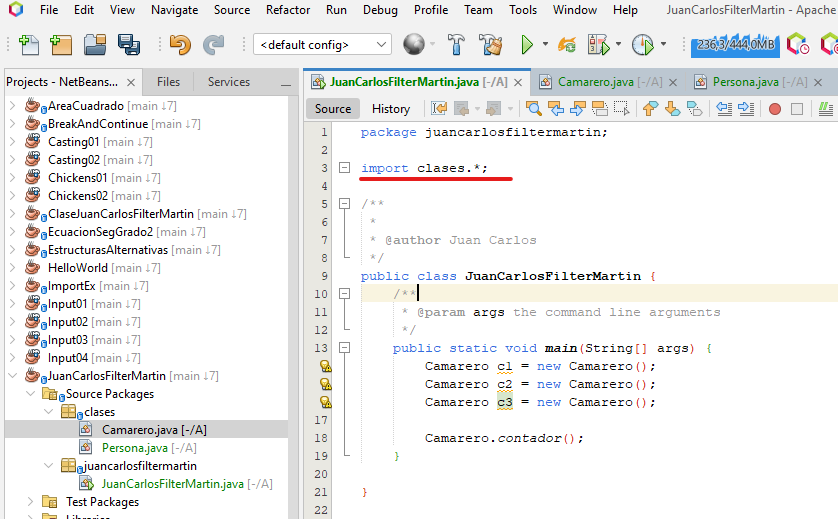
*→ Para ello realizaremos las siguientes acciones:*

### Puesto que las clases están en un paquete distinto al del proyecto, incorporar ese paquete a la aplicación.

Indicamos en la clase main (encima y fuera de la propia clase) con la palabra reservada import el paquete de clases:

**import.clases.\***

(*Al poner el \* indicamos que importe todo lo que se encuentra en ese paquete)*

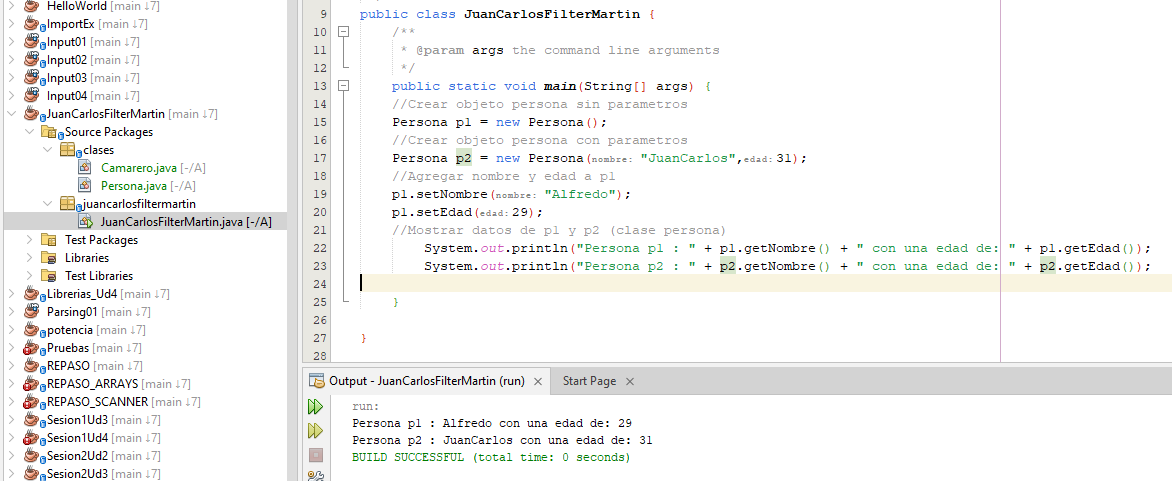


### Crear un objeto que nos permita crear una persona.

Se ha creado 2 objetos persona (p1 y p2) uno sin parámetro y otro con parámetros.

Se le han indicarlo los paramentos a p1 mediante los métodos set

y por ultimo se muestra por pantalla con un **System.out.println** y los métodos get



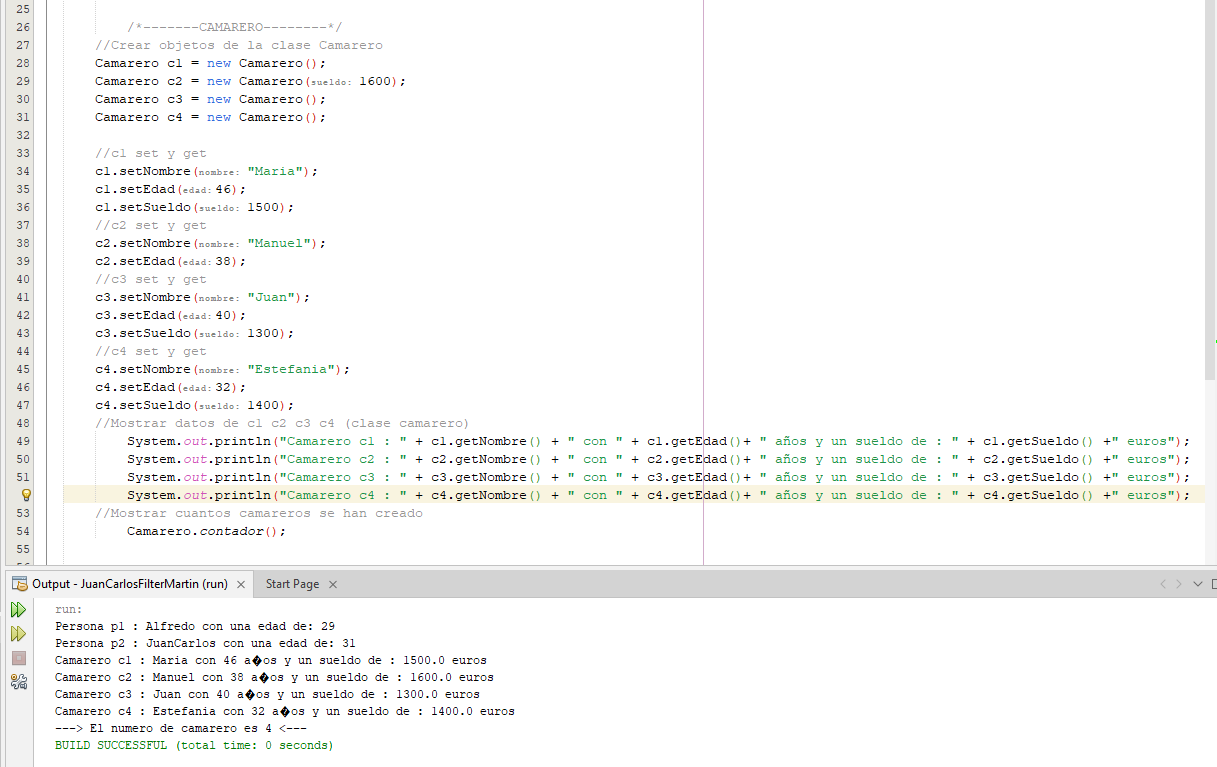
### Crear varios objetos que nos permita crear varios camareros.

Se ha creado 4 objetos camarero (c1, c2, c3 y c4) sin parámetros y uno de ellos con el parámetro que nos permite la clase camarero

(como *los atributos de la clase persona son private la clase camarero no tiene acceso directo y solamente puede acceder a ellos mediante los métodos set y get. Para heredar estas variables deberían ser protected)*

Se le han indicarlo los paramentos mediante los métodos set

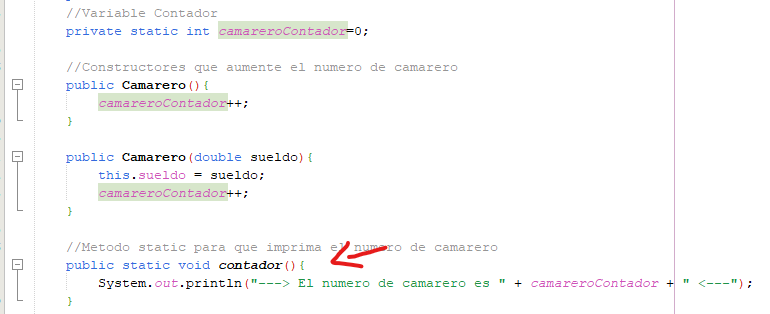
y por ultimo se muestra por pantalla con un **System.out.println** y los métodos get

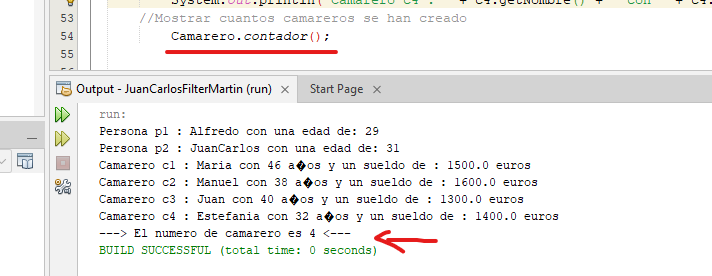


### A través del método estático mostrar el número de camareros creado.

Mediante el método estático creado vamos la clase main y si tenemos objetos camarero creado introducimos:

Nombre de la clase y el método → **Camarero.contador();**

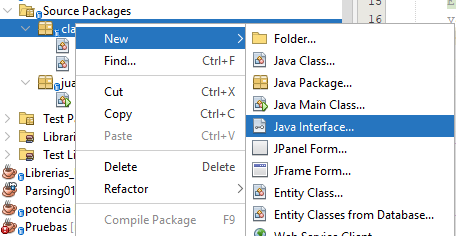




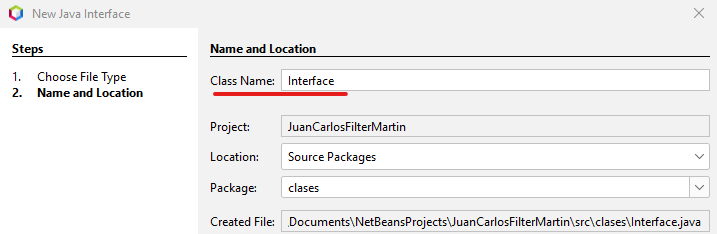
### Indica en el informe, qué sentencia usarías para crear una interface.

Lo primero para crear una interface desde el propio paquete de clases como puede ser el de camarero y persona, mediante el botón derecho tenemos la opción de java interface

**new > java interface**

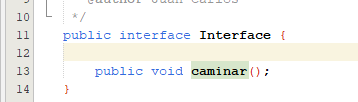


Aquí podemos crear una nueva interface asignándole un nombre a esta



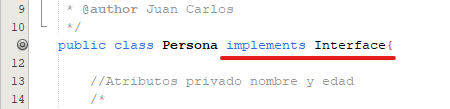
Con la interface creada podemos indicarle métodos que varías clases puede utilizar y tienen en común o usarlo solamente para 2 clases de 3 que tiene el package

Por ejemplo el método caminar

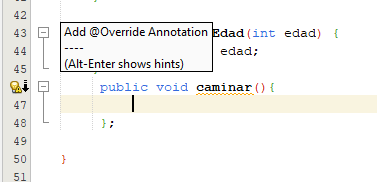


Este método se puede usar en las clases agregándole **implements**  y seguido el nombre de la interface

(Se puede poner tantas interfaces diferentes como necesites)



Una vez implementada podemos escribir el método de esa interface haciendo una sobre escritura.



Por último decir que si por ejemplo tenemos una clase avión (este avión no puede camino) entonces no es necesario agregarle la interface)

Las interfaces se utilizan para cuando se necesita acceder a esos métodos que contiene

### Ejecutar la aplicación con estos valores

Realiza las capturas de los resultados obtenidos que deberán ir dentro del informe que se redacte, para comprobar los resultados en cada caso.

