



**CICLO: [DAM]**  
**MÓDULO DE [PROGRAMACIÓN]**

## **[Tarea N° 04]**

**Alumno:**  
**[Juan Carlos Filter Martín]**  
**[15456141A]**

## Contenido

<b>1. Documentos que se adjuntan a este informe.....</b>	<b>3</b>
<b>2. RA04_e) Se han desarrollado programas que instancie y utilicen objetos de las clases creadas anteriormente.....</b>	<b>3</b>
Crear proyecto con nombre y apellidos.....	3
Dentro del proyecto crear un paquete donde se creen las clases que se piden.....	4
Clase Persona.....	5
<b>3. RA04_f) Se han utilizado mecanismos para controlar la visibilidad de las clases y de sus miembros.....</b>	<b>6</b>
Crear atributos privados: nombre y edad.....	6
<b>4. RA04_d) Se han creado constructores.....</b>	<b>6</b>
Crear dos constructores (con parámetros y sin parámetros).....	6
Crear getter y setter para los atributos de la clase.....	7
<b>5. RA04_g) Se han definido y utilizado clases heredadas.....</b>	<b>7</b>
Crear una clase Camarero que herede de Persona.....	7
<b>6. RA04_h) Se han creado y utilizado métodos estáticos.....</b>	<b>8</b>
Crear un método estático en la clase camarero que muestre un contador.....	8
<b>7. RA04_i) Se han creado y utilizado conjuntos y librerías de clases.....</b>	<b>10</b>
La función main será la que usemos para probar que todo funciona correctamente.....	10
Puesto que las clases están en un paquete distinto al del proyecto, incorporar ese paquete a la aplicación.....	10
Crear un objeto que nos permita crear una persona.....	11
Crear varios objetos que nos permita crear varios camareros.....	12
A través del método estático mostrar el número de camareros creado.....	13
Indica en el informe, qué sentencia usarías para crear una interface.....	14
Ejecutar la aplicación con estos valores.....	16

## 1. Documentos que se adjuntan a este informe.

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

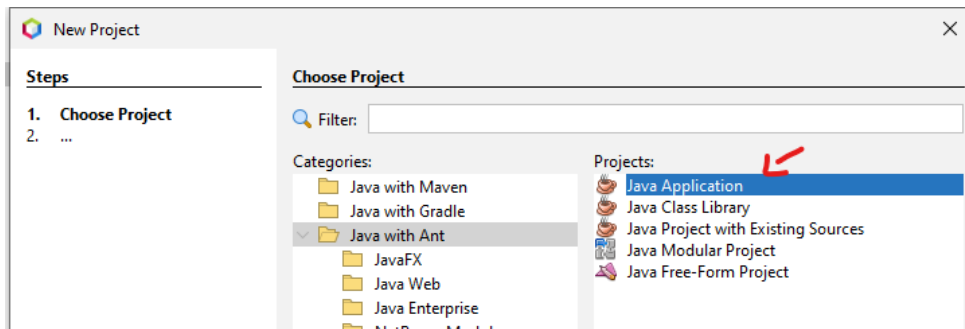
1. Informe de elaboración de la tarea.
2. Proyecto NetBeans

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

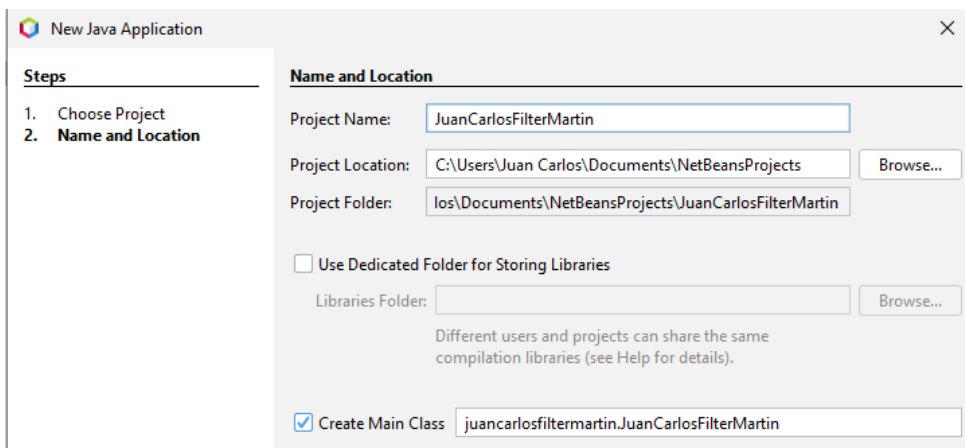
### Crear proyecto con nombre y apellidos.

Para ello vamos a New project

En mi caso voy a elegir Ant y java application

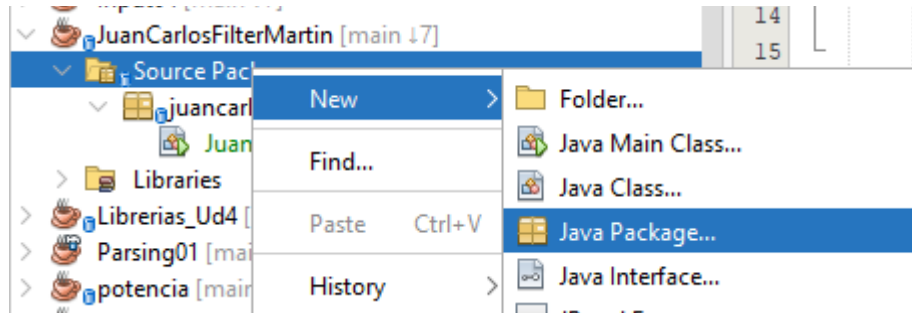


Ahora en esta ventana pondremos el nombre de proyecto:  
**JuanCarlosFilterMartin**

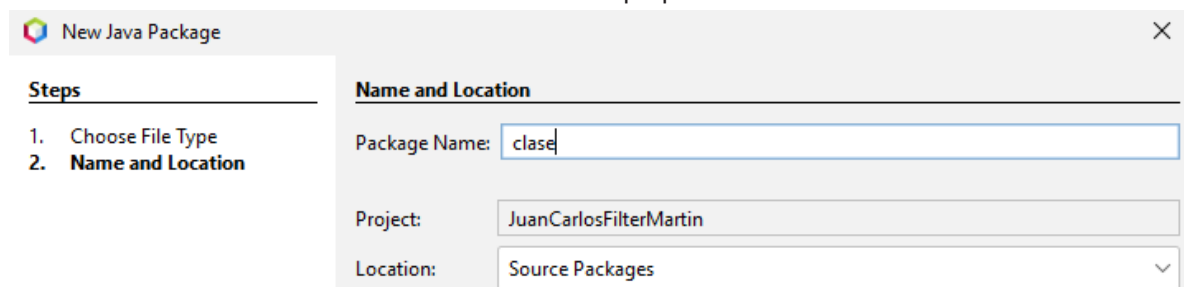


**Dentro del proyecto crear un paquete donde se creen las clases que se piden.**

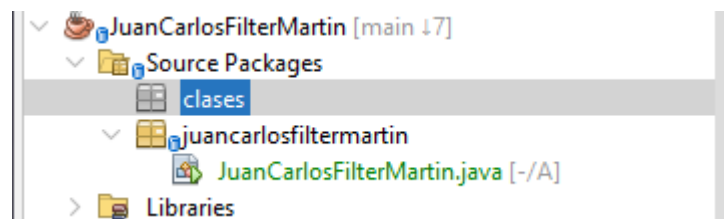
Con botón derecho donde se encuentra la carpeta de los paquetes  
**New > Java Package..**



Ahora introducimos el nombre del paquete **clase**

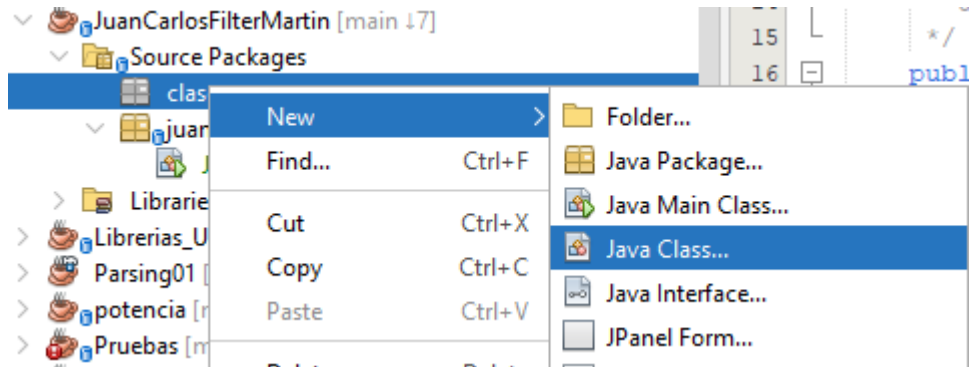


Ya tendríamos el paquete **Clases** creado

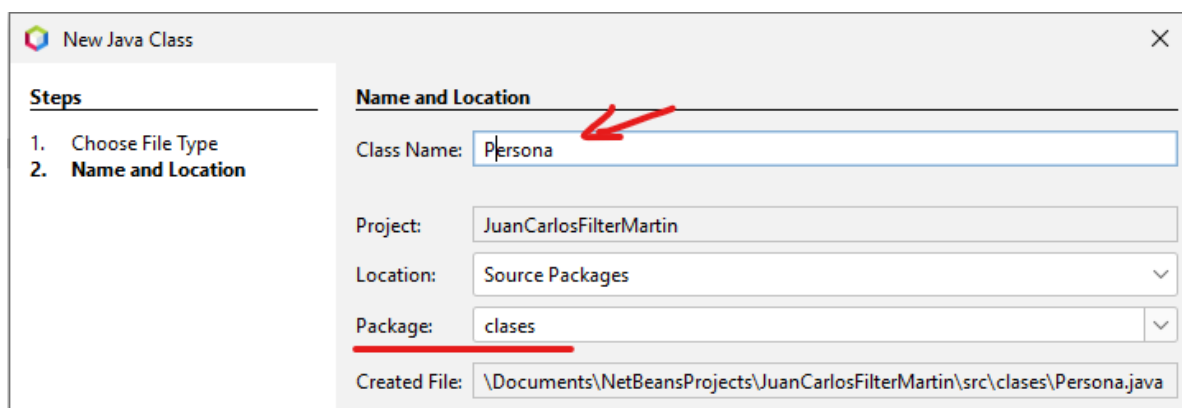


## Clase Persona

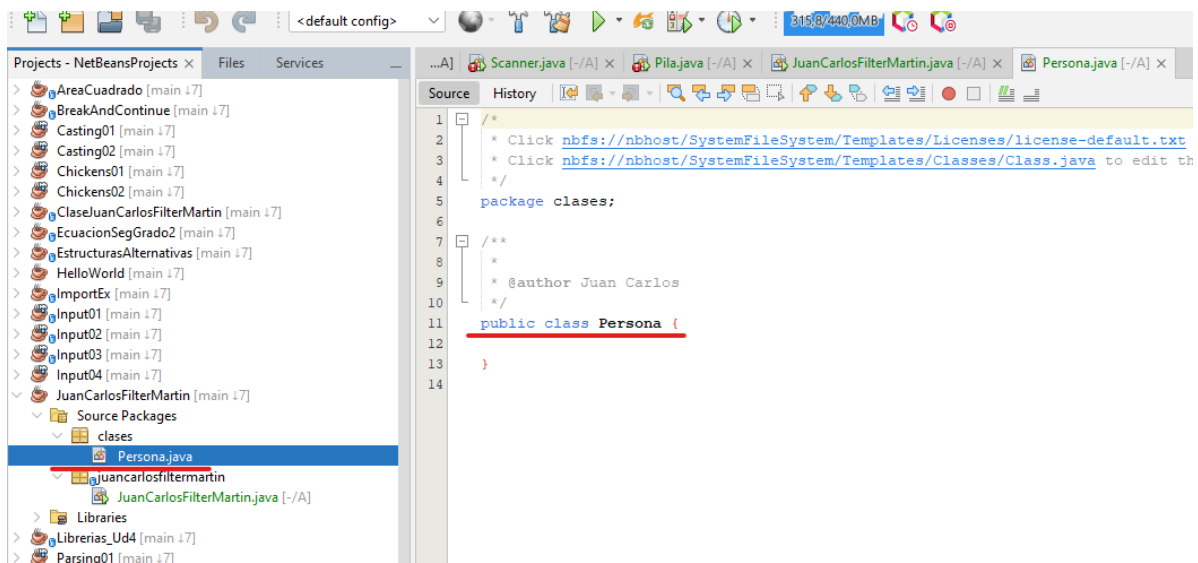
Para crear la clase **Persona** vamos al paquete **clases** > botón derecho > **java Class..** (Para que podamos crear una clase)



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



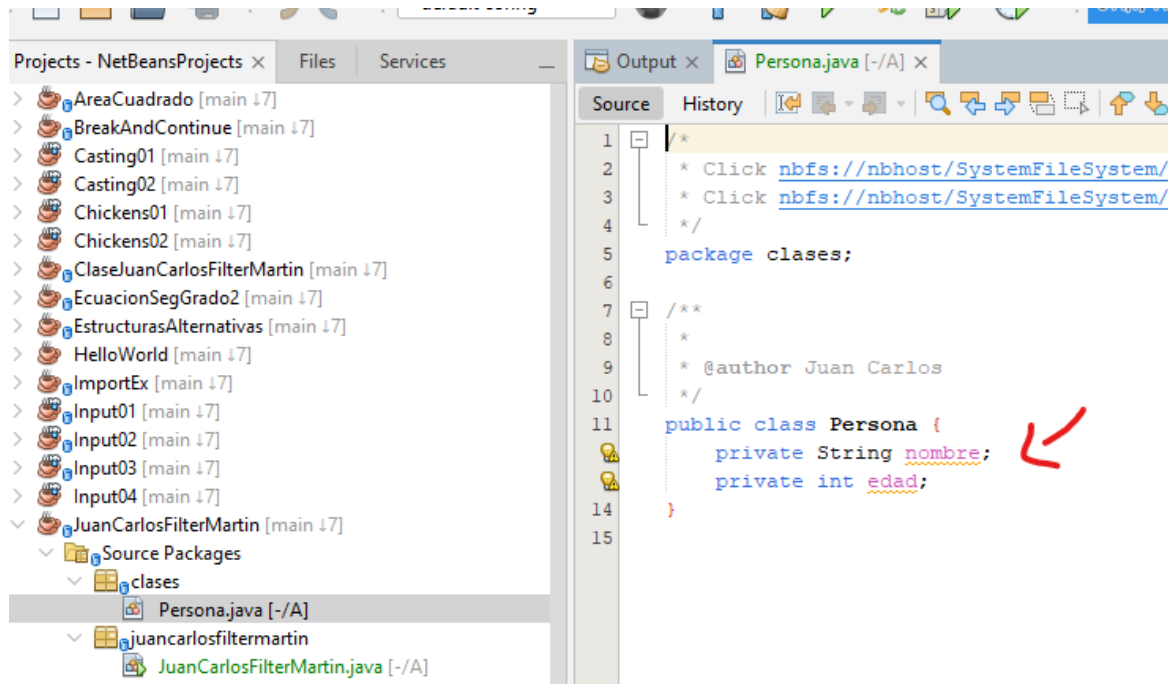
Con esto ya tendríamos la clase **Persona** creada



### 3. 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)

### 4. 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)

```
11  public class Persona {
12
13      //Atributos privado nombre y edad
14      private String nombre;
15      private int edad;
16
17      //Constructores
18      public Persona(String nombre, int edad) {
19          this.nombre = nombre;
20          this.edad = edad;
21      }
22      public Persona() {
23      }
```

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

→ Atributos 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

```
16
17 //Constructores
18 public Persona(String nombre, int edad) {
19     this.nombre = nombre;
20     this.edad = edad;
21 }
22 public Persona() {
23 }
24
25 //Metodos setter y getter
26 public String getNombre() {
27     return nombre;
28 }
29
30 public void setNombre(String nombre) {
31     this.nombre = nombre;
32 }
33
34 public int getEdad() {
35     return edad;
36 }
37
38 public void setEdad(int edad) {
39     this.edad = edad;
40 }
41
42
43 }
```

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

### Crear una clase Camarero que herede de Persona.

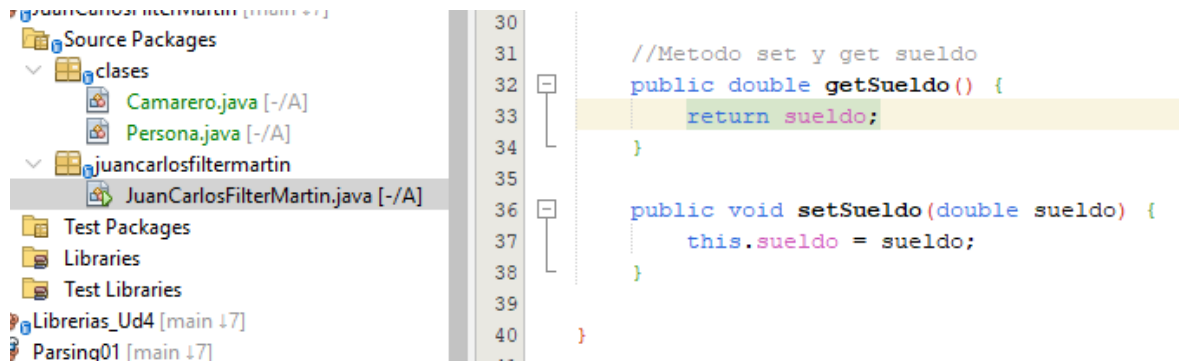
→ Con un atributo llamado sueldo, donde introduciremos el sueldo del camarero.

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

```
ImportEx [main 17]
Input01 [main 17]
Input02 [main 17]
Input03 [main 17]

10 */
11 public class Camarero extends Persona{
12     private double sueldo;
```

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



The screenshot shows an IDE with a project structure on the left and a code editor on the right. The project structure includes 'Source Packages' with 'clases' containing 'Camarero.java' and 'Persona.java', and 'juancarlosfiltermartin' containing 'JuanCarlosFilterMartin.java'. The code editor shows the implementation of the 'getSueldo' method in 'Camarero.java'.

```
30
31 //Metodo set y get sueldo
32 public double getSueldo() {
33     return sueldo;
34 }
35
36 public void setSueldo(double sueldo) {
37     this.sueldo = sueldo;
38 }
39
40 }
```

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

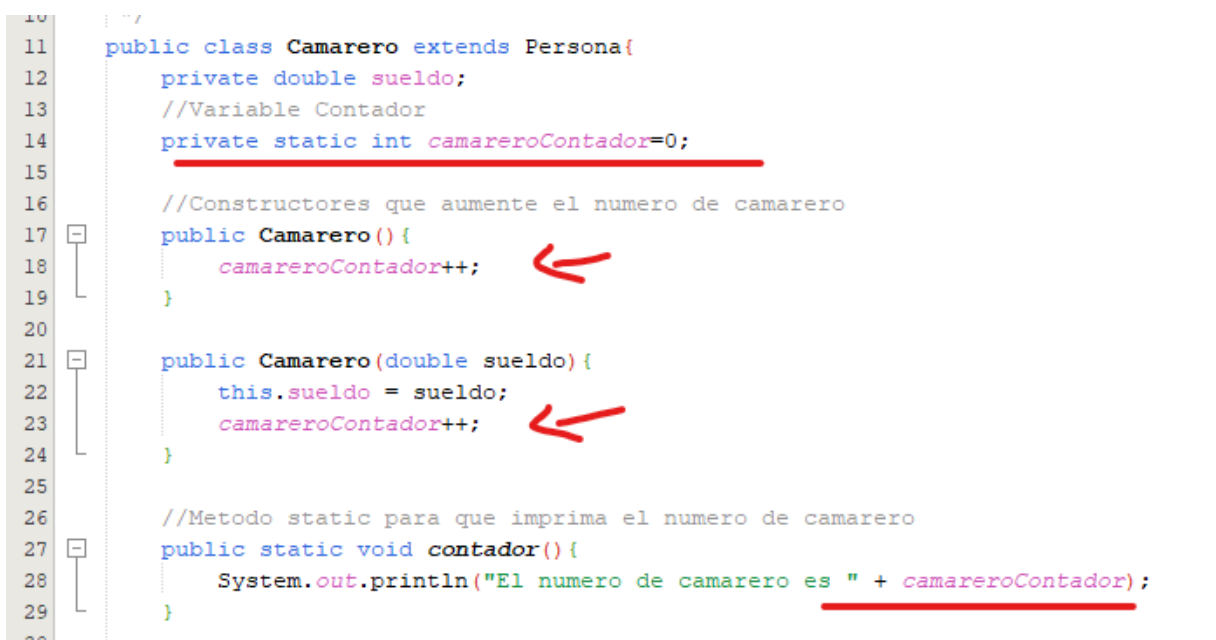
Crear un método estático en la clase camarero que muestre un contador.

→ Se incrementará cada vez que creamos 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**



The screenshot shows the implementation of the 'Camarero' class in 'Camarero.java'. The class extends 'Persona'. It has a private double variable 'sueldo' and a private static int variable 'camareroContador' initialized to 0. There are two constructors: a no-argument constructor and a constructor that takes a 'double sueldo' parameter. Both constructors increment 'camareroContador'. There is also a static method 'contador' that prints the value of 'camareroContador'.

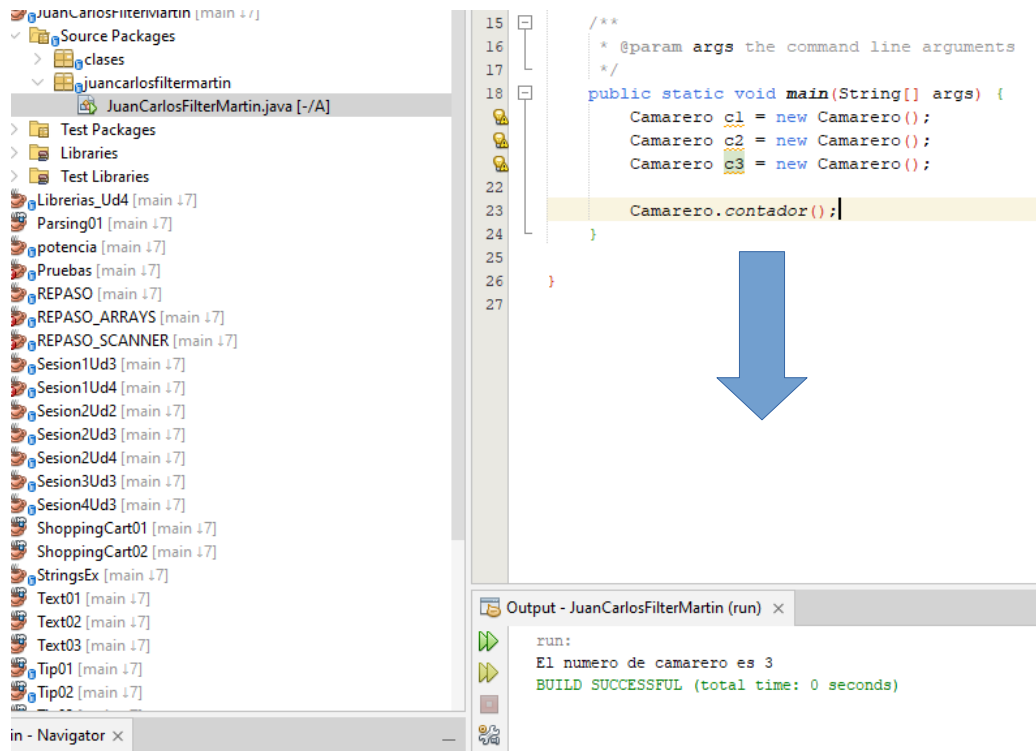
```
10
11 public class Camarero extends Persona{
12     private double sueldo;
13     //Variable Contador
14     private static int camareroContador=0;
15
16     //Constructores que aumente el numero de camarero
17     public Camarero(){
18         camareroContador++;
19     }
20
21     public Camarero(double sueldo){
22         this.sueldo = sueldo;
23         camareroContador++;
24     }
25
26     //Metodo static para que imprima el numero de camarero
27     public static void contador(){
28         System.out.println("El numero de camarero es " + camareroContador);
29     }
30 }
```



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



The screenshot shows an IDE with the following components:

- Navigator:** A tree view on the left showing the project structure. The file `JuanCarlosFilterMartin.java` is selected under the `clases` package.
- Editor:** The main window displays the code for `JuanCarlosFilterMartin.java`. The code is as follows:

```
15  /**  
16   * @param args the command line arguments  
17   */  
18  public static void main(String[] args) {  
19      Camarero c1 = new Camarero();  
20      Camarero c2 = new Camarero();  
21      Camarero c3 = new Camarero();  
22  
23      Camarero.contador();  
24  }  
25  
26  
27
```

  
A large blue arrow points from the `Camarero.contador();` line to the output window.
- Output:** A window at the bottom right titled "Output - JuanCarlosFilterMartin (run)" shows the execution results:

```
run:  
El numero de camarero es 3  
BUILD SUCCESSFUL (total time: 0 seconds)
```

## 7. 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.

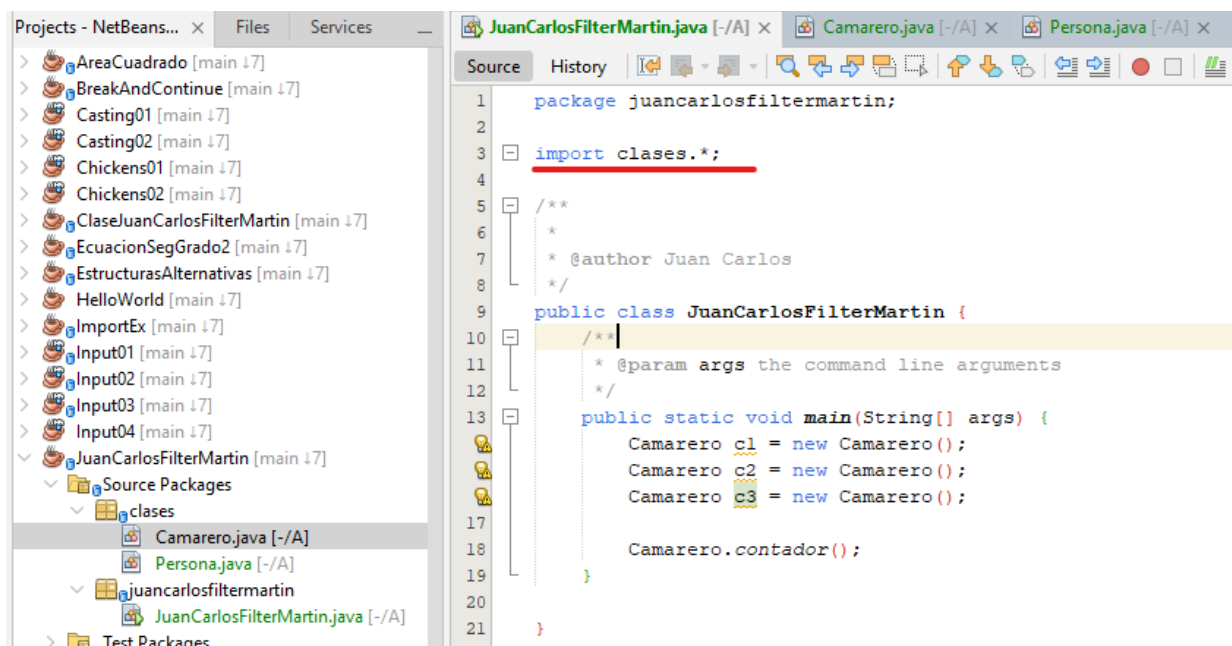
→ Para ello realizaremos las siguientes acciones:

- 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)

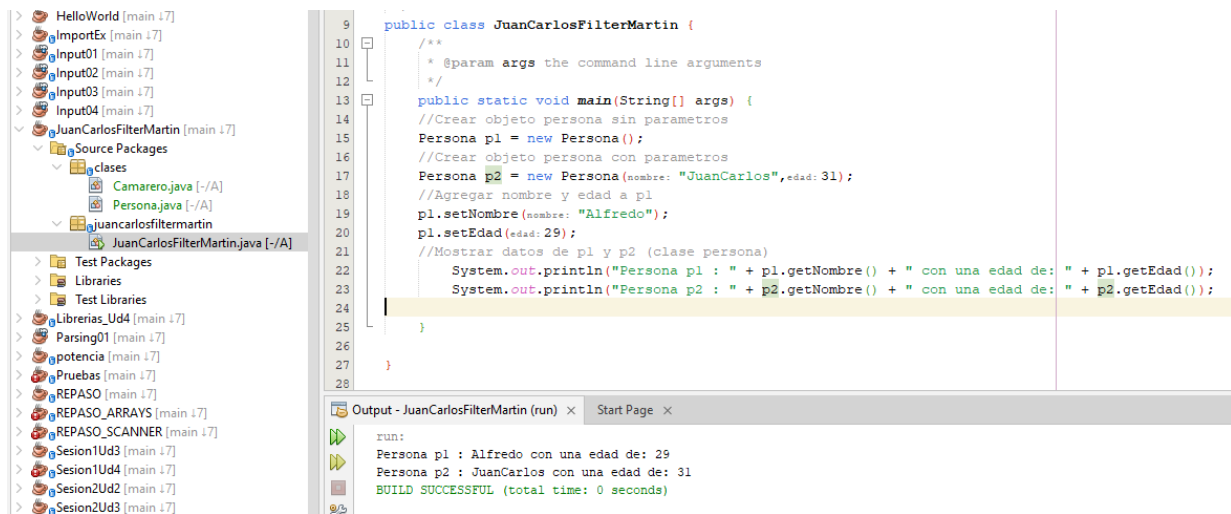


- **Crear un objeto que nos permita crear una persona.**

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

Se le han indicado los parámetros a p1 mediante los métodos set

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



The screenshot shows an IDE with a project named 'JuanCarlosFilterMartin'. The left sidebar displays the project structure, including 'Source Packages' with 'clases' containing 'Camarero.java' and 'Persona.java', and 'JuanCarlosFilterMartin' containing 'JuanCarlosFilterMartin.java'. The main editor shows the code for 'JuanCarlosFilterMartin.java'.

```
9 public class JuanCarlosFilterMartin {
10     /**
11      * @param args the command line arguments
12      */
13     public static void main(String[] args) {
14         //Crear objeto persona sin parametros
15         Persona p1 = new Persona();
16         //Crear objeto persona con parametros
17         Persona p2 = new Persona(nombre: "JuanCarlos", edad: 31);
18         //Agregar nombre y edad a p1
19         p1.setNombre(nombre: "Alfredo");
20         p1.setEdad(edad: 29);
21         //Mostrar datos de p1 y p2 (clase persona)
22         System.out.println("Persona p1 : " + p1.getNombre() + " con una edad de: " + p1.getEdad());
23         System.out.println("Persona p2 : " + p2.getNombre() + " con una edad de: " + p2.getEdad());
24     }
25 }
26
27
28
```

The output window shows the following text:

```
run:
Persona p1 : Alfredo con una edad de: 29
Persona p2 : JuanCarlos con una edad de: 31
BUILD SUCCESSFUL (total time: 0 seconds)
```

- 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 indicado los parámetros mediante los métodos set

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

```
25
26      /*-----CAMARERO-----*/
27      //Crear objetos de la clase Camarero
28      Camarero c1 = new Camarero();
29      Camarero c2 = new Camarero(sueldo: 1600);
30      Camarero c3 = new Camarero();
31      Camarero c4 = new Camarero();
32
33      //c1 set y get
34      c1.setNombre(nombre: "Maria");
35      c1.setEdad(edad: 46);
36      c1.setSueldo(sueldo: 1500);
37      //c2 set y get
38      c2.setNombre(nombre: "Manuel");
39      c2.setEdad(edad: 38);
40      //c3 set y get
41      c3.setNombre(nombre: "Juan");
42      c3.setEdad(edad: 40);
43      c3.setSueldo(sueldo: 1300);
44      //c4 set y get
45      c4.setNombre(nombre: "Estefania");
46      c4.setEdad(edad: 32);
47      c4.setSueldo(sueldo: 1400);
48      //Mostrar datos de c1 c2 c3 c4 (clase camarero)
49      System.out.println("Camarero c1 : " + c1.getNombre() + " con " + c1.getEdad() + " años y un sueldo de : " + c1.getSueldo() + " euros");
50      System.out.println("Camarero c2 : " + c2.getNombre() + " con " + c2.getEdad() + " años y un sueldo de : " + c2.getSueldo() + " euros");
51      System.out.println("Camarero c3 : " + c3.getNombre() + " con " + c3.getEdad() + " años y un sueldo de : " + c3.getSueldo() + " euros");
52      System.out.println("Camarero c4 : " + c4.getNombre() + " con " + c4.getEdad() + " años y un sueldo de : " + c4.getSueldo() + " euros");
53
54      //Mostrar cuantos camareros se han creado
55      Camarero.contador();
```

Output - JuanCarlosFilterMartin (run) × Start Page ×

```
run:
Persona p1 : Alfredo con una edad de: 29
Persona p2 : JuanCarlos con una edad de: 31
Camarero c1 : Maria con 46 años y un sueldo de : 1500.0 euros
Camarero c2 : Manuel con 38 años y un sueldo de : 1600.0 euros
Camarero c3 : Juan con 40 años y un sueldo de : 1300.0 euros
Camarero c4 : Estefania con 32 años y un sueldo de : 1400.0 euros
---> El numero de camarero es 4 <---
BUILD SUCCESSFUL (total time: 0 seconds)
```

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

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

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

```
//Variable Contador
private static int camareroContador=0;

//Constructores que aumente el numero de camarero
public Camarero() {
    camareroContador++;
}

public Camarero(double sueldo) {
    this.sueldo = sueldo;
    camareroContador++;
}

//Metodo static para que imprima el numero de camarero
public static void contador() {
    System.out.println("---> El numero de camarero es " + camareroContador + " <---");
}
```

```
53 //Mostrar cuantos camareros se han creado
54 Camarero.contador();
55
```

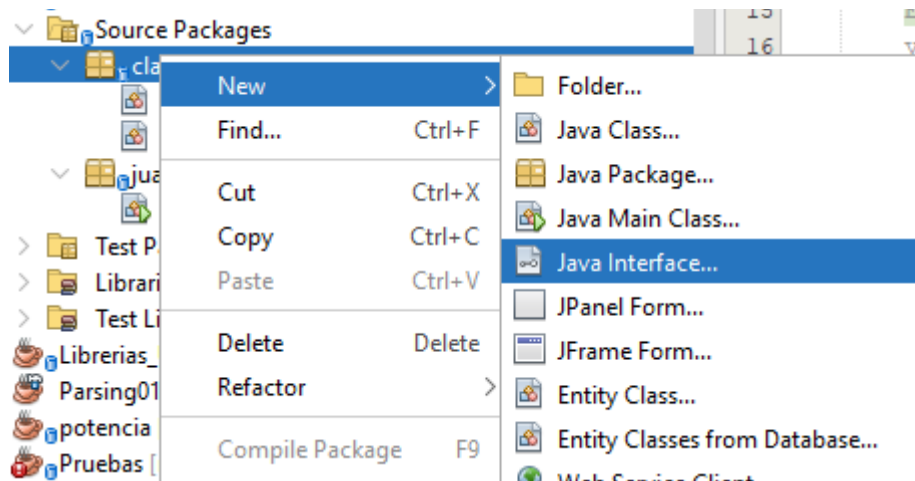
Output - JuanCarlosFilterMartin (run) × Start Page ×

```
run:
Persona p1 : Alfredo con una edad de: 29
Persona p2 : JuanCarlos con una edad de: 31
Camarero c1 : Maria con 46 años y un sueldo de : 1500.0 euros
Camarero c2 : Manuel con 38 años y un sueldo de : 1600.0 euros
Camarero c3 : Juan con 40 años y un sueldo de : 1300.0 euros
Camarero c4 : Estefania con 32 años y un sueldo de : 1400.0 euros
---> El numero de camarero es 4 <---
BUILD SUCCESSFUL (total time: 0 seconds)
```

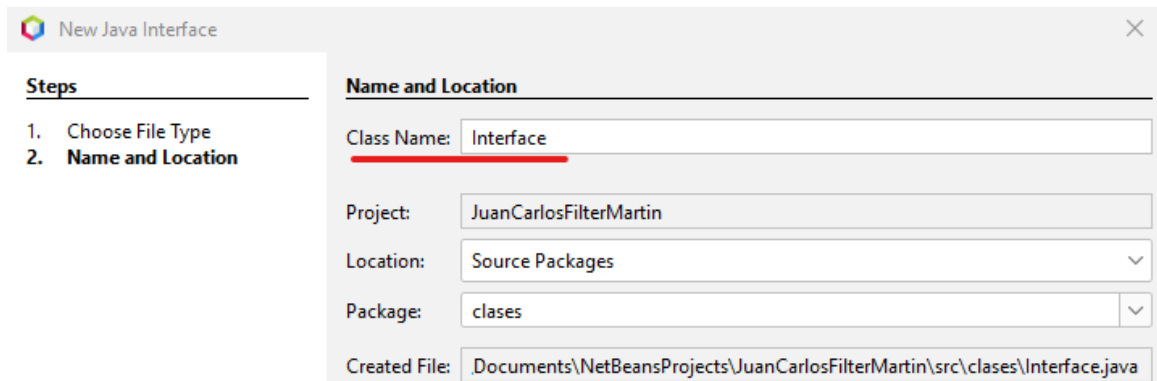
- Indica en el informe, qué sentencia usarías para crear una interface.

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 varias clases puede utilizar y tienen en común o usarlo solamente para 2 clases de 3 que tiene el package

Por ejemplo el método caminar

```

10  L  */
11  public interface Interface {
12
13      public void caminar();
14  }

```

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)

```
9      * @author Juan Carlos
10     */
11     public class Persona implements Interface{
12
13         //Atributos privado nombre y edad
14         /*
```

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

```
42
43     Add @Override Annotation Edad(int edad) {
44         ----
45         (Alt-Enter shows hints)
46
47     public void caminar() {
48
49
50
51     };
```

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.

## PERSONA

```

5 package clases;
6
7 /**
8  *
9  * @author Juan Carlos
10  */
11 public class Persona {
12
13     //Atributos privado nombre y edad
14     private String nombre;
15     private int edad;
16
17     //Constructores
18     public Persona(String nombre, int edad) {
19         this.nombre = nombre;
20         this.edad = edad;
21     }
22     public Persona() {
23     }
24
25     //Metodos setter y getter
26     public String getNombre() {
27         return nombre;
28     }
29
30     public void setNombre(String nombre) {
31         this.nombre = nombre;
32     }
33
34     public int getEdad() {
35         return edad;
36     }
37
38     public void setEdad(int edad) {
39         this.edad = edad;
40     }
41
42 }

```

## CAMARERO

```

8
9  *
10  * @author Juan Carlos
11  */
12 public class Camarero extends Persona{
13     private double sueldo;
14     //Variable Contador
15     private static int camareroContador=0;
16
17     //Constructores que aumente el numero de camarero
18     public Camarero(){
19         camareroContador++;
20     }
21
22     public Camarero(double sueldo){
23         this.sueldo = sueldo;
24         camareroContador++;
25     }
26
27     //Metodo static para que imprima el numero de camarero
28     public static void contador(){
29         System.out.println("---> El numero de camarero es " + cama
30     }
31
32     //Metodo set y get sueldo
33     public double getSueldo() {
34         return sueldo;
35     }
36
37     public void setSueldo(double sueldo) {
38         this.sueldo = sueldo;
39     }
40
41 }

```

## MAIN CLASS

```

9 public class JuanCarlosFilterMartin {
10
11     * @param args the command line arguments
12     */
13     public static void main(String[] args) {
14         /*-----PERSONA-----*/
15         //Crear objeto persona sin parametros
16         Persona p1 = new Persona();
17         //Crear objeto persona con parametros
18         Persona p2 = new Persona(nombre: "JuanCarlos", edad: 31);
19         //Agregar nombre y edad a p1
20         p1.setNombre(nombre: "Alfredo");
21         p1.setEdad(edad: 29);
22         //Mostrar datos de p1 y p2 (clase persona)
23         System.out.println("Persona p1 : " + p1.getNombre() + " con una edad de: " + p1.getEdad());
24         System.out.println("Persona p2 : " + p2.getNombre() + " con una edad de: " + p2.getEdad());
25
26         /*-----CAMARERO-----*/
27         //Crear objetos de la clase Camarero
28         Camarero c1 = new Camarero();
29         Camarero c2 = new Camarero(sueldo: 1600);
30         Camarero c3 = new Camarero();
31         Camarero c4 = new Camarero();
32
33         //c1 set y get
34         c1.setNombre(nombre: "Maria");
35         c1.setEdad(edad: 46);
36         c1.setSueldo(sueldo: 1500);
37
38         //c2 set y get
39         c2.setNombre(nombre: "Manuel");
40         c2.setEdad(edad: 38);
41
42         //c3 set y get
43         c3.setNombre(nombre: "Juan");
44         c3.setEdad(edad: 40);
45         c3.setSueldo(sueldo: 1300);
46
47         //c4 set y get
48         c4.setNombre(nombre: "Estefania");
49         c4.setEdad(edad: 32);
50         c4.setSueldo(sueldo: 1400);
51
52         //Mostrar datos de c1 c2 c3 c4 (clase camarero)
53         System.out.println("Camarero c1 : " + c1.getNombre() + " con " + c1.getEdad() + " años y un sueldo de : " + c1.getSueldo() + " euros");
54         System.out.println("Camarero c2 : " + c2.getNombre() + " con " + c2.getEdad() + " años y un sueldo de : " + c2.getSueldo() + " euros");
55         System.out.println("Camarero c3 : " + c3.getNombre() + " con " + c3.getEdad() + " años y un sueldo de : " + c3.getSueldo() + " euros");
56         System.out.println("Camarero c4 : " + c4.getNombre() + " con " + c4.getEdad() + " años y un sueldo de : " + c4.getSueldo() + " euros");
57
58         //Mostrar cuantos camareros se han creado
59         Camarero.contador();
60     }
61 }

```

```

run:
Persona p1 : Alfredo con una edad de: 29
Persona p2 : JuanCarlos con una edad de: 31
Camarero c1 : Maria con 46 años y un sueldo de : 1500.0 euros
Camarero c2 : Manuel con 38 años y un sueldo de : 1600.0 euros
Camarero c3 : Juan con 40 años y un sueldo de : 1300.0 euros
Camarero c4 : Estefania con 32 años y un sueldo de : 1400.0 euros
---> El numero de camarero es 4 ---
BUILD SUCCESSFUL (total time: 0 seconds)

```