



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

## **[Tarea N° 02]**

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

## Contenido

1. Documentos que se adjuntan a este informe.....	3
2. RA2_b) Se han escrito programas simples.....	3
3. RA2_c) Se han instanciado objetos a partir de clases predefinidas.....	4
4. RA2_e) Se han escrito llamadas a métodos estáticos.....	8
5. RA2_f). Se han utilizado parámetros en la llamada a métodos.....	9
6. RA2_g) Se han incorporado y utilizado librerías de objetos.....	10
7. RA2_h) Se han utilizado constructores.....	11
8. RA2_i) Se ha utilizado el entorno integrado de desarrollo en la creación y compilación de programas simples.....	12

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

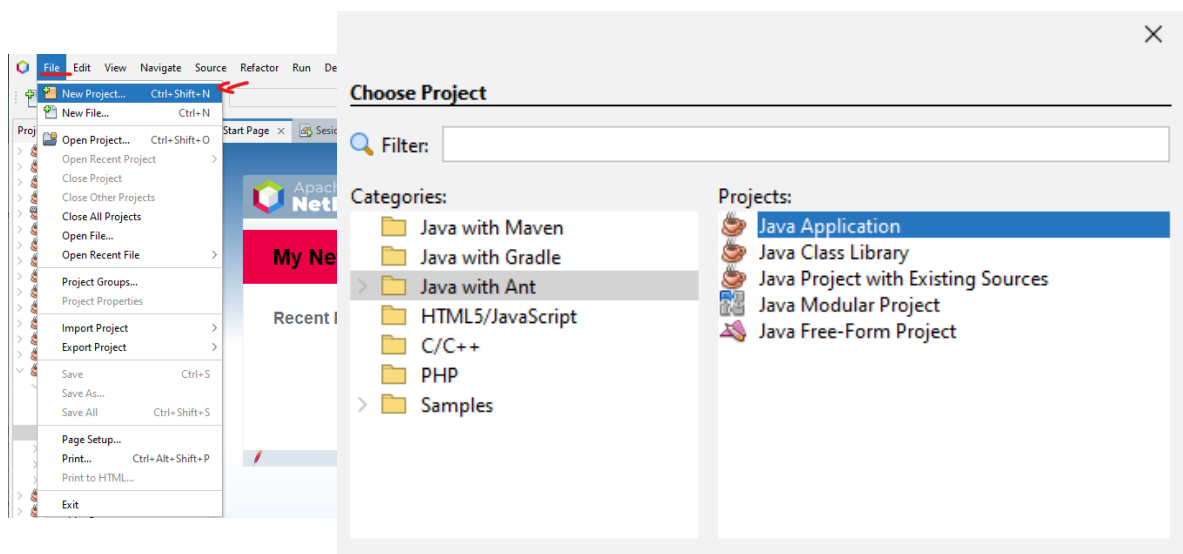
A continuación se detallan los documentos que componen la presente entrega de la tarea:

1. Informe de elaboración de la tarea.
2. Proyecto Java completo.

## 2. RA2\_b) Se han escrito programas simples.

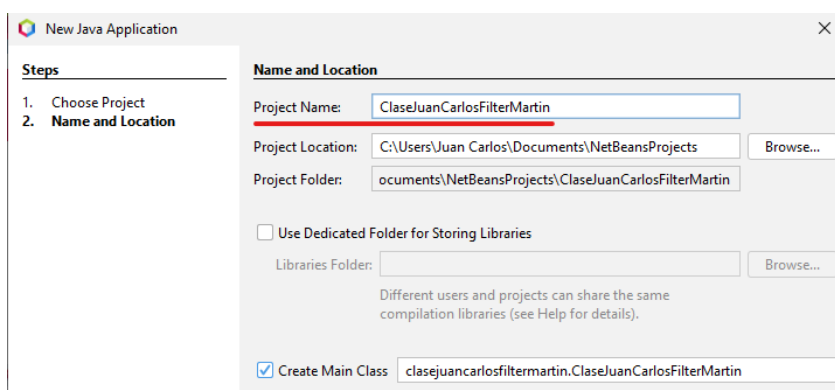
Para crear un Proyecto de Java en NetBeans nos dirigimos a **File > New Project...**

y en la siguiente ventana elegimos nuestro proyecto en nuestro caso como es Java vamos a **Java with Ant > Java Class Library**



Ahora tendremos que asignarle un nombre al proyecto que en mi caso sería **ClaseJuanCarlosFilterMartin** en el apartado **Project Name**.

También podemos cambiar la localización del proyecto, crear automáticamente la clase main, etc.



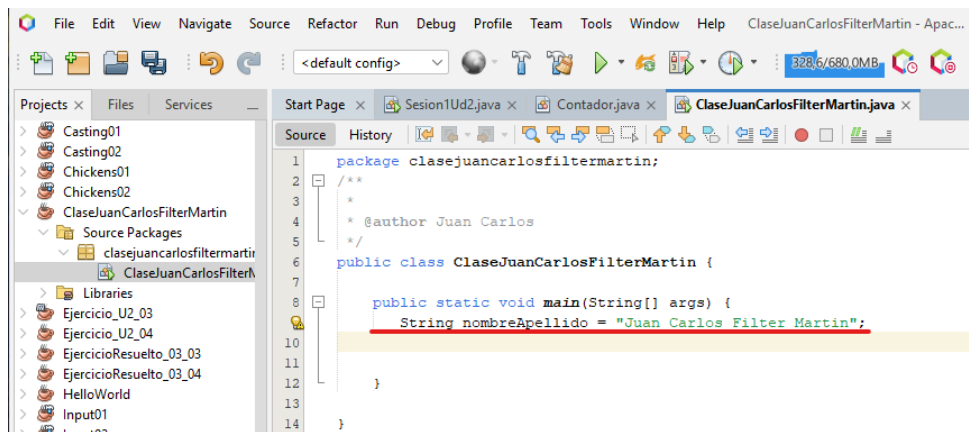
Como podemos ver con esto ya tendríamos nuestro proyecto con la clase main creada.



### 3. RA2\_c) Se han instanciado objetos a partir de clases predefinidas.

**Crear variable con nombre y apellidos.**

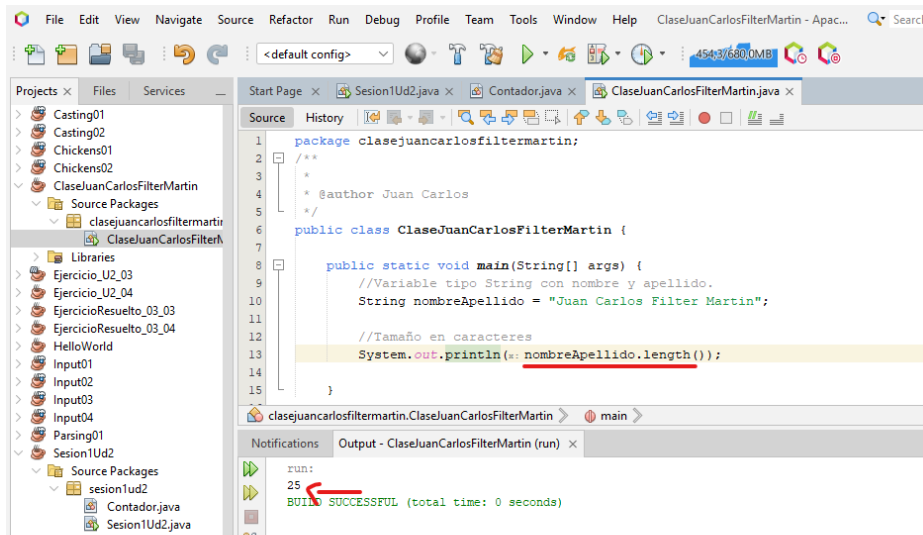
Tiene que ser una variable de tipo String ya que es una cadena de caracteres.



**Extraer y emprimir la siguiente información:**

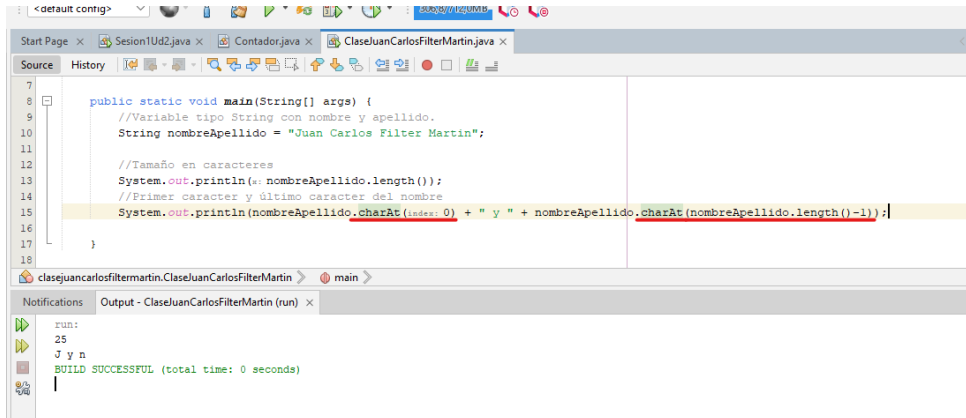
Tamaño en caracteres.

Para ello usamos el método **length** en nuestra variable que es de tipo String.



### Primer caracter y último caracter del nombre.

Usaremos el método: **charAt (0)** indicándole que empiece en el primer carácter y concatenándolo con **charAt(nombreApellido.length()-1)** para indicarle que se posicione en el último carácter restándole -1 ya que el método charAt comienza en 0 y todo esto **pasárselo al método charAt** para imprimir el ultimo carácter de la variable.



The screenshot shows an IDE with a Java file named `ClaseJuanCarlosFilterMartin.java`. The code is as follows:

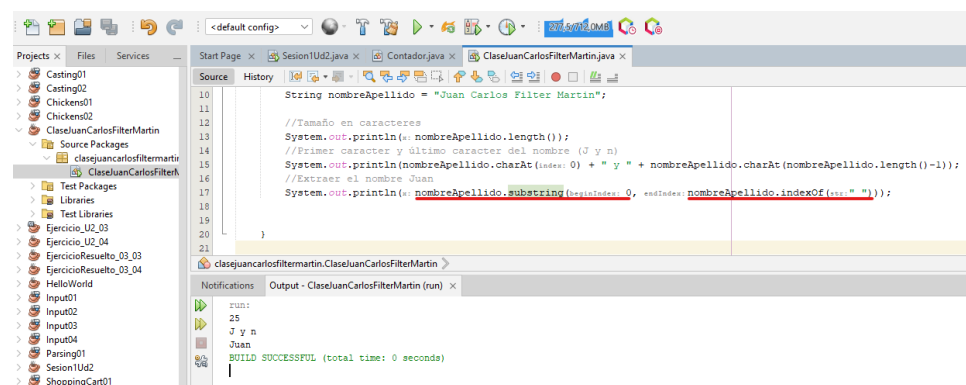
```
7
8 public static void main(String[] args) {
9     //Variable tipo String con nombre y apellido.
10    String nombreApellido = "Juan Carlos Filter Martin";
11
12    //Tamaño en caracteres
13    System.out.println(nombreApellido.length());
14    //Primer caracter y último caracter del nombre
15    System.out.println(nombreApellido.charAt(index: 0) + " y " + nombreApellido.charAt(nombreApellido.length()-1));
16
17 }
18
```

The output window shows the following results:

```
run:
25
J y n
BUILD SUCCESSFUL (total time: 0 seconds)
```

### El nombre. Si tiene varias partes solo se devuelve la primera.

Con el método **substring(0)** imprimiremos la cadena de caracteres desde la posición 0 hasta el método **indexOf(" ")** Indicándole que se detenga antes del primer espacio.



The screenshot shows the same IDE with the same Java file. The code is updated to use the `substring` method:

```
10 String nombreApellido = "Juan Carlos Filter Martin";
11
12 //Tamaño en caracteres
13 System.out.println(nombreApellido.length());
14 //Primer caracter y último caracter del nombre (J y n)
15 System.out.println(nombreApellido.charAt(index: 0) + " y " + nombreApellido.charAt(nombreApellido.length()-1));
16 //Extraer el nombre Juan
17 System.out.println(nombreApellido.substring(beginIndex: 0, endIndex: nombreApellido.indexOf(" ")));
18
19
20
21
```

The output window shows the following results:

```
run:
25
J y n
Juan
BUILD SUCCESSFUL (total time: 0 seconds)
```

### El segundo apellido. Si tiene varias partes solo mostrar la última.

En este caso **utilizamos el método substring** y le indicamos que **comience a partir del ultimo espacio** para ello voy a **utilizar lastIndexOf**. *Este método comienza desde la ultima posición hasta el carácter indicado (" ").* Entonces **ponemos el lastIndexOf dentro del substring enviándole la posición del ultimo espacio sumándole +1** para que nos imprima a partir de la primera letra del ultimo apellido

```
6 public class ClaseJuanCarlosFilterMartin {
7
8     public static void main(String[] args) {
9         //Variable tipo String con nombre y apellido
10        String nombreApellido = "Juan Carlos Filter Martin";
11
12        //Tamaño en caracteres
13        System.out.println(" nombreApellido.length()");
14        //Primer caracter y último caracter del nombre (J y n)
15        System.out.println(nombreApellido.charAt(index: 0) + " y " + nombreApellido.charAt(nombreApellido.length()-1));
16        //Extraer el nombre Juan
17        System.out.println(nombreApellido.substring(beginIndex: 0, endIndex: nombreApellido.indexOf(" ")));
18        //El segundo apellido Martin
19        System.out.println(nombreApellido.substring(nombreApellido.lastIndexOf(" ") + 1));
20    }
21 }
22
23
24
```

run:  
25  
J y n  
Juan  
Martin  
BUILD SUCCESSFUL (total time: 0 seconds)

Cambiar todas las "a" por " \_".

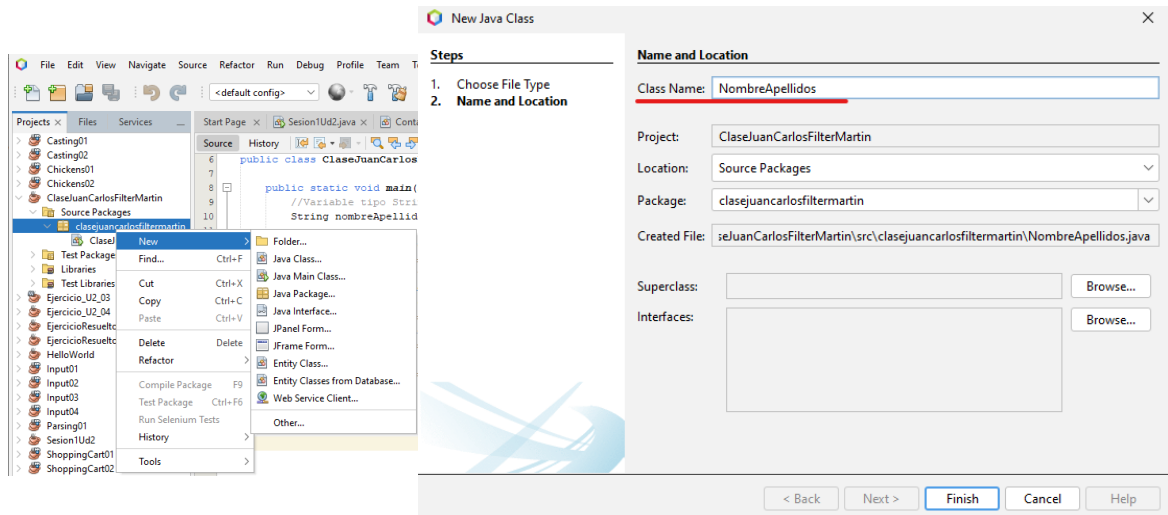
Para esto utilizamos .replace y le indicamos que "a" se remplace por " \_"  
**nombreApellido.replace("a", " \_")**

```
6 public class ClaseJuanCarlosFilterMartin {
7
8     public static void main(String[] args) {
9         //Variable tipo String con nombre y apellido
10        String nombreApellido = "Juan Carlos Filter Martin";
11
12        //Tamaño en caracteres
13        System.out.println(nombreApellido.length());
14        //Primer caracter y último caracter del nombre (J y n)
15        System.out.println(nombreApellido.charAt(index: 0) + " y " + nombreApellido.charAt(nombreApellido.length()-1));
16        //Extraer el nombre Juan
17        System.out.println(nombreApellido.substring(beginIndex: 0, endIndex: nombreApellido.indexOf(" ")));
18        //El segundo apellido Martin
19        System.out.println(nombreApellido.substring(nombreApellido.lastIndexOf(" ") + 1));
20        //Cambiar todas las "a" por " _"
21        System.out.println(nombreApellido.replace("a", " _"));
22    }
23 }
24
25
26
```

run:  
25  
J y n  
Juan  
Martin  
Ju\_n C\_los Filter M\_rtin  
BUILD SUCCESSFUL (total time: 0 seconds)

Dentro del proyecto crear una clase que se denomine, "NombreApellidos.java".

Hay varias formas de crear una clase pero por ejemplo podemos pulsar boton derecho sobre el paquete del proyecto> new > Java Class... e introducimos el nombre de la clase en Class Name.

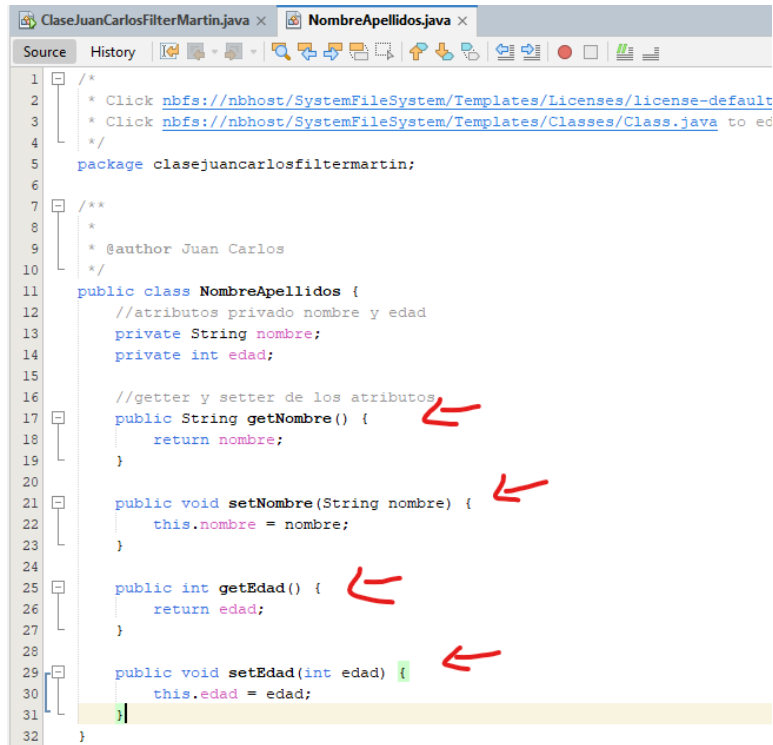


Con esto ya tendríamos nuestro proyecto con la:  
clase main **ClaseJuanCarlosFilterMartin** y la clase **NombreApellidos**

Clase con 2 atributos privados (nombre y edad)



## Construir getter y setter para los atributos de la clase

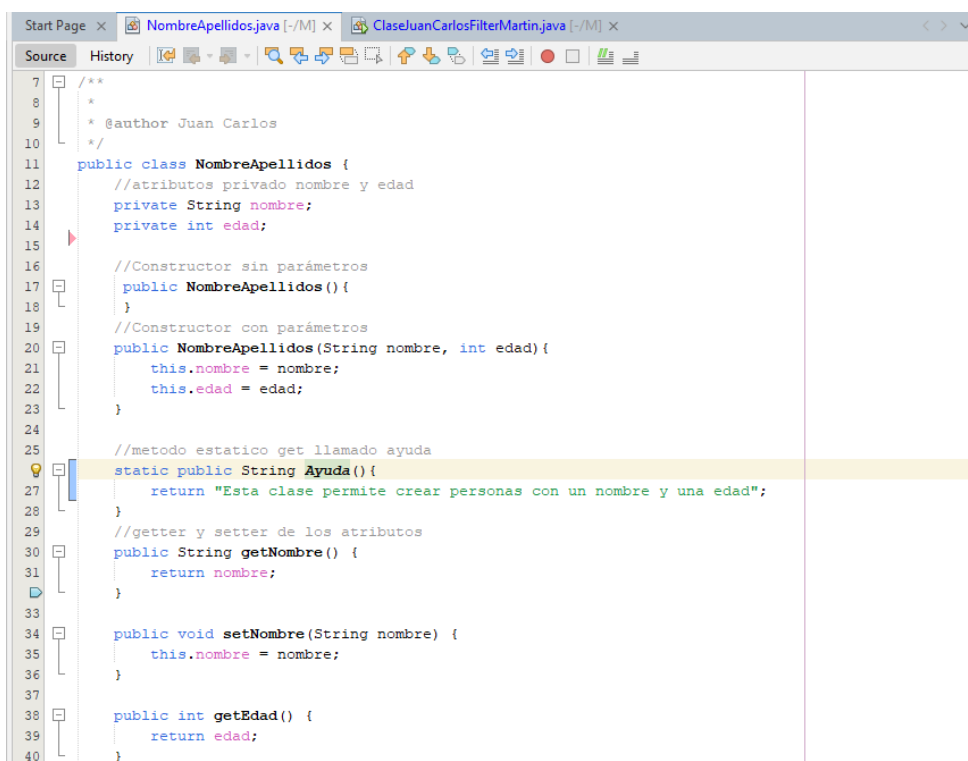


```
1  /*
2  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default
3  * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to ed
4  */
5  package clasejuancarlosfiltermartin;
6
7  /**
8   *
9   * @author Juan Carlos
10  */
11  public class NombreApellidos {
12      //atributos privado nombre y edad
13      private String nombre;
14      private int edad;
15
16      //getter y setter de los atributos
17      public String getNombre() {
18          return nombre;
19      }
20
21      public void setNombre(String nombre) {
22          this.nombre = nombre;
23      }
24
25      public int getEdad() {
26          return edad;
27      }
28
29      public void setEdad(int edad) {
30          this.edad = edad;
31      }
32  }
```

## 4. RA2\_e) Se han escrito llamadas a métodos estáticos.

Crear un método estático llamado ayuda que único que hará será mostrar un mensaje diciendo: "Esta clase permite crear personas con un nombre y una edad"

Creamos un método **static public String Ayuda()** diciéndole que devuelva:  
"Esta clase permite crear personas con un nombre y una edad"

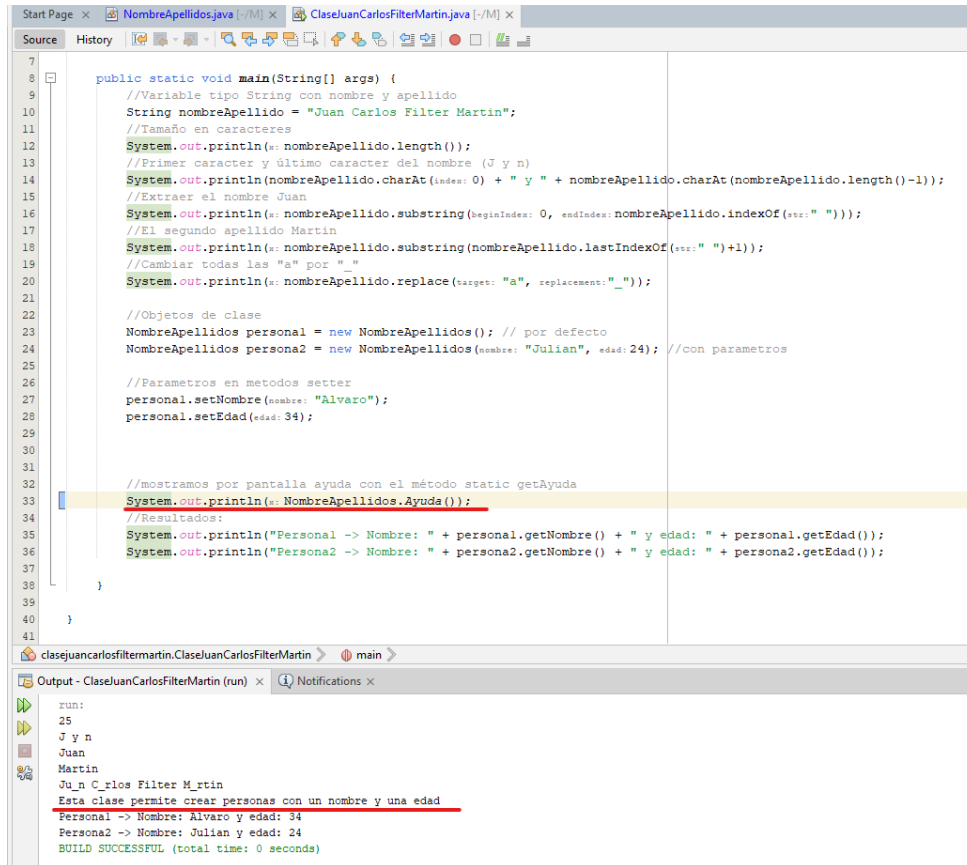


```
7  /**
8   *
9   * @author Juan Carlos
10  */
11  public class NombreApellidos {
12      //atributos privado nombre y edad
13      private String nombre;
14      private int edad;
15
16      //Constructor sin parámetros
17      public NombreApellidos() {
18      }
19      //Constructor con parámetros
20      public NombreApellidos(String nombre, int edad) {
21          this.nombre = nombre;
22          this.edad = edad;
23      }
24
25      //metodo estatico get llamado ayuda
26      static public String Ayuda() {
27          return "Esta clase permite crear personas con un nombre y una edad";
28      }
29      //getter y setter de los atributos
30      public String getNombre() {
31          return nombre;
32      }
33
34      public void setNombre(String nombre) {
35          this.nombre = nombre;
36      }
37
38      public int getEdad() {
39          return edad;
40      }
41  }
```



Nos vamos a la **clase main** y con un **System.out.println** llamando a nuestra clase **NombreApellidos** y el método donde está la variable ayuda podemos ver el resultado por pantalla:

**System.out.println(NombreApellido.getAyuda());**



```
7 public static void main(String[] args) {
8     //Variable tipo String con nombre y apellido
9     String nombreApellido = "Juan Carlos Filter Martin";
10    //Tamaño en caracteres
11    System.out.println(s: nombreApellido.length());
12    //Primer caracter y último caracter del nombre (J y n)
13    System.out.println(nombreApellido.charAt(index: 0) + " y " + nombreApellido.charAt(nombreApellido.length()-1));
14    //Extraer el nombre Juan
15    System.out.println(s: nombreApellido.substring(beginIndex: 0, endIndex: nombreApellido.indexOf(" ")));
16    //El segundo apellido Martin
17    System.out.println(s: nombreApellido.substring(nombreApellido.lastIndexOf(" ")+1));
18    //Cambiar todas las "a" por " "
19    System.out.println(s: nombreApellido.replace(target: "a", replacement: " "));
20
21    //Objetos de clase
22    NombreApellidos personal = new NombreApellidos(); // por defecto
23    NombreApellidos persona2 = new NombreApellidos(nombre: "Julian", edad: 24); //con parametros
24
25    //Parametros en metodos setter
26    personal.setNombre(nombre: "Alvaro");
27    personal.setEdad(edad: 34);
28
29
30
31
32    //mostramos por pantalla ayuda con el método static getAyuda
33    System.out.println(s: NombreApellidos.Ayuda());
34    //Resultados:
35    System.out.println("Personal -> Nombre: " + personal.getNombre() + " y edad: " + personal.getEdad());
36    System.out.println("Persona2 -> Nombre: " + persona2.getNombre() + " y edad: " + persona2.getEdad());
37
38 }
39
40
41
```

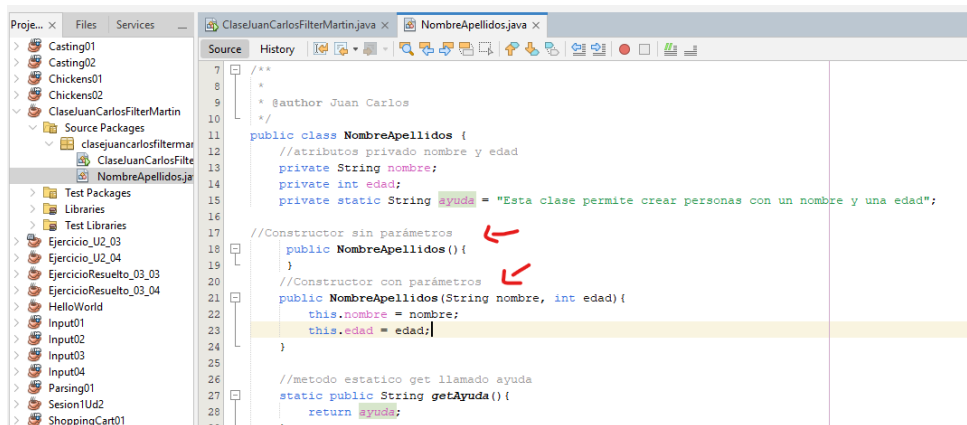
Output - ClaseJuanCarlosFilterMartin (run) x Notifications x

```
run:
25
J y n
Juan
Martin
Ju_n_Carlos Filter M_rtin
Esta clase permite crear personas con un nombre y una edad
Personal -> Nombre: Alvaro y edad: 34
Persona2 -> Nombre: Julian y edad: 24
BUILD SUCCESSFUL (total time: 0 seconds)
```

## 5. RA2\_f). Se han utilizado parámetros en la llamada a métodos.

Tanto los métodos setter como uno de los constructores, deberán llevar parámetros donde recibirán los datos de la persona que vamos a crear (nombre y edad).

**Constructores por defecto y con parámetros en la clase Nombreapellidos**



```
7 /**
8  *
9  * @author Juan Carlos
10  */
11 public class NombreApellidos {
12     //atributos privado nombre y edad
13     private String nombre;
14     private int edad;
15     private static String ayuda = "Esta clase permite crear personas con un nombre y una edad";
16
17     //Constructor sin parámetros
18     public NombreApellidos() {
19     }
20     //Constructor con parámetros
21     public NombreApellidos(String nombre, int edad) {
22         this.nombre = nombre;
23         this.edad = edad;
24     }
25
26     //metodo estatico get llamado ayuda
27     static public String getAyuda() {
28         return ayuda;
29     }
30 }
```

Ahora en la clase main creamos un objeto **persona1** con el constructor por defecto y otro objeto **persona2** con el constructor indicándoles parámetros.

Seguido de los **parámetros** asignados a los **metodos setter** al objeto **persona1**

```
21 System.out.println(s: nombreApellido.replace(target: "a", replacement: "_"));
22
23 //Objetos de clase
24 NombreApellidos personal = new NombreApellidos(); // por defecto
25 NombreApellidos persona2 = new NombreApellidos(nombre: "Julian", edad: 24); //con parametros
26
27 //Parametros en metodos setter
28 personal.setNombre(nombre: "Alvaro");
29 personal.setEdad(edad: 34);
30
31
32
```

Resultado al imprimir por pantalla:

```
15 System.out.println(nombreApellido.charAt(index: 0) + " y " + nombreApellido.charAt(nombreApellido.length()-1));
16 //Extraer el nombre Juan
17 System.out.println(s: nombreApellido.substring(beginIndex: 0, endIndex: nombreApellido.indexOf(" "));
18 //El segundo apellido Martin
19 System.out.println(s: nombreApellido.substring(nombreApellido.lastIndexOf(" ") + 1));
20 //Cambiar todas las "a" por "_"
21 System.out.println(s: nombreApellido.replace(target: "a", replacement: "_"));
22
23 //Objetos de clase
24 NombreApellidos personal = new NombreApellidos(); // por defecto
25 NombreApellidos persona2 = new NombreApellidos(nombre: "Julian", edad: 24); //con parametros
26
27 //Parametros en metodos setter
28 personal.setNombre(nombre: "Alvaro");
29 personal.setEdad(edad: 34);
30
31
32 //mostramos por pantalla ayuda con el método static getAyuda
33 System.out.println(s: personal.getAyuda());
34 System.out.println("Personal -> Nombre: " + personal.getNombre() + " y edad: " + personal.getEdad());
35 System.out.println("Persona2 -> Nombre: " + persona2.getNombre() + " y edad: " + persona2.getEdad());
36
37 }
38
39
40
41
```

Output - ClaseJuanCarlosFilterMartin (run)

```
25
J y n
Juan
Martin
Juan Carlos Filter Martin
Esta clase permite crear personas con un nombre y una edad
Personal -> Nombre: Alvaro y edad: 34
Persona2 -> Nombre: Julian y edad: 24
BUILD SUCCESSFUL (total time: 0 seconds)
```

## 6. RA2\_g) Se han incorporado y utilizado librerías de objetos.

Al estar usando la clase String no hace falta importarla ya que se importa de forma predeterminada en todos los programas java. De todas forma se importaría de esta forma:

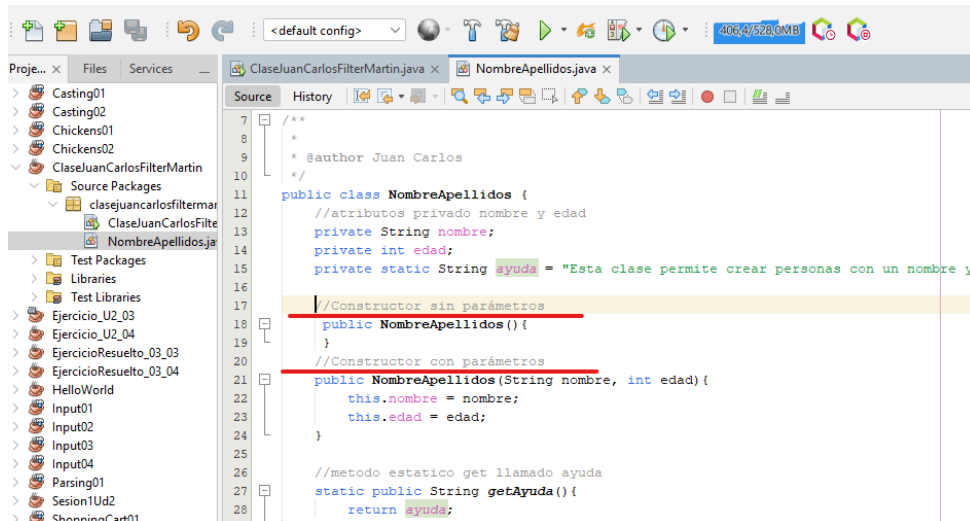
```
1 package clasejuancarlosfiltermartin;
2 import java.lang.String;
3
4 /**
5  * @author Juan Carlos
6  */
7 public class ClaseJuanCarlosFilterMartin {
```

## 7. RA2\_h) Se han utilizado constructores.

Dos constructores:

1º sin parámetros.

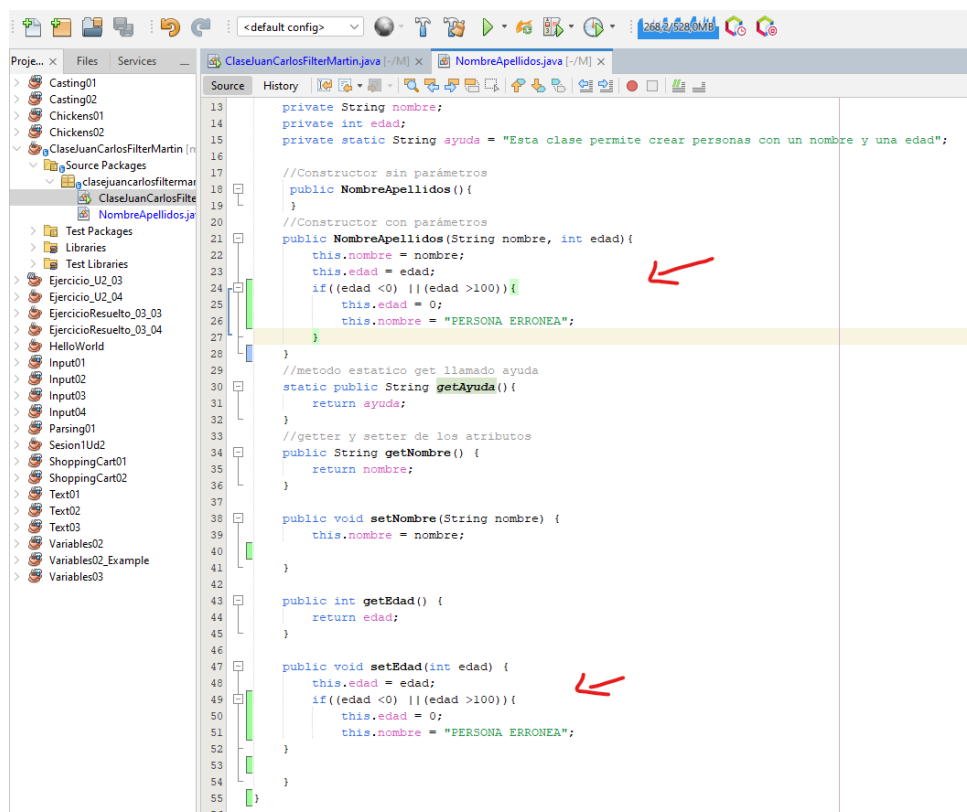
2º constructor recibirá como parámetros nombre y edad que usará para inicializar los atributos.



```
7  /**
8   *
9   * @author Juan Carlos
10  */
11  public class NombreApellidos {
12      //atributos privado nombre y edad
13      private String nombre;
14      private int edad;
15      private static String ayuda = "Esta clase permite crear personas con un nombre y
16
17      //Constructor sin parámetros
18      public NombreApellidos() {
19      }
20      //Constructor con parámetros
21      public NombreApellidos(String nombre, int edad) {
22          this.nombre = nombre;
23          this.edad = edad;
24      }
25
26      //metodo estatico get llamado ayuda
27      static public String getAyuda() {
28          return ayuda;
29      }
30  }
```

Tanto en las funciones set como en los constructores se comprobará que la edad sea correcta (0-100 años). Si intentamos introducir una edad incorrecta, se pondrá a 0 y el nombre se pondrá a **"PERSONA ERRONEA"**.

Para ello vamos a indicar tanto en el constructor como en el set edad con un if que si edad es menor a 0 o mayor a 100 cambie la variable edad = 0 y la variable nombre = "PERSONA ERRONEA"



```
13  private String nombre;
14  private int edad;
15  private static String ayuda = "Esta clase permite crear personas con un nombre y una edad";
16
17  //Constructor sin parámetros
18  public NombreApellidos() {
19  }
20  //Constructor con parámetros
21  public NombreApellidos(String nombre, int edad) {
22      this.nombre = nombre;
23      this.edad = edad;
24      if ((edad < 0) || (edad > 100)) {
25          this.edad = 0;
26          this.nombre = "PERSONA ERRONEA";
27      }
28  }
29
30  //metodo estatico get llamado ayuda
31  static public String getAyuda() {
32      return ayuda;
33  }
34
35  //getter y setter de los atributos
36  public String getNombre() {
37      return nombre;
38  }
39
40  public void setNombre(String nombre) {
41      this.nombre = nombre;
42  }
43
44  public int getEdad() {
45      return edad;
46  }
47
48  public void setEdad(int edad) {
49      this.edad = edad;
50      if ((edad < 0) || (edad > 100)) {
51          this.edad = 0;
52          this.nombre = "PERSONA ERRONEA";
53      }
54  }
55  }
```

Podemos ver los resultados si vamos a nuestra clase main e introducimos un campo que salga del rango de edad entre 0 y 100:

```
20      System.out.println(n: nombreApellido.replace(target: "a", replacement: "_"));
21
22      //Objetos de clase
23      NombreApellidos personal = new NombreApellidos(); // por defecto
24      NombreApellidos persona2 = new NombreApellidos(nombre: "Julian", edad: -5); //con parametros
25
26      //Parametros en metodos setter
27      personal.setNombre(nombre: "Alvaro");
28      personal.setEdad(edad: 101);
29
30
31
32      //mostramos por pantalla ayuda con el método static getAyuda
33      System.out.println(n: personal.getAyuda());
34      //Resultados:
35      System.out.println("Personal -> Nombre: " + personal.getNombre() + " edad: " + personal.getEdad());
36      System.out.println("Persona2 -> Nombre: " + persona2.getNombre() + " edad: " + persona2.getEdad());
37
38  }
39
40  }
41
```

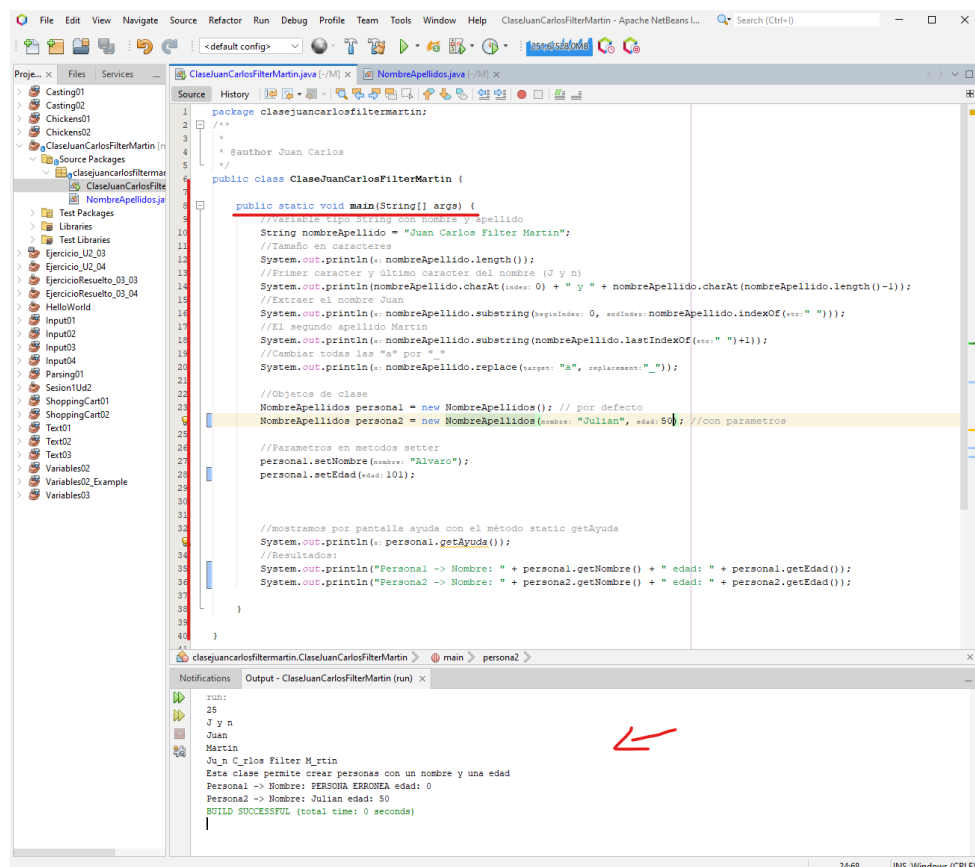
clasejuncarlosfiltermartin.ClaseJuanCarlosFilterMartin > main >

Notifications Output - ClaseJuanCarlosFilterMartin (run) x

```
25
J y n
Juan
Martin
Ju_n_C_rios Filter M_rtin
Esta clase permite crear personas con un nombre y una edad
Personal -> Nombre: PERSONA ERRONEA edad: 0
Persona2 -> Nombre: PERSONA ERRONEA edad: 0
BUILD SUCCESSFUL (total time: 0 seconds)
```

## 8. RA2\_i) Se ha utilizado el entorno integrado de desarrollo en la creación y compilación de programas simples.

Todas las pruebas, tanto las referentes a la clase String como a la clase que vamos a crear nosotros se harán desde la función main.



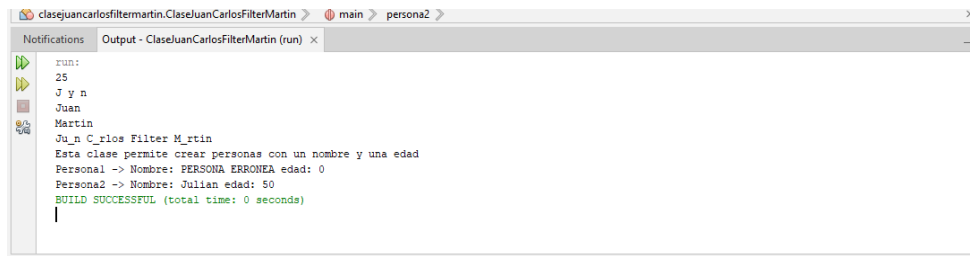
```
1 package clasejuncarlosfiltermartin;
2
3 /**
4  *
5  * @author Juan Carlos
6  */
7 public class ClaseJuanCarlosFilterMartin {
8
9     public static void main(String[] args) {
10         //Variable tipo String con nombre y apellido
11         String nombreApellido = "Juan Carlos Filter Martin";
12         //Tamaño en caracteres
13         System.out.println(n: nombreApellido.length());
14         //Primer caracter y último caracter del nombre (J y n)
15         System.out.println(nombreApellido.charAt(index: 0) + " y " + nombreApellido.charAt(nombreApellido.length()-1));
16         //Extraer el nombre Juan
17         System.out.println(n: nombreApellido.substring(beginIndex: 0, endIndex: nombreApellido.indexOf(" ")));
18         //El segundo apellido Martin
19         System.out.println(n: nombreApellido.substring(nombreApellido.lastIndexOf(" ") + 1));
20         //Cambiar todas las "a" por "_"
21         System.out.println(n: nombreApellido.replace(target: "a", replacement: "_"));
22
23         //Objetos de clase
24         NombreApellidos personal = new NombreApellidos(); // por defecto
25         NombreApellidos persona2 = new NombreApellidos(nombre: "Julian", edad: 50); //con parametros
26
27         //Parametros en metodos setter
28         personal.setNombre(nombre: "Alvaro");
29         personal.setEdad(edad: 101);
30
31
32         //mostramos por pantalla ayuda con el método static getAyuda
33         System.out.println(n: personal.getAyuda());
34         //Resultados:
35         System.out.println("Personal -> Nombre: " + personal.getNombre() + " edad: " + personal.getEdad());
36         System.out.println("Persona2 -> Nombre: " + persona2.getNombre() + " edad: " + persona2.getEdad());
37
38     }
39
40 }
41
```

clasejuncarlosfiltermartin.ClaseJuanCarlosFilterMartin > main > persona2 >

Notifications Output - ClaseJuanCarlosFilterMartin (run) x

```
Run:
25
J y n
Juan
Martin
Ju_n_C_rios Filter M_rtin
Esta clase permite crear personas con un nombre y una edad
Personal -> Nombre: PERSONA ERRONEA edad: 0
Persona2 -> Nombre: Julian edad: 50
BUILD SUCCESSFUL (total time: 0 seconds)
```

Al ejecutar deberemos de obtener unos mensajes que debemos de capturar y mostrar en el informe como prueba de que la aplicación funciona.



```
run:
25
J y n
Juan
Martin
Ju_n C_rios Filter M_rtin
Esta clase permite crear personas con un nombre y una edad
Personal -> Nombre: PERSONA ERRONEA edad: 0
Persona2 -> Nombre: Julian edad: 50
BUILD SUCCESSFUL (total time: 0 seconds)
```

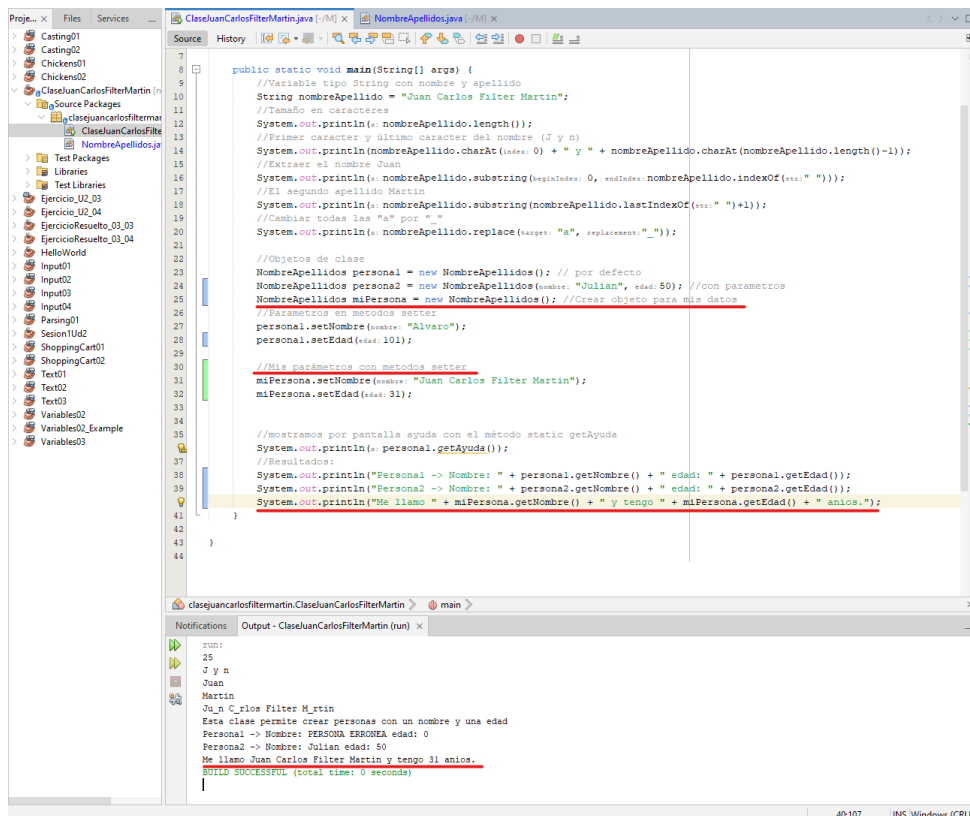
Para probar nuestra clase, vamos a crear un objeto con los valores: vuestro nombre y vuestra edad

Para ello:

1º Instanciamos un objeto con nombre miPersona de la clase NombreApellidos

2º Le asignamos los parámetros del nombre y la edad mediante los metodos set “Juan Carlos Filter Martin” y edad 31.

3º Escribimos un System.out.print con los el objeto miPersona.getNombre y getEdad



```
public static void main(String[] args) {
    //Variable tipo String con nombre y apellido
    String nombreApellido = "Juan Carlos Filter Martin";
    //Tamaño en caracteres
    System.out.println(nombreApellido.length());
    //Primer caracter y último caracter del nombre (J y n)
    System.out.println(nombreApellido.charAt(0) + " y " + nombreApellido.charAt(nombreApellido.length()-1));
    //Extraer el nombre Juan
    System.out.println(nombreApellido.substring(0, nombreApellido.indexOf(" ")));
    //2º segundo apellido Martin
    System.out.println(nombreApellido.substring(nombreApellido.lastIndexOf(" ") + 1));
    //Cambiar todas las "a" por "e"
    System.out.println(nombreApellido.replace("a", "e"));

    //Objetos de clase
    NombreApellidos personal = new NombreApellidos(); // por defecto
    NombreApellidos persona2 = new NombreApellidos(nombre, edad: 50); //con parametros
    NombreApellidos miPersona = new NombreApellidos(); //Crear objeto para mis datos

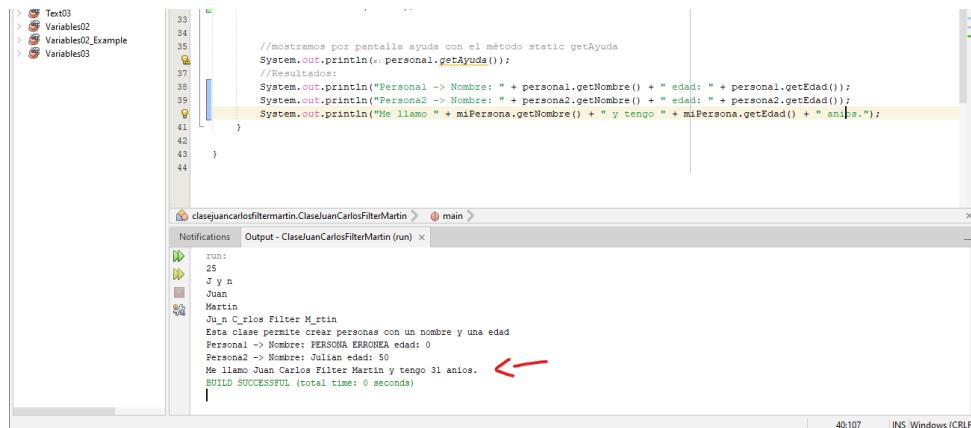
    //Parametros en metodos setter
    personal.setNombre(nombre: "Alvaro");
    personal.setEdad(edad: 101);

    //Mis parámetros con metodos setter
    miPersona.setNombre(nombre: "Juan Carlos Filter Martin");
    miPersona.setEdad(edad: 31);

    //mostramos por pantalla ayuda con el método static getAyuda
    System.out.println(personal.getAyuda());
    //Resultados
    System.out.println("Personal -> Nombre: " + personal.getNombre() + " edad: " + personal.getEdad());
    System.out.println("Persona2 -> Nombre: " + persona2.getNombre() + " edad: " + persona2.getEdad());
    System.out.println("Me llamo " + miPersona.getNombre() + " y tengo " + miPersona.getEdad() + " años.");
}
```

```
run:
25
J y n
Juan
Martin
Ju_n C_rios Filter M_rtin
Esta clase permite crear personas con un nombre y una edad
Personal -> Nombre: PERSONA ERRONEA edad: 0
Persona2 -> Nombre: Julian edad: 50
Me llamo Juan Carlos Filter Martin y tengo 31 años.
BUILD SUCCESSFUL (total time: 0 seconds)
```

Mostrar por pantalla el nombre y edad del objeto creado.



```
33 //mostramos por pantalla ayuda con el método static getAyuda
34 System.out.println(("- personal.getAyuda()));
35 //Resultados:
36 System.out.println("Personal -> Nombre: " + personal.getNombre() + " edad: " + personal.getEdad());
37 System.out.println("Persona2 -> Nombre: " + persona2.getNombre() + " edad: " + persona2.getEdad());
38 System.out.println("Me llamo " + miPersona.getNombre() + " y tengo " + miPersona.getEdad() + " años.");
39 }
40 }
41 }
42 }
43 }
44 }
```

claseJuanCarlosFilterMartin.ClaseJuanCarlosFilterMartin (run) x

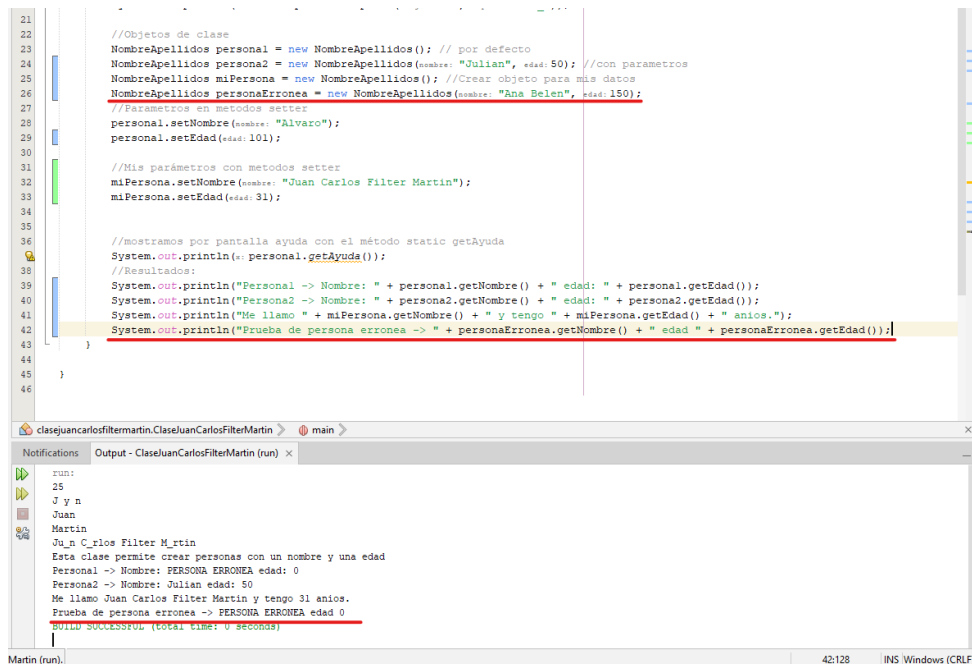
Notifications Output - ClaseJuanCarlosFilterMartin (run) x

run: 25  
J y n  
Juan  
Martin  
Ju\_n\_C\_rios Filter\_M\_rtin  
Esta clase permite crear personas con un nombre y una edad  
Personal -> Nombre: PERSONA ERRONEA edad: 0  
Persona2 -> Nombre: Julian edad: 50  
Me llamo Juan Carlos Filter Martin y tengo 31 años. ←  
BUILD SUCCESSFUL (total time: 0 seconds)

Crear un objeto con datos erróneos (edad fuera del rango permitido), para ver qué mensaje aparece.

Creamos un objeto en el que vamos a indicarle una edad mayor a 100 en el propio constructor

Usamos el System.out.println y podemos en el resultado que nos ha cambiado el nombre a PERSONA ERRONEA y con edad de 0.



```
21 //Objetos de clase
22 NombreApellidos personal = new NombreApellidos(); // por defecto
23 NombreApellidos persona2 = new NombreApellidos(nombre: "Julian", edad: 50); //con parametros
24 NombreApellidos miPersona = new NombreApellidos(); //Crear objeto para mis datos
25 NombreApellidos personaErronea = new NombreApellidos(nombre: "Ana Belen", edad: 150);
26 //Parametros en metodos setter
27 personal.setNombre(nombre: "Alvaro");
28 personal.setEdad(edad: 101);
29 //Mis parametros con metodos setter
30 miPersona.setNombre(nombre: "Juan Carlos Filter Martin");
31 miPersona.setEdad(edad: 31);
32 //mostramos por pantalla ayuda con el método static getAyuda
33 System.out.println(("- personal.getAyuda()));
34 //Resultados:
35 System.out.println("Personal -> Nombre: " + personal.getNombre() + " edad: " + personal.getEdad());
36 System.out.println("Persona2 -> Nombre: " + persona2.getNombre() + " edad: " + persona2.getEdad());
37 System.out.println("Me llamo " + miPersona.getNombre() + " y tengo " + miPersona.getEdad() + " años.");
38 System.out.println("Prueba de persona erronea -> " + personaErronea.getNombre() + " edad " + personaErronea.getEdad());
39 }
40 }
41 }
42 }
43 }
44 }
```

claseJuanCarlosFilterMartin.ClaseJuanCarlosFilterMartin (run) x

Notifications Output - ClaseJuanCarlosFilterMartin (run) x

run: 25  
J y n  
Juan  
Martin  
Ju\_n\_C\_rios Filter\_M\_rtin  
Esta clase permite crear personas con un nombre y una edad  
Personal -> Nombre: PERSONA ERRONEA edad: 0  
Persona2 -> Nombre: Julian edad: 50  
Me llamo Juan Carlos Filter Martin y tengo 31 años.  
Prueba de persona erronea -> PERSONA ERRONEA edad 0  
BUILD SUCCESSFUL (total time: 0 seconds)