FPII 2021/2022 - Laboratorio Semana 1 (Lab0):

PRIMEROS PROGRAMAS en Iava

Contenido

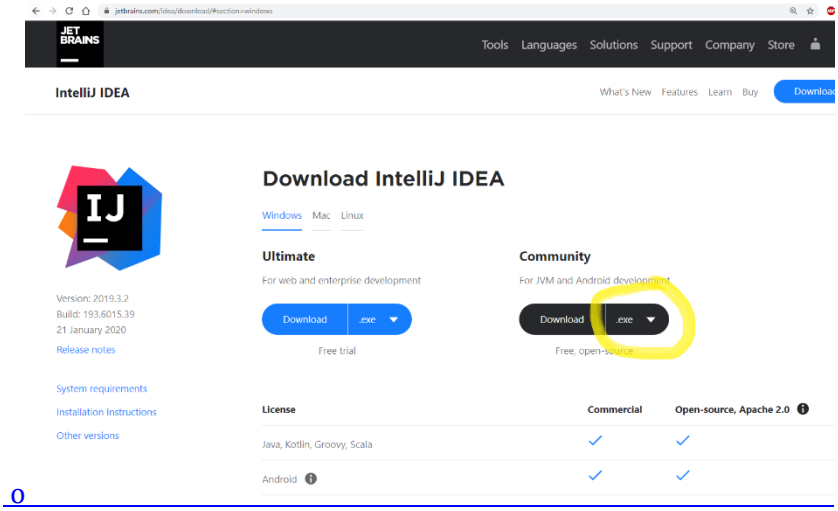
1. Entorno de trabajo

En esta asignatura se trabajará con el IDE de programación IntelliJ IDEA. Es un entorno muy completo que nos permite crear proyectos en Java, con sistemas de carpetas (paquetes) y depurar programas. Sin embargo, en esta primera clase de laboratorio intentaremos simplificar su uso al máximo para centrarnos en lo que es escribir código Java.

Si no está instalado IntelliJ donde estás trabajando, debes instalarlo siguiendo el apartado 1 A) (*Instalar IntelliJ IDEA*). Si ya lo tienes instalado, salta al apartado 2 (*Traducción de un pequeño programa en C)*.

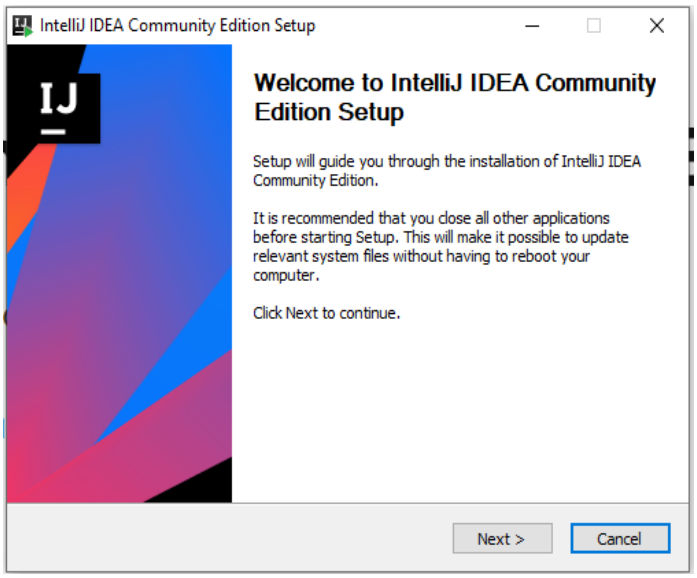
Instalar IntelliJ IDEA

Te lo puedes descargar desde https://www.jetbrains.com/idea/download/



Usamos la versión *Community*,

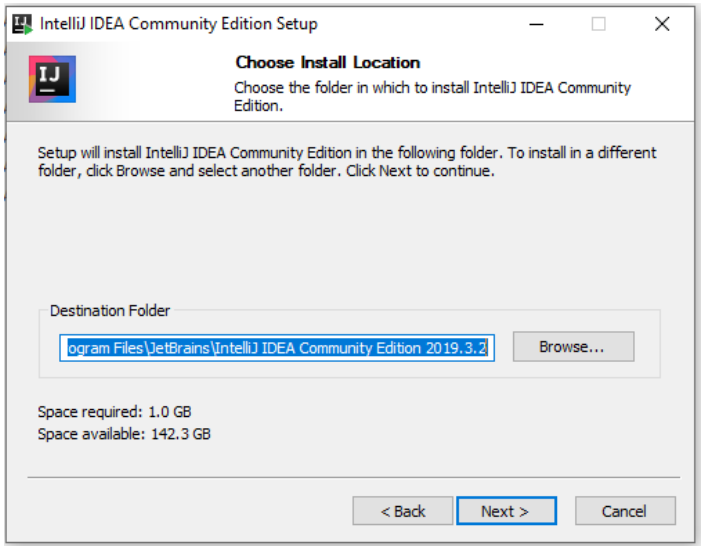
(por ejem. archivo *ideaIC-2020.3.1*)

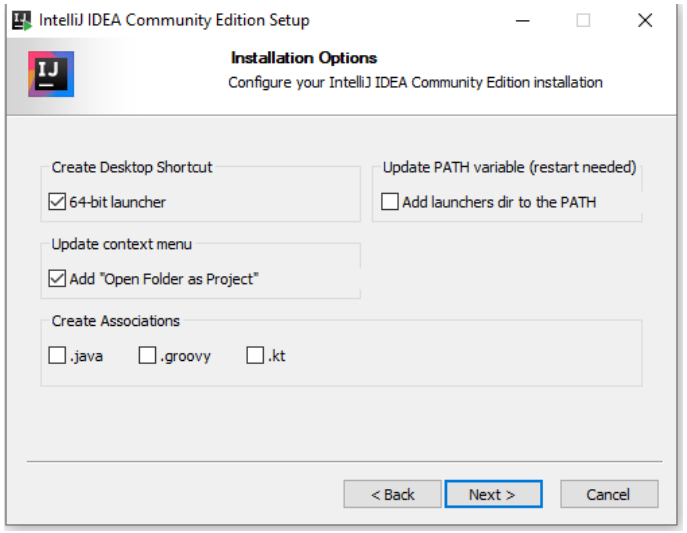


Al ejecutarlo, nos aparecerá la ventana de bienvenida.

Pulsamos *Next*

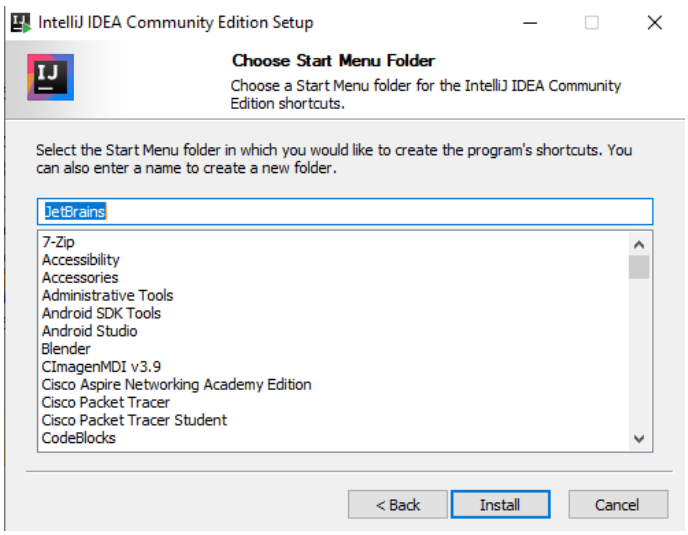
Pulsamos *Next*



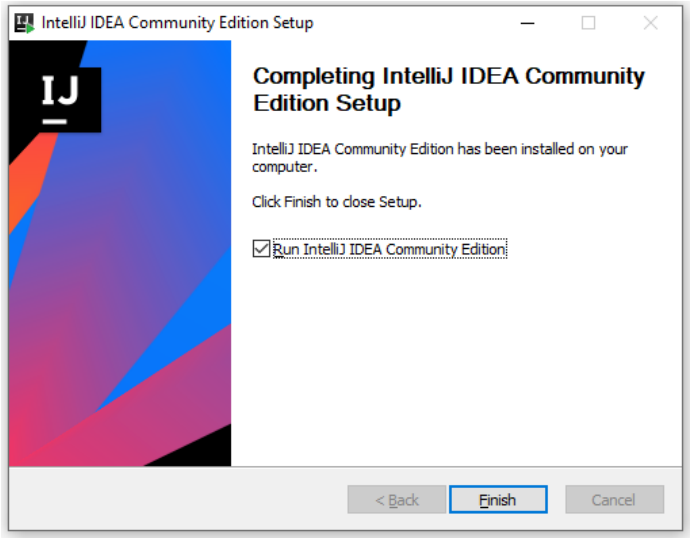


Podemos marcar esas dos opciones.

Pulsamos *Next*

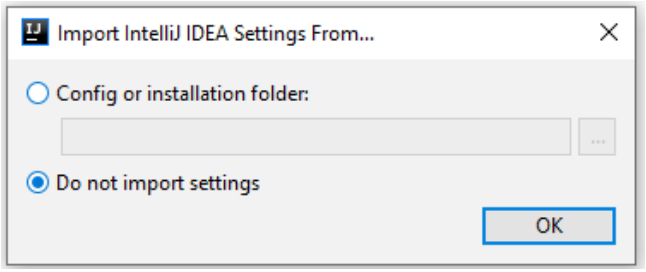


Pulsamos *Install*



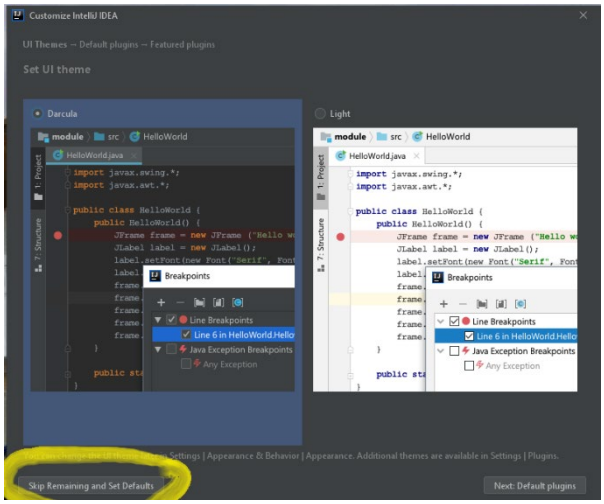
Pulsamos *Finish*

Ya hemos terminado la instalación, y al ejecutar *IntelliJ Idea*, nos pedirá datos para la configuración:



No importamos ningún ajuste

Pulsamos *OK*



Aquí elegimos el tema de la interface (fondo negro o blanco).

Pulsamos *Skip Remaining and set Defaults*, para poner por defecto el resto de los valores de configuración.

2. Traducción de un pequeño programa de C (FP1) a Java

2 A) Suma de los n primeros números

Un ejercicio que podéis haber hecho en C, en la asignatura de FP I, podría ser el que pide implementar una función que, dado un número *n*, retorne:

$$f(n) = \sum_{i=0}^{n} i$$

- Solución en C:

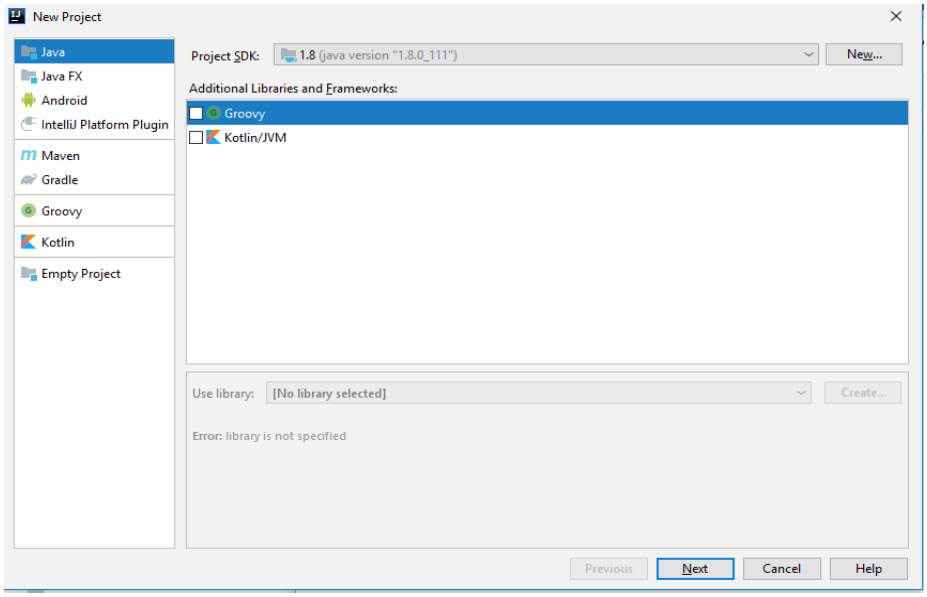
```
int sum1(int n){
    int sum = 0, i;

    for (i=0;i<=n;i++)
       sum = sum + i;

    return sum;
}
```

- Vamos a realizar lo mismo en Java:

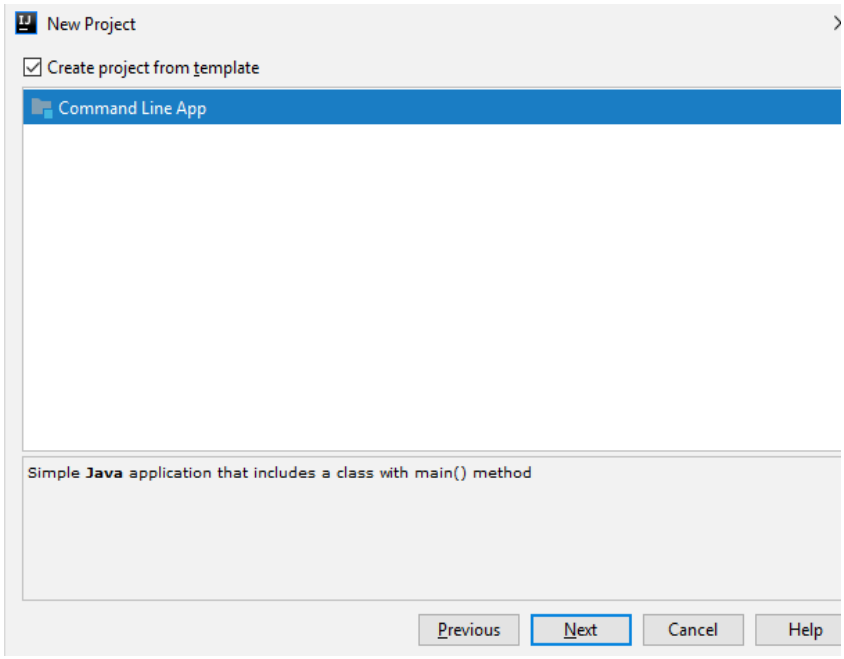
  Lo primero que necesitamos es el *esqueleto* de una clase que nos sirva de programa. Usando IntelliJ IDEA, crearemos un proyecto donde trabajar. Elegimos *Create New Project*



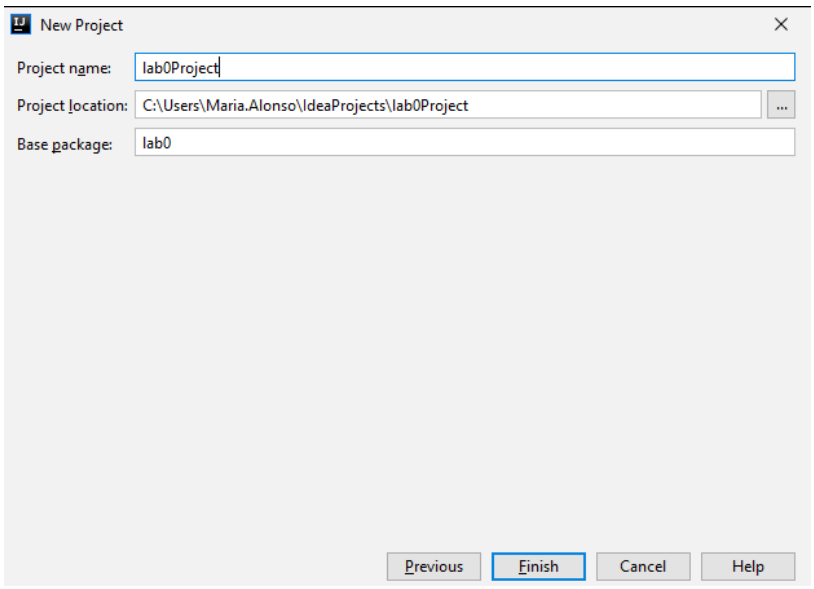
Marcamos Java, y como estamos creando un proyecto simple, no necesitamos complementos o extras para este proyecto.

No clicamos ni Groovy no Kotlin/JVM.

Pulsamos **Next**.



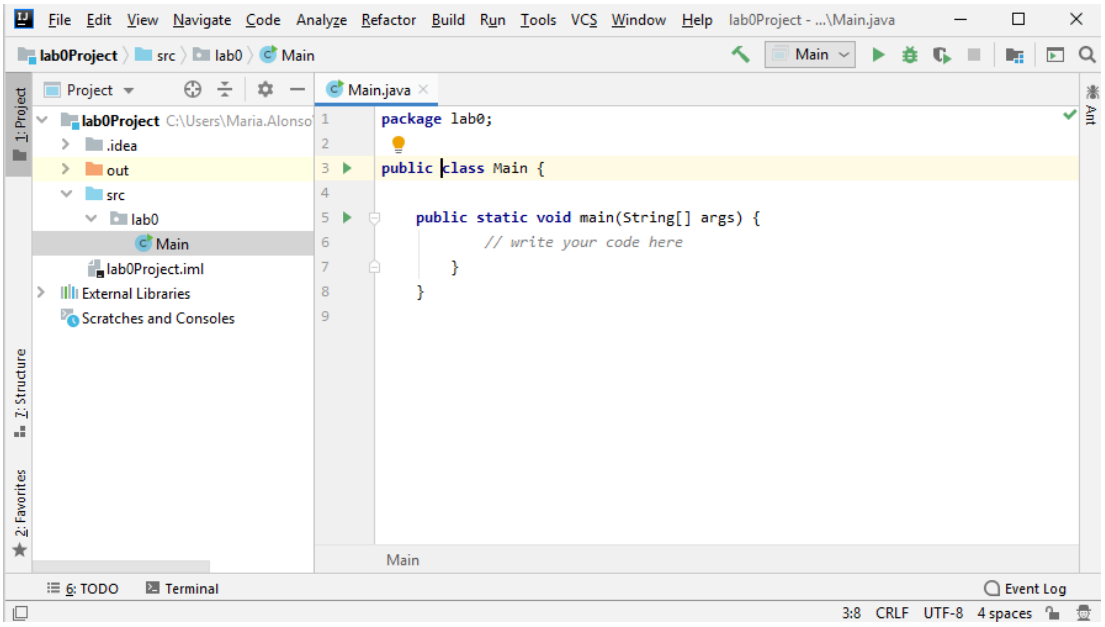
Usamos la plantilla para una aplicación sencilla (*Command Line App*). Pulsamos Next.



Ahora le daremos el nombre al proyecto a crear, la ubicación y el paquete donde se encontrará.

- El proyecto puede llamarse *Lab0Project.*
- Ubicarlo en una carpeta, teniendo en cuenta que en el laboratorio trabajan muchos compañeros de todos los grupos en el mismo equipo, y los nombres que usamos son iguales. Al terminar, debéis guardaros para vosotros vuestro trabajo y eliminarlo del ordenador del laboratorio.
- El paquete (contenedor de clases, para organizar el código) se llamará ***lab0***.

A continuación, pulsamos el botón Next, y tendremos una ventana como:

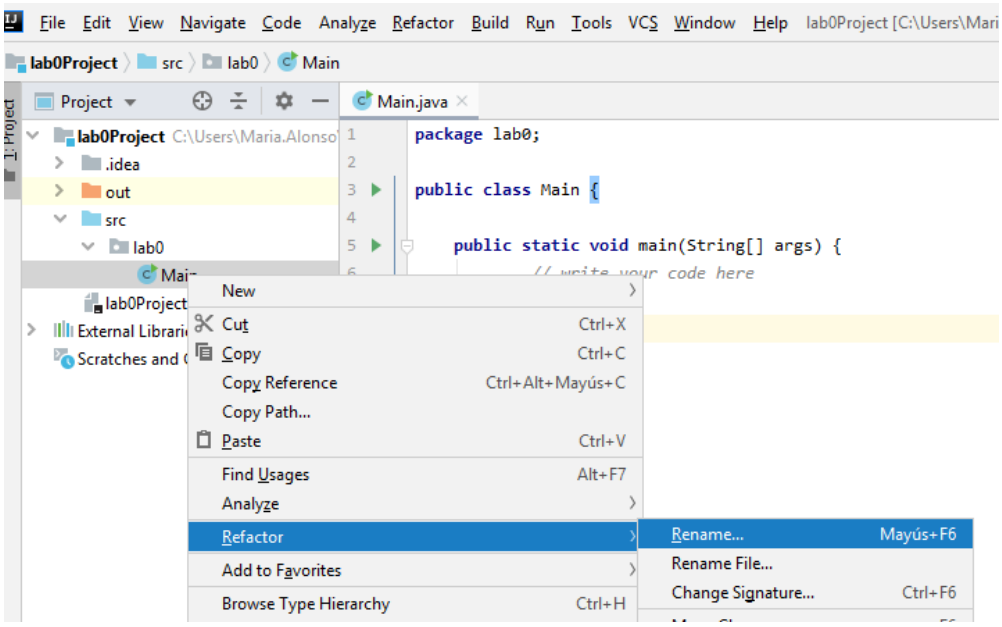


En la pestaña de proyectos, a la izquierda, tendremos en `Lab0Project`, la carpeta `src` que contiene la carpeta `lab0` que es el paquete con los fuentes .java que creemos dentro del paquete.

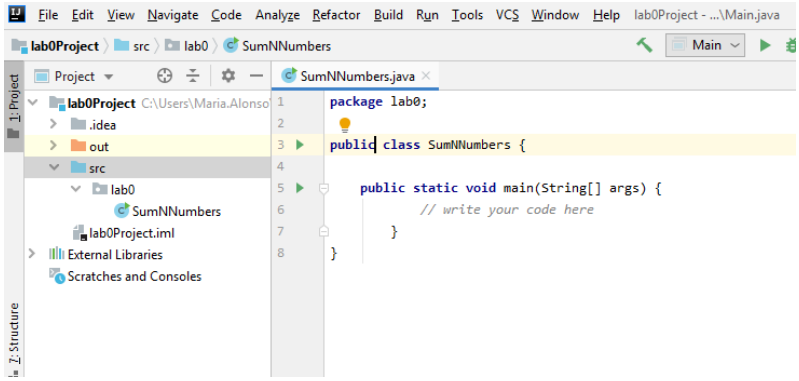
En principio, nos ha creado en el archivo Main.java una clase llamada `Main`, que contiene un método llamado `main`.

Vamos a cambiarle el nombre a la clase. En vez de `Main`, se va a llamar `SumNNumbers`. (Daos cuenta de que el nombre del archivo .java

también debe cambiar). Para ello en la ventana `Project` con el botón derecho sobre la clase `Main`, elegimos `Refactor, Rename` y le damos el nuevo nombre .



Y tu código quedará ahora así:



Dentro del método *main* irá lo que se ejecute en el programa principal, de momento no nos interesa, pues sólo queremos hacer la función. La vamos a escribir justo antes de la función main. Probemos a escribir lo que teníamos en C:

```
public class SumNNumbers {

   int sum1(int n) {
      int sum = 0, i;

      for (i = 0; i <= n; i++) {
         sum = sum + i;
      }

      return sum;
   }

   public static void main(String[] args) {
      // Escribe aquí el código
   }
}
```

Es importante que veas tu código bien indentado: Desde el menú *Code*, pinchamos *Reformat Code* (`Ctrl+Alt+I`). También podemos hacer *Auto-Indent Lines* (`Ctrl+Alt+I`).

Y ahora intenta compilar y ejecutar. Podemos compilar sólo (desde el menú *Build* → *Recompile*) y después ejecutar, o bien ejecutar directamente y en ese caso compilará antes (desde el menú *Run*).

Observa qué sucede. ¿Hay errores?

Una vez que consigamos que el código compile, intentaremos hacer una llamada a la función en el *main*, para comprobar que funciona, por ejemplo, añade en el main:

```
int salida = sum1(11);
```

Para ello tendremos que seguir estos dos sencillos pasos:

- añadir la palabra `static` al principio de la declaración la función (ya veremos qué significa más adelante)
- ejecutar el fichero (parecido a antes, botón derecho y '*Run*', o también puedes ejecutar pulsando la flecha verde desde el editor).

Observa qué sucede: el programa se ejecuta, pero no "hace nada". Era de esperar, ya que para poder ver que el código funciona, tendremos que mostrar el resultado, y para ello es necesario mostrar el resultado de la llamada a sum1. Por ejemplo, dentro del método main, escribe las siguientes dos líneas:

```
int salida = sum1(11);
System.out.println("La suma de 1 a 11 es " + salida);
```

Nota: `System.out.println` es el *equivalente* a printf de C pero en general su uso es más sencillo.

Observa qué sucede ahora. ¿Qué valor obtienes? ¿Es correcto?

Para hacer el programa más elegante (y más genérico), usaremos una variable también a la hora de llamar al método. Por ello, sustituye el código anterior por esta versión *mejor escrita*:

```
int entrada = 11;
int salida = sum1 (entrada);
System.out.println("La suma de 1 a " + entrada + " es " + salida);
```

Comprobemos que la salida sigue siendo la misma. Puedes hacer pequeñas pruebas cambiando el número, por ejemplo.

2 B) Implementación de otra función

Repitamos el ejercicio anterior añadiendo en la misma clase (`SumNNumbers`) la función `sum2` (también `static`) que usa esta fórmula.

$$S = \frac{n\,(n+1)}{2}$$

Nota: El cuerpo de esta función se puede escribir con sólo una línea de código, o dos si queremos primero calcular el valor y guardarlo en una variable, y después retornarla.

Su resultado debería ser equivalente a la función que ya creamos.

Para hacer dicha comprobación, crea el código en el main que calcule sum1 y sum2 con varias entradas, y comprueba que las dos salidas son iguales para cada una de las entradas.

Como sugerencia, al final, podríais usar este código que se os copia abajo. Es muy importante que lo entendáis todo:

```
int[] array = {3,4,13,21,67,102,155,365,1007};
for(int i=0;i<array.length;i++) {
    System.out.println("sum1 para " + array[i] + " = " + sum1(array[i]));
    System.out.println("sum2 para " + array[i] + " = " + sum2(array[i]));
    System.out.println("-----------------");
}
```

¿Se te ocurre alguna manera de automatizar el hecho de comprobar que para cada entrada dada la salida de las dos sumas es la misma? En otras palabras, se trataría de que, en lugar de andar examinando el output por pantalla y comprobando 'a mano', que es el mismo, por ejemplo, para el 365...

```
-----------------
sum1 para 365 = 66795
sum2 para 365 = 66795
-----------------
```

... pudiéramos obtener una respuesta del programa de tipo sí/no (true/false) sobre si devuelven lo mismo. Piensa cómo se podría conseguir.

3. Depurar programas con IntelliJ IDEA.

En la sesión de prácticas el profesor te explicará cómo funciona el depurador en IntelliJ, herramienta que te será muy útil cuando tengas errores de ejecución en tus programas.

IMPORTANTE: Los ejercicios que hemos resuelto en esta sesión se han resuelto haciendo uso de la misma técnica que se usó en Fundamentos de Programación I, esto es, la programación imperativa. En esta sesión se ha usado Java como lenguaje de programación, pero **no se ha hecho uso** del paradigma orientado a objetos.