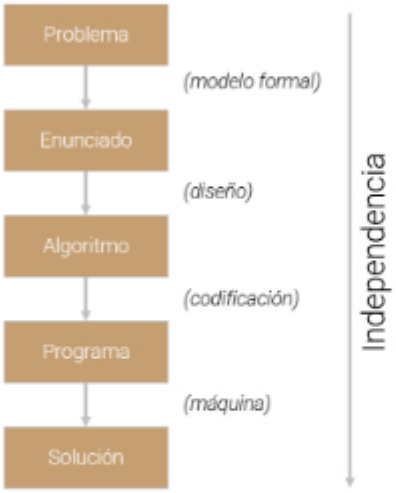
4. Nuestro primer programa: «¡Hola, mundo!»

Ya hemos presentado los tipos de datos y los algoritmos (en diagrama de flujo y pseudocódigo). Con ambos conseguimos esta fórmula:

PROGRAMAS = TIPOS DE DATOS + ALGORITMOS

Pero ahora nos deberíamos preguntar: ¿por qué tenemos que aprender a programar? Muchas personas influyentes señalan que esta es una competencia digital que, en el futuro, todo el mundo deberá aprender, al igual que ocurre con el aprendizaje de otros idiomas. Además, ¿por qué usar programas ajenos si puedo hacérmelos yo?

Recapitulando el proceso que venimos explicando —desde la presentación del problema hasta dar con la solución por medio de un sistema informático—, tendríamos lo siguiente:



1. Dado un **problema,** intentaremos encontrar un modelo formal que nos permita formularlo como un enunciado.
2. Mediante una técnica de diseño (que acabáis de aprender), realizaremos un algoritmo que trate de resolver el problema.
3. Mediante el lenguaje de programación, en nuestro caso **Java,** obtendremos el programa. Podemos utilizar cualquier otro existente que nos permita alcanzar la solución igualmente.
4. Una vez ejecutado el programa por alguien o por algún medio, obtendremos un resultado que nos dará la solución.

Figura 1.11. Pasos que se deben seguir entre el problema y la solución

Llegados a este punto, ya nos toca empezar a programar. En el mundo de la programación, el primer problema que se trata de resolver cuando te adentras en un nuevo lenguaje de programación es ver cuál es el código mínimo que se necesita para escribir la expresión **«¡Hola, mundo!»** (en inglés, ***«Hello World!»***) y mostrarla por pantalla. Por tanto, nuestro algoritmo en **pseudocódigo** no recibiría ninguna entrada y se trataría de **escribir «¡Hola, mundo!» por pantalla**.

Como muestra, aquí podéis ver cómo sería la instrucción que realiza esa acción, aunque igual va a requerir de más instrucciones para poder conformar un programa completo que podamos ejecutar en nuestro dispositivo para obtener el resultado del problema planteado. En el caso práctico ampliado programaremos todo el código necesario para ejecutarlo en **Java**.

C#	Java	Python
`Console.WriteLine("¡Hola, mundo!");`	`System.out.println("¡Hola, mundo!");`	`print "¡Hola, mundo!"`

Tabla 1.1. Instrucción para escribir «¡Hola, mundo!» y mostrarlo por pantalla

El resto del código fuente que se tenga que escribir va a depender de las instrucciones de cada lenguaje de programación. Sus creadores han decidido cuál va a ser su léxico, al que llamaremos **palabras reservadas**, y que podremos ir utilizando para confeccionar nuestro programa. En la siguiente unidad veremos que nosotros podemos ampliar esos **tokens** (palabras o fragmentos de caracteres que están separados por un símbolo especial, normalmente el espacio en blanco) con las palabras o frases juntas —que llamaremos **identificadores**— que nos interese crear en cada momento para dar nombre a nuestras variables y/o al nombre de las funciones, métodos y programas.

IMPORTANTE

En el módulo de Entornos de Desarrollo aprenderéis los tipos de clasificaciones de los lenguajes de programación, así como la filosofía de cada uno de ellos. Actualmente, los lenguajes de programación más utilizados, como Java, suelen ser multiparadigma, es decir, admiten su utilización para más de uno de los paradigmas definidos tradicionalmente. También estudiaréis el ciclo de vida de desarrollo de los programas.

En nuestro caso, vamos a centrarnos en la versión en Java. El programa mínimo necesario para poder mostrar por la consola el texto «¡Hola, mundo!» es este:

```java
public class HolaMundo
{
    public static void main(String[] args) {
        System.out.println("¡Hola Mundo!");
    }
}
```

Figura 1.22. Código «¡Hola, mundo!» en Java

4.1. Explicación del código de *HolaMundo.java*

Las palabras reservadas del lenguaje aparecen en color naranja:

- `public` hace referencia a si va a poder ser visto por otros objetos. En nuestro ejemplo, de momento todo es público y visible para todos los usuarios.
- `class` define cada clase que vayamos necesitando. El código fuente que escribiremos como programadores se ha de guardar en archivos con extensión .java. Y el nombre del archivo será el mismo que el nombre de la clase. En nuestro caso, hemos decidido elegir *HolaMundo* como

nombre de la clase (*class* en inglés). Java es un lenguaje orientado a objetos. Por tanto, tendremos que ir definiendo diferentes tipos de objetos según las necesidades del problema que deseamos resolver. Utilizaremos el par de llaves { y }, tanto al principio como al final de la clase, para indicar todo lo que estará contenido dentro de la misma.

- `static` define el método como estático o perteneciente a la clase en función de lo que puede ser invocado, aunque no creemos objetos para dicha clase. Java tomó como sintaxis para su lenguaje algunas de las características y funcionalidades de C, C++ y Objective C. Estos eran los lenguajes más utilizados a principios de la década de los 90, llamado originalmente Oak, y que acabaría incurriendo en problemas legales relacionados con las patentes de ese nombre, por lo que pasó a denominarse Java poco después de su creación.
- `void` (que significa «vacío») indica que el método no va a devolver ningún valor tras su ejecución.

Los identificadores en rojo son clases que ya existen en Java, y que están disponibles en bibliotecas (también llamadas librerías) que nos proporcionan un código escrito por el fabricante. En nuestro ejemplo tenemos:

- `String`. En Java lo tenemos que utilizar como un objeto. En nuestro código estamos definiendo el tipo de datos que va a almacenar el vector (o *array*) de los argumentos que nos puedan pasar de la ejecución del programa desde la consola, como parámetros de inicialización del programa.
- `System` es una clase del sistema. Nos va a permitir, en el ejemplo, enviar información a la consola o pantalla de salida (out). Para ello, utilizamos el método println(datos), que mostrará por la salida estándar los datos (del tipo que sea: texto, números o ambos). En este ejemplo esos datos son la cadena de texto «¡Hola, mundo!». Y para indicar que nuestra instrucción ha terminado, colocaremos el símbolo «;» al final de la línea.

Ahora solo nos queda el nombre o identificador del método **main**. Utilizaremos el par de llaves { al principio del método y el } para indicar que todas las instrucciones se ejecutarán en el mismo método. En nuestro ejemplo solo hay una instrucción. Este método **main** va a ser siempre la puerta de entrada a nuestros programas. La máquina virtual de Java sabe por dónde empezar a ejecutar nuestros programas, ya que solo vamos a poder tener un único método **main**. Aunque nuestro programa vaya creciendo y tenga cientos, miles o millones de archivos .java, solo en uno de ellos podrá existir este método especial de inicio (al igual que teníamos la instrucción de inicio en nuestro diagrama de flujo).