



universidad
cenfotec_
La U de la informática

Fundamentos de Programación

Introducción al lenguaje de programación Java.

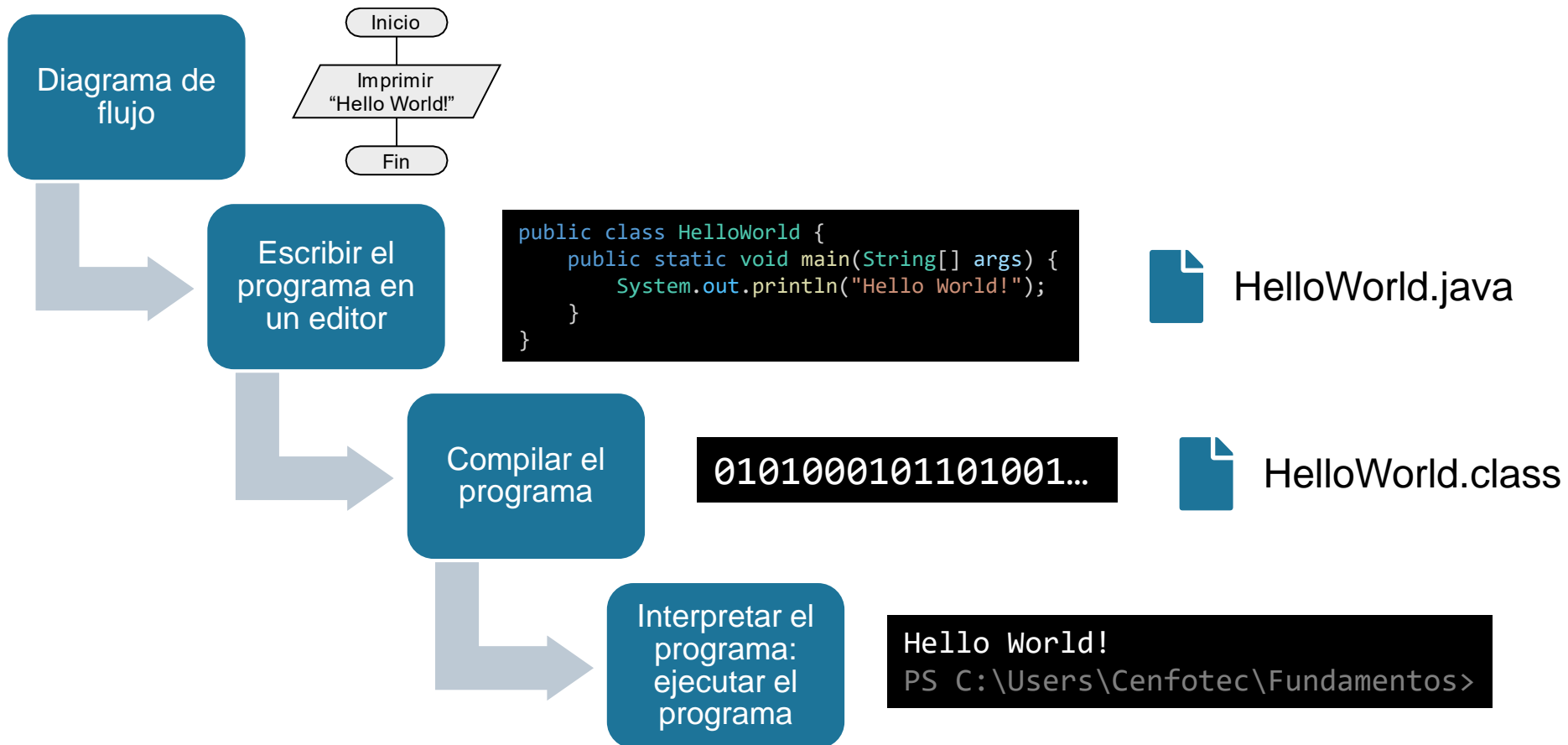
Objetivos

- Conocer y aprender a instalar el compilador del lenguaje de programación Java.
- Comprender y usar los conceptos de editar, compilar y ejecutar un programa escrito en Java.
- Comprender el concepto de flujo de entrada y salida de datos.
- Representar en el lenguaje de programación Java cualquier algoritmo con estructuras secuenciales.

Metodología de solución de un problema:

1. Definición del problema.
2. Análisis del problema: entender el problema y hacer modelado.
3. Diseño de la solución: hacer el algoritmo que solucione el problema.
4. Implementación: del programa correspondiente al algoritmo en lenguaje de programación.
5. Principios de control de calidad: hacer revisiones, pruebas y mantenimiento si es necesario.

Implementación en Java



El lenguaje Java



Fue desarrollado en *Sun Microsystems* en 1991 como parte del proyecto Green.

Entre las múltiples ventajas que ofrece Java tenemos:

- **Es pequeño:** el kit de desarrollo es sencillo, relativamente fácil de instalar y fácil de usar.
- **Es confiable:** permite implementar aplicaciones que van a funcionar correctamente sin problemas en todas las arquitecturas.
- **Es portable:** permite ser ejecutado en cualquier plataforma.

El lenguaje Java

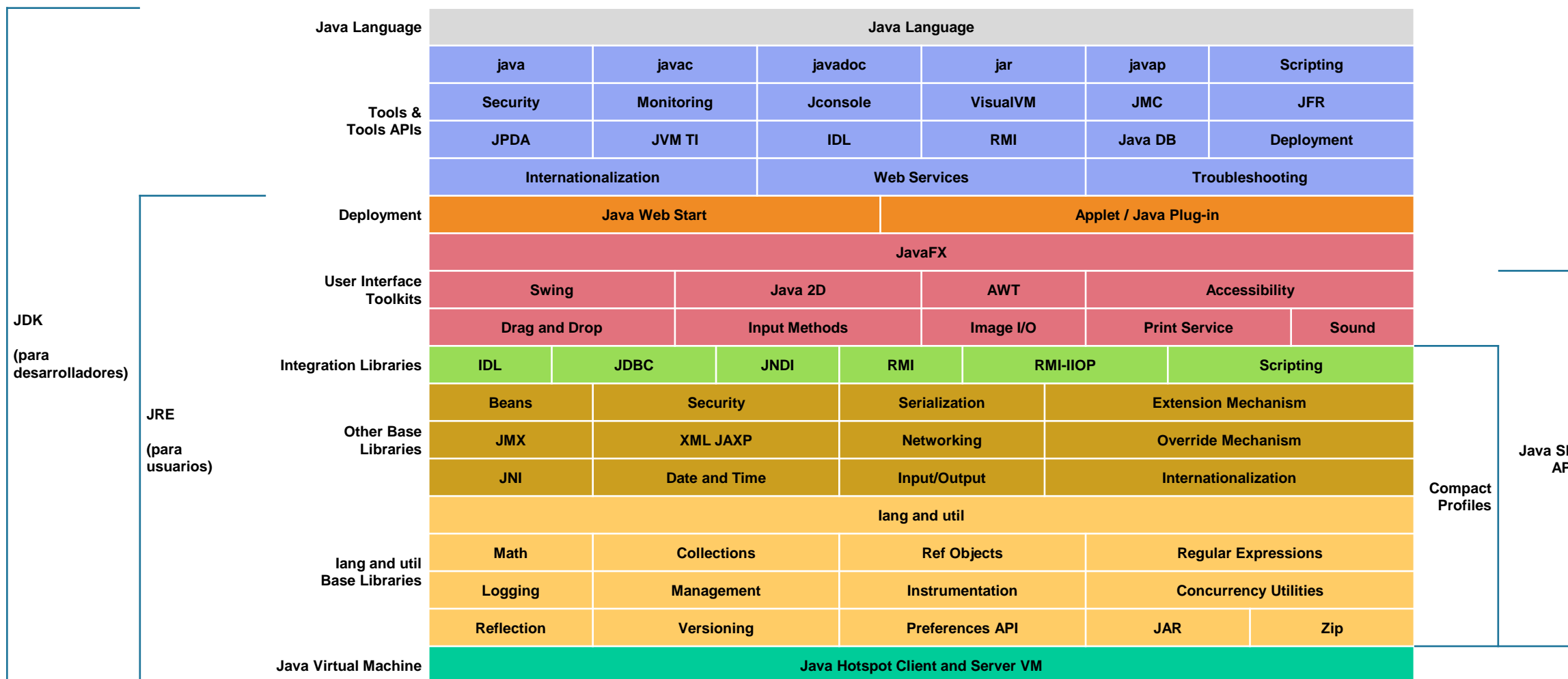
Los números de versión del lenguaje corresponden al software primario de desarrollo de Sun Microsystems: el JDK (Kit de Desarrollo de Java).

Comúnmente denominado como el JDK, actualmente está disponible en versión 14: Java SE Development Kit 14.0.2 (agosto 2020).

Se encuentra sin costo en el sitio: <https://www.oracle.com/>

Sitio de interés para consulta: <http://www.javahispano.org/>

El lenguaje Java

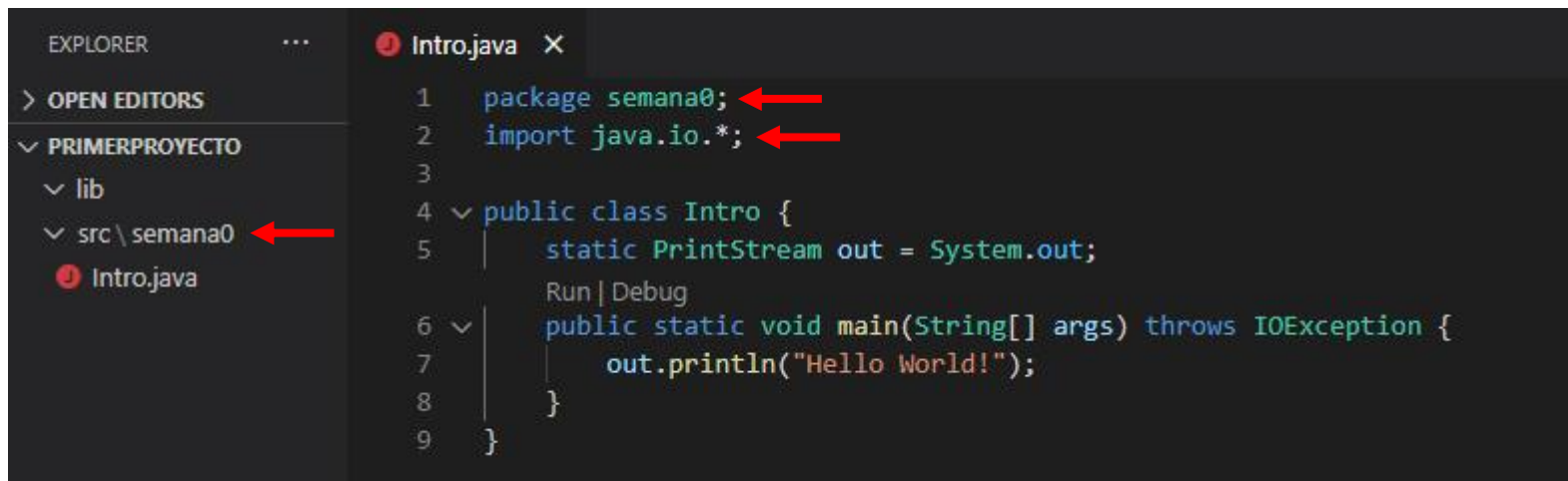


Aclaración

En el curso **Fundamentos de Programación** se utilizará **Java** con un enfoque en la programación estructurada. Aún así, es importante mencionar que Java es un lenguaje **orientado a objetos**, esto significa que está diseñado para usarse bajo ese paradigma.

Por lo tanto, hay conceptos del lenguaje que va a comprender a lo largo del presente curso y se continuará en el curso de *Programación Orientada a Objetos*.

Conceptos importantes de Java



```
EXPLORER
> OPEN EDITORS
v PRIMERPROYECTO
  v lib
    v src \ semana0
      Intro.java

Intro.java x
1 package semana0;
2 import java.io.*;
3
4 public class Intro {
5     static PrintStream out = System.out;
6     public static void main(String[] args) throws IOException {
7         out.println('Hello World!');
8     }
9 }
```

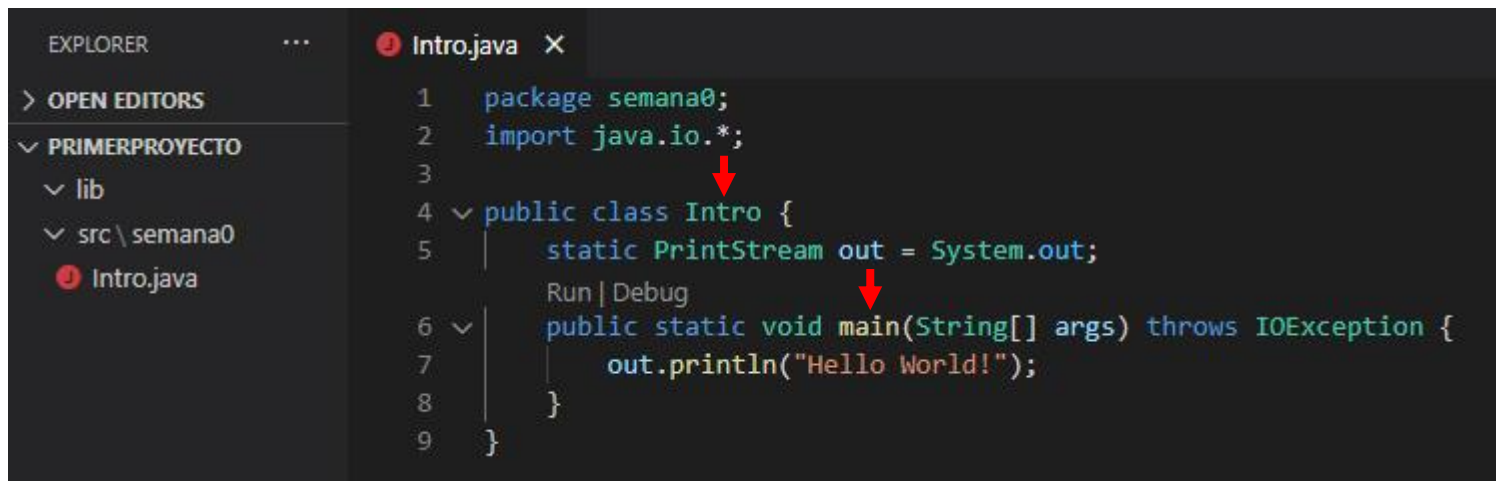
package

Los **paquetes de Java** son una forma de **organizar** los archivos de código fuente, muy similares a carpetas. Es de suma importancia que el paquete indicado en la línea 1 debe coincidir con la carpeta que contiene el archivo que se muestra al lado izquierdo en el Explorador de Archivos. En este caso, **Intro.java** está en el paquete **semana0**.

import

Java tiene una gran cantidad de paquetes ya creados con funcionalidades distintas, organizados en varias **librerías**. Para utilizar estos recursos se debe importar en el archivo fuente utilizando la palabra reservada **import**. En este caso, para ejecutar el programa **Intro**, se está importando **java.io.***, que corresponde a las funcionalidades de entrada y salida de datos.

Conceptos importantes de Java



```
1 package semana0;
2 import java.io.*;
3
4 public class Intro {
5     static PrintStream out = System.out;
6     public static void main(String[] args) throws IOException {
7         out.println('Hello World!');
8     }
9 }
```

class

Una **clase** es un contenedor donde se va a codificar un programa.

El nombre del archivo **Intro.java** en el Explorador de Archivos debe coincidir con el nombre que se escribe después de las palabras reservadas **public class**. Por estándar, es importante que su primera letra esté en mayúscula.

main

El **main** se refiere al programa principal del código. Todas las instrucciones que se encuentran dentro de las llaves del **main** serán ejecutadas.

Conceptos importantes de Java

```
1  package semana0;  
2  import java.io.*;  
3  public class Intro { ←  
4  
5      static PrintStream out = System.out;  
6      public static void main(String[] args) throws IOException { ←  
7          out.println("Hello World!");  
8      } ←  
9  } ←
```

llaves

Las **llaves** se utilizan para delimitar bloques de código o un conjunto de instrucciones.

Tal como se mencionó anteriormente, la clase contiene el programa y dentro de esta clase, se encuentra el programa principal (**main**). En el ejemplo se puede ver que la clase **Intro** empieza en la línea 3 y termina en la línea 9, y el **main** da inicio en la línea 6 y finaliza en la línea 8.

Conceptos importantes de Java

```
1 package semana0;  
2 import java.io.*;  
3 public class Intro {  
4  
5     static PrintStream out = System.out;  
6     public static void main(String[] args) throws IOException {  
7         out.println("Hello World!");  
8     }  
9 }
```

Indentación

Indentación es una práctica que consiste en agregar **tabulaciones** o **espacios en blanco** al inicio de cada instrucción con el objetivo que el contenido del programa se estructure y así mejorar la comprensión del mismo. Este se puede comparar con el concepto de sangría en documentos.

En Java estos espacios en blanco **no afectan** la ejecución del programa pero en otros lenguajes como Python, la **indentación** juega un papel muy similar a las llaves.

Conceptos importantes de Java

```
Run | Debug
6 public static void main(String[] args) throws IOException {
7     out.println("Hello World!");
8 }
```

<code>public static void</code>	Son modificadores del programa principal (main). A medida que va avanzado en el curso podrá aprender el significado de cada uno de ellos.
<code>(String[] args)</code>	Se refiere a un conjunto de argumentos que Java envía al main para su ejecución. Más adelante podrá comprender mejor su objetivo.
<code>throws IOException</code>	En Java se utiliza para capturar los errores durante la ejecución, conocidos como excepciones.

Conceptos importantes de Java

```
1  package semana0;  
2  
3  import java.io.BufferedReader;  
4  import java.io.IOException;  
5  import java.io.InputStreamReader;  
6  import java.io.PrintStream;  
7  
8  public class Intro {  
9  
10     static BufferedReader in = new BufferedReader(new InputStreamReader(System.in));  
11     static PrintStream out = System.out;  
12     Run | Debug  
13     public static void main(String[] args) throws IOException {  
14         String nombre = in.readLine();  
15         out.println("Hola " + nombre);  
16     }  
17 }
```

Este es un ejemplo de un programa funcional. El código al igual que los diagramas, se define como una serie de pasos que se van a ejecutar línea tras línea. Como se mencionaba anteriormente, solo lo que está dentro del **main** será ejecutado, entonces la lógica del programa es únicamente lo que está en las líneas 13 y 14, es decir, en el cuerpo de **main**, delimitado por sus llaves.

Conceptos importantes de Java

```
3  import java.io.BufferedReader;
4  import java.io.IOException;
5  import java.io.InputStreamReader;
6  import java.io.PrintStream;
```

Es importante recordar que **Java** tiene una gran cantidad de paquetes ya creados con funcionalidades distintas, organizados en varias **librerías**. Anteriormente se mencionó que para utilizar estos recursos se debe importar en el archivo fuente utilizando la palabra reservada **import**. En este caso, en las líneas 3, 4, 5 y 6, se están importando paquetes específicos de la librería de **entrada** y **salida** de datos de Java (**java.io**), los cuales son necesarios para usar el teclado que es la entrada estándar de datos y la pantalla que es la salida estándar de datos.

Anteriormente se utilizó **import java.io.***, que permite importar todos los paquetes que se encuentran dentro de la librería **java.io**, por lo que no es necesario especificar cada uno de ellos dentro de su programa.

Conceptos importantes de Java

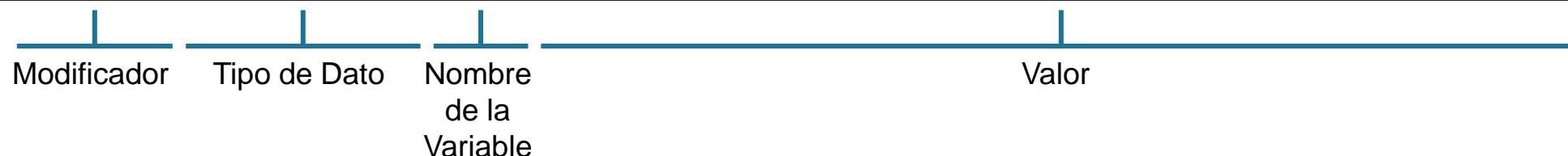
```
10    static BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
11    static PrintStream out = System.out;
```

Estas dos instrucciones van entre las llaves de la clase **Intro** y antes del **main**.

En la línea 10 se declara e inicializa una variable con el nombre **in**, la cual le permitirá obtener datos del teclado. Además, en la línea 11 se declara e inicializa una variable con el nombre **out**, la que le permitirá imprimir datos en pantalla.

A estas variables que no están dentro de **main** pero sí dentro de **Intro** les vamos a llamar **globales**, son variables que podremos usar en cualquier lugar dentro del programa.

```
10    static BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
```



Conceptos importantes de Java

```
12     public static void main(String[] args) throws IOException {  
13         String nombre = in.readLine();  
14         out.println("Hola " + nombre);  
15     }
```

En las instrucciones de las líneas 13 y 14, se están utilizando las variables globales **in** y **out** para leer valores del teclado y posteriormente imprimir esos valores en la pantalla.

<code>in.readLine();</code>	Es una instrucción que recupera una línea escrita usando el teclado, esta instrucción recupera una cadena de caracteres y la guarda en la variable nombre .
<code>out.println();</code>	Esta es una instrucción que imprime mensajes por pantalla, en este caso se imprime el mensaje Hola seguido del contenido de la variable nombre .

Conceptos importantes de Java

```
1  String nombre;  
2  int numero1;  
3  double numero2;  
4  float numero3;  
5  
6  nombre = in.readLine();  
7  numero1 = Integer.parseInt(in.readLine());  
8  numero2 = Double.parseDouble(in.readLine());  
9  numero3 = Float.parseFloat(in.readLine());
```

En este ejemplo se puede ver que en la línea número 1 se declara una variable de tipo **String** (cadena de caracteres), en la línea 2 una variable de tipo entero, en la línea número 3 una variable de tipo real (de mayor tamaño) y en la línea 4 una variable de tipo real (de menor tamaño).

En la línea número 6 se lee del teclado una cadena de caracteres y se asigna ese valor directamente a la variable **nombre**.

En la línea número 7 se lee también una cadena de caracteres, pero **numero1** al ser una variable de tipo entero, es necesario convertir dicha cadena a un valor entero mediante la instrucción **Integer.parseInt()**;

Este proceso se repite para las variables **numero2** y **numero3**, convirtiendo las cadenas de caracteres a sus respectivos tipos de datos, mediante las instrucciones **Double.parseDouble()** y **Float.parseFloat()** respectivamente.



universidad
cenfotec_

La U de la informática