



Java Foundations

3-5

Entrada del teclado

ORACLE
Academy



Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

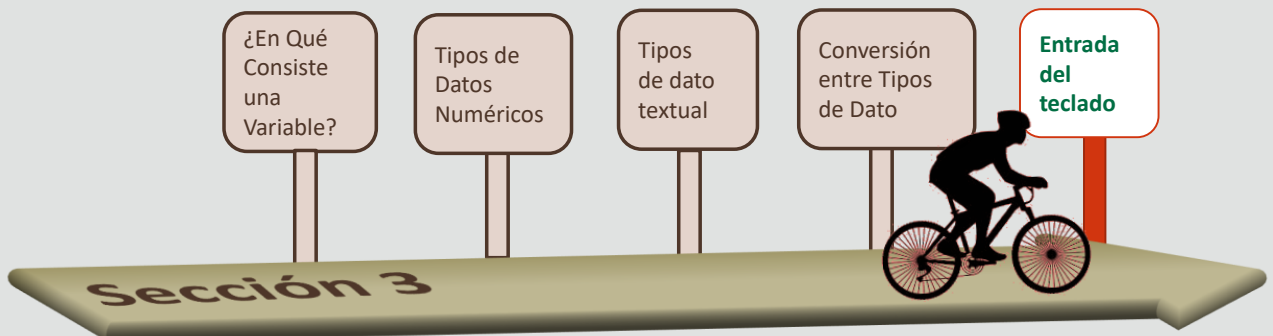
Objetivos

- En esta lección se abordan los siguientes objetivos:
 - Comprender las entradas de los usuarios
 - Crear una clase JOptionPane para recopilar entradas de los usuarios
 - Usar una clase Scanner para recopilar entradas de la consola
 - Usar una clase Scanner para recopilar entradas de un archivo
 - Comprender cómo gestionar los tokens y delimitadores mediante una clase Scanner



Temas

- **Entrada del usuario**
- JOptionPane
- Scanner



ORACLE
Academy

JFo 3-5
Entrada del teclado

Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

4

¿Por qué debe obtener la entrada del usuario?

- La asignación manual de los valores a las variables es lo que se conoce como codificación de valores:

```
String input = "This is a String";
```

- Puede cambiar fácilmente valores codificados, ya que tiene el código fuente y NetBeans:

```
String input = "This is a different String";
```

- Pero al distribuir software, los usuarios no tendrán la misma facilidad

Tipos de entrada del usuario

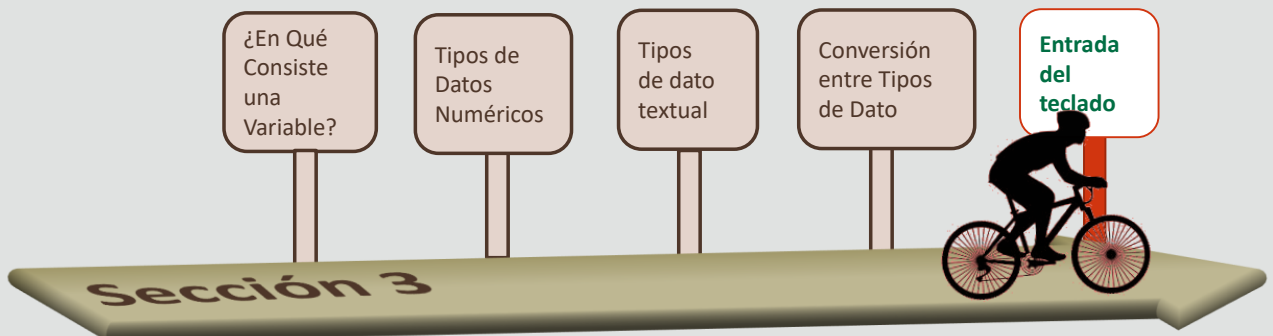
- Entre los ejemplos de entrada del usuario se incluye...
 - Pulsar un botón de un controlador de juego
 - Introducir una dirección en un GPS
 - Introducir números y funciones en una calculadora
 - Decir a las personas su nombre
- Pero sin la entrada del usuario...
 - ¿Cuándo hará el juego que su personaje salte?
 - ¿Dónde le guiará el GPS?
 - ¿Qué números devorará la calculadora?
 - ¿Qué le llamará la gente?

Cómo obtener la entrada del usuario

- Hay muchas formas de obtener la entrada del usuario:
 - Botones (físicos o virtuales)
 - Ruedas y diales
 - Reconocimiento de voz
 - Cuadros de diálogo de texto
 - Archivos de propiedades
- Java ofrece varias maneras de obtener la entrada del usuario, incluidos...
 - Swing JOptionPane
 - JavaFX (un sucesor de Swing, que trataremos más adelante)
 - Scanner

Temas

- Entrada del usuario
- **JOptionPane**
- Scanner



ORACLE
Academy

JFo 3-5
Entrada del teclado

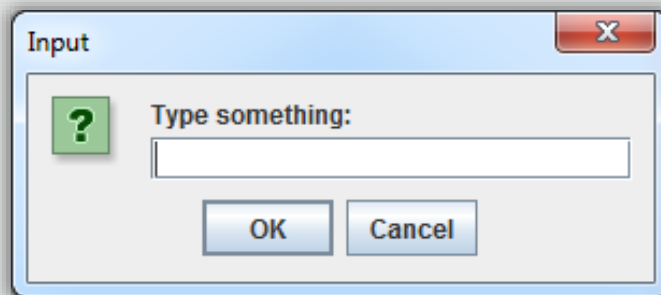
Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

8

JOptionPane

- Se trata de una forma sencilla de obtener la entrada de los usuarios:

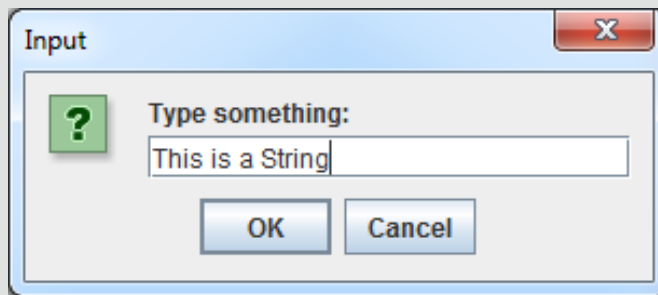
```
JOptionPane.showInputDialog("Type something:");
```



JOptionPane devuelve cadenas

- La entrada se puede almacenar como una cadena:

```
String input = JOptionPane.showInputDialog("Type something:");
```



- Esto equivale a escribir:

```
String input = "This is a String";
```



Ejercicio 1, parte 1

- Importe y edite el proyecto `Input01`
- Cree un `JOptionPane`:
 - NetBeans detectará un error
 - Siga las sugerencias de NetBeans de importación de `javax.swing.JOptionPane`
 - Trataremos la importación en otra sección



Ejercicio 1, parte 2

- Almacene esta entrada como una String
- Imprima la variable String
- Analice String como una variable int independiente
 - Tendrá que introducir un valor que se pueda analizar
 - Imprima este valor +1
- Intente crear un cuadro de diálogo, analizarlo e inicializar un int en una sola línea Debe tener solo un punto y coma (;)

Código condensado

- Puede distribuir las entradas, analizar y calcular en varias líneas:

```
String inputString = JOptionPane.showInputDialog("??");  
int input = Integer.parseInt(inputString);  
input++;
```

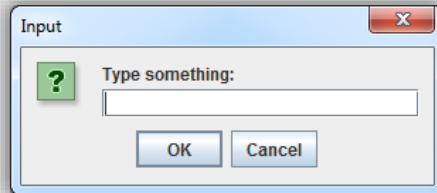
- O condensarlo en una sola línea:

```
int input = Integer.parseInt(JOptionPane.showInputDialog("??")) +1;
```

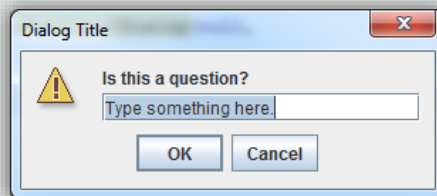
- Esta opción es una cuestión de preferencia personal
 - Pero si necesita hacer referencia a determinados valores posteriormente, sería útil guardar estos valores en una variable

Diferentes InputDialogs

- Hemos creado un InputDialog simple:



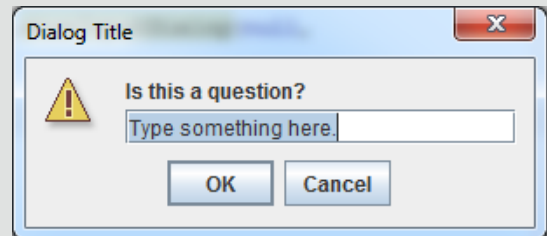
- Con un código más complicado, podemos personalizar el InputDialog más:



Más opciones con InputDialogs

- Esta versión de un InputDialog no devuelve una String
- El resultado se debe convertir en String para que se pueda utilizar:

Conversión

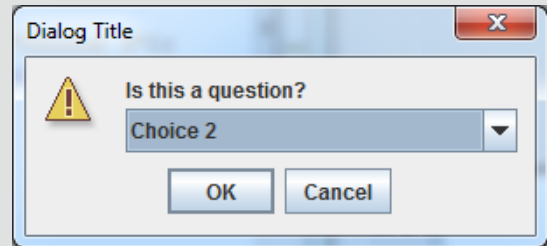


```
String input = (String)JOptionPane.showInputDialog(null,  
"Is this a question?",  
"Dialog Title",  
2,  
null,  
null,  
"Type something here.");
```

¿Confuso con este código? No se preocupe. Incluso los programadores con experiencia se pueden sentir confusos cuando ven un nuevo código. Una forma muy útil para desarrollar su comprensión consiste en modificar el código existente y ver qué ocurre. Haremos esto en el siguiente ejercicio.

Más opciones con InputDialogs

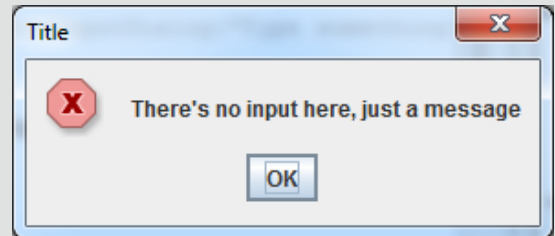
- Para evitar entradas no deseadas, es posible proporcionar solo los valores aceptables a los usuarios
- Alguna de esta sintaxis se examina con más detalle en la sección 8



```
String[] acceptableValues = {"Choice 1", "Choice 2", "Choice 3"};  
InputDialog string= (String)JOptionPane.showInputDialog(null,  
    "Is this a question?",  
    "Dialog Title",  
    2,  
    null,  
    acceptableValues,  
    acceptableValues[1]);
```


showMessageDialog


- Un showMessageDialog no proporciona un campo para la entrada
- Existen muchas otras variaciones de JOptionPane



```
JOptionPane.showMessageDialog(  
    null,  
    "There's no input here, just a message",  
    "Title",  
    0);
```



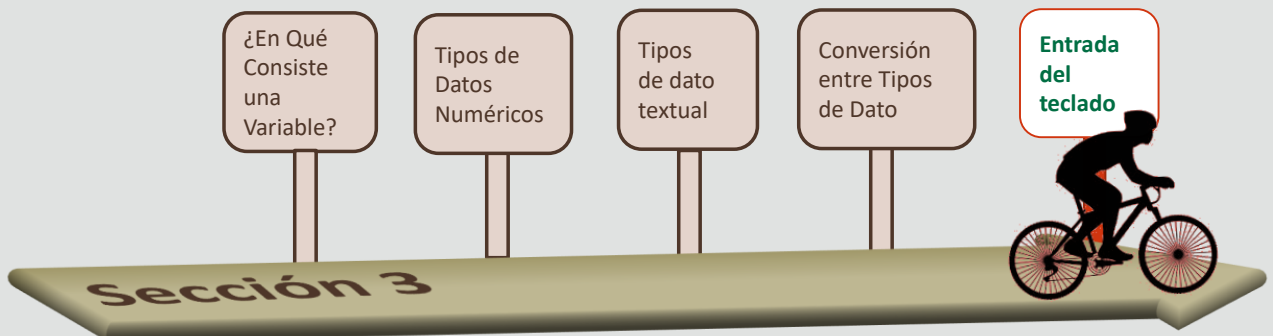
Ejercicio 2

- Importe y edite el proyecto Input02
- Experimente con el código e intente volver a cambiar...
 - El título del mensaje
 - El mensaje
 - Cualquier texto de entrada por defecto
 - El icono del cuadro de diálogo 
- Analizar, manipular e imprimir cualquier entrada

Indicación: Ignore los valores nulos. Si necesita ayuda, la documentación de Java podría ser útil:
<http://docs.oracle.com/javase/8/docs/api/javax/swing/JOptionPane.html>.

Temas

- Entrada del usuario
- JOptionPane
- **Scanner**



ORACLE
Academy

JFo 3-5
Entrada del teclado

Copyright © 2020, Oracle y/o sus filiales. Todos los derechos reservados.

19

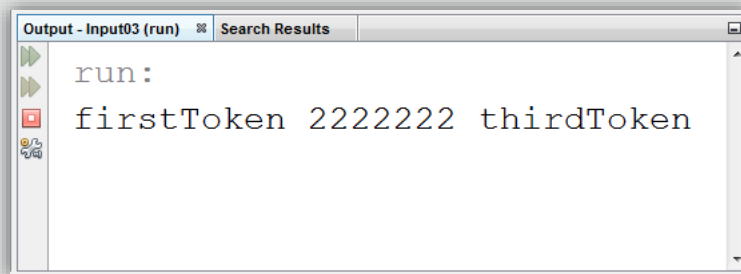
Obtención de una entrada con un objeto Scanner

- Un objeto Scanner abre un flujo para recopilar entradas:
 - System.in prepara Scanner para recopilar entradas de la consola
 - Escriba la entrada en la ventana de salida de NetBeans
 - También se puede utilizar el objeto Scanner sin IDE
- Se trata de una de las mejores prácticas para cerrar el flujo de Scanner cuando haya terminado

```
public static void main(String[] args) {  
    Scanner sc = new Scanner(System.in);  
  
    sc.close();  
} //end method main
```

Lectura de entradas con un Scanner

- El objeto Scanner busca tokens
- Los tokens están separados por un delimitador
 - El delimitador por defecto es un espacio



The screenshot shows a Java IDE output window titled "Output - Input03 (run)". The window contains the following text:

```
run:  
firstToken 2222222 thirdToken
```

La clase Scanner

- La clase Scanner, como cualquier otra clase, tiene campos y métodos
- Unos métodos de clase Scanner útiles...
 - nextInt() lee el siguiente token como un valor int
 - nextDouble() lee el siguiente token como un valor double
 - next() lee el siguiente token como un valor String

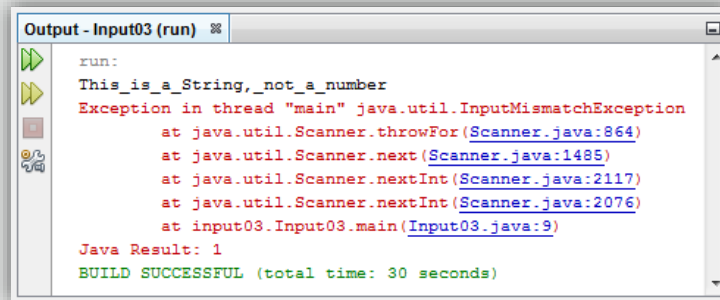
```
public static void main(String[] args) {  
    Scanner sc = new Scanner(System.in);  
    int x = sc.nextInt();  
    double y = sc.nextDouble();  
    String z = sc.next();  
    sc.close();  
} //end method main
```



Ejercicio 3

- Importe y edite el proyecto `Input03`
- Cree una clase `Scanner`:
 - NetBeans detectará un error
 - Siga las sugerencias de NetBeans de importación de `java.util.Scanner`
 - Recuerde cerrar la clase `Scanner`
- Utilice la clase `Scanner` y `System.in` para escribir un programa que...
 - Busque e imprima la suma de tres números enteros introducidos por el usuario
- Intente introducir menos de tres tokens
- Intente introducir un token que no se pueda analizar como un `int`

Excepciones: InputMismatchException



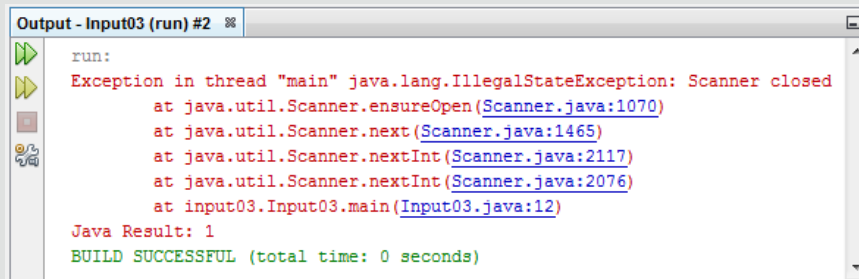
The screenshot shows a Java IDE output window titled "Output - Input03 (run)". The output text is as follows:

```
run:
This_is_a_String,_not_a_number
Exception in thread "main" java.util.InputMismatchException
    at java.util.Scanner.throwFor(Scanner.java:864)
    at java.util.Scanner.next(Scanner.java:1485)
    at java.util.Scanner.nextInt(Scanner.java:2117)
    at java.util.Scanner.nextInt(Scanner.java:2076)
    at input03.Input03.main(Input03.java:9)
Java Result: 1
BUILD SUCCESSFUL (total time: 30 seconds)
```

- Se produce cuando la entrada no se analiza como el tipo esperado:

```
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    System.out.println(sc.nextInt());
    sc.close();
} //end method main
```


Excepciones: IllegalStateException

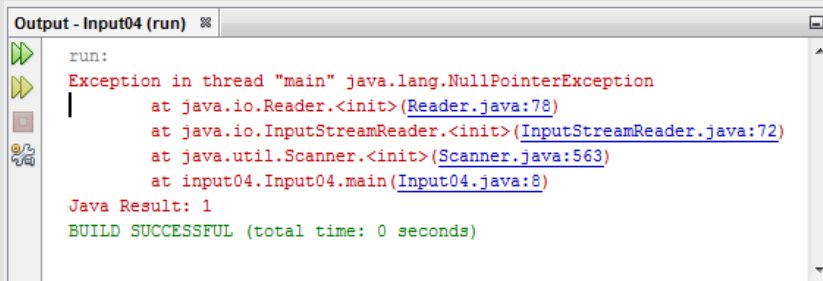


```
run:
Exception in thread "main" java.lang.IllegalStateException: Scanner closed
    at java.util.Scanner.ensureOpen(Scanner.java:1070)
    at java.util.Scanner.next(Scanner.java:1465)
    at java.util.Scanner.nextInt(Scanner.java:2117)
    at java.util.Scanner.nextInt(Scanner.java:2076)
    at input03.Input03.main(Input03.java:12)
Java Result: 1
BUILD SUCCESSFUL (total time: 0 seconds)
```

- Se produce porque se accede al flujo después de que se haya cerrado:

```
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    sc.close();
    System.out.println(sc.nextInt());
} //end method main
```

Excepciones: NullPointerException



```
run:
Exception in thread "main" java.lang.NullPointerException
    at java.io.Reader.<init>(Reader.java:78)
    at java.io.InputStreamReader.<init>(InputStreamReader.java:72)
    at java.util.Scanner.<init>(Scanner.java:563)
    at input04.Input04.main(Input04.java:8)
Java Result: 1
BUILD SUCCESSFUL (total time: 0 seconds)
```

- Se produce porque "fakeFile.txt" no existe. Además, es un error común olvidar la extensión .txt

```
public static void main(String[] args) {
    Scanner sc = new Scanner(
        Input04.class.getResourceAsStream("fakeFile.txt"));
    sc.close();
} //end method main
```

Recuerde la extensión

Lectura desde un archivo

- Java ofrece varias formas para leer archivos
- Más métodos de Scanner útiles incluyen:
 - `nextLine()` avanza la clase Scanner más allá de la línea actual y devuelve la entrada que se ha omitido
 - `findInLine("StringToFind")` intenta buscar la siguiente incidencia de un patrón creado a partir de la cadena especificada, ignorando los delimitadores

```
public static void main(String[] args) {  
    Scanner sc = new Scanner(  
        Input04.class.getResourceAsStream("fakeFile.txt"));  
    int x = sc.nextInt();  
    String entireLine = sc.nextLine();  
    sc.close();  
} //end method main
```



Ejercicio 4, parte 1

- Importe y edite el proyecto `Input04`
- Ejecute el código y examine la salida
- Lea cada línea siguiente hasta que encuentre "BlueBumper"
- Los dos números después de "BlueBumper" son `xPositon` e `yPosition` del objeto `Almacene` estas coordenadas como números enteros e imprímalos
- Examine `input04text.txt`, si es necesario



Ejercicio 4, parte 2

- Examine Level05.txt, si tiene curiosidad:
 - Así es como los datos de nivel se almacenan para Java Puzzle Ball
 - Leer y analizar los datos de nivel es un poco más complicado que lo que ha hecho en este ejercicio
 - Pero si ha finalizado este ejercicio, está cerca de entender cómo se hace

Resumen

- En esta lección, debe haber aprendido lo siguiente:
 - Comprender las entradas de los usuarios
 - Crear una clase JOptionPane para recopilar entradas de los usuarios
 - Usar una clase Scanner para recopilar entradas de la consola
 - Usar una clase Scanner para recopilar entradas de un archivo
 - Comprender cómo gestionar los tokens y delimitadores mediante una clase Scanner



