

Ejercicio 6.4 Excepciones (p. 399)

Las excepciones son un mecanismo especial para gestionar errores y permiten separar el tratamiento de errores del código normal de un programa (Kölling y Barnes, 2013).

El formato para escribir un bloque en el que se gestionan excepciones es:

```
try {  
    instrucciones  
} catch {  
    instrucciones  
} finally {  
    instrucciones  
}
```

Dentro del bloque try se coloca el código que podría generar una excepción. Los bloques catch capturan y tratan una excepción cuando esta ocurre. Pueden existir varios bloques catch. Estos se definen directamente después del bloque try. Ningún código puede estar entre el final del bloque try y el comienzo del primer bloque catch. Los catch se evalúan por orden, si un catch atrapa la excepción que ha ocurrido, se ejecuta y los demás no.

Por último, el bloque finally es opcional e incluye código que se ejecuta siempre, independientemente si se ha producido una excepción o no (Altadill-Izurra y Pérez-Martínez, 2017).

Objetivos de aprendizaje

Al finalizar este ejercicio, el lector tendrá la capacidad para:

- Identificar bloques de código donde se pueden generar excepciones.
- Identificar los bloques catch que capturan una excepción específica.
- Reconocer y diferenciar los propósitos de los bloques try, catch y finally para la gestión de excepciones.

Diagrama de Casos de uso

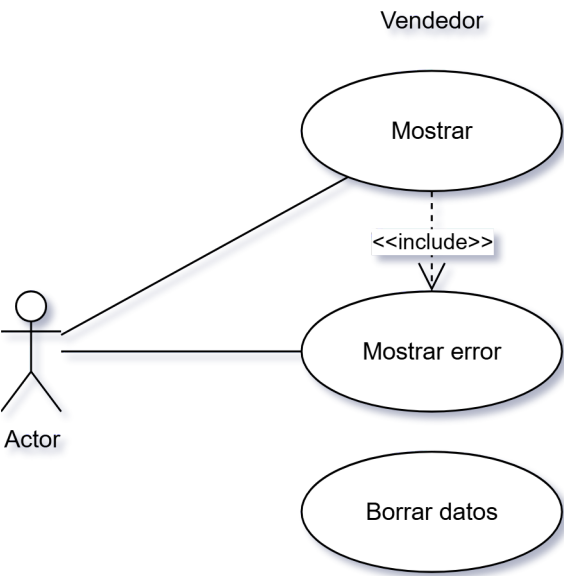
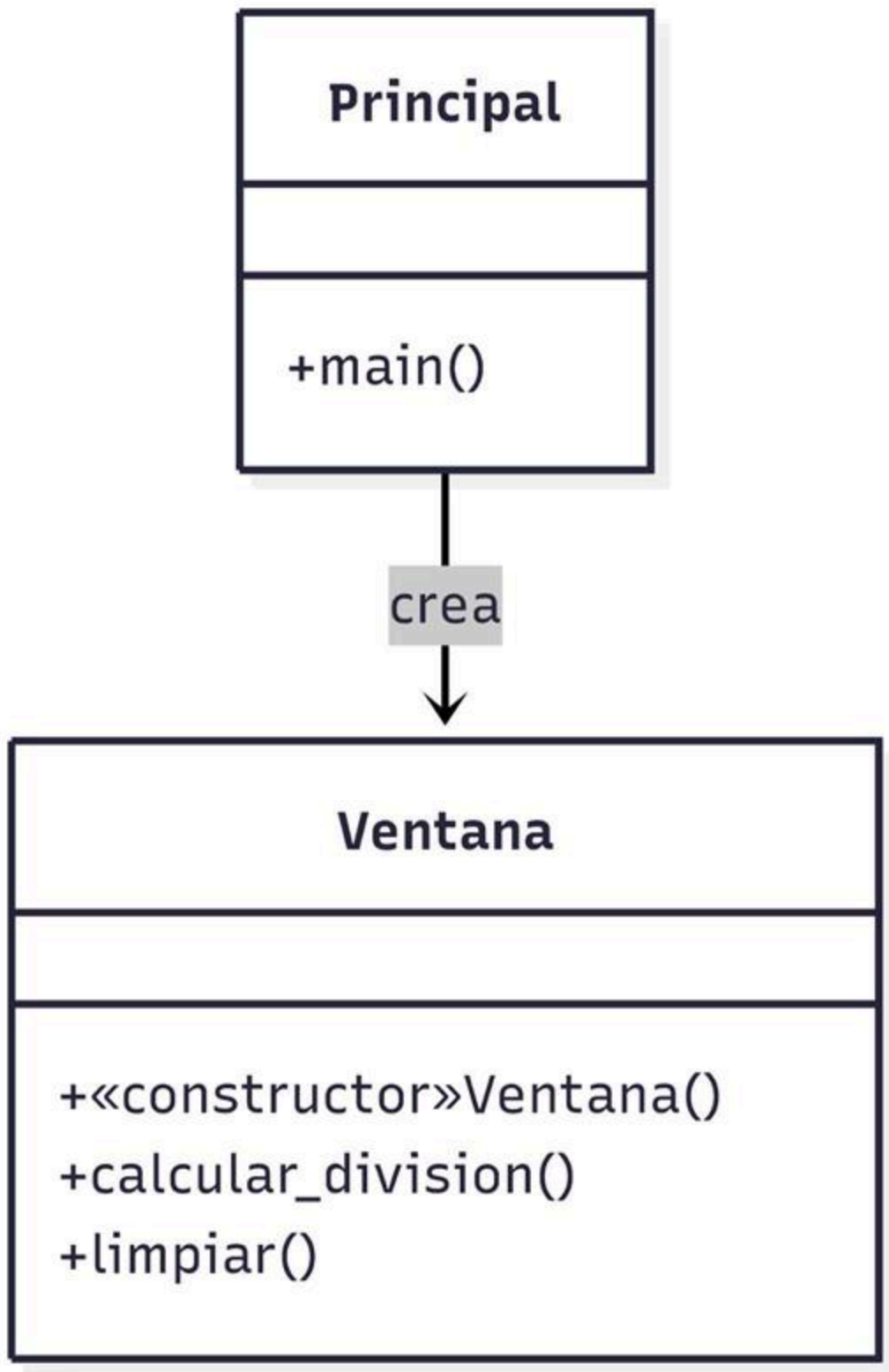


Diagrama de Clases

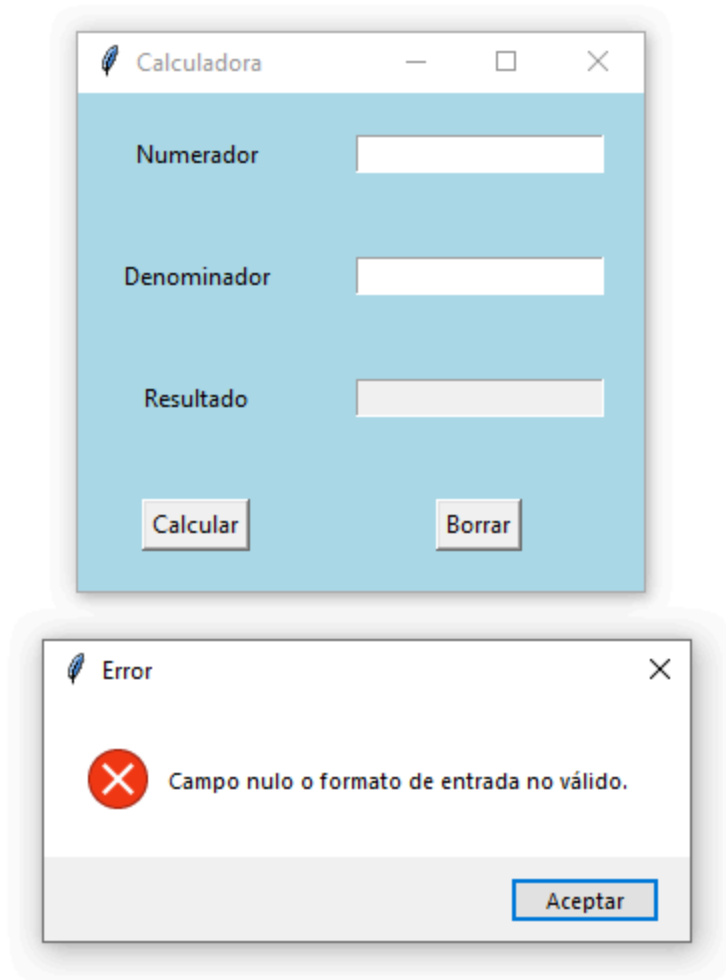


Solución

[Click para ver código fuente](#)

Ejecución del programa

Sin ingresar un valor



Ingresando solamente un valor no válido

Calculadora


Numerador

Denominador

Resultado

Calcular

Error

 Campo nulo o formato de entrada no válido.

Ingresando solamente un valor válido/numérico

Calculadora


Numerador

Denominador

Resultado

Calcular

Error

 Campo nulo o formato de entrada no válido.

Aceptar

Ingresando 0 como denominador


Calculadora

Numerador

Denominador

Resultado

Error

 No es posible realizar una división por 0.

Ingresando valores válidos

Calculadora

Numerador

Denominador

Resultado

Ejercicio 6.5. Lanzamiento de excepciones (p. 404)

Objetivos de aprendizaje

Al finalizar este ejercicio, el lector tendrá la capacidad para:

- Lanzar excepciones específicas en métodos de una clase.
- Conocer y aplicar la sentencia `throw` para el lanzamiento de excepciones.

Enunciado: clase Vendedor

Se requiere implementar una clase vendedor que posee los siguientes atributos: nombre (tipo `String`), apellidos (tipo `String`) y edad (tipo `int`).

La clase contiene un constructor que inicialice los atributos de la clase. Además, la clase posee los siguientes métodos:

- Imprimir: muestra por pantalla los valores de sus atributos.
- Verificar edad: este método recibe como parámetro un valor entero que representa la edad del vendedor. Para que un vendedor pueda desempeñar sus labores se requiere que sea mayor de edad (mayor de 18 años). Si esta condición no se cumple, se lanza una excepción de tipo `IllegalArgumentException` con el mensaje "El vendedor debe ser mayor de 18 años". Además, se evalúa si la edad se encuentra en el rango de 0 a 120, si no se cumple, se genera una excepción de tipo `IllegalArgumentException` con el mensaje "La edad no puede ser negativa ni mayor a 120". Si la edad cumple estos requerimientos se pueden instanciar el objeto vendedor.

Además, se requiere que los datos del vendedor se ingresen por teclado.

Diagrama de Casos de uso

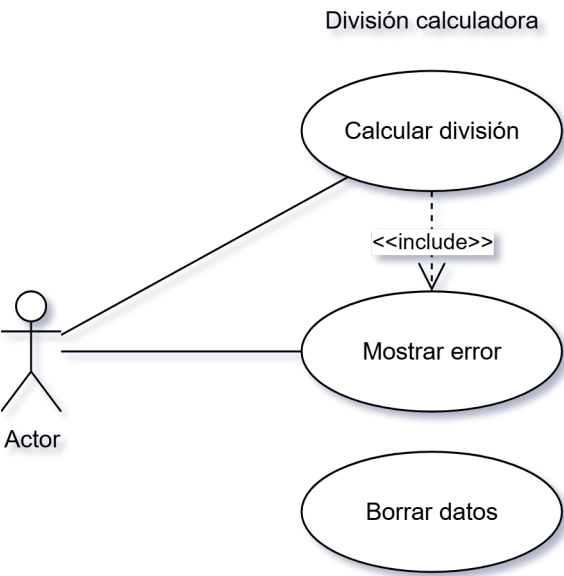
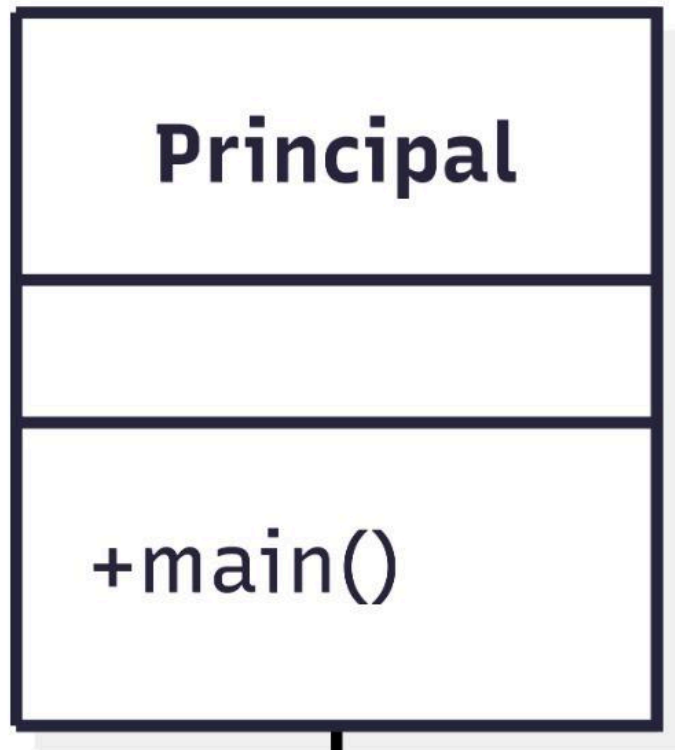


Diagrama de Clases



crea



+limpiar()

Solución

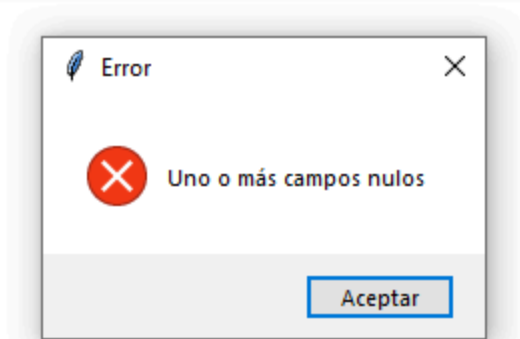
[Click para ver código fuente](#)

Ejecución del programa

Sin ingresar ningún valor



The screenshot shows a window titled "Interfaz de Vendedor" with a light blue background. It contains three input fields for "Nombre", "Apellido", and "Edad", each with a white text box and a grey placeholder box. Below the fields are two buttons: "Mostrar" and "Borrar". At the bottom, a message reads "La edad no puede ser negativa ni mayor a 120".



Interfaz de Vendedor


Nombre

Apellido


Edad

La edad no puede ser negativa ni mayor a 120

Error

 Campo de edad nulo o formato de edad no válido.

Ingresando solamente un valor

 Interfaz de Vendedor

Nombre


Apellido


Edad

Mostrar

Borrar

La edad no puede ser negativa ni mayor a 120

 Error

 Uno o más campos nulos

Aceptar

Interfaz de Vendedor


Nombre

Apellido

Edad

La edad no puede ser negativa ni mayor a 120

Error

 Campo de edad nulo o formato de edad no válido.

Ingresando un número negativo

Interfaz de Vendedor


Nombre

Apellido

Edad

La edad no puede ser negativa ni mayor a 120

Error

 La edad no puede ser negativa ni mayor a 120

Ingresando un número menor a 18 y mayor o igual a 0

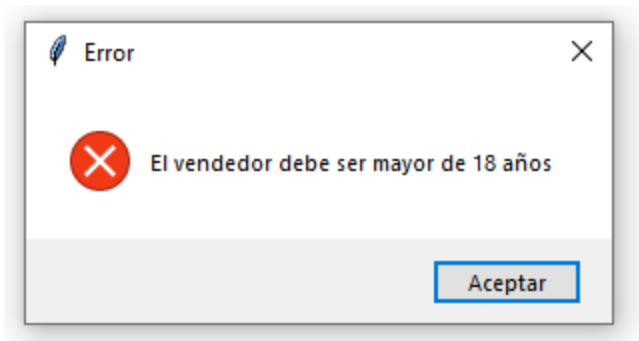
Interfaz de Vendedor

Nombre

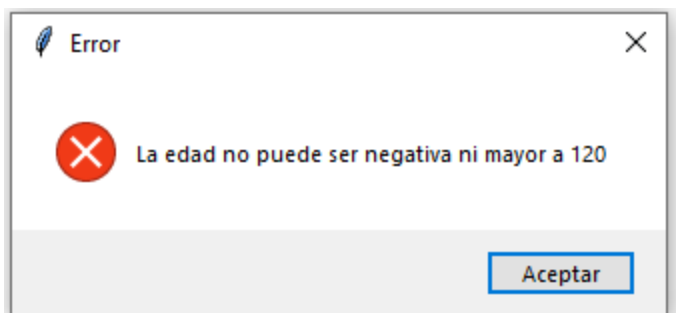
Apellido

Edad

El vendedor debe ser mayor de 18 años



Ingresando un número mayor a 120

A window titled "Interfaz de Vendedor" with a light blue background. It contains three input fields: "Nombre" with the value "Pedro", "Apellido" with the value "Pérez", and "Edad" with the value "121". To the right of each input field is a corresponding disabled grey field. Below the fields are two buttons: "Mostrar" and "Borrar". At the bottom, there is a text label: "La edad no puede ser negativa ni mayor a 120".


Ingresando un valor no válido

Interfaz de Vendedor

Nombre	<input type="text" value="Pedro"/>	<input type="text" value="Pedro"/>
Apellido	<input type="text" value="Pérez"/>	<input type="text" value="Pérez"/>
Edad	<input type="text" value="asdsada"/>	<input type="text"/>

La edad no puede ser negativa ni mayor a 120

Error

 Campo de edad nulo o formato de edad no válido.

Ingresando un valor válido

Interfaz de Vendedor

Nombre	<input type="text" value="Pedro"/>	<input type="text" value="Pedro"/>
Apellido	<input type="text" value="Pérez"/>	<input type="text" value="Pérez"/>
Edad	<input type="text" value="19"/>	<input type="text" value="19"/>

La edad no puede ser negativa ni mayor a 120

Ejercicio 6.6. catches múltiples (p. 410)

Objetivos de aprendizaje

Al finalizar este ejercicio, el lector tendrá la capacidad para:

- Definir múltiples catch para el tratamiento de excepciones.
- Definir gestores para el tratamiento de excepciones aritméticas.

Enunciado: clase CálculosNuméricos

Se requiere definir una clase denominada CálculosNuméricos que realice las siguientes operaciones:

- Calcular el logaritmo neperiano recibiendo un valor double como parámetro. Este método debe ser estático. Si el valor no es positivo se genera una excepción aritmética.
 - Calcular la raíz cuadrada recibiendo un valor double como parámetro. Este método debe ser estático. Si el valor no es positivo se genera una excepción aritmética.
- Se debe crear un método main que utilice dichos métodos ingresando un valor por teclado.

Diagrama de Casos de uso

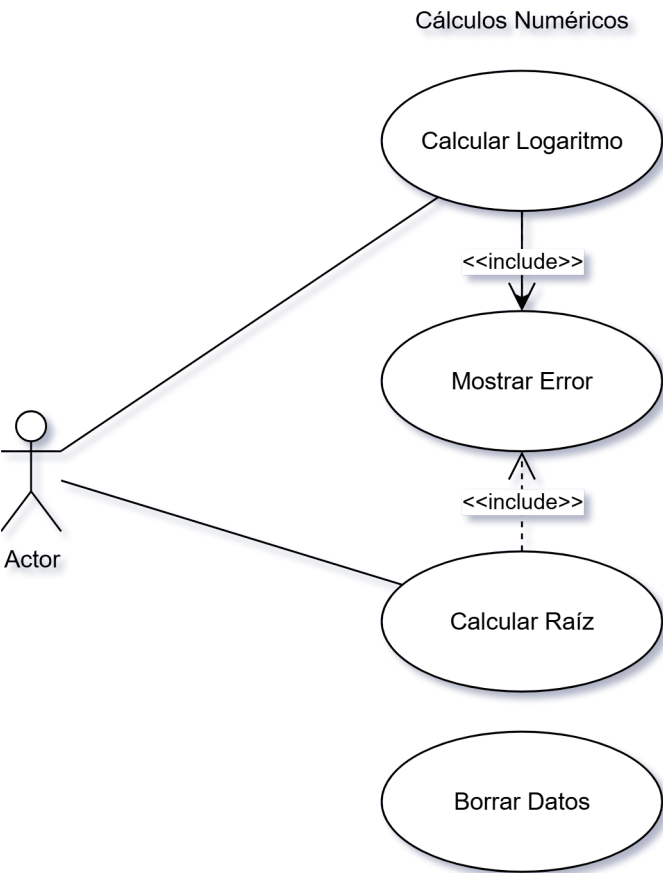


Diagrama de Clases

Principal

+main()

crea

Ventana

+«constructor»Ventana()

+calcular_log()

+calcular raiz cuadrada()

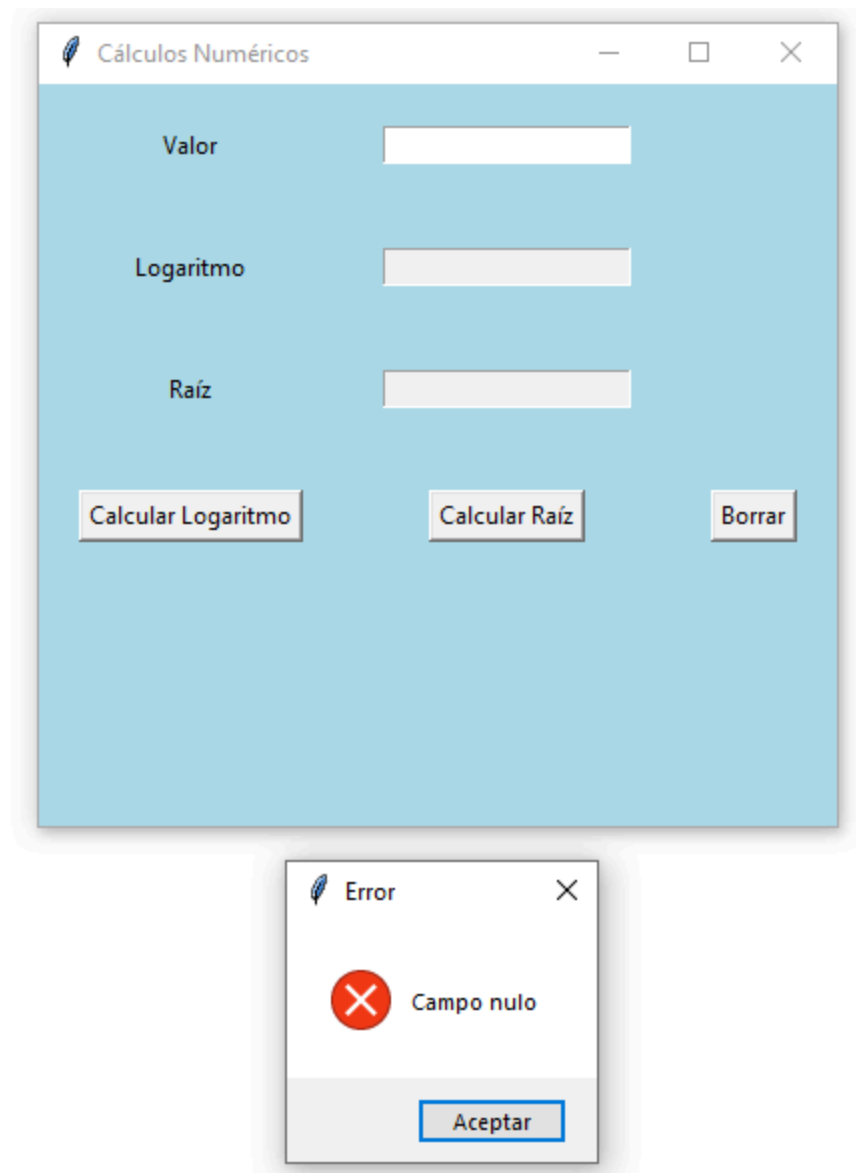
```
calcular_raiz_cuadrada()  
+limpiar()
```

Solución

[Click para ver código fuente](#)

Ejecución del programa

Sin ingresar un valor



Ingresando un valor negativo

Cálculos Numéricos

Valor

Logaritmo

Raíz

Error: el valor debe ser un número positivo para calcular el logaritmo

Error: el valor debe ser un número positivo para calcular la raíz cuadrada

Ingresando 0 como valor

Cálculos Numéricos

Valor

Logaritmo

Raíz

Error: El valor debe ser distinto de 0 para calcular el logaritmo

Ingresando un valor no válido

Cálculos Numéricos

Valor

Logaritmo

Raíz

Error: el valor debe ser numérico para calcular el logaritmo

Error: el valor debe ser numérico para calcular la raíz cuadrada

Ingresando un valor válido

Cálculos Numéricos

Valor

Logaritmo

Raíz

Ejercicio 6.7. Validación de campos (p. 417)

Las excepciones se pueden utilizar para validar el formato de campos de entrada. Por ejemplo:

- Restringir que un campo de entrada de datos sea numérico, alfabético, alfanumérico, etc.
- Restringir la cantidad de caracteres que debe tener un campo de entrada de datos.
- Impedir que se ingresen fechas anteriores a la fecha actual.
- Definir campos obligatorios en un formulario de tal manera que no reciba valores vacíos.
- Definir restricciones especiales de los campos de entrada. Por ejemplo, si es un correo electrónico, que tenga el carácter @.

Para estos casos, si se presentan situaciones que no cumplen dichas condiciones se debe generar la excepción correspondiente.

Objetivo de aprendizaje

Al finalizar este ejercicio, el lector tendrá la capacidad para definir métodos que realicen validación de campos y generen las excepciones apropiadas en caso de que no se cumplan las condiciones estipuladas en los requisitos de un programa.

Enunciado: clase EquipoMaratónProgramación

Un equipo de programadores desea participar en una maratón de programación. El equipo tiene los siguientes atributos:

- Nombre del equipo (tipo String).
- Universidad que está representando el equipo (tipo String).
- Lenguaje de programación que va a utilizar el equipo en la competencia (tipo String).
- Tamaño del equipo (tipo int).

Se requiere un constructor que inicialice los atributos del equipo. El equipo está conformado por varios programadores, mínimo dos y máximo tres. Cada programador posee nombre y apellidos (de tipo String). Se requieren además los siguientes métodos:

- Un método para determinar si el equipo está completo.
- Un método para añadir programadores al equipo. Si el equipo está lleno se debe imprimir la excepción correspondiente.
- Un método para validar los atributos nombre y apellidos de un programador para que reciban datos que sean solo texto. Si se reciben datos numéricos se debe generar la excepción

correspondiente. Además, no se permiten que los campos String tengan una longitud igual o superior a 20 caracteres.

- En un método main se debe crear un equipo solicitando sus datos por teclado y se validan los nombres y apellidos de los programadores.

Diagrama de Casos de uso

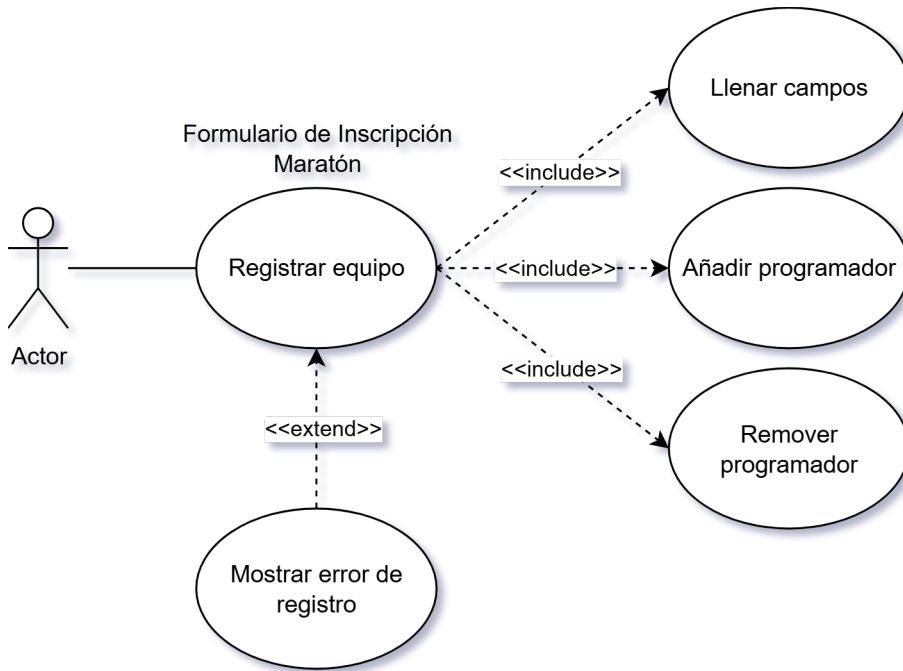
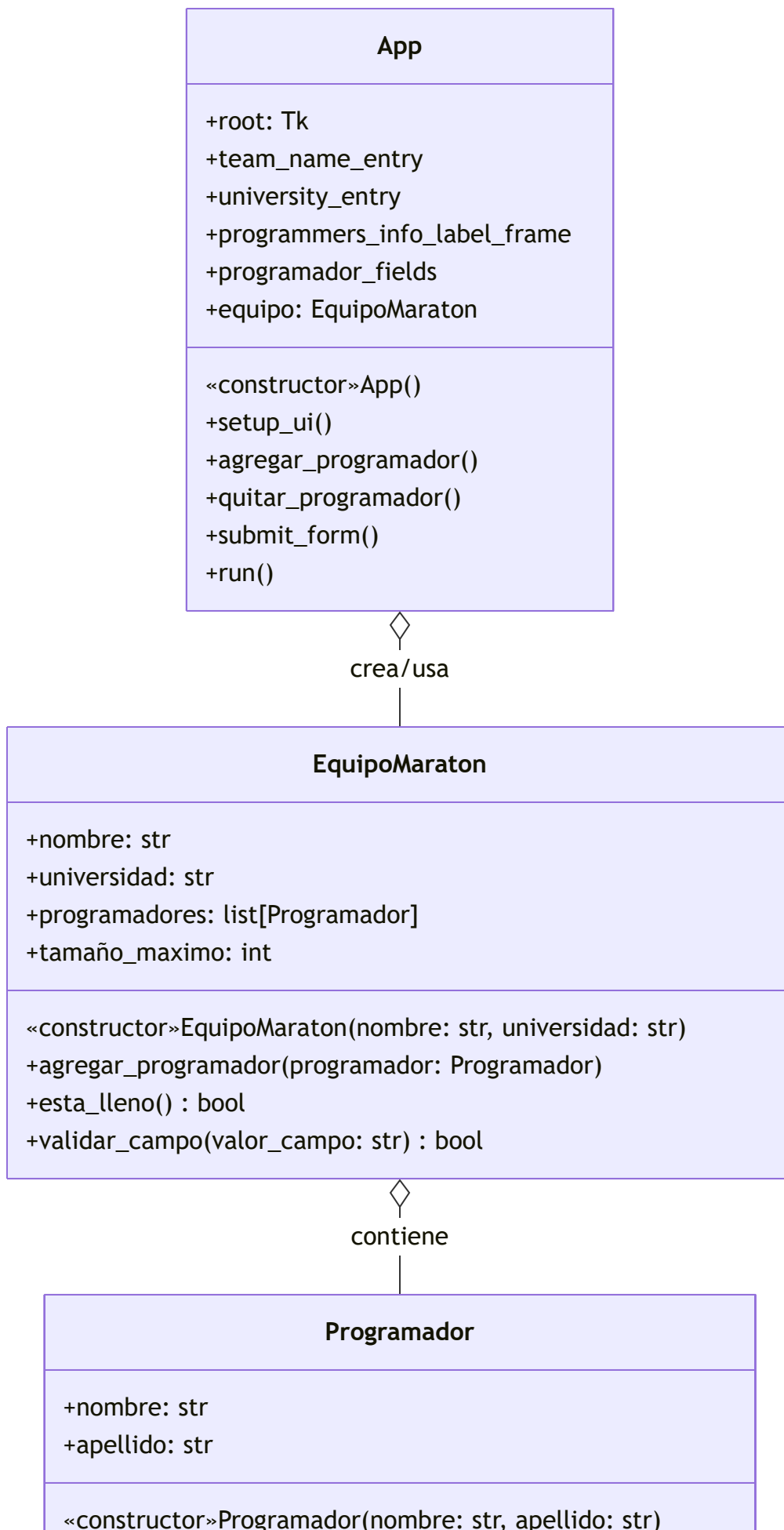


Diagrama de Clases



Solución

[Click para ver código fuente](#)

Ejecución del programa

Cuando no se llena la información del equipo

Formulario de Inscripción - Maratón de Programación

Información del Equipo

Nombre del Equipo

Universidad

Universidad Nacional de Colombia (UNAL)

Información de los integrantes

Programador 1

Nombre

Apellido

Programador 2

Nombre

Apellido

Programador 3

Nombre

Apellido

Acciones

+

-

Enviar

Error

Por favor, complete la información del equipo.

Aceptar

Cuando se deja vacío alguno de los campos de los integrantes

Información del Equipo

Nombre del Equipo

Fsociety

Universidad

Universidad Nacional de Colombia (UNAL) ▾

Información de los integrantes

Programador 1

Nombre

Apellido

Programador 2

Nombre

Apellido

Programador 3

Nombre

Apellido

Acciones

+

-

Enviar



Error



El nombre "" no es válido. No puede ser vacío, debe contener menos de 20 caracteres y no debe tener números.

Aceptar

Cuando alguno de los campos de los integrantes contiene dígitos

Información del Equipo

Nombre del Equipo

Fsociety

Universidad

Universidad Nacional de Colombia (UNAL) ▾

Información de los integrantes

Programador 1

Nombre

123

Apellido

Perez

Programador 2

Nombre

Elliot

Apellido

Alderson

Programador 3

Nombre

Darlene

Apellido

Alderson

Acciones

+

-

Enviar

Error



El nombre '123' no es válido. No puede ser vacío, debe contener menos de 20 caracteres y no debe tener números.

Aceptar

Cuando alguno de los campos de los integrantes tiene mas de 19 caracteres

Información del Equipo

Nombre del Equipo

Fsociety

Universidad

Universidad Nacional de Colombia (UNAL)

Información de los integrantes

Programador 1

Nombre

Apellido

Otorrinolaringologia

Perez

Programador 2

Nombre

Apellido

Elliot

Alderson

Programador 3

Nombre

Apellido

Darlene

Alderson

Acciones

+

-

Enviar

Error



El nombre 'Otorrinolaringologia' no es válido. No puede ser vacío, debe contener menos de 20 caracteres y no debe tener números.

Aceptar

Cuando la ejecucion es correcta

Información del Equipo

Nombre del Equipo

Universidad

Información de los integrantes

Programador 1

Nombre

Apellido

Programador 2

Nombre

Apellido


Programador 3

Nombre

Apellido

Acciones

Éxito

 Equipo 'Fsociety' de la Universidad Nacional de Colombia (UNAL) inscrito correctamente con 3 programadores.

Ejercicio 6.8. Lectura de archivos (p. 425)

Al finalizar este ejercicio, el lector tendrá la capacidad para:

- Crear un flujo de bytes para leer archivos de texto.
- Conocer y aplicar las clases `InputStreamReader` y `BufferedReader` para la creación del flujo de bytes que facilita la lectura de archivos.

Enunciado: clase LeerArchivo

Se tiene un archivo de texto denominado prueba.txt en una cierta localización en un sistema de archivos. Se requiere desarrollar un programa que lea dicho archivo de texto utilizando un flujo de bytes que muestre los contenidos del archivo en pantalla.

Diagrama de Casos de uso

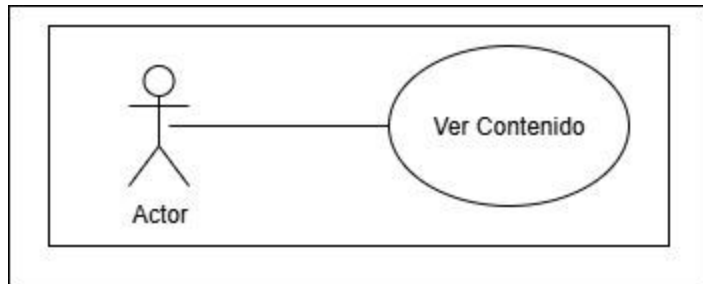
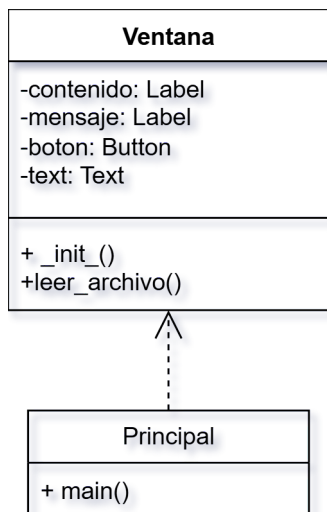


Diagrama de Clases

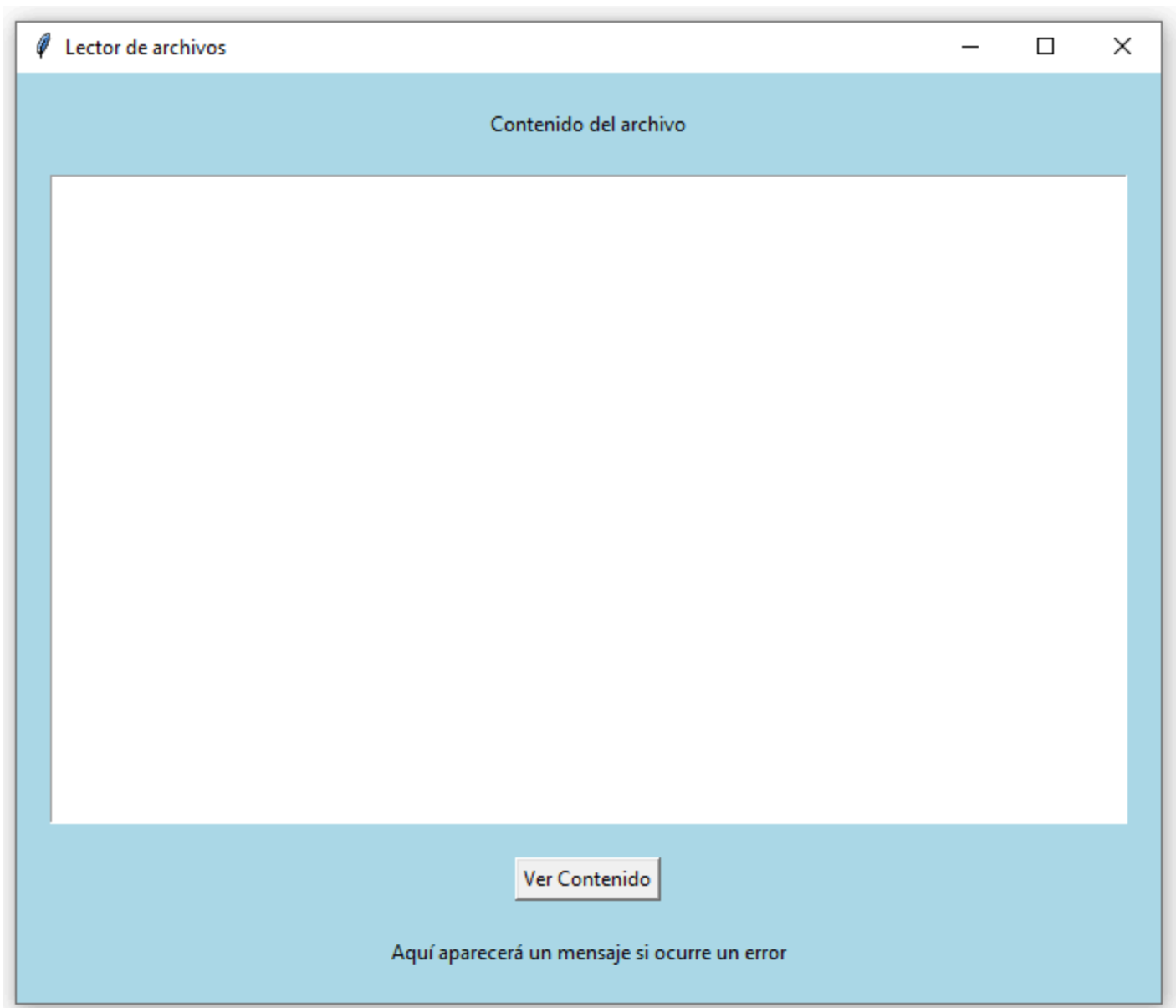


Solución

[Click para ver código fuente](#)

Ejecución del programa

Interfaz gráfica



Cuando el archivo de prueba está en la misma ubicación del trabajo con el código



Cuando el archivo de prueba NO está en la misma ubicación del trabajo con el código

Contenido del archivo



Ver Contenido

No se pudo leer el archivo