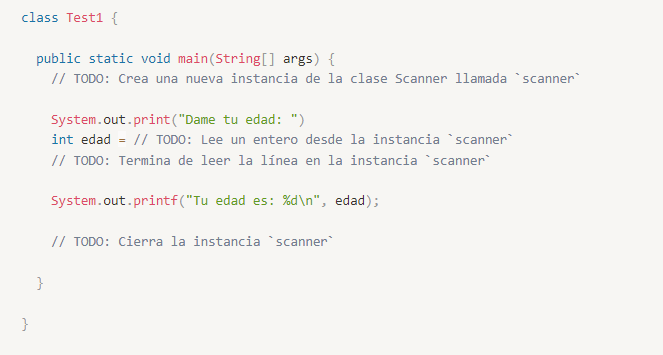
Examen Final

**Problema 1** - Uso del **Scanner**

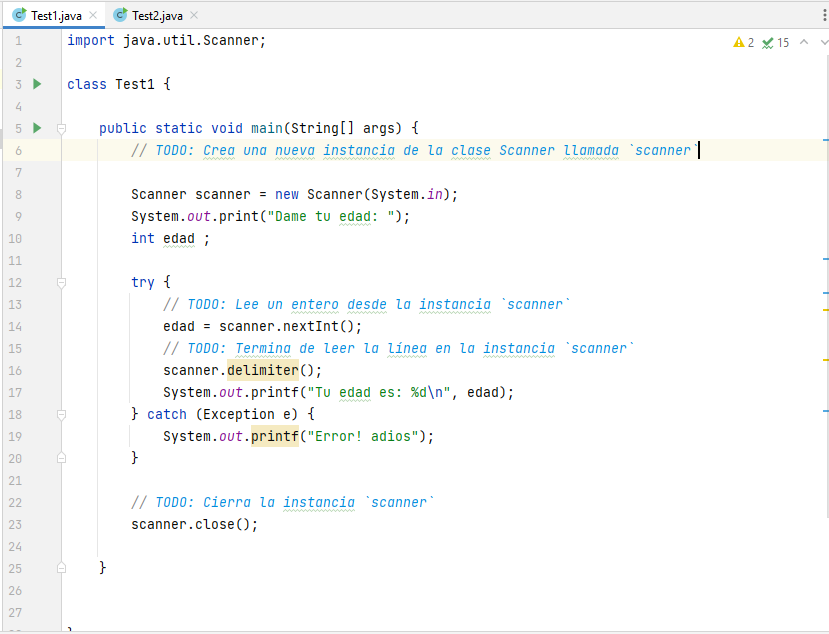


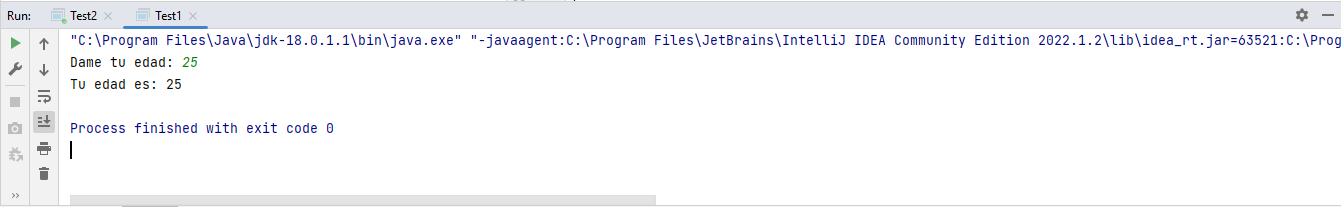
**¿Qué error hay en el código?**

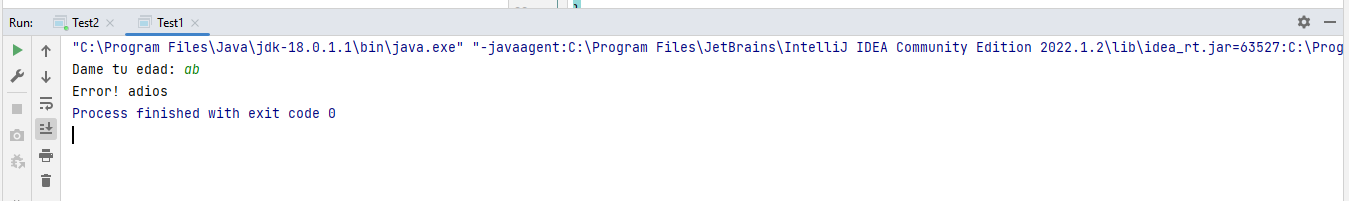
a) Falta un punto y coma en la primer impresión a consola

b) Falta marcar como public la clase Test

c) No hay ningún error en el código







**Problema 2** - Crear una **Ventana** con **Swing**

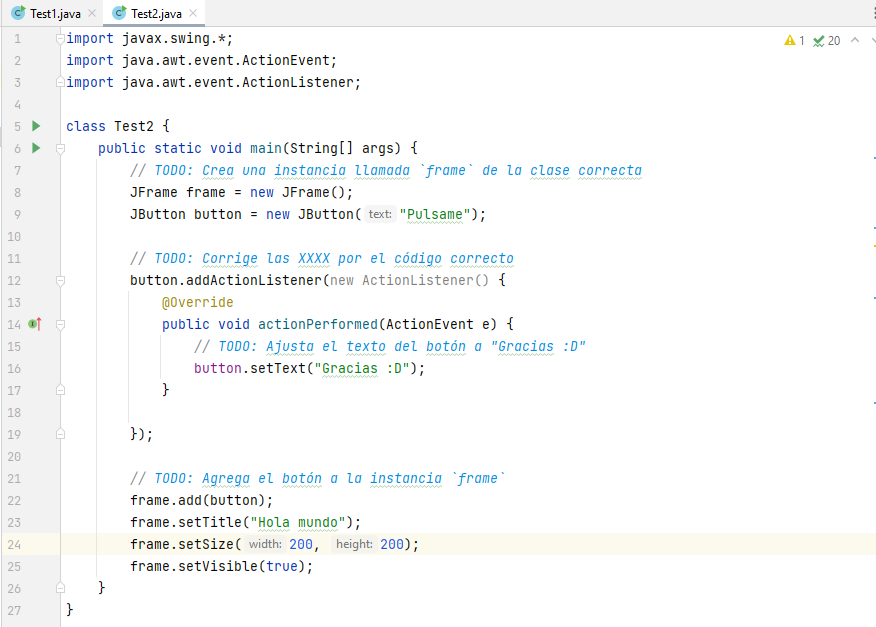


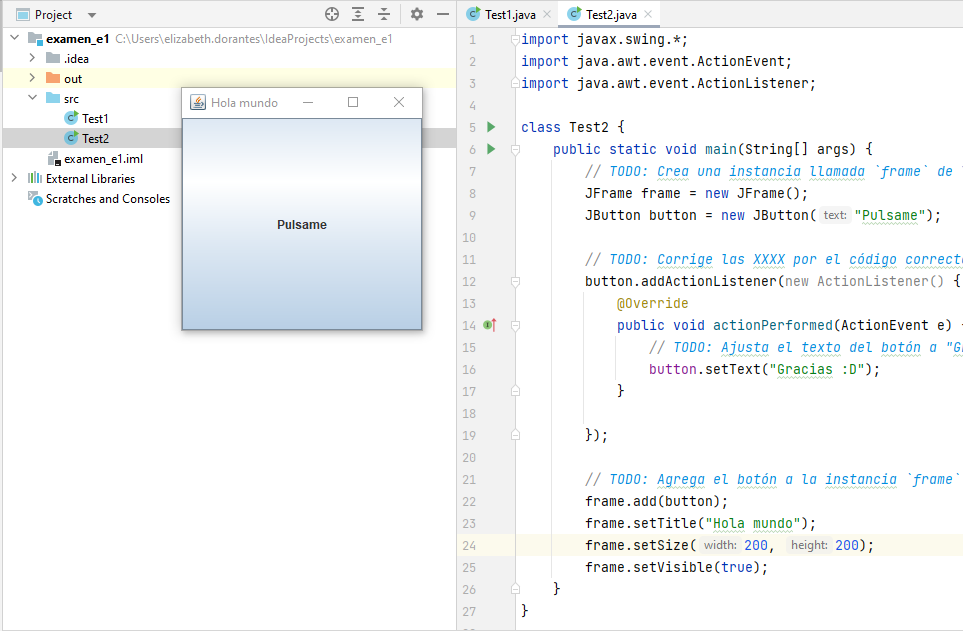
**¿Qué faltó para que se mostrara la ventana?**

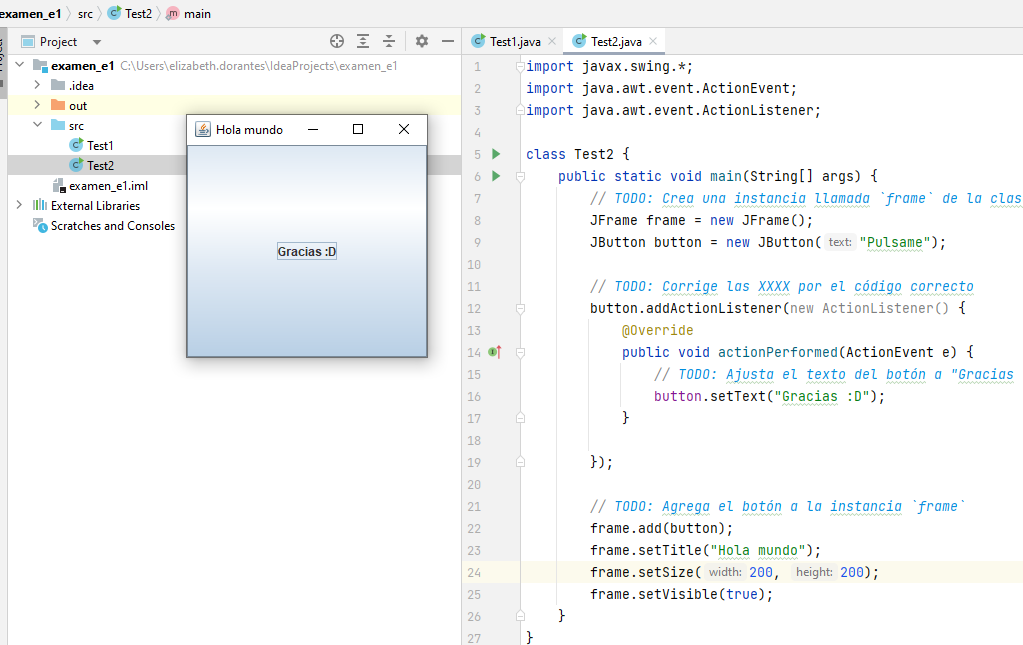
a) No faltó nada, la ventana se mostró correctamente

b) Faltó llamar al método show()

c) Faltó llamar al método setVisible(true)







**Problema 3** - **Conectarse** a una Base de Datos **MySQL**

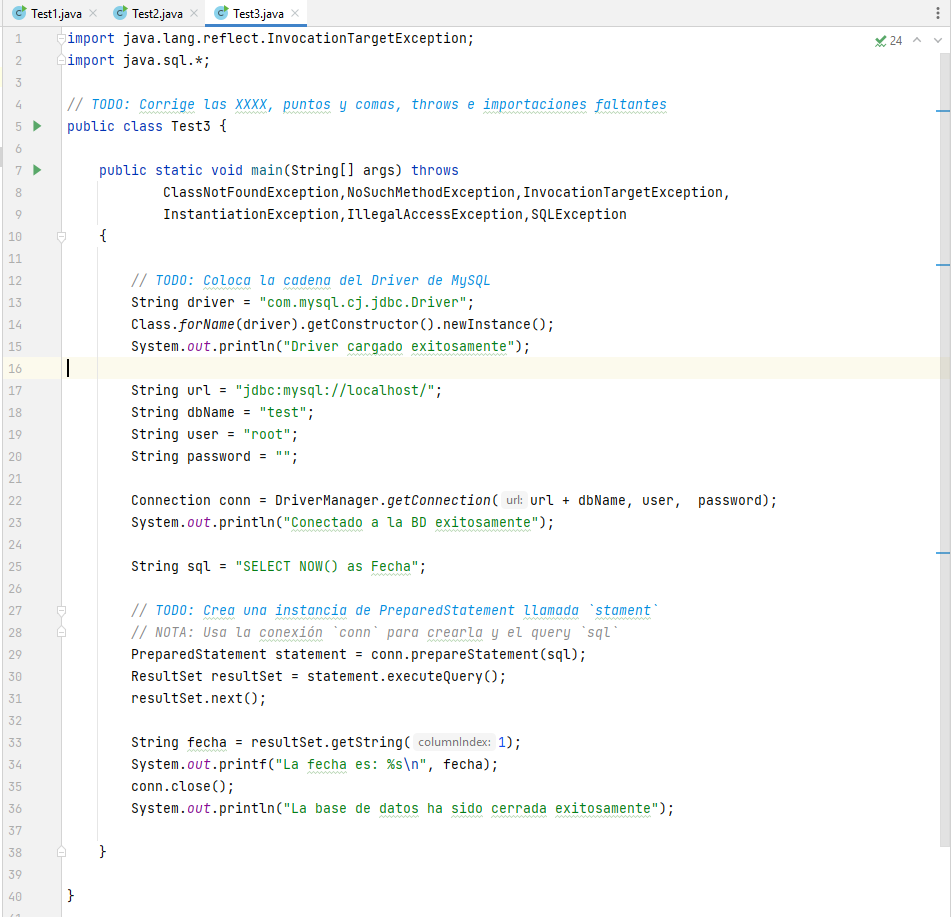


**¿Cómo corregiste obtener la fecha del resultSet.getString(...)?**

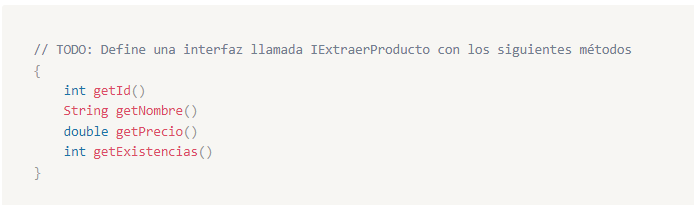
a) No tuve que corregirlo, funcionó correctamente

b) Sólo tuve que cambiar el 0 por 1

c) Sólo tuve que cambiar el 0 por "Fecha"



**Problema 4** - Diseña la **Interfaz** del **Modelo *IExtraerProducto***

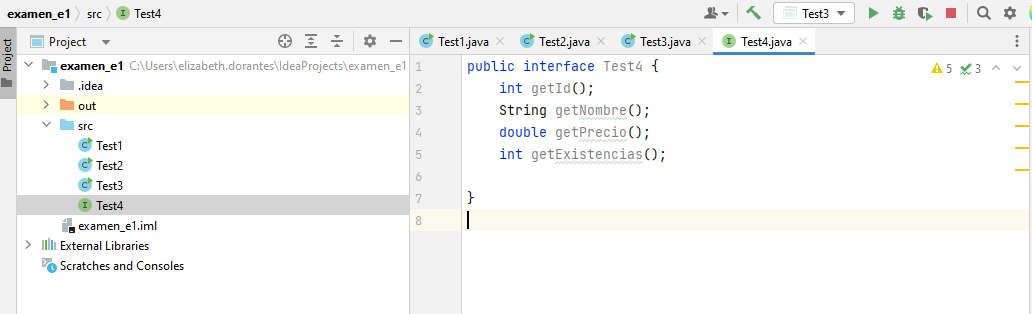


**¿Cuál sería la mejor explicación de lo que es una interface?**

a) Se utilizan para modelar estereotipos que abstraen la funcionalidad, y esta puede ser implementada posteriormente en clases

b) Se utilizan para definir los métodos que tendrá que implementar una clase para que no haya error en el código

c) Se utilizan para agrupar clases por funcionalidad y poder pasarlas como parámetros o usarlas como objetos similares



**Problema 5** - Uso de ***InputStream*** y ***OutputStream***

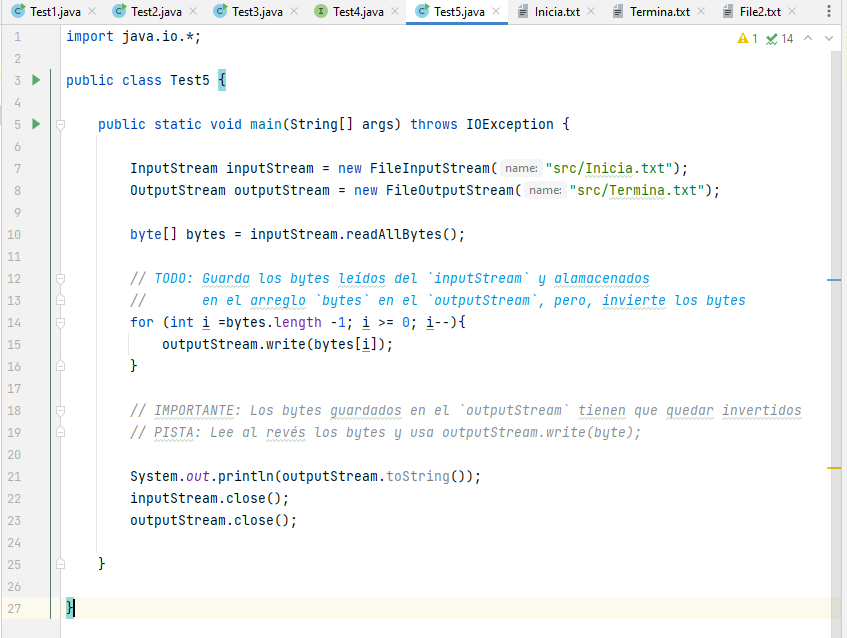


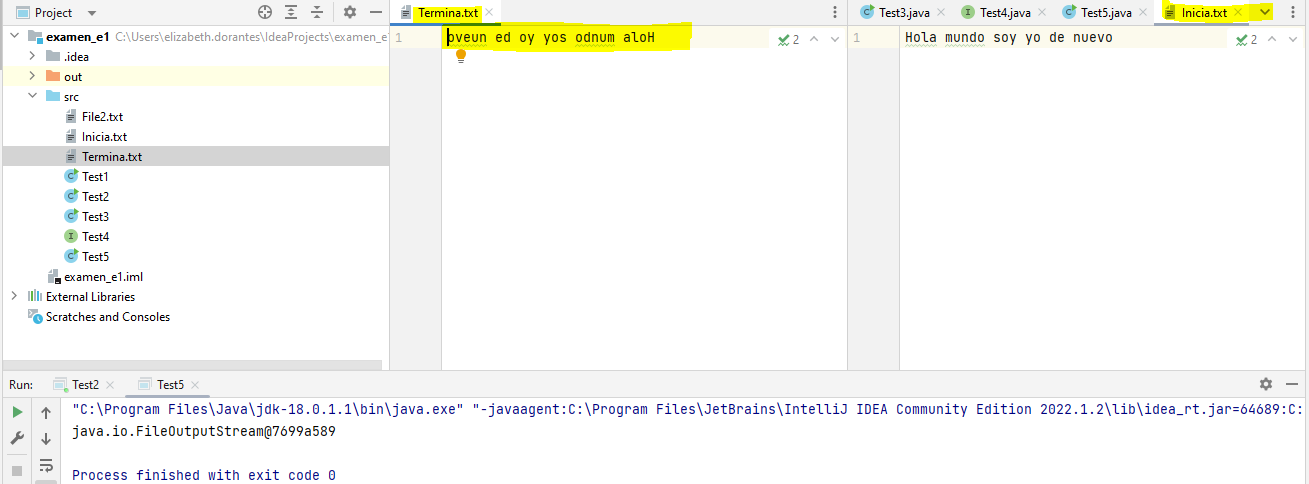
**¿Para qué sirven InputStream y OutputStream?**

a) Son dos clases que nos permiten leer y escribir bytes en la fuente conectada, la cuál puede ser un archivo, un socket o algún otro

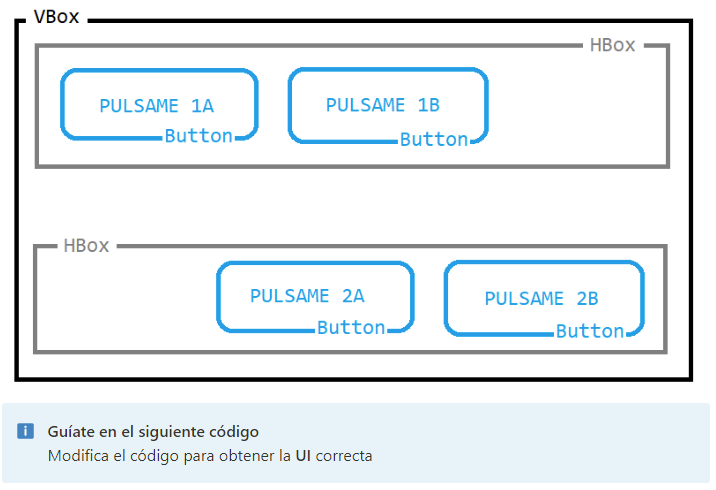
b) Son dos clases que nos permiten leer y escribir bytes de forma temporal sobre archivos, sockets o algunos otros, sin modificar los flujos originales

c) Son dos clases que nos permiten leer y escribir bytes directamente en la memoria RAM para almacenar variables en forma de bytes





**Problema 6** - Diseña la siguiente **UI** en **FXML** para **JavaFX**





**¿Qué mejoras tiene *JavaFx* respecto a *Swing*?**

a) *JavaFx* separa la *Vista* mediante archivos *FXML* de forma nativa, a diferencia de *Swing* que tiene que definir sus componentes directamente en el código

b) *JavaFx* separa el *Controlador* mediante una clase tradicional de *Java* y utiliza los *JavaBeans* para conectar eventos que surgen desde la interfaz, a diferencia de *Swing*

c) No hay diferencias, ambos se pueden interoperar

