

**PROGRAMACION ORIENTADA A OBJETOS II**

**SEMANA 4**

Jorge Peñaloza Guerra

Lunes 3 de julio de 2023

ingeniería en informática

# **DESARROLLO**

Contexto

Un club náutico Victorias CA., posee un sistema informático donde lleva todos los datos correspondientes a sus instalaciones, empleados, socios y embarcaciones que se encuentran en dicho club. El sistema está presentando una falla, ya que no cuenta con los niveles de seguridad por usuario y se quiere controlar los que pueden ingresar al sistema con sus respectivas claves. Éstas, tienen que ser numéricas y en el caso que no lo sean, el sistema realice el manejo de esa excepción para que no se visualice el error que arroja Java y no exista una caída de este. A continuación, responde y realiza las actividades siguientes:

A continuación, responde las preguntas y realiza la siguiente actividad:

1. ¿Qué tipo de excepción utilizarás para resolver el caso planeado y así evitar errores y caídas del sistema? Justifica tu respuesta.

El caso menciona que el club náutico Victorias CA. tiene un sistema informático que registra datos relacionados con sus instalaciones, empleados, socios y embarcaciones. Sin embargo, el sistema presenta una falla, ya que no cuenta con niveles de seguridad por usuario y se desea controlar el acceso al sistema mediante claves numéricas. Si la clave ingresada no cumple con este requisito, se debe manejar la excepción correspondiente para evitar mostrar el error de Java y prevenir una caída del sistema.

* 1. Para solucionar esta situación, se propone implementar un mecanismo de validación de las claves ingresadas por los usuarios. A continuación se detalla el enfoque a seguir:
     1. Análisis del caso: Se identifica que el sistema necesita incorporar niveles de seguridad por usuario y que las claves deben ser numéricas. Además, se destaca la importancia de manejar adecuadamente las excepciones para evitar errores visibles por parte de los usuarios.
     2. Diseño de la solución: Se plantea implementar una funcionalidad en el sistema que valide las claves ingresadas por los usuarios. Para ello, se utilizará una estructura de control que verifique si la clave cumple con el formato numérico requerido. En caso contrario, se lanzará una excepción personalizada que capturará el error y mostrará un mensaje adecuado al usuario.
     3. Implementación del código: Se agrega la validación de las claves en el código del sistema. Cuando un usuario intente ingresar una clave no numérica, se lanzará una excepción personalizada, específicamente diseñada para manejar este tipo de error. La excepción capturará el error y mostrará un mensaje indicando que la clave debe ser numérica.
     4. Pruebas y validación: Se llevan a cabo pruebas exhaustivas para verificar que la validación de las claves y el manejo de excepciones funcionen correctamente. Se asegura que el sistema acepte correctamente las claves numéricas y que se muestre el mensaje de error apropiado cuando se ingrese una clave no numérica.

1. ¿De qué manera utilizarás el manejo, devolución, captura y omisión de excepciones en clases y métodos en el caso planteado? Explica.

De la clase UserCredentials se tiene el constructor en se asigna un mensaje a la excepción de conversión de tipo de datos cuando ocurre una excepción de conversión de tipo de dato y permite delegar el maneja de esta vista en la figura 1. Además, en a la figura 2 se puede como se gestiona la excepción al crear la clase con un tipo de dato diferente del solicitado. En la figura 3 se puede ver como se omite una excepción que se produce cuando la clase no ha sido correctamente construida en la figura 2, pero como se manejó la excepción el programa sigue funcionando y producto de eso se produce una segunda excepción por que el método de obtención de password devuelve int, pero password tiene NULL en estos momentos.

Figura 1

Constructor con delegación de excepción

Texto

Descripción generada automáticamente

Fuente: Elaboración propia

Figura 2

Manejo de excepción

Texto

Descripción generada automáticamente

Fuente: Elaboración propia

Figura 3

Omisión de excepción

Texto

Descripción generada automáticamente

Fuente: Elaboración propia

1. Desarrolla el código en Java donde se maneje la excepción para evitar los errores y caída del sistema de la tarea planteada.

En código Java 1, 2 y 3 se observan lasa clases programas en java para comprobar la funcionalidad. Mientras que en código fxml se presenta la estructura de la venta diseñada para JavaFX. En la figura 4, 5, 6 y 7 se pueden observar las ventana de control de acceso, manejo de excepción, mensaje indicando error de acceso y finalmente se puede observar un mensaje que valida el acceso del usuario.

Código Java 1

Clase: LoginAplication.java

package cl.evolutec.tareas4v2;  
  
import javafx.application.Application;  
import javafx.fxml.FXMLLoader;  
import javafx.scene.Scene;  
import javafx.stage.Stage;  
  
import java.io.IOException;  
  
public class LoginApplication extends Application  
{  
 @Override  
 public void start(Stage stage) throws IOException  
 {  
 FXMLLoader fxmlLoader = new FXMLLoader(LoginApplication.class.getResource("VentanaLogin.fxml"));  
 Scene scene = new Scene(fxmlLoader.load(), 660, 340);  
 stage.setTitle("Hello!");  
 stage.setScene(scene);  
 stage.show();  
 }  
  
 public static void main(String[] args)  
 {  
 *launch*();  
 }  
}

Fuente: Elaboración propia

Código Java 2

Clase: LoginController.java

package cl.evolutec.tareas4v2;  
import javafx.application.Platform;  
import javafx.event.ActionEvent;  
import javafx.fxml.FXML;  
import javafx.scene.Node;  
import javafx.scene.control.Button;  
import javafx.scene.control.PasswordField;  
import javafx.scene.control.TextField;  
import javafx.stage.Stage;  
import javafx.scene.control.Alert;  
  
public class LoginController {  
  
 @FXML  
 private Button btnAceptar;  
  
 @FXML  
 private Button btnCancelar;  
  
 @FXML  
 private PasswordField txtPass;  
  
 @FXML  
 private TextField txtUser;  
  
 @FXML  
 void terminarPrograma(ActionEvent event) {  
 // Cierra la ventana actual  
 Stage stage = (Stage) ((Node) event.getSource()).getScene().getWindow();  
 stage.close();  
  
 // Cierra la aplicación JavaFX  
 Platform.*exit*();  
 System.*exit*(0);  
 }  
  
 @FXML  
 void verificarLogin(ActionEvent event) {  
 String userN1 = "jorge";  
 int passN1 = 1234;  
  
 String userN2 = "ricardo";  
 int passN2 = 1234;  
  
 String txtuser = txtUser.getText();  
 String txtpass = txtPass.getText();  
  
  
 UserCredentials userCredentials = null;  
 try {  
 userCredentials = new UserCredentials(txtuser, txtpass);  
 } catch (NumberFormatException e) {  
 showAlert("Error", "Entrada inválida",  
 e.getMessage());  
  
 }  
  
 //Obtener datos de clase UserCredentials  
 String user = "";  
 int password = 0;  
 try {  
 user = userCredentials.getUsername();  
 password = userCredentials.getPassword();  
 } catch (NullPointerException e){  
 //System.out.println(e.getMessage());  
 }  
  
  
 if(txtuser.equals(userN1) && password == passN1){  
 showMessage("Acceso", "Acceso comprobado",  
 "Presione aceptar para continuar a Nivel 1");  
 }  
 else {  
 if(txtuser.equals(userN2) && password == passN2){  
 showMessage("Acceso", "Acceso comprobado",  
 "Presione aceptar para continuar a Nivel 2");  
 }  
 else {  
 showMessage("Acceso", "Acceso invalido",  
 "Presione aceptar para seguir intentando");  
 }  
 }  
  
  
 System.*out*.println("Usuario: " + txtuser);  
 System.*out*.println("Contraseña: " + txtpass);  
  
 txtUser.setText("");  
 txtPass.setText("");  
 }  
 private void showAlert(String title, String header, String content) {  
 Alert alert = new Alert(Alert.AlertType.*ERROR*);  
 alert.setTitle(title);  
 alert.setHeaderText(header);  
 alert.setContentText(content);  
 alert.showAndWait();  
 }  
  
 private void showMessage(String title, String header, String content) {  
 Alert alert = new Alert(Alert.AlertType.*INFORMATION*);  
 alert.setTitle(title);  
 alert.setHeaderText(header);  
 alert.setContentText(content);  
 alert.showAndWait();  
 }  
  
}

Fuente: Elaboración propia

Código Java 3

Clase: UserCredentials.java

package cl.evolutec.tareas4v2;  
  
public class UserCredentials {  
  
 private String username;  
 private int password;  
  
 public UserCredentials(String username, String password) throws NumberFormatException{  
 this.username = username;  
 try {  
 this.password = Integer.*parseInt*(password);  
 } catch (NumberFormatException e) {  
 throw new NumberFormatException("La contraseña debe ser numérica.");  
 }  
 }  
  
 public String getUsername() {  
 return username;  
 }  
 public int getPassword() {  
 return password;  
 }  
}

Fuente: Elaboración propia.

Código fxml 1

Clase: VentanaLogin.fxml

<?xml version="1.0" encoding="UTF-8"?>  
  
<?import javafx.scene.control.\*?>  
<?import javafx.scene.layout.\*?>  
<?import javafx.scene.text.\*?>  
  
<VBox maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="312.0" prefWidth="630.0" xmlns="http://javafx.com/javafx/17.0.2-ea" xmlns:fx="http://javafx.com/fxml/1" fx:controller="cl.evolutec.tareas4v2.LoginController">  
 <children>  
 <Label alignment="CENTER" prefHeight="78.0" prefWidth="610.0" text="Control de acceso">  
 <font>  
 <Font size="36.0" />  
 </font>  
 </Label>  
 <GridPane prefHeight="135.0" prefWidth="533.0">  
 <columnConstraints>  
 <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />  
 <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />  
 </columnConstraints>  
 <rowConstraints>  
 <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />  
 <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />  
 </rowConstraints>  
 <children>  
 <Label alignment="CENTER\_RIGHT" prefHeight="51.0" prefWidth="255.0" text="Usuario">  
 <font>  
 <Font size="24.0" />  
 </font>  
 </Label>  
 <Label alignment="CENTER\_RIGHT" prefHeight="51.0" prefWidth="255.0" text="Password" GridPane.rowIndex="1">  
 <font>  
 <Font size="24.0" />  
 </font>  
 </Label>  
 <TextField fx:id="txtUser" maxWidth="-Infinity" prefHeight="51.0" prefWidth="217.0" GridPane.columnIndex="1">  
 <font>  
 <Font size="24.0" />  
 </font>  
 </TextField>  
 <PasswordField fx:id="txtPass" maxWidth="-Infinity" prefHeight="51.0" prefWidth="217.0" GridPane.columnIndex="1" GridPane.rowIndex="1">  
 <font>  
 <Font size="24.0" />  
 </font>  
 </PasswordField>  
 </children>  
 </GridPane>  
 <GridPane alignment="CENTER" prefHeight="73.0" prefWidth="600.0">  
 <columnConstraints>  
 <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />  
 <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />  
 <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />  
 <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />  
 <ColumnConstraints hgrow="SOMETIMES" minWidth="10.0" prefWidth="100.0" />  
 </columnConstraints>  
 <rowConstraints>  
 <RowConstraints minHeight="10.0" prefHeight="30.0" vgrow="SOMETIMES" />  
 </rowConstraints>  
 <children>  
 <Button fx:id="btnAceptar" alignment="CENTER" contentDisplay="CENTER" mnemonicParsing="false" onAction="#verificarLogin" prefHeight="54.0" prefWidth="138.0" text="Aceptar" textAlignment="CENTER" GridPane.columnIndex="1">  
 <font>  
 <Font size="24.0" />  
 </font>  
 </Button>  
 <Button fx:id="btnCancelar" alignment="CENTER" contentDisplay="CENTER" mnemonicParsing="false" onAction="#terminarPrograma" prefHeight="54.0" prefWidth="143.0" text="Cancelar" textAlignment="CENTER" GridPane.columnIndex="3">  
 <font>  
 <Font size="24.0" />  
 </font>  
 </Button>  
 </children>  
 </GridPane>  
 </children>  
</VBox>

Fuente: Elaboración propia

Figura 4

Ventana Login

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Fuente: Elaboración propia

Figura 5

Excepción de tipo de numero controlada

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Fuente: Elaboración propia

Figura 6

Acceso invalido

Interfaz de usuario gráfica, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

Fuente: Elaboración propia

Figura 6

Acceso comprobado

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

Fuente: Elaboración propia

# **REFERENCIAS BIBLIOGRÁFICAS**

IACC. (2022). Control de Excepciones. En *Programación orientada a objetos II. Semana 4.*