Nº refatoração: 1	Classe: ExcluiEstudanteControlller.java	**Nome:** Extract Method
Código antes de refatorar:		Código após refatorar

private void btnConfOnAction(ActionEvent event) {
try {
int estId = Integer.parseInt(txtIDEst.getText());
estudanteDAO.excluir(estId);
Alert alert;
alert = new Alert(AlertType.INFORMATION, "Você clicou no botão Confirmar", ButtonType.OK);
alert.setTitle("Estudante removido com sucesso!");
alert.setHeaderText("Informação");
alert.show();
} catch (SQLException e1) {
Alert alert;
alert = new Alert(AlertType.INFORMATION, "Você clicou no botão Cancelar", ButtonType.OK);
alert.setTitle("Estudante não foi removido com sucesso!");
alert.setHeaderText("Informação");
alert.show();
e1.printStackTrace();
}
}
		private void btnConfOnAction(ActionEvent event) {
		try {
		int estId = Integer.parseInt(txtIDEst.getText());
		estudanteDAO.excluir(estId);
		mostrarAlerta("Estudante removido com sucesso!", "Você clicou no botão Confirmar");
		} catch (SQLException e1) {
		mostrarAlerta("Estudante não foi removido com sucesso!", "Você clicou no botão Cancelar");
		e1.printStackTrace();
		}
		}
		
		private void mostrarAlerta(String titulo, String mensagem) {
		Alert alert = new Alert(AlertType.INFORMATION, mensagem, ButtonType.OK);
		alert.setTitle(titulo);
		alert.setHeaderText("Informação");
		alert.show();
		}	

Nº refatoração: 2	**Classe:** ExcluiEstudanteControlller.java	**Nome:** Replace Temp with Query
Código antes de refatorar:		Código após refatorar:

private void btnConfOnAction(ActionEvent event) {
try {
int estId = Integer.parseInt(txtIDEst.getText());
estudanteDAO.excluir(estId);
mostrarAlerta("Estudante removido com sucesso!", "Você clicou no botão Confirmar");
} catch (SQLException e1) {
mostrarAlerta("Estudante não foi removido com sucesso!", "Você clicou no
		private void btnConfOnAction(ActionEvent event) {
		try {
		
		estudanteDAO.excluir(getEstudanteId());
		mostrarAlerta("Estudante removido com sucesso!", "Você clicou no botão Confirmar");
		} catch (SQLException e1) {
		mostrarAlerta("Estudante não foi removido com sucesso!", "Você clicou no botão Cancelar");

botão Cancelar");
e1.printStackTrace();
}
}
	e1.printStackTrace();
	}
	}
	
	private int getEstudanteId() {
	return Integer.parseInt(txtIDEst.getText());
	}

Nº refatoração: 3	**Classe:** ExcluiEstudanteController.java	**Nome:** Introduce Field
Código antes de refatorar:		Código após refatorar:

private void btnConfOnAction(ActionEvent event) {
try {
int estId = Integer.parseInt(txtIDEst.getText());
estudanteDAO.excluir(estId);
mostrarAlerta("Estudante removido com sucesso!", "Você clicou no botão Confirmar");
} catch (SQLException e1) {
mostrarAlerta("Estudante não foi removido com sucesso!", "Você clicou no botão Cancelar");
e1.printStackTrace();
}
}
		private int estudanteId;
		
		private void btnConfOnAction(ActionEvent event) {
		try {
		estudanteId = Integer.parseInt(txtIDEst.getText());
		estudanteDAO.excluir(estudanteId);
		mostrarAlerta("Estudante removido com sucesso!", "Você clicou no botão Confirmar");
		} catch (SQLException e1) {
		mostrarAlerta("Estudante não foi removido com sucesso!", "Você clicou no botão Cancelar");
		e1.printStackTrace();
		}
		}

Nº refatoração: 4	**Classe:** ExcluiEstudanteController.java	**Nome:** Inline Method
Código antes de refatorar:		Código após refatorar:

private void btnVoltarOnAction(ActionEvent event) {
Stage stageAtual = getStageAtual(event);
stageAtual.close();
}
 	
private Stage getStageAtual(ActionEvent event) {
return (Stage) ((Node) event.getSource()).getScene().getWindow();
}
		private void btnVoltarOnAction(ActionEvent event) {
		((Stage) ((Node) event.getSource()).getScene().getWindow()).close();
		}

<table>
<tr><td>Nº refatoração: 5</td><td>Classe: IncluiEstudanteControlle r.java</td><td>Nome: Extract Method</td></tr>
<tr><td colspan="2">Código antes de refatorar:</td><td>Código após refatorar</td></tr>
<tr><td colspan="2">@FXML
   private void btnConfOnAction(ActionEvent event) {
      Estudante estudante = new Estudante();

estudante.setNome(txtNomEst.getText());

estudante.setIdade(Integer.parseInt(txtIdadEst.getText()));
   …
   }</td><td>@FXML
   private void btnConfOnAction(ActionEvent event) {
      Estudante estudante = criarEstudante();
  …
  private Estudante criarEstudante() {
      Estudante estudante = new Estudante();
      estudante.setNome(txtNomEst.getText());

estudante.setIdade(Integer.parseInt(txtIdadEst.getText()));
      return estudante;
   }</td></tr>
</table>

<table>
<tr><td>Nº refatoração: 6</td><td>Classe: MenusController.java</td><td>Nome: Extract Method</td></tr>
<tr><td colspan="2">Código antes de refatorar:</td><td>Código após refatorar</td></tr>
<tr><td colspan="2">

```
@FXML
    void incluiOnAction(ActionEvent event) {
        try {
            FXMLLoader loader = new
FXMLLoader(getClass().getResource("/exem
plo/curso1/view/IncluiEstudante.fxml"));
            Parent root = loader.load();
            Stage newStage = new Stage();
            Scene newScene = new Scene(root);
            newStage.setScene(newScene);
            newStage.setTitle("Estudante");
            newStage.setResizable(false);
            newStage.show();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

</td><td>

```
   @FXML
    void incluiOnAction(ActionEvent event) {

carregarTela("/exemplo/curso1/view/IncluiEstu
dante.fxml", "Estudante");
    }

private void carregarTela(String fxmlFile,
String titulo) {
        try {
            FXMLLoader loader = new
FXMLLoader(getClass().getResource(fxmlFile)
);
            Parent root = loader.load();
            Stage newStage = new Stage();
            Scene newScene = new Scene(root);
            newStage.setScene(newScene);
            newStage.setTitle(titulo);
            newStage.setResizable(false);
            newStage.show();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
```

</td></tr>
</table>

Nº refatoração: 7	Classe: iProfessor.java	**Nome:** Remove Redundant Modifier
Código antes de refatorar:		Código após refatorar

public interface IProfessor {
public void inserir(Professor professor) throws SQLException;
public void atualizar(Professor professor) throws SQLException;
public List<Professor> buscarTodos() throws SQLException;
public void excluir(int id) throws SQLException;
}
		public interface IProfessor {
		
		void inserir(Professor professor) throws SQLException;
		
		void atualizar(Professor professor) throws SQLException;
		
		List<Professor> buscarTodos() throws SQLException;
		
		void excluir(int id) throws SQLException;
		}

<table>
<tr><td>Nº refatoração: 8</td><td>Classe: MenusController.java</td><td>Nome: Replace Magic Numbers</td></tr>
<tr><td colspan="2">Código antes de refatorar:</td><td>Código após refatorar</td></tr>
<tr><td colspan="2">public class ProfessorDAO implements IProfessor, IConst {
    private String sql;

    public void inserir(Professor professor) throws SQLException {
        sql = "INSERT INTO professor (nome,idade) VALUES (?,?)";
…
}</td><td>public class ProfessorDAO implements IProfessor, IConst {

    private static final String INSERT_SQL = "INSERT INTO professor (nome, idade) VALUES (?, ?)";
…
   public void inserir(Professor professor) throws SQLException {
        try (Connection conexao = getConnection();
             PreparedStatement pstmt = conexao.prepareStatement(INSERT_SQL)) {
            pstmt.setString(1, professor.getNome());
            pstmt.setInt(2, professor.getIdade());
            pstmt.executeUpdate();
        }
    }
}</td></tr>
</table>