2° Trabalho de Programação Orientada à Objetos II - Refatoração

Grupo: Guilherme K. Trajanoski, Jean Rafael G. W. e Matheus Gonzaga.

N° refatoração: 1	**Classe:** DepartamentoController.java	**Nome:** Dead Code Removal
Código antes de refatorar	**Código após refatorar**	

`@FXML`
`private Button excluirButton;`
 	
`@FXML`
`private MenuItem goToDepartamento;`
 	
`@FXML`
`private MenuItem goToDisciplina;`
	…	

N° refatoração: 2	**Classe:** DisciplinaController.java	**Nome:** Dead Code Removal
Código antes de refatorar	**Código após refatorar**	

`@FXML`
`private Button excluirButton;`
 	
`@FXML`
`private MenuItem goToDepartamento;`
 	
`@FXML`
`private MenuItem goToDisciplina;`
 	
`@FXML`
`private MenuItem goToEstudante;`
 	
`@FXML`
`private Button saveButton;`
	…	

N° refatoração: 3	**Classe:** EstudanteController.java	**Nome:** Dead Code Removal
Código antes de refatorar	**Código após refatorar**	

`@FXML`
`private Button editarDisciplinasButton;`
 	
`@FXML`
`private Button excluirButton;`
 	
`@FXML`
`private MenuItem goToDepartamento;`
 	
`@FXML`
`private MenuItem goToDisciplina;`
 	
`@FXML`
`private MenuItem goToEstudante;`
 	
`@FXML`
`private Button saveButton;`
	…	

N° refatoração: 4	**Classe:** EstudanteDisciplinaController.java	**Nome:** Dead Code Removal
Código antes de refatorar	**Código após refatorar**	

`@FXML`
`private Button excluirButton;`
 	
`@FXML`
`private Button saveButton;`
	…	

N° refatoração: 5	**Classe:** DepartamentoController.java	**Nome:** Extract Method

Código antes de refatorar

```
@FXML
public void initialize() {
   departamentoDBDAO = new DepartamentoDBDAO();

   // Configurar as colunas
   colId.setCellValueFactory(new
PropertyValueFactory<>("departamentoId")); // Ensure
the property name matches
   colNome.setCellValueFactory(new
PropertyValueFactory<>("nome")); // Ensure the
property name matches

   // Carregar dados do banco de dados
   try {
       departamentos =
departamentoDBDAO.listaTodos();

       observableDepartamentos =
FXCollections.observableArrayList(departamentos);

tabelaDepartamentos.setItems(observableDepartamentos
);

   } catch (SQLException e) {
       e.printStackTrace();
       Alert alert = new
Alert(Alert.AlertType.ERROR);
       alert.setHeaderText("Erro no banco de
dados.");
       alert.setContentText("Não foi possível
recuperar os dados do banco.");
       alert.showAndWait();
   }
…
}
```

Código após refatorar

```
@FXML
public void initialize() {
   departamentoDBDAO = new DepartamentoDBDAO();


   initTables();
…
}

private void initTables(){
// Configurar as colunas
colId.setCellValueFactory(new
PropertyValueFactory<>("departamentoId"));
colNome.setCellValueFactory(new
PropertyValueFactory<>("nome"));

observableDepartamentos =
FXCollections.observableArrayList(new
ArrayList<>());
tabelaDepartamentos.setItems(observableDepartamentos)
;

atualizarTabelaDepartamentos();

}
```

N° refatoração: 6	**Classe:** DisciplinaController.java	**Nome:** Extract Method

Código antes de refatorar

```
@FXML
    public void initialize() {
        disciplinaDBDAO = new DisciplinaDBDAO();

        // Configurar as colunas
        colId.setCellValueFactory(new
PropertyValueFactory<>("disciplinaId"));
        colNome.setCellValueFactory(new
PropertyValueFactory<>("nome"));

        // Carregar disciplinas do banco de dados
        try {
            disciplinas =
disciplinaDBDAO.listaTodas();

            observableDisciplinas =
FXCollections.observableArrayList(disciplinas);

tabelaDisciplinas.setItems(observableDisciplinas);

        } catch (SQLException e) {
            e.printStackTrace();
            Alert alert = new
Alert(Alert.AlertType.ERROR);
            alert.setHeaderText("Erro no banco de
dados.");
            alert.setContentText("Não foi possível
recuperar os dados (disciplinas) do banco.");
            alert.showAndWait();
        }

…
}
```

Código após refatorar

```
@FXML
public void initialize() {
   disciplinaDBDAO = new DisciplinaDBDAO();

   initTables();
…
}

private void initTables(){
   // Configurar as colunas
   colId.setCellValueFactory(new
PropertyValueFactory<>("disciplinaId"));
   colNome.setCellValueFactory(new
PropertyValueFactory<>("nome"));

   observableDisciplinas =
FXCollections.observableArrayList(new
ArrayList<>());
tabelaDisciplinas.setItems(observableDisciplinas);
   atualizarTabelaDisciplinas();
}
```

N° refatoração: 7	**Classe:** EstudanteController.java	**Nome:** Extract Method

Código antes de refatorar

```
@FXML
public void initialize() {
   estudanteDBDAO = new EstudanteDBDAO();

   // Configurar as colunas
   colId.setCellValueFactory(new
PropertyValueFactory<>("estudanteId")); // Ensure
the property name matches
   colNome.setCellValueFactory(new
PropertyValueFactory<>("nome"));       // Ensure the
property name matches

   // Carregar dados do banco de dados
   try {
       estudantes = estudanteDBDAO.listaTodos();

       // Use ObservableList directly from Estudante
       observableEstudantes =
FXCollections.observableArrayList(estudantes);

tabelaEstudantes.setItems(observableEstudantes);

   } catch (SQLException e) {
       e.printStackTrace();
       Alert alert = new
Alert(Alert.AlertType.ERROR);
       alert.setHeaderText("Erro no banco de
dados.");
       alert.setContentText("Não foi possível
recuperar os dados do banco.");
       alert.showAndWait();
   }

…
```

Código após refatorar

```
@FXML
public void initialize() {
   estudanteDBDAO = new EstudanteDBDAO();

   initTables();
…
```

```
private void initTables(){
   // Configurar as colunas
   colId.setCellValueFactory(new
PropertyValueFactory<>("estudanteId"));
   colNome.setCellValueFactory(new
PropertyValueFactory<>("nome"));

   observableEstudantes =
FXCollections.observableArrayList(new
ArrayList<>());
   atualizarTabelaEstudantes();
}
```

N° refatoração: 8	**Classe:** EstudanteController.java	**Nome:** Move Field

Código antes de refatorar

```
public class EstudanteController extends Controller
{
   private EstudanteDBDAO estudanteDBDAO;
   private List<Estudante> estudantes;
   private ObservableList<Estudante>
observableEstudantes;

…
```

```
private void atualizarTabelaEstudantes() {
   try {
       estudantes = estudanteDBDAO.listaTodos();
       observableEstudantes.setAll(estudantes);

…
```

Código após refatorar

```
public class EstudanteController extends Controller {
   private EstudanteDBDAO estudanteDBDAO;
   private ObservableList<Estudante>
observableEstudantes;

…
```

```
private void atualizarTabelaEstudantes() {
   try {
       List<Estudante> estudantes =
estudanteDBDAO.listaTodos();
       observableEstudantes.setAll(estudantes);

…
```

N° refatoração: 9	**Classe:** DepartamentoController.java	**Nome:** Extract Method

Código antes de refatorar

```
Alert confirmAlert = new
Alert(Alert.AlertType.CONFIRMATION);
confirmAlert.setTitle("Confirmar Atualização");
confirmAlert.setContentText("Departamento já
cadastrado, deseja atualizar os dados deste
departamento?");
```

Código após refatorar

```
public class Controller {

…

boolean getConfirmacao(String titulo, String
mensagem) {
```

```
// Captura a resposta do usuário
if (confirmAlert.showAndWait().get() ==
ButtonType.OK) {
…



Alert confirmAlert = new
Alert(Alert.AlertType.CONFIRMATION);
confirmAlert.setTitle("Confirmar Remoção");
confirmAlert.setContentText("Deseja remover este
departamento?");

   // Aguardar confirmação
if (confirmAlert.showAndWait().get() ==
ButtonType.OK) {
…
```

```
   Alert confirmAlert = new
Alert(Alert.AlertType.CONFIRMATION);
   confirmAlert.setTitle(titulo);
   confirmAlert.setContentText(mensagem);
   return confirmAlert.showAndWait().filter(response
-> response == ButtonType.OK).isPresent();
}

if (getConfirmacao("Confirmar Atualização",
"Departamento já cadastrado, deseja atualizar os
dados deste departamento?")) {
…



if (getConfirmacao("Confirmar Remoção", "Deseja
remover este departamento?")) {
…
```

N° refatoração: 10	**Classe:** DisciplinaController.java	**Nome:** Extract Method

Código antes de refatorar

```
Alert confirmAlert = new
Alert(Alert.AlertType.CONFIRMATION);
confirmAlert.setTitle("Confirmar Remoção");
confirmAlert.setContentText("Deseja remover esta
disciplina?");

if (confirmAlert.showAndWait().get() ==
ButtonType.OK) {
…



Alert confirmAlert = new
Alert(Alert.AlertType.CONFIRMATION);
confirmAlert.setTitle("Confirmar Atualização");
confirmAlert.setContentText("Disciplina já
cadastrada, deseja atualizar os dados desta
disciplina?");

// Captura a resposta do usuário
if (confirmAlert.showAndWait().get() ==
ButtonType.OK) {
…
```

Código após refatorar

```
public class Controller {

…

boolean getConfirmacao(String titulo, String
mensagem) {
   Alert confirmAlert = new
Alert(Alert.AlertType.CONFIRMATION);
   confirmAlert.setTitle(titulo);
   confirmAlert.setContentText(mensagem);
   return confirmAlert.showAndWait().filter(response
-> response == ButtonType.OK).isPresent();
}

if (getConfirmacao("Confirmar Atualização",
"Disciplina já cadastrada, deseja atualizar os dados
desta disciplina?")) {
…



if (getConfirmacao("Confirmar Remoção", "Deseja
remover esta disciplina?")) {
…
```

N° refatoração: 11	**Classe:** EstudanteController.java	**Nome:** Extract Method

Código antes de refatorar

```
Alert confirmAlert = new
Alert(Alert.AlertType.CONFIRMATION);
confirmAlert.setTitle("Confirmar Remoção");
confirmAlert.setContentText("Deseja remover este
estudante?");

if (confirmAlert.showAndWait().get() ==
ButtonType.OK) {
…



Alert confirmAlert = new
Alert(Alert.AlertType.CONFIRMATION);
confirmAlert.setTitle("Confirmar Atualização");
confirmAlert.setContentText("Estudante já
cadastrado, deseja atualizar os dados deste
estudante?");

// Captura a resposta do usuário
if (confirmAlert.showAndWait().get() ==
ButtonType.OK) {
…
```

Código após refatorar

```
public class Controller {

…

boolean getConfirmacao(String titulo, String
mensagem) {
   Alert confirmAlert = new
Alert(Alert.AlertType.CONFIRMATION);
   confirmAlert.setTitle(titulo);
   confirmAlert.setContentText(mensagem);
   return confirmAlert.showAndWait().filter(response
-> response == ButtonType.OK).isPresent();
}

if (getConfirmacao("Confirmar Atualização",
"Estudante já cadastrado, deseja atualizar os dados
deste estudante?")) {
…



if (getConfirmacao("Confirmar Remoção", "Deseja
remover este estudante?")) {
```

	…

N° refatoração: 12	**Classe:** EstudanteDisciplinaController.java	**Nome:** Extract Method

Código antes de refatorar

```
Alert confirmAlert = new
Alert(Alert.AlertType.CONFIRMATION);
confirmAlert.setTitle("Confirmar Atualização");
confirmAlert.setContentText("Disciplina já
cadastrada, deseja atualizar os dados desta
disciplina?");

// Captura a resposta do usuário
if (confirmAlert.showAndWait().get() ==
ButtonType.OK) {
…



Alert confirmAlert = new
Alert(Alert.AlertType.CONFIRMATION);
confirmAlert.setTitle("Confirmar Remoção");
confirmAlert.setContentText("Deseja remover esta
disciplina?");

if (confirmAlert.showAndWait().get() ==
ButtonType.OK) {
…
```

Código após refatorar

```
public class Controller {

…

boolean getConfirmacao(String titulo, String
mensagem) {
   Alert confirmAlert = new
Alert(Alert.AlertType.CONFIRMATION);
   confirmAlert.setTitle(titulo);
   confirmAlert.setContentText(mensagem);
   return confirmAlert.showAndWait().filter(response
-> response == ButtonType.OK).isPresent();
}
```

```
if (getConfirmacao("Confirmar Remoção", "Deseja
remover esta disciplina?")) {
…



if (getConfirmacao("Confirmar Atualização",
"Disciplina já cadastrada, deseja atualizar os dados
desta disciplina?")) {
…
```