Exemplo 1: MVCSimplesSerializado

Modelo:
Neste exemplo, o pacote modelo tem apenas uma classe (Aluno). Cada aluno tem só duas informações no seu estado: o nome (nome) e o número de registro (dre).

Controlador:
O pacote contem apenas uma classe para o controle. A classe só permite a criação de um único objeto controlador, para isso utiliza o padrão de projeto Singleton.

O controlador permite a criação de alunos, e pode informar o nome do aluno que possui um determinado número de registro fornecido. O controlador usa apenas uma "regra de negócio":
– Não pode haver mais de um aluno com o mesmo número de registro.

O controlador mantém na memória (um HashMap) o cadastro dos alunos já criados, e também pode salvar em disco e recuperar do disco os alunos que foram criados em sessões anteriores. Essa persistência é conseguida por meio de serialização do mapa.

O controlador pode lançar dois tipos de exceção:
- se for solicitado que crie um aluno com número de registro já utilizado.
- se for solicitado o nome de um aluno para um número de registro ainda não utilizado.

Vista:
O exemplo mostra uma vista que utiliza uma interface gráfica construída com Swing.

Nota que a vista deve inclui uma variável de instância que referencia uma instância do controlador. Ela traduz as solicitações do usuário em mensagens para os métodos do controlador. A vista também está preparada para capturar as exceções lançadas pelo controlador e traduzi-las em avisos apropriados para o usuário.

A vista já inclui um método main para lançar a aplicação.

Segue abaixo o código completo:

Pacote modelo:

```
package modelo;

public class Aluno implements java.io.Serializable{

	private static final long serialVersionUID = 1L;
	private String dre;
	private String nome;

	public Aluno(String dre, String nome) {
		super();
		this.dre = dre;
```

```
		this.nome = nome;
	}
	public String getDre() {
		return dre;
	}
	public void setDre(String dre) {
		this.dre = dre;
	}
	public String getNome() {
		return nome;
	}
	public void setNome(String nome) {
		this.nome = nome;
	}
}
```

Pacote controle:

```
package controlador;

import java.util.Collection;
import java.util.HashMap;
import java.io.*;
import modelo.Aluno;

public class ControladorAlunoSerializado {

	public static ControladorAlunoSerializado controladorAlunoSerializado = null;

	public static ControladorAlunoSerializado getControladorAlunoSerializado(){
		if (controladorAlunoSerializado==null)controladorAlunoSerializado = new
ControladorAlunoSerializado();
		return controladorAlunoSerializado;
	}

	HashMap<String, Aluno> alunos;

	private ControladorAlunoSerializado(){
		alunos = new HashMap<String, Aluno>();
	}


	public Aluno getAluno(String dre) throws AlunoInexistenteException{
		if (alunos.get(dre)== null)throw new AlunoInexistenteException(dre);
		else return alunos.get(dre);
	}

	public void criaAluno(String dre, String nome)throws DreDuplicadoException{
		if(alunos.get(dre)== null){
			alunos.put(dre, new Aluno(dre, nome));
		}
		else throw new DreDuplicadoException();
	}

	public Collection<Aluno> getAlunos(){
		return alunos.values();
	}

	public void salvarAlunos()throws IOException{
			FileOutputStream fos = new FileOutputStream("alunos.ser");
			ObjectOutputStream oos = new ObjectOutputStream(fos);
			oos.writeObject(alunos);
			oos.close();
	}

	public void recuperarAlunos() throws IOException, ClassNotFoundException{
		ObjectInputStream ois =
			new ObjectInputStream(new FileInputStream("alunos.ser"));
		Object objeto = ois.readObject();
```

```
            alunos = (HashMap<String, Aluno>)objeto;
        }

        public void limparDados(){
            // destroi todos os dados da memória
            alunos = new HashMap<String, Aluno>();
        }
}
```

Pacote vista

```
package vista;

import java.awt.BorderLayout;
import java.awt.FlowLayout;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;
import java.io.IOException;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;

import controlador.AlunoInexistenteException;
import controlador.ControladorAlunoSerializado;
import controlador.DreDuplicadoException;

public class GuiSimplesSalvando {
        private String dre, nome;
        private JFrame janela;
        private JPanel painelGeral, pCentro, pDisplay, pDre, pNome, pBotoes, pMensagem;
        private JLabel labelDre, labelNome, labelMensagem;
        private JButton botCriar, botObterNome, botSalvar, botLimpar;
        private JTextField tfDre, tfNome, tfMensagem;
        private ControladorAlunoSerializado controlador =
            ControladorAlunoSerializado.getControladorAlunoSerializado();

        public GuiSimplesSalvando() {
            janela = new JFrame("GUI Simples Persistente");
            painelGeral = new JPanel(new BorderLayout());
            pCentro = new JPanel(new BorderLayout());

            pDisplay = new JPanel(new GridLayout(2,1));
            pDre = new JPanel();
            pNome = new JPanel();
            pDre.setLayout(new FlowLayout(FlowLayout.LEFT));
            pNome.setLayout(new FlowLayout(FlowLayout.LEFT));
            pDisplay.add(pDre);
            pDisplay.add(pNome);
            labelDre = new JLabel("DRE: ");
            pDre.add(labelDre);
            tfDre = new JTextField(10);
            tfDre.addKeyListener(new OuvinteTfDre());
            pDre.add(tfDre);
            labelNome = new JLabel("Nome: ");
            pNome.add(labelNome);
            tfNome = new JTextField(30);
            pNome.add(tfNome);
            pBotoes = new JPanel();

            botCriar = new JButton("Criar Aluno");
            botCriar.addActionListener(new OuvinteCriar());
            pBotoes.add(botCriar);

            botObterNome = new JButton("Obter nome");
```

```
		botObterNome.addActionListener(new OuvinteObterNome());
		pBotoes.add(botObterNome);

		botSalvar = new JButton("Salvar dados");
		botSalvar.addActionListener(new OuvinteSalvar());
		pBotoes.add(botSalvar);

		botLimpar = new JButton("Limpar dados");
		botLimpar.addActionListener(new OuvinteLimpar());
		pBotoes.add(botLimpar);

		pMensagem = new JPanel();
		labelMensagem = new JLabel("Mensagem: ");
		tfMensagem = new JTextField(40);
		tfMensagem.setEditable(false);
		pMensagem.add(labelMensagem);
		pMensagem.add(tfMensagem);
		pMensagem.setLayout(new FlowLayout(FlowLayout.LEFT));

		pCentro.add(pDisplay, BorderLayout.CENTER);
		pCentro.add(pBotoes, BorderLayout.SOUTH);
		painelGeral.add(pCentro, BorderLayout.CENTER);
		painelGeral.add(pMensagem, BorderLayout.SOUTH);

		janela.add(painelGeral);
		janela.setBounds(0, 0, 600, 400);
		janela.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		janela.setVisible(true);

		try{
			controlador.recuperarAlunos();
			tfMensagem.setText("Os dados dos alunos foram recuperados do arquivo");
		}
		catch(IOException ioe){
			tfMensagem.setText
			   ("Não foi possível recuperar os dados dos alunos: IOException");
		}
		catch (ClassNotFoundException cnf){
			tfMensagem.setText
		  ("Não foi possível recuperar os dados dos alunos: ClassNotFoundException");
		}
	}


	class OuvinteTfDre extends KeyAdapter{
		public void keyTyped(KeyEvent ev){
			tfMensagem.setText("");
			tfNome.setText("");
		}
	}

	class OuvinteCriar implements ActionListener {
		public void actionPerformed(ActionEvent aev){
			nome = tfNome.getText();
			dre = tfDre.getText();
			try{
				controlador.criaAluno(dre, nome);
				tfMensagem.setText("Aluno " + nome + " criado OK, com DRE " +
dre);
			}
			catch(DreDuplicadoException ex){
				tfMensagem.setText
			  ("Não foi possível criar o aluno. O DRE " + dre + " já foi alocado");
			}
		}
	}

	class OuvinteObterNome implements ActionListener {
		public void actionPerformed(ActionEvent aev){
			dre = tfDre.getText();
			try{
				nome = controlador.getAluno(dre).getNome();
```

```
                        tfMensagem.setText("O aluno com DRE = " + dre + " chama-se " +
nome);
                }
                catch(AlunoInexistenteException ex){
                        tfMensagem.setText("Não existe aluno com DRE = " + dre);
                }
        }
    }

    class OuvinteSalvar implements ActionListener{
        public void actionPerformed(ActionEvent aev){
                try{
                        controlador.salvarAlunos();
                        tfMensagem.setText("Dados dos alunos salvos com sucesso");
                }
                catch (IOException ioe){
                        tfMensagem.setText
                    ("Não foi possível salvar os dados dos alunos no arquivo");
                }
        }
    }


    class OuvinteLimpar implements ActionListener{
        public void actionPerformed(ActionEvent aev){
                controlador.limparDados();
        }
    }


    public static void main(String[] args) {
        new GuiSimplesSalvando();
    }
}
```