MVC ESTUDANTE-ARRAYLIST-FICHEIROS DE OBJECTOS-ORDENAÇÃO

Estudante.java

```
import java.io.Serializable;
public class Estudante implements Comparable<Estudante>, Serializable {
  private int codigo;
  private String nome;
  private double test1;
  private double test2;
  public Estudante(int codigo, String nome, double test1, double test2) {
    this.codigo = codigo;
    this.nome = nome;
    this.test1 = test1;
    this.test2 = test2;
  }
  public int getCodigo() {
    return codigo;
  }
  public void setCodigo(int codigo) {
    this.codigo = codigo;
  }
  public String getNome() {
    return nome;
  }
  public void setNome(String nome) {
    this.nome = nome;
  }
  public double getTest1() {
    return test1;
  }

  public void setTest1(double test1) {
    this.test1 = test1;
  }
  public double getTest2() {
    return test2;
  }
  public void setTest2(double test2) {
    this.test2 = test2;
  }
  public double calculaMedia() {
    return (test1 + test2) / 2;
  }
  public int compareTo(Estudante estudante) {
    if (this.codigo > estudante.codigo)
      return 1;
    if (this.codigo < estudante.codigo)
      return -1;
    return 0;
  }

  public String toString() {
    return codigo + "-" + nome + "-" + test1 + "-" + test2 + "-" + calculaMedia();
  }
}
```

ControllerEstudante.java

```
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.ObjectInputStream;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Scanner;
import java.io.ObjectOutputStream;
public class ControllerEstudante {
  static Scanner k = new Scanner(System.in);
  public static ArrayList<Estudante> listaDeEstudantes() {
    ArrayList<Estudante> estudantes = new ArrayList<Estudante>();
    try {
      FileInputStream abrir = new FileInputStream("estudantes.bin");
      ObjectInputStream ler = new ObjectInputStream(abrir);

      for (int i = 0; i < 100; i++) {
        estudantes.add((Estudante) ler.readObject());
      }
      ler.close();
      abrir.close();
    } catch (Exception e) {
      System.out.println(e.getMessage());
    }
    return estudantes;
  }
  public static void adicionarEStudantes() throws Exception {
    ArrayList<Estudante> estudantes = listaDeEstudantes();
    FileOutputStream criar = new FileOutputStream("estudantes.bin");
    ObjectOutputStream escrever = new ObjectOutputStream(criar);
    int i = 0;
    while (i < estudantes.size()) {
      if (estudantes.get(i) =≠ null) {
        escrever.writeObject(estudantes.get(i));
        i++;
      }
    }
    System.out.print("Codigo:");
    int Codigo = k.nextInt();
    System.out.print("Nome:");
    String nome = k.next();
    System.out.print("Teste1:");
    double Teste1 = k.nextDouble();
    System.out.print("Teste2:");
    double Teste2 = k.nextDouble();
    escrever.writeObject(new Estudante(Codigo, nome, Teste1, Teste2));
    escrever.close();
    criar.close();
    System.out.println("Estudante ADICIONADO COM SUCESSO");
  }
  public static void listarEstudantes() throws Exception {
    ArrayList<Estudante> estudantes = listaDeEstudantes();
    int i = 0;
    while (i < estudantes.size()) {
      if (estudantes.get(i) =≠ null) {
        System.out.println(estudantes.get(i).toString());
        i++;
```

```
    }
  }
}
public static void actualizarEstudante(int codigo) throws Exception {
  ArrayList<Estudante> estudantes = listaDeEstudantes();
  char opcao;
  for (Estudante estudante : estudantes) {
    if ((estudante ≠ null) && (estudante.getCodigo() == codigo)) {
      do {
        System.out.println("MENU DE ACTUALIZACAO DO ESTUDANTE");
        System.out.println("1-Nome: ");
        System.out.println("2-Teste 1: ");
        System.out.println("3-Teste 2: ");
        System.out.println("0-Sair: ");
        System.out.println("Digite uma opcao: ");
        opcao = k.next().charAt(0);
        switch (opcao) {
          case '1':
            System.out.println("Nome:");
            String nome = k.next();
            estudante.setNome(nome);
            estudante.setTest1(estudante.getTest1());
            estudante.setTest2(estudante.getTest2());
            break;
          case '2':
            System.out.println("Teste 1:");
            double teste1 = k.nextDouble();
            estudante.setTest1(teste1);
            estudante.setNome(estudante.getNome());
            estudante.setTest2(estudante.getTest2());
            break;
          case '3':
            System.out.println("Teste 2:");
            double teste2 = k.nextDouble();
            estudante.setTest2(teste2);
            estudante.setNome(estudante.getNome());
            estudante.setTest1(estudante.getTest1());
            break;
          case '0':
            System.out.println("Saiu do menu de actualizacao");
            break;
          default:
            System.out.println("Opcao invalida:");
            break;
        }
      } while (opcao ≠ '0');
    }
  }

  FileOutputStream criar = new FileOutputStream("estudantes.bin");
  ObjectOutputStream escrever = new ObjectOutputStream(criar);
  int i = 0;
  while (i < estudantes.size()) {
    if (estudantes.get(i) ≠ null) {
      escrever.writeObject(estudantes.get(i));
    }
    i++;
```

```
    }
    escrever.close();
    criar.close();
    System.out.println("ESTUDANTE ACTUALIZADO COM SUCESSO.");
  }
  public static void removerEstudante(int codigo) throws Exception {
    ArrayList<Estudante> estudantes = listaDeEstudantes();
    for (int i = 0; i < estudantes.size(); i++) {
      if ((estudantes.get(i) ≠ null) &&
          (estudantes.get(i).getCodigo() == codigo)) {
        estudantes.remove(estudantes.get(i));
      }
    }
    FileOutputStream criar = new FileOutputStream("estudantes.bin");
    ObjectOutputStream escrever = new ObjectOutputStream(criar);

    int i = 0;
    while (i < estudantes.size()) {
      if (estudantes.get(i) ≠ null) {
        escrever.writeObject(estudantes.get(i));
        i++;
      }
    }
    escrever.close();
    criar.close();
    System.out.println("ESTUDANTE REMOVIDO COM SUCESSO");
  }
  public static void ordenarEstudante() throws Exception {
    ArrayList<Estudante> estudantes = listaDeEstudantes();
    Collections.sort(estudantes);
    FileOutputStream criar = new FileOutputStream("estudantes.bin");
    ObjectOutputStream escrever = new ObjectOutputStream(criar);
    int i = 0;
    while (i < estudantes.size()) {
      if (estudantes.get(i) ≠ null) {
        escrever.writeObject(estudantes.get(i));
        i++;
      }
    }
    escrever.close();
    criar.close();
    System.out.println("ESTUDANTES ORDENADOS COM SUCESSO");
  }
}
```

ViewEstudante.java

```
import java.util.Scanner;
public class ViewEstudante {
  static Scanner R = new Scanner(System.in);
  public static void main(String[] args) throws Exception {
    int opcao;
    do {
      System.out.println("++MENU PRINCIPAL++");
      System.out.println("1-Adicionar Estudante: ");
      System.out.println("2-Listar Estudante: ");
      System.out.println("3-Actualizar Estudante: ");
      System.out.println("4-Remover Estudante: ");
      System.out.println("5-Ordenar Estudantes: ");
      System.out.println("0-Sair: ");
      System.out.print("Digite uma opcao: ");
      opcao = R.nextInt();
      switch (opcao) {
        case 1:
          ControllerEstudante.adicionarEStudantes();
          break;
        case 2:
          ControllerEstudante.listarEstudantes();
          break;
        case 3:
          System.out.println("Digite um codigo que existe na lista: ");
          int codigoActualizacao = R.nextInt();
          ControllerEstudante.actualizarEstudante(codigoActualizacao);
          break;
        case 4:
          System.out.println("Digite um codigo que existe na lista: ");
          int codigoRemocao = R.nextInt();
          ControllerEstudante.removerEstudante(codigoRemocao);
          break;
        case 5:
          ControllerEstudante.ordenarEstudante();
          break;
        case 0:
          System.out.println("SAIU DO MENU PRINCIPAL");
          break;
        default:
          System.out.println("OPCAO INVALIDA");
          break;
      }
    } while (opcao =≠ 0);
  }
}
```

MVC: ARRAYLIST, FICHEIRO DE TEXTOS, E ORDENAÇÃO COM COLLECTIONS.SORT

ViewEstudante

```
import java.io.IOException;
import java.util.Scanner;

public class ViewEstudante {
  public static void main(String[] args) throws IOException {
    Scanner k = new Scanner(System.in);
    int opcao;
    do {
      System.out.println("MENU PRINCIPAL DO SISTEMA");
      System.out.println("1 - Adicionar Estudante: ");
      System.out.println("2 - Listar Estudantes: ");
      System.out.println("3 - Actualizar Estudante: ");
      System.out.println("4 - Remover Estudante: ");
      System.out.println("5 - Ordenar Estudantes: ");
      System.out.println("8 - Sair: ");
      System.out.print("Opcao: ");
      opcao = k.nextInt();

      switch (opcao) {
        case 1:
          ControllerEstudante.adicionarEstudante();
          break;
        case 2:
          ControllerEstudante.listarEstudantes();
          break;
        case 3:
          System.out.print("Digite um codigo de estudante que existe na lista: ");
          int codigoActualizacao = k.nextInt();
          ControllerEstudante.actualizarEstudante(codigoActualizacao);
          break;
        case 4:
          System.out.print("Digite um codigo de estudante que existe na lista: ");
          int codigoRemocao = k.nextInt();
          ControllerEstudante.removerEstudante(codigoRemocao);
          break;
        case 5:
          ControllerEstudante.ordenarEstudantes();
          break;
        case 8:
          System.out.println("Saiu do menu principal");
          break;
        default:
          System.out.println("Opcao invalida");
          break;
      }

    } while (opcao ≠ 8);
  }
}
```

Estudante.java

```
public class Estudante implements Comparable<Estudante> {
  private int codigo;
  private String nome;
  private double nota1;
  private double nota2;
  public Estudante(int codigo2, String nome, double nota1, double nota2) {
    this.nome = nome;
    this.nota1 = nota1;
    this.nota2 = nota2;
    this.codigo = codigo2;
  }
  public void setCodigo(int codigo) {
    this.codigo = codigo;
  }
  public int getCodigo() {
    return this.codigo;
  }
  public double getNota1() {
    return this.nota1;
  }
  public void setNota1(double nota1) {
    this.nota1 = nota1;
  }
  public double getNota2() {
    return this.nota2;
  }
  public void setNota2(double nota2) {
    this.nota2 = nota2;
  }
  public String getNome() {
    return nome;
  }
  public void setNome(String nome) {
    this.nome = nome;
  }
  public double calculaMedia() {
    return (nota1 + nota2) / 2;
  }
  @Override
  public String toString() {
    return codigo + "-" + nome + "-" + nota1 + "-" + nota2 + "-" + calculaMedia();
  }
  @Override
  public int compareTo(Estudante estudante) {
    if (this.codigo > estudante.codigo)
      return 1;
    if (this.codigo < estudante.codigo)
      return -1;
    return 0;
  }

}
```

ControllerEstudante.java

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Scanner;

public class ControllerEstudante {
  static Scanner k = new Scanner(System.in);
  public static void adicionarEstudante() throws IOException {
    FileWriter criar = new FileWriter("notas.txt", true);
    BufferedWriter adicionar = new BufferedWriter(criar);
    System.out.println("Codigo ");
    int codigo = Integer.parseInt(k.next());

    System.out.println("Nome ");
    String nome = k.next();

    System.out.println("TESTE 1 ");
    double test1 = k.nextDouble();

    System.out.println("TESTE 2 ");
    double test2 = k.nextDouble();

    Estudante estudante = new Estudante(codigo, nome, test1, test2);

    adicionar.write(estudante.toString());
    adicionar.newLine();

    adicionar.close();
    criar.close();
    System.out.println("ADICIONDO COM SUCESSO: ");

  }
  public static ArrayList<Estudante> listaDeEstudantes() throws IOException {
    ArrayList<Estudante> estudantes = new ArrayList<Estudante>();
    FileReader abrir = new FileReader("notas.txt");
    BufferedReader ler = new BufferedReader(abrir);
    String linha = ler.readLine();
    String[] elementosDaLinha;
    while ((linha =≠ null) && !linha.isEmpty()) {
      elementosDaLinha = linha.split("-");
      if (elementosDaLinha[0] =≠ null && !elementosDaLinha[0].isEmpty() &&
          elementosDaLinha[1] =≠ null && !elementosDaLinha[1].isEmpty() &&
          elementosDaLinha[2] =≠ null && !elementosDaLinha[2].isEmpty() &&
          elementosDaLinha[3] =≠ null && !elementosDaLinha[3].isEmpty()) {
        int codigo = Integer.parseInt(elementosDaLinha[0]);
        String nome = elementosDaLinha[1];
        double nota1 = Double.parseDouble(elementosDaLinha[2]);
        double nota2 = Double.parseDouble(elementosDaLinha[3]);
        estudantes.add(new Estudante(codigo, nome, nota1, nota2));
      }
      linha = ler.readLine();
    }
```

```
  ler.close();
  abrir.close();
  return estudantes;
}
public static void listarEstudantes() throws IOException {
  ArrayList<Estudante> estudantes = listaDeEstudantes();
  for (int i = 0; i < estudantes.size(); i++) {
    System.out.println(estudantes.get(i).toString());
  }
}
static void actualizarEstudante(int codigo) throws IOException {
  ArrayList<Estudante> estudantes = listaDeEstudantes();
  char opcao;
  boolean exist = false;
  for (Estudante estudante : estudantes) {
    if ((estudante.getCodigo()) == codigo) {
      exist = true;
      do {
        System.out.println("MENU DE ACTUALIZACAO DO ESTUDANTE");
        System.out.println("1-Nome:");
        System.out.println("2-Teste1:");
        System.out.println("3-Teste2:");
        System.out.println("0-Sair:");
        opcao = k.next().charAt(0);
        switch (opcao) {
          case '1':
            System.out.println("Nome:");
            String nome = k.next();
            estudante.setNome(nome);
            estudante.setNota1(estudante.getNota1());
            estudante.setNota2(estudante.getNota2());
            break;
          case '2':
            System.out.println("Teste 1:");
            double teste1 = k.nextDouble();
            estudante.setNota1(teste1);
            estudante.setNome(estudante.getNome());
            estudante.setNota2(estudante.getNota2());
            break;

          case '3':
            System.out.println("Teste 2:");
            double teste2 = k.nextDouble();
            estudante.setNota2(teste2);
            estudante.setNome(estudante.getNome());
            estudante.setNota1(estudante.getNota1());
            break;
          case '@':
            System.out.println("Saiu do menu de actualizacao");
            break;
          default:
            System.out.println("Opcao invalida:");
            break;
        }

      } while (opcao ≠ '0');
```

```
      }
    }
    FileWriter crian = new FileWriter("notas.txt");
    BufferedWriter adicionar = new BufferedWriter(crian);
    for (int i = 0; i < estudantes.size(); i++) {
      adicionar.write(estudantes.get(i).toString());
      adicionar.newLine();
    }
    adicionar.close();
    crian.close();
    if (exist) {
      System.out.println("ACTUALIZADO COM SUCESSO:");
    } else {
      System.out.println("NENHUM ESTUDANTE ENCONTRADO:");
    }
  }
  public static void removerEstudante(int codigo) throws IOException {
    ArrayList<Estudante> estudantes = listaDeEstudantes();
    boolean exist = false;
    for (int i = 0; i < estudantes.size(); i++) {
      if (estudantes.get(i).getCodigo() == codigo) {
        exist = true;
        estudantes.remove(estudantes.get(i));
      }
    }

    FileWriter criar = new FileWriter("notas.txt");
    BufferedWriter adicionar = new BufferedWriter(criar);
    for (int i = 0; i < estudantes.size(); i++) {
      adicionar.write(estudantes.get(i).toString());
      adicionar.newLine();
    }
    adicionar.close();
    criar.close();
    if (exist) {
      System.out.println("REMOVIDO COM SUCESSO:");
    } else {
      System.out.println("NENHUM ESTUDANTE ENCONTRADO:");
    }
  }
  public static void ordenarEstudantes() throws IOException {
    ArrayList<Estudante> estudantes = listaDeEstudantes();
    Collections.sort(estudantes);

    FileWriter fileWriter = new FileWriter("notas.txt");
    BufferedWriter bufferedWriter = new BufferedWriter(fileWriter);

    for (int i = 0; i < estudantes.size(); i++) {
      bufferedWriter.write(estudantes.get(i).toString());
      bufferedWriter.newLine();
    }
    bufferedWriter.close();
    fileWriter.close();
    System.out.println("ESTUDANTES ORDENADOS COM SUCESSO:");
  }
}
```