

```

// Solution to Project 9.6

import java.util.Scanner;

public class TestScoresView{

    private TestScoresModel model;

    public TestScoresView(TestScoresModel m){
        model = m;
        run();
    }

    // Menu-driven command loop
    private void run(){
        while (true){

            System.out.println("Number of students: " + model.size());
            System.out.println("Index of current student: " +
                model.currentPosition());

            displayMenu();
            int command = getCommand("Enter a number [1-11]: ", 1, 11);
            if (command == 11)
                break;
            runCommand(command);

            System.out.print("Press <ENTER> to continue...");
            Scanner reader = new Scanner(System.in);
            reader.nextLine();
            for(int x=0; x<=10; x++)System.out.println("");
            //Process p = Runtime.getRuntime().exec("cls");
        }
    }

    private void displayMenu(){
        System.out.println("MAIN MENU");
        System.out.println(" 1. Display the current student");
        System.out.println(" 2. Display the class average");
        System.out.println(" 3. Display the student with the highest grade");
        System.out.println(" 4. Display all of the students");
        System.out.println(" 5. Edit the current student");
        System.out.println(" 6. Add a new student");
        System.out.println(" 7. Move to the first student");
        System.out.println(" 8. Move to the last student");
        System.out.println(" 9. Move to the next student");
        System.out.println("10. Move to the previous student");
        System.out.println("11. Quit the program");
    }
}

```

```

// Prompts the user for a command number and runs until
// the user enters a valid command number
// Parameters: prompt is the string to display
//             low is the smallest command number
//             high is the largest command number
// Returns: a valid command number
private int getCommand(String prompt, int low, int high){
    // Exercise: recover from all input errors
    Scanner reader = new Scanner(System.in);
    int command = low - 1;
    while (command < low || command > high){
        System.out.print(prompt);
        command = reader.nextInt();
        if (command < low || command > high)
            System.out.println("Error: command must be between " +
                               low + " and " + high);
    }
    return command;
}

// Selects a command to run based on a command number,
// runs the command, and asks the user to continue by
// pressing the Enter key
private void runCommand(int command){
    if (command == 1)
        displayStudent();
    else if (command == 2)
        System.out.println("Average score = " + model.getClassAverage());
    else if (command == 3)
        displayHighScore();
    else if (command == 4)
        System.out.println(model);
    else if (command == 5)
        editStudent();
    else if (command == 6)
        addStudent();
    else if (command == 7)
        model.first();
    else if (command == 8)
        model.last();
    else if (command == 9)
        model.next();
    else if (command == 10)
        model.previous();
}

private void displayStudent(){
    Student s = model.currentStudent();
    if (s == null)
        System.out.println("No student is currently available");
    else
        System.out.println(s);
}

```

```
private void displayHighScore() {
    Student s = model.getHighScore();
    if (s == null)
        System.out.println("No student is currently available");
    else
        System.out.println(s);
}

private void addStudent() {
    String nametemp;
    int howmany;

    Scanner reader = new Scanner(System.in);
    System.out.print("Enter the name: ");
    nametemp = reader.nextLine();
    System.out.print("How many grades will you enter: ");
    howmany = reader.nextInt();

    Student s = new Student(nametemp, howmany);

    for (int i = 1; i <= s.getNumberOfTests(); i++){
        System.out.print("Enter the score for test " + i + ": ");
        s.setScore(i, reader.nextInt());
    }
    String message = s.validateData();
    if (message != null)
        System.out.println(message);
    else{
        message = model.add(s);
        if (message != null)
            System.out.println(message);
        else
            System.out.println("Student added");
    }
}
```

```

private void editStudent(){
    Student s = model.currentStudent();
    if (s == null)
        System.out.println("No student is currently available");
    else{
        // Work on a temporary copy
        Student temp = new Student(s);
        String menu = "EDIT MENU\n" +
            "1. Change the name\n" +
            "2. Change a score\n" +
            "3. View the student\n" +
            "4. Quit this menu\n";
        Scanner reader = new Scanner(System.in);
        int command = 1;
        while (command != 4){
            System.out.print(menu);
            command = getCommand("Enter a number [1-4]: ", 1, 4);
            if (command == 1){
                System.out.print("Enter the student name: ");
                String name = reader.nextLine();
                temp.setName(name);
            }else if (command == 2){
                System.out.print("Enter the number of the test [1-" +
                    temp.getNumberOfTests() + "]: ");
                int i = reader.nextInt();
                System.out.print("Enter the test score: ");
                temp.setScore(i, reader.nextInt());
            }else if (command == 3){
                System.out.println(temp);
            }else{
                // Check for valid data before writing to database
                String message = temp.validateData();
                if (message != null)
                    System.out.println(message);
                else
                    model.replace(temp);
            }
        }
    }
}

```