

PROGRAMMING EXAMPLE: Grade Report

PART II: The first part of this programming example illustrated how to display the output in the Windows console environment. In this section, we create the GUI, as shown in **STUDENT** Figure 10-20, to show the output.
GRADE REPORT:
GUI DESIGN

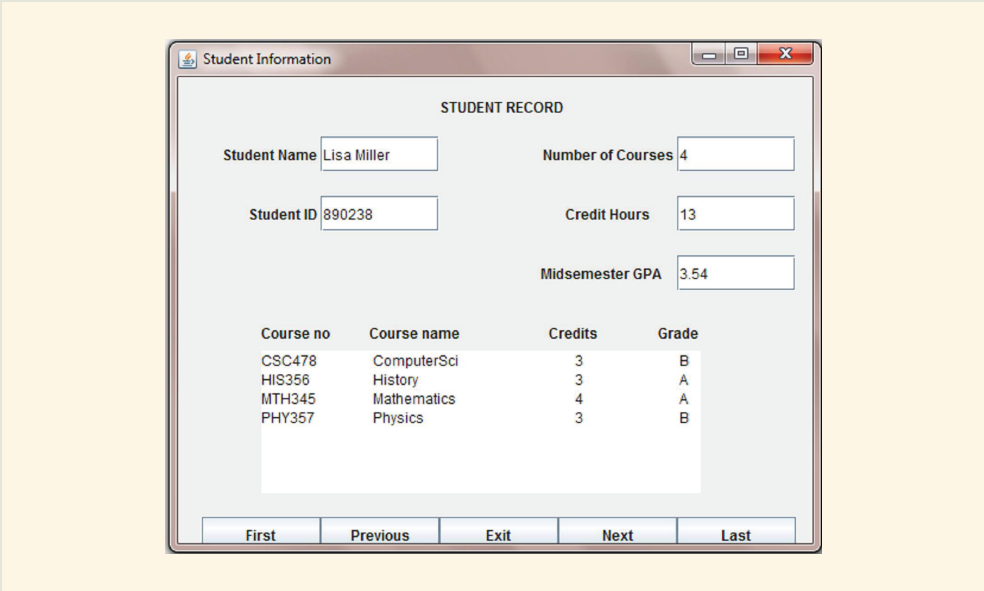


FIGURE 10-20 GUI to show a student’s record

From the desired output, it is clear that the GUI contains various labels, text fields, and buttons. Moreover, when you click one of the buttons on the bottom row, an action event is generated. Therefore, we also need to write the code necessary to handle the events.

The students’ records are stored in an array. When the user clicks the button labeled **First**, the program must display the record of the first student in the list, that is, in the array. Similarly, when the user clicks the **Next** button, the program must display the record of the next student in the list.

NOTE An object of the `class JTextField` can display only one line of text. However, the text showing the course information contains multiple lines. To display multiline text, we make use of the `class JTextArea`. We will create an object of the type `JTextArea` and use the appropriate methods of this class to display the courses as desired. This is a predefined Java class. We will introduce the methods as we need them.

As in earlier programs, GUI components, such as labels, are placed in the content pane of the window. Thus, the program contains a window, a container, and various labels, text fields, a text area, and buttons. Let's first declare the reference variables to create the labels, text fields, and buttons.

```
//GUI components
//Each of the items to be displayed needs a label and
//text field. Labels end with L, text fields end with TF,
//and buttons end with B.

private JLabel headingL, stNameL, stIDL, noCoursesL,
               courseListL, creditHrsL, gpaL;

private JTextField stNameTF, stIDTF, noCoursesTF,
                  creditHrsTF, gpaTF;

private JTextArea courseListTA; //to display multiline text

private JButton exitB, nextB, prevB, firstB, lastB;
```

The following statements instantiate these objects:

```
//instantiate labels
headingL = new JLabel("STUDENT RECORD");
stNameL = new JLabel("Student Name ", SwingConstants.RIGHT);
stIDL = new JLabel("Student ID ", SwingConstants.RIGHT);
noCoursesL = new JLabel("Number of Courses ",
                        SwingConstants.RIGHT);
courseListL = new JLabel("Course no          Course name " +
                        "                      Credits " +
                        "                      Grade");
creditHrsL = new JLabel("Credit Hours ", SwingConstants.RIGHT);
gpaL = new JLabel("Midsemester GPA ", SwingConstants.RIGHT);

//instantiate the text fields
stNameTF = new JTextField(10);
stIDTF = new JTextField(10);
noCoursesTF = new JTextField(10);
creditHrsTF = new JTextField(10);
gpaTF = new JTextField(10);

//instantiate the text area
courseListTA = new JTextArea(6, 20); //6 rows and 20
                                     //columns wide
courseListTA.setAutoscrolls(true);   //set the scroll
                                     //bars to show

//instantiate the buttons
exitB = new JButton("Exit");
nextB = new JButton("Next");
```

```
prevB = new JButton("Previous");
firstB = new JButton("First");
lastB = new JButton("Last");
```

We will create a class containing the application program by extending the `class JFrame`. We will also create an inner class to handle the action events.

The following statements set the title and the size of the window and declare a reference variable that points to the content pane of the window:

```
setTitle("Student Information"); //set title of the window
setSize(WIDTH, HEIGHT);         //set size of the window using
                                //the named constants
                                //WIDTH and HEIGHT
Container pane = getContentPane(); //get the container
```

From the GUI design, it follows that GUI components are placed at a specific position in the container. To create such an interface, we set the container layout to `null`. The following statement accomplishes this:

```
pane.setLayout(null); //set the container layout to null
```

We use the method `setSize` to specify the size of the GUI components. Recall from Chapter 6 that the heading of the method `setSize` is:

```
public void setSize(int width, int height)
```

The following statements set the size of each component:

```
headingL.setSize(200, 30); //label headingL is 200 pixels wide
                           //and 30 pixels high
stNameL.setSize(100, 30);
stNameTF.setSize(100, 30);
stIDL.setSize(100, 30);
stIDTF.setSize(100, 30);
noCoursesL.setSize(120, 30);
noCoursesTF.setSize(100, 30);
creditHrsL.setSize(100, 30);
creditHrsTF.setSize(100, 30);
gpaL.setSize(110, 30);
gpaTF.setSize(100, 30);
courseListL.setSize(370, 30);
courseListTA.setSize(370, 120);
firstB.setSize(100, 30);
prevB.setSize(100, 30);
exitB.setSize(100, 30);
nextB.setSize(100, 30);
lastB.setSize(100, 30);
```

Similarly, we use the method `setLocation` to specify the location of the GUI components in the container. The heading of the method `setLocation` is:

```
public void setLocation(int xCoordinate, int yCoordinate)
```

The following statements set the location of the GUI components in the container:

```
headingL.setLocation(220, 10);
stNameL.setLocation(20, 50);
stNameTF.setLocation(120, 50);
stIDL.setLocation(20, 100);
stIDTF.setLocation(120, 100);
noCoursesL.setLocation(300, 50);
noCoursesTF.setLocation(420, 50);
creditHrsL.setLocation(300, 100);
creditHrsTF.setLocation(420, 100);
gpaL.setLocation(300, 150);
gpaTF.setLocation(420, 150);
courseListL.setLocation(70, 200);
courseListTA.setLocation(70, 230);
firstB.setLocation(20, 370);
prevB.setLocation(120, 370);
exitB.setLocation(220, 370);
nextB.setLocation(320, 370);
lastB.setLocation(420, 370);
```

We will use the method `add` to put the labels, text fields, text area, and buttons into the container `pane`. The following statements accomplish this:

```
//add labels, text fields, and buttons to the pane
pane.add(headingL);
pane.add(stNameL);
pane.add(stNameTF);
pane.add(stIDL);
pane.add(stIDTF);
pane.add(noCoursesL);
pane.add(noCoursesTF);
pane.add(courseListL);
pane.add(courseListTA);
pane.add(creditHrsL);
pane.add(creditHrsTF);
pane.add(gpaL);
pane.add(gpaTF);
pane.add(firstB);
pane.add(prevB);
pane.add(exitB);
pane.add(nextB);
pane.add(lastB);
```

Method Each student's record is stored in the array `studentList`. Therefore, to display a student's grade report, the relevant information is retrieved from the array `studentList` and displayed in the respective text fields and text area. To simplify displaying a student's record, we write the method `displayGradeReports`. This method takes as its parameter an `int` variable specifying the position (index) in the array where the student's data is stored. Moreover, because the buttons `Next` and

Previous show the data of the student preceding and following, respectively, the current student, the program must store the index of the current student, which we store in the instance variable `displayedStudentIndex`. Essentially, the definition of the method `displayGradeReports` is:

```
public void displayGradeReports(int stNo)
{
    displayedStudentIndex = stNo;
    String CourseListing = "";

    boolean isPaid = studentList[stNo].getIsTuitionPaid();

    stNameTF.setText(studentList[stNo].getFirstName() + " "
                    + studentList[stNo].getLastName());
    stIDTF.setText("" + studentList[stNo].getStudentId());
    noCoursesTF.setText(""
                    + studentList[stNo].getNumberOfCourses());
    creditHrsTF.setText(""
                    + studentList[stNo].getHoursEnrolled());

    if (isPaid)
        gpaTF.setText("" + String.format("%.2f",
            studentList[stNo].getGpa()));
    else
        gpaTF.setText("" + "****");

    for (int count = 0;
        count < studentList[stNo].getNumberOfCourses();
        count++)
    {
        String str;
        Course temp;

        temp = studentList[stNo].getCourse(count);

        str = temp.getCourseNumber() + "\t "
            + temp.getCourseName() + "\t\t"
            + temp.getCredits() + "\t";

        if (isPaid)
            str = str + studentList[stNo].getGrade(count);
        else
            str = str + "****";

        if (count == 0)
            CourseListing = str;
        else
            CourseListing = CourseListing + "\n" + str;
    }
}
```

```

        if (!isPaid)
            CourseListing = CourseListing + "\n"
                        + "**** Grades are being held for "
                        + "not paying the tuition. ****\n"
                        + "Amount Due: $"
                        + String.format("%.2f",
                                studentList[stNo].billingAmount(tuitionRate));

        courseListTA.setText(CourseListing);
    }

```

Event Handling The GUI contains five buttons, each generating a specific event. To process these events, we create the `class` `ButtonHandler` implementing the `interface` `ActionListener`. The method `getActionCommand` of the `class` `ActionEvent` is used to identify the button generating the event. The definition of the `class` `ButtonHandler` and the method `actionPerformed` is:

```

private class ButtonHandler implements ActionListener
{
    public void actionPerformed(ActionEvent e)
    {
        if (e.getActionCommand().equals("Previous"))
            if (displayedStudentIndex > 0)
                displayGradeReports(displayedStudentIndex - 1);
            else
                displayGradeReports(displayedStudentIndex);
        else if (e.getActionCommand().equals("Next"))
            if (displayedStudentIndex + 1 < noOfStudents)
                displayGradeReports
                    (displayedStudentIndex + 1);
            else
                displayGradeReports(displayedStudentIndex);
        else if (e.getActionCommand().equals("First"))
            displayGradeReports(0);
        else if (e.getActionCommand().equals("Last"))
            displayGradeReports(noOfStudents - 1);
        else
            System.exit(0);
    }
}

```

As usual, we declare and instantiate an object of the type `ButtonHandler` and then register this object to each button. The following statements declare and instantiate the object handler:

```

private ButtonHandler bHandler; //declare the event handler
bHandler = new ButtonHandler(); //instantiate the event handler

```

The following statements register the listener object `bHandler` with the buttons:

```
exitB.addActionListener(bHandler);
nextB.addActionListener(bHandler);
prevB.addActionListener(bHandler);
firstB.addActionListener(bHandler);
lastB.addActionListener(bHandler);
```

MAIN PROGRAM

As in the first part of this program, we declare an array of type `Student` to hold the students' data. We must also store the number of students registered and the tuition rate. Thus, we need the following variables:

```
Student[] studentList;
```

```
int noOfStudents;
double tuitionRate;
```

Because we are creating a GUI, in order to place the data stored in `studentList` in GUI components, such as the text field and text area, we declare these variables as instance variables of the class containing the application program. Moreover, because they will be declared as instance variables, these variables need not be passed as parameters to the method `getStudentData`.

The data will be read from a file, so we need the following `Scanner` object to access the input file:

```
Scanner inFile = new Scanner(new FileReader("stData.txt"));
```

Method get Student Data

Because `studentList` and `noOfStudents` are instance variables, this method can directly access them. Therefore, this method has only one parameter: a parameter to access the input file. In pseudocode, the definition of this method is the same as the definition of this method for the non-GUI application program we designed in the first part. In fact, except for the heading of this method, the body of this method is similar to the one given in the first part of this program. Therefore, we give only the heading and leave the body as an exercise for you (see Programming Exercise 1 at the end of this section).

```
public static void getStudentData(Scanner inFile)
{
    //Write the body of this method.
}
```

PROGRAM LISTING

The preceding sections defined the necessary algorithms, methods, classes, and GUI components to complete the design of the Java program to create the GUI to display a student's data.

The program listing is as follows:

```
//*****
// Author: D.S. Malik
//
// Program: Student Grade Report -- GUI version
// This program reads students' data from file and outputs
// the grades. If a student has not paid the tuition, the
// grades are not shown, and an appropriate message is
// output.
//*****

import java.io.*;
import java.util.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class GradeReportProgram extends JFrame
{
    private static final int WIDTH = 550;
    private static final int HEIGHT = 430;

    private Student[] studentList;
    private int noOfStudents;
    private double tuitionRate;
    private int displayedStudentIndex = 0;

    //GUI components
    //Each of the items to be displayed needs
    //a label and text field.
    //Labels end with L and text fields end with TF.
    private JLabel headingL, stNameL, stIDL, noCoursesL,
        courseListL, creditHrsL, gpaL;
    private JTextField stNameTF, stIDTF, noCoursesTF,
        creditHrsTF, gpaTF;
    private JTextArea courseListTA;
    private JButton exitB, nextB, prevB, firstB, lastB;

    private ButtonHandler bHandler;

    public GradeReportProgram()
    {
        setTitle("Student Information");    //set the window title
    }
}
```



```

setSize(WIDTH, HEIGHT); //set the window size
Container pane = getContentPane(); //get the container
pane.setLayout(null); //set the container's layout to null

bHandler = new ButtonHandler(); //instantiate the button
//event handler

//instantiate the labels
headingL = new JLabel("STUDENT RECORD");
stNameL = new JLabel("Student Name ",
    SwingConstants.RIGHT);
stIDL = new JLabel("Student ID ", SwingConstants.RIGHT);
noCoursesL = new JLabel("Number of Courses ",
    SwingConstants.RIGHT);
courseListL = new JLabel("Course no "
    + "Course name "
    + "Credits "
    + "Grade");
creditHrsL = new
    JLabel("Credit Hours ", SwingConstants.RIGHT);
gpaL = new JLabel("Midsemester GPA ",
    SwingConstants.RIGHT);

//instantiate the text fields
stNameTF = new JTextField(10);
stIDTF = new JTextField(10);
noCoursesTF = new JTextField(10);
creditHrsTF = new JTextField(10);
gpaTF = new JTextField(10);

//instantiate the text area
courseListTA = new JTextArea(6, 20);
courseListTA.setAutoscrolls(true);

//instantiate the buttons and register the listener
exitB = new JButton("Exit");
exitB.addActionListener(bHandler);

nextB = new JButton("Next");
nextB.addActionListener(bHandler);

prevB = new JButton("Previous");
prevB.addActionListener(bHandler);

firstB = new JButton("First");
firstB.addActionListener(bHandler);

lastB = new JButton("Last");
lastB.addActionListener(bHandler);

```

```

    //set the size of the labels, text fields, and buttons
    headingL.setSize(200, 30);
    stNameL.setSize(100, 30);
    stNameTF.setSize(100, 30);
    stIDL.setSize(100, 30);
    stIDTF.setSize(100, 30);
    noCoursesL.setSize(120, 30);
    noCoursesTF.setSize(100, 30);
    creditHrsL.setSize(100, 30);
    creditHrsTF.setSize(100, 30);
    gpaL.setSize(110, 30);
    gpaTF.setSize(100, 30);
    courseListL.setSize(370, 30);
    courseListTA.setSize(370, 120);
    firstB.setSize(100, 30);
    prevB.setSize(100, 30);
    exitB.setSize(100, 30);
    nextB.setSize(100, 30);
    lastB.setSize(100, 30);

    //set the location of the labels, text fields,
    //and buttons
    headingL.setLocation(220, 10);
    stNameL.setLocation(20, 50);
    stNameTF.setLocation(120, 50);
    stIDL.setLocation(20, 100);
    stIDTF.setLocation(120, 100);
    noCoursesL.setLocation(300, 50);
    noCoursesTF.setLocation(420, 50);
    creditHrsL.setLocation(300, 100);
    creditHrsTF.setLocation(420, 100);
    gpaL.setLocation(300, 150);
    gpaTF.setLocation(420, 150);
    courseListL.setLocation(70, 200);
    courseListTA.setLocation(70, 230);
    firstB.setLocation(20, 370);
    prevB.setLocation(120, 370);
    exitB.setLocation(220, 370);
    nextB.setLocation(320, 370);
    lastB.setLocation(420, 370);

    //add the labels, text fields, and buttons to the pane
    pane.add(headingL);
    pane.add(stNameL);
    pane.add(stNameTF);
    pane.add(stIDL);
    pane.add(stIDTF);
    pane.add(noCoursesL);
    pane.add(noCoursesTF);
    pane.add(courseListL);

```

```

        pane.add(courseListTA);
        pane.add(creditHrsL);
        pane.add(creditHrsTF);
        pane.add(gpaL);
        pane.add(gpaTF);
        pane.add(firstB);
        pane.add(prevB);
        pane.add(exitB);
        pane.add(nextB);
        pane.add(lastB);

        setVisible(true); //show the window
        setDefaultCloseOperation(EXIT_ON_CLOSE);
    } //end constructor

    public static void main(String[] args) throws
        FileNotFoundException
    {
        GradeReportProgram gradeProg =
            new GradeReportProgram();
        Scanner inFile =
            new Scanner(new FileReader("stData.txt"));

        gradeProg.noOfStudents =
            inFile.nextInt(); //get the number of students
        gradeProg.tuitionRate =
            inFile.nextDouble(); //get the tuition rate

        gradeProg.studentList =
            new Student[gradeProg.noOfStudents];

        for (int i = 0; i < gradeProg.noOfStudents; i++)
            gradeProg.studentList[i] = new Student();

        gradeProg.getStudentData(inFile);
        gradeProg.displayGradeReports(0);
    } //end main

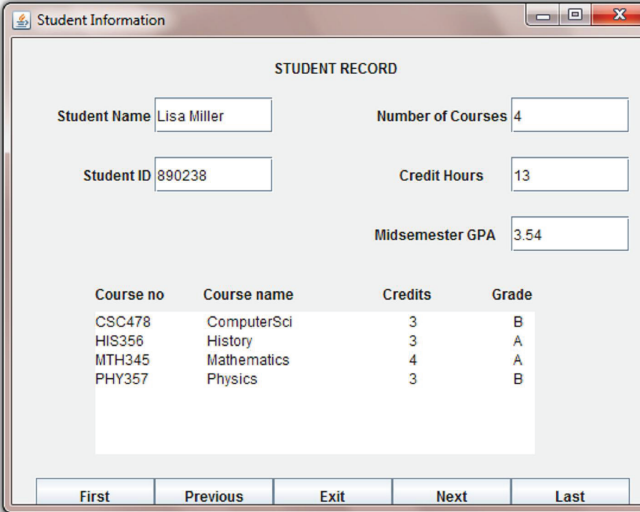
    //Write and place the definition of the method
    //getStudentData here.

    //Place the definition of the class ButtonHandler here.

    //Place the definition of the method displayGradeReports here.
}

```

Sample Output: (Programming Exercise 1, at the end of this section, asks you to write the definition of the method `getStudentData`. When you have completed Programming Exercise 1 and executed your program, it should produce the output shown in Figures 10-21 and 10-22.)



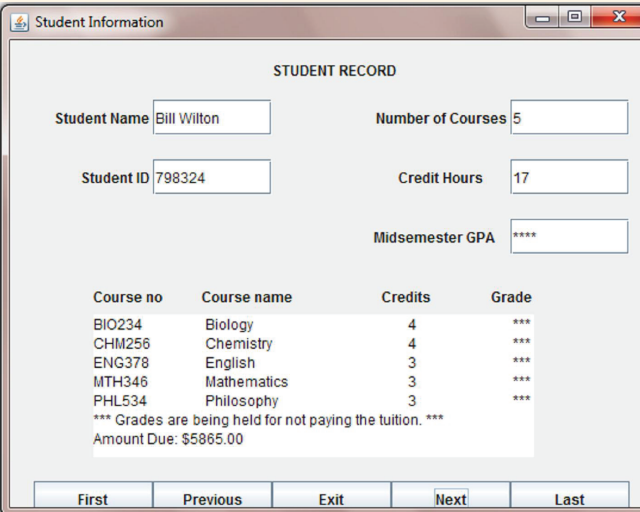
The screenshot shows a window titled "Student Information" with a "STUDENT RECORD" section. It contains input fields for Student Name (Lisa Miller), Student ID (890238), Number of Courses (4), Credit Hours (13), and Midsemester GPA (3.54). Below these is a table of courses.

Course no	Course name	Credits	Grade
CSC478	ComputerSci	3	B
HIS356	History	3	A
MTH345	Mathematics	4	A
PHY357	Physics	3	B

At the bottom are buttons: First, Previous, Exit, Next, and Last.

FIGURE 10-21 Sample GUI output of student grade program

Clicking the **Next** button produces the output shown in Figure 10-22.



The screenshot shows the same "Student Information" window, but now displaying the record for Bill Wilton. The input fields show Student Name (Bill Wilton), Student ID (798324), Number of Courses (5), Credit Hours (17), and Midsemester GPA (****). The table of courses is updated.

Course no	Course name	Credits	Grade
BIO234	Biology	4	***
CHM256	Chemistry	4	***
ENG378	English	3	***
MTH346	Mathematics	3	***
PHL534	Philosophy	3	***

Below the table, a message reads: "*** Grades are being held for not paying the tuition. ***
Amount Due: \$5865.00". The "Next" button is highlighted.

FIGURE 10-22 Sample output after clicking Next

PROGRAMMING EXERCISE

1. Write the definition of the method `getStudentData` of the GUI version of the Student Grade Report programming example. Also, complete the definition of the main program. When you execute your program, it should produce the output shown in Figures 10-21 and 10-22.