Labs

Lab 7.1: Displaying Calendars

What is the purpose?

In Lab 4.2, you wrote code to print a calendar. In this lab exercise, you will rewrite the PrintCalendar class from Lab 4.2 to display a calendar for a specified month using the Calendar and GregorianCalendar classes. The program receives the month and year from the command line, for example, to display a calendar for August, 2009.

What are the steps?

- Task 1:

  Procedure

  1. Create a subclass of GregorianCalendar called MyCalendar.
  2. Create the following methods:
     - public int daysInMonth()
     - public String getMonthName()

     OR, complete the code shown in Figure 7-1-1.

```
// MyCalendar.java: A subclass of GregorianCalendar
import java.util.*;

public class MyCalendar extends GregorianCalendar {
  /**Find the number of days in a month*/
  public int daysInMonth() {

  }

  /**return month name in English*/
  public String getMonthName() {

  }
}
```

Figure 7-1-1

  3. Create a java class called PrintCalendar to display calendar from command line.
  4. Create the following methods:
     a. static void printMonth(int year, int month)
     b. static int getStartDay()
     c. static void printMonthBody(int startDay, int numOfDaysInMonth)

     OR, complete the code picture in Figure 7-1-2.

Figure 7-1-2

```
// Display calendar from command line
import java.util.*;

public class PrintCalendar {
  /** MyCalendar is a subclass of GregorianCalendar,
      provided in a separate companion class.
   */
  static MyCalendar calendar = new MyCalendar();

  public static void main(String[] args) {
    /** declare month and year with default value for current
        month and year
     */
    int month = calendar.get(MyCalendar.MONTH) + 1;
    int year = calendar.get(MyCalendar.YEAR);

    //set date to the first day in a month
    calendar.set(Calendar.DATE, 1);

    //print calendar for the month
    printMonth(year, month);
  }

  static void printMonth(int year, int month) {
    //get start day of the week for the first date in the month
    int startDay = getStartDay();

    //get number of days in the month
    int numOfDaysInMonth = calendar.daysInMonth();

    //print headings
    printMonthTitle(year, month);

    //print body
    printMonthBody(startDay, numOfDaysInMonth);
  }

  static int getStartDay() {
    return calendar.get(Calendar.DAY_OF_WEEK);
  }

  static void printMonthBody(int startDay, int numOfDaysInMonth) {
    //print padding space before the first day of the month

  }
}
```

5. Compile the java files using the javac command.
6. Execute the PrintCalendar class using the java command.

7. Save screenshots of the output similar to Figures 7-1-3 through 7-1-5 and submit them to your instructor.

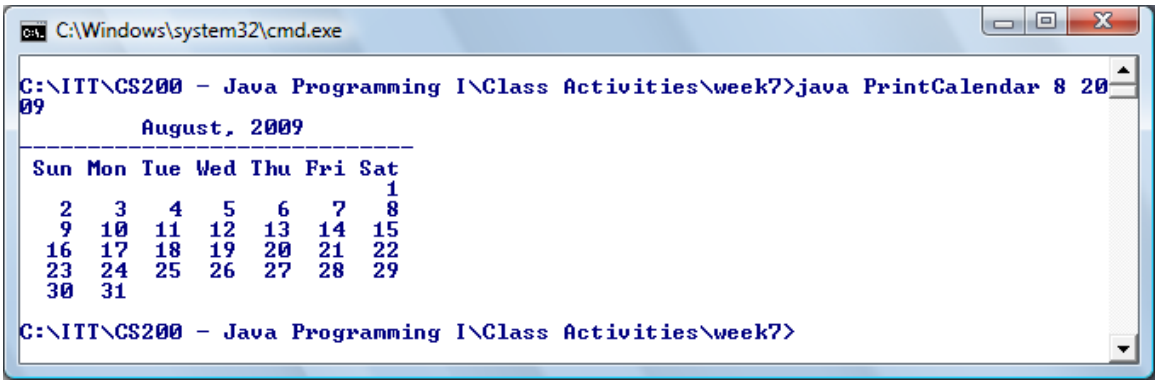

Figure 7-1-3

The program can also be run without the year. In this case, the year is the current year, for example, to display a calendar for October of the current year.

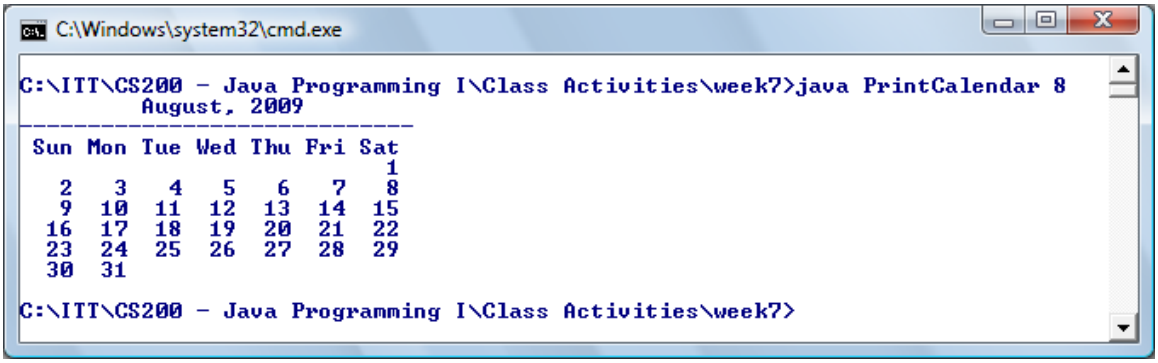

Figure 7-1-4

If the program is run without specifying a month and a year, the month is the current month, for example, to display a calendar for the current month and year.

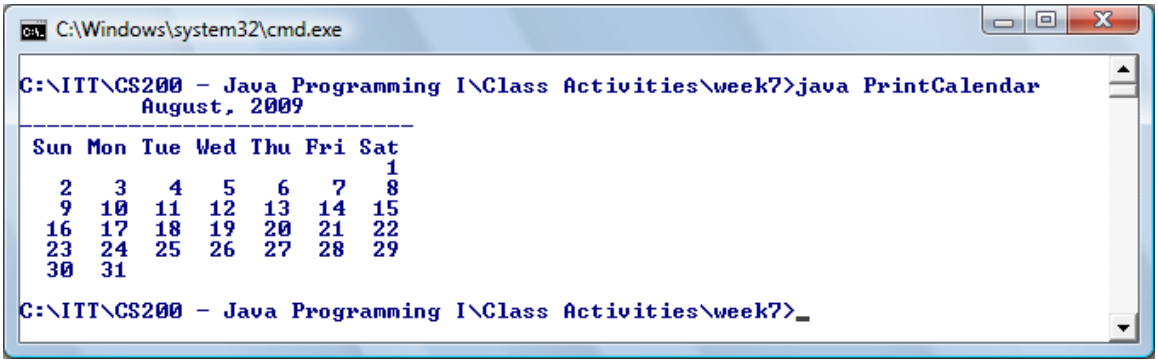

Figure 7-1-5

Did it work?

- Were you able to—

- Display current monthly calendar by the PrintCalendar class with parameters of current month and current year.
- Display current monthly calendar by the PrintCalendar class with parameter of current month.
- Display current monthly calendar by the PrintCalendar class without any parameters.

Lab 7.2: The Person and Student Classes

What is the purpose?

Design the Student class that extends Person. Implement the compareTo method in the Person class to compare persons in alphabetical order of their last name. Implement the compareTo method to compare students in alphabetical order of their major and last name.

Write a test program with the following four methods:

```
/** Sort an array of comparable objects **/
public static void sort(Object[] list)

/** Print an array of objects **/
public static void printList(Object[] object)

/** Return the max object in an array of comparable objects **/
public static Object max(Object[] list)
```

Main method: Test the sort, printList, and max methods using an array of four students, an array of 4 strings, an array of 100 random rationals, and an array of 100 random integers.

What are the steps?

- Task 1:

  Procedure

  1. Create a Java class called School.
  2. Complete the code shown in Figure 7-2-1.

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

public School() {
  // Define the arrays to be tested.
  Student[] s = new Student[3];
  s[0] = new Student(new Name("Derek", 'S', "Dexony"));
```

```
        s[1] = new Student(new Name("Stacy", 'M', "Waters"));
        s[2] = new Student(new Name("Adamo", 'U', "Leetmz"));

        String[] str = {"Orange", "Jackblade", "Apple"};
        Integer[] i = {new Integer(5), new Integer(0), new Integer(3)};

        // Display current array of students
        printList(s);
        System.out.println();

        // Display max of students
        System.out.println("Max is " + max(s));
        System.out.println();

        // Display sorted students
        sort(s);
        printList(s);
        System.out.println();

        // Display current array of strings
        printList(str);
        System.out.println();

        // Display max of strings
        System.out.println("Max is " + max(str));
        System.out.println();

        // Display sorted strings
        sort(str);
        printList(str);
        System.out.println();

        // Display current array of intergers
        printList(i);
        System.out.println();

        // Display max of ints
        System.out.println("Max is " + max(i));
        System.out.println();

        // Display sorted ints
        sort(i);
        printList(i);

    }
```

```
/** Pirnt an array of objects */
public static void printList(Object[] object) {

  }
} // End of method printList(Object[])

public static Object max(Object[] object) {
  // Find max of array object

  // Return results.

    } // End of method max(Object[])

public static void sort(Object[] object) {
  // Set the variables for the current min, minIndex, and compare value

  // Begin loop to work through the list.

    // Swap list[i] with list[currentMaxIndex] if needed.

} // End of instance method sort(Object[])

class Student extends Person implements Comparable, Cloneable {
  // Variables
  private String major;

  // Constructors
  public Student(Name name) {
    super(name);
    major = "Computer Science";
  }

  // Default Constructor
  public Student() {

  }

  // Instance Methods
  public String getMajor() {
    return major;
  }

  public void setMajor(String major) {
    this.major = major;
  }
```

```
  public String toString() {
    Name name = this.getName();
    return name.getFullName() + '\n' + "Major: " + major;
  }

  public boolean equals(Object object) {
    Name name = this.getName();
    Name otherName = ((Student)object).getName();

    if ((name.getFullName().equals(otherName.getLastName()))
        && (major.equals(((Student)object).getMajor()))) {
      return true;
    }
    return false;
  }

  /** Compare studnet's major, then last name */
  public int compareTo(Object object) {
}

public class Person {
  // Variables
  private Name name;

  // Constructors
  public Person(Name name) {
    this.name = name;
  }

  // Default Constructor
  public Person() {
  }

  // Instance Methods
  public Name getName() {
    return name;
  }

  public void setName(Name name) {
    this.name = name;
  }

  public String toString() {
    return name.getFullName();
  }
```

```
  public boolean equals(Object object) {
    Name otherName = ((Person)object).getName();
    if (name.getFullName().equals(otherName.getFullName())) {
      return true;
    }
    return false;
  }

  /** Compare person's last name */
  public int compareTo(Object object) {

}

class Name {
  // Variables
  private String firstName;
  private char mi;
  private String lastName;

  // Constructors
  public Name(String firstName, char mi, String lastName) {
    // Set new variables to instance.
    this.firstName = firstName;
    this.mi = mi;
    this.lastName = lastName;
  }

  // Default Constructor
  public Name() {

  }

  // Instance Methods
  public String getFirstName() {
    return firstName;
  }

  public void setFirstName(String firstName) {
    this.firstName = firstName;
  }

  public char getMi() {
    return mi;
  }

  public void setMi(char mi) {
```

```
        this.mi = mi;
      }

      public String getLastName() {
        return lastName;
      }

      public void setLastName(String lastName) {
        this.lastName = lastName;
      }

      public String getFullName() {
        return firstName + " " + mi + " " + lastName;
      }
    }
  }
```

Figure 7-2-1

3. Express the result to the console.
4. Compile the java file using javac command.
5. Execute the School class using java command.
6. Save a screenshot of the output similar to that shown in Figure 7-2-2 and submit it to your instructor.

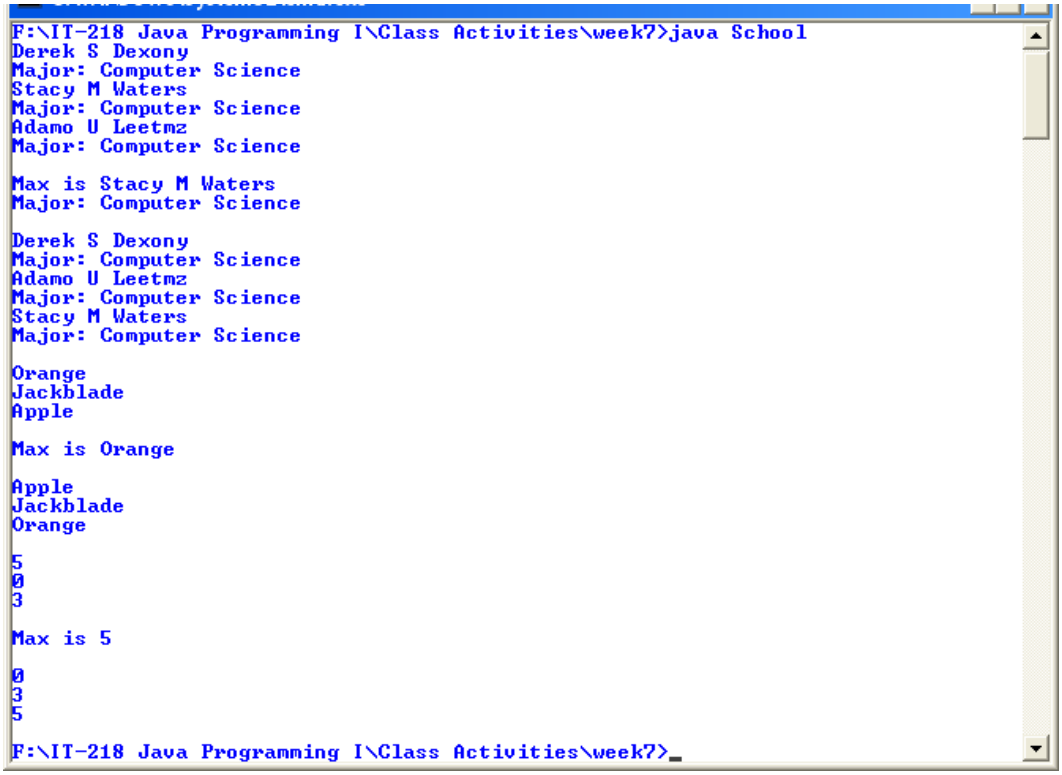

Figure 7-2-2

Did it work?

Were you able to—

- Display an array of student names?
- Display an array of sorted student names?
- Display the max value of an array of student names?
- Display an array of strings?
- Display an array of sorted strings?
- Display the max value of an array of strings?
- Display an array of integers?
- Display an array of sorted integers?
- Display the max value of an array of integers?