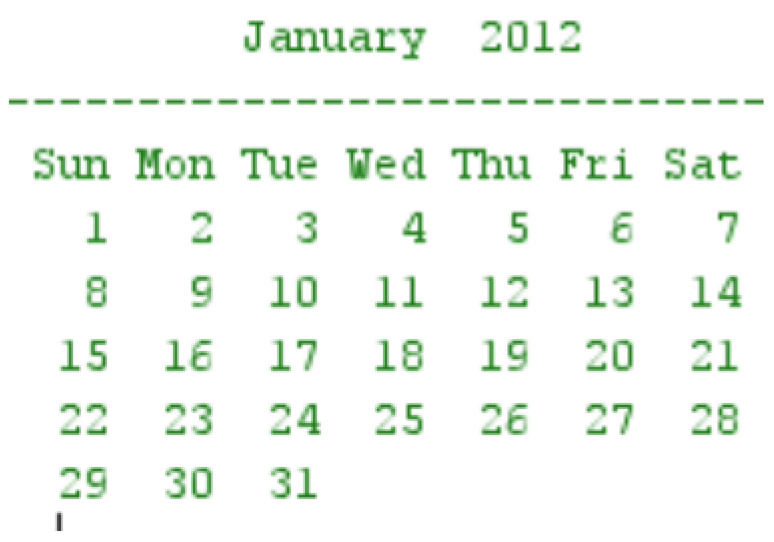
Module Four - Assignment Submission

The Assignment

1. Write a program that prompts the user to enter a month (1-12) and a year (e.g., 2012), and then displays a calendar for that month and year as illustrated below:



```
          January  2012
-----------------------------
 Sun Mon Tue Wed Thu Fri Sat
   1   2   3   4   5   6   7
   8   9  10  11  12  13  14
  15  16  17  18  19  20  21
  22  23  24  25  26  27  28
  29  30  31
```

2. Write a program that prompts the user to specify a year (e.g., 2012) and then displays a calendar for each month in that year. You must reuse the methods from part one

Design

```
When approaching the design of the first problem, it is important to notice
that the second problem is simply a caller of the functional output of the
first program, for each month January - December of that given year. Thus,
we need to make the main of the initial program be a caller of some
function which takes a month and a year, and prints the calendar out.

From the assignment, we know that we have to use the following methods in
our implementation:

void printMonthCalendar(int month, int year)
    Displays a calendar like the onevabove for a specified month and year.

void printMonthHeader(int month, int year)
    Displays the header information (month, year, line separator, 3-
character day names) for a calendar.

void printMonthBody(int month, int year)
    Displays the days in the calendar associated with the corresponding
days of the week.

String getMonthName(int month)
```

```
    Returns the name of the month for a specified month number (e.g.,
returns March for m=3).

int getStartDay(int month, int year)
Returns the day of week number (1=Monday,..., 7= Sunday) for the specified
month and year.

int getNumDaysInMonth(int month, int year)
    Returns the number of days in specified month and year. Leap years are
accounted for.

boolean isLeapYear(int year) Returns true if the specified year is a leap
year, and returns false otherwise

With this in mind, we'll have to design a program which:
1. Get user input
2. Call printMonthCalendar
3. Call printMonthHeader
    call getMonthName
4. Call printMonthBody
    call getStartDay
    call getNumDaysInMonth
        call isLeapYear

For the second program, we'll just just call in a for loop to get all
calendar dates.

The getMonthName method can be implemented as a simple lookup table.

The isLeapYear method can be implemented using the gregorian calendar
formula of divisible by four, unless its divisible by 100, then it must
also be divisible by 400.

The printMonthHeader can be implemnted using a series of print statements,
with a method to determine how many spaces must be prepended to center the
header file

The printMonthCalendar method can be implemented using a while loop which
breaks when the number of days > the number of days in a month (also
implemented as a LUT), and then tabs a newline after we run out of columns
in a row.
```

Implementation

Print Month

```
package mod4;
/* PrintMonth
 *
 * Program which takes in a month and year and prints out the calendar for
that month
```

```
 *
 * @author Duncan Parke
 */

import java.util.Scanner;

public class PrintMonth {
    // Print the calendar for a given month
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Please enter the month you wish to see the
calendar of (1-12): ");
        int month = input.nextInt();
        System.out.print("Please enter the year you want to see the
calendar for (YYYY): ");
        int year = input.nextInt();
        printMonthCalendar(month, year);
        input.close();
    }

    public static void printMonthCalendar(int month, int year) {
        printMonthHeader(month, year);
        printMonthBody(month, year);
    }

    /* printMonthHeader
     *
     * The printMonthHeader method prints the header of the calendar for a
given month
     * and year. It does so in a format perscribed by the module four
assignment
     * of the 605.201 course.
     *
     * Pre-Conditions: The month value, m, is 1-12 The day value, d, is 1-
31, or
     *      1-28, 1-29 for February The year value, y, is the full year
(e.g., 2012)
     *
     * @param month
     * @param year
     *
     */
    static void printMonthHeader(int month, int year) {
        // Assumes 29 total characters per row
        // Counted manually from the assignment sheet
        String headerString = getMonthName(month) + " " + year;
        // Pad with spaces in order to center the header
        int padding = (29 - headerString.length())/2;
        for (int i = 0; i < padding; i++) {
            System.out.print(" ");
        }
        System.out.print(headerString);
        for (int i = 0; i < padding; i++) {
            System.out.print(" ");
        }
        System.out.println();
        System.out.println("----------------------------");
        System.out.println(" Sun Mon Tue Wed Thu Fri Sat ");
    }

    /* printMonthBody
     *
     * The printMonthBody method prints the body of the calendar for a
given month
     * and year. It prints the days of the month in a grid format, with the
first
     * day of the month starting in the correct column.
     *
     * Pre-Conditions: The month value, m, is 1-12 The day value, d, is 1-
31, or
     *      1-28, 1-29 for February The year value, y, is the full year
(e.g., 2012)
     *
     * @param month
     * @param year
     *
     */

    static void printMonthBody(int month, int year) {
        // Get the start day of the week, and the number
        // of days in the month
        int numDays = getNumDaysInMonth(month, year);
        int startDay = getStartDay(month, year);
        int realStartDay = 0;
        int colCount = 0;
        int dayCounter = 1; // Start on day 1
        // 0 index using realStartDay
        if(startDay != 7) realStartDay = startDay;
        // Prints, always assuming a space to start
        // Single digit days are space_space_num
        // double digit days are "space_num_num"
        // System.out.print(" "); // space to start
        // Print spaces until the start day
        // Initialze with one space offset
        for(int i = 0; i < realStartDay; i++) {
            System.out.print("    ");
        }
        colCount = realStartDay;
        while(dayCounter < numDays + 1) {
            if(dayCounter < 10) {
                System.out.print("   " + dayCounter);
            }
            else {
                System.out.print("  " + dayCounter);
            }
            dayCounter += 1;
            colCount += 1;
            if ( colCount == 7 ) {
                colCount = 0;
                System.out.println(" "); // zero pad
            }
        }
        System.out.println(); // newline for preference
    }

    /* getMonthName
     * Get month name implements a look up table that returns the string
value of a
     * month given the integer value of the month
     *
     * Pre-Conditions: The month value, m, is 1-12
     *
     * Post-Conditions: A string value is returned, representing the name
of the month
     *
     * @param month
     * @return monthName (String)
     */

    static String getMonthName(int month) {
        switch(month) {
            case 1: return "January";
            case 2: return "February";
            case 3: return "March";
            case 4: return "April";
            case 5: return "May";
            case 6: return "June";
            case 7: return "July";
            case 8: return "August";
            case 9: return "September";
            case 10: return "October";
            case 11: return "November";
            case 12: return "December";
            default: return "Invalid month";
        }
    }

    /** getStartDay
    * The method getStartDay() implements Zeller's Algorithm for
determining the
    * day of the week the first day of a month is. The method adjusts
Zeller's
    * numbering scheme for day of week ( 0=Saturday, ..., 6=Friday ) to
conform
    * to day of week number that corresponds to ISO conventions (i.e.,
    * 1=Monday, ..., 7=Sunday)
    *
    * Pre-Conditions: The month value, m, is 1-12 The day value, d, is 1-
31, or
    * 1-28, 1-29 for February The year value, y, is the full year (e.g.,
2012)
    *
```

```
     * Post-Conditions: A value of 1-7 is returned, representing the first
day of
     * the month 1 = Monday, ..., 7 = Sunday
     *
     * Reference: https://codereview.stackexchange.com/questions/67722/its-
friday-zellers-congruence-revisited
     *
     * @param month
     * @param year
     * @return dayNum (1 = Mon, ... , 7 = Sun )
     *
     * precondition: month, year, and day are integer types
     *
     * @author: rolfl, modified by Josh Lafond
     */
    static int getStartDay(int month, int year) {
        // Adjust month number & year to fit Zeller's numbering system
        if (month < 3) {
           month += 12;
           year -= 1;
        }

        int centuryYear = year % 100;     // Calculate year within century
        int centuryTerm = year / 100;     // Calculate century term
        int firstDayInMonth = 0;          // Day number of first day in
month 'm'

        firstDayInMonth = (1 + // to shift index 0 to the 1-7 return range
           (13 * (month + 1) / 5)
           + centuryYear +
           (centuryYear / 4) +
           (centuryTerm / 4) +
           (5 * centuryTerm)) % 7;

        // Convert Zeller's value to ISO value (1 = Mon, ... , 7 = Sun )
        int dayNum = ((firstDayInMonth + 5) % 7) + 1;

        return dayNum;
     } // end getStartDay()

    /** getNumDaysInMonth
     * The method getStartDay implements a lookup table which returns the
number of
     * days in a specific month. The method adjusts for leap years.
     *
     * Pre-Conditions: The month value, m, is 1-12 The day value, d, is 1-
31, or
     * 1-28, 1-29 for February The year value, y, is the full year (e.g.,
2012)
     *
     * Post-Conditions: An integer value is returned, representing the
number of
     * days in a month
     *
     *
     * @param month
     * @param year
     * @return numDays(28, 29, 30, or 31)
     *
     * precondition: month and year are integer types
     *
     * @author: dparke
     */
    static int getNumDaysInMonth(int month, int year) {
        switch(month) {
            // Jan, Mar, May, Jul, Aug, Oct, Dec == 31
            case 1:
            case 3:
            case 5:
            case 7:
            case 8:
            case 10:
            case 12: return 31;
            // Apr, Jun, Sep, Nov == 30
            case 4:
            case 6:
            case 9:
            case 11: return 30;
            // If Feb, return 28 if not a leap year
            // else, return 29
            case 2: return isLeapYear(year) ? 29 : 28;
            // error case
            default: return 0;
        }

    }
    /** isLeapYear
     * The method isLeapYear determines if a given year of the Gregorian
calendar is a
     * leap year or not
     *
     * Pre-Conditions: The year is of integer type
     *
     * Post-Conditions: A value of true or false is returned, indicating if
the year is a leap year
     *
     * Reference: https://codereview.stackexchange.com/questions/67722/its-
friday-zellers-congruence-revisited
     *
     * @param year
     * @return boolean (true if leap year, false if not)

     *
     * @author: dparke
     */
    static boolean isLeapYear(int year) {
        if ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0)) {
            return true;
        } else {
            return false;
        }
    }

}
```

Print Year

```
package mod4;
/* PrintYear
*
* Program which takes in a year and prints out the calendar for that year
*
* @author Duncan Parke
*/
import java.util.Scanner;

// Class to print the calendar for a given year
public class PrintYear {
    // Function entry point
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Please enter the year you want to see the
calendar for (YYYY): ");
        int year = input.nextInt();
        for(int i = 1; i <= 12; i++) {
            PrintMonth.printMonthCalendar(i, year);
        }
        input.close();
    }
}
```

Output

```
Script started on 2025-02-16 16:55:48-05:00 [TERM="xterm-256color"
TTY="/dev/pts/5" COLUMNS="120" LINES="13"]
[?2004h]0;arkosh@Mjolnir:
~/dev/605_201/module4/mod4[01;32markosh@Mjolnir[00m:[01;34m~/dev/605_20
1/module4/mod4[00m$ java/P[K[K
PrintMom[Knth.[K[K[K[Knth.java
[?2004l
Please enter the month you wish to see the calendar of (1-12): 3
Please enter the year you want to see the calendar for (YYYY): 2022
         March 2022
-----------------------------
 Sun Mon Tue Wed Thu Fri Sat
           1   2   3   4   5
   6   7   8   9  10  11  12
```

```
  13  14  15  16  17  18  19
  20  21  22  23  24  25  26
  27  28  29  30  31
[?2004h]0;arkosh@Mjolnir:
~/dev/605_201/module4/mod4[01;32markosh@Mjolnir[00m:[01;34m~/dev/605_20
1/module4/mod4[00m$ exit
[?2004l
exit

Script done on 2025-02-16 16:56:12-05:00 [COMMAND_EXIT_CODE="0"]
```

```
Script started on 2025-02-16 16:54:40-05:00 [TERM="xterm-256color"
TTY="/dev/pts/5" COLUMNS="120" LINES="13"]
[?2004h]0;arkosh@Mjolnir:
~/dev/605_201[01;32markosh@Mjolnir[00m:[01;34m~/dev/605_201[00m$ ^C[?
2004l
[?2004h
[?2004l

[?2004h]0;arkosh@Mjolnir:
~/dev/605_201[01;32markosh@Mjolnir[00m:[01;34m~/dev/605_201[00m$  cd
/home/arkosh/dev/605_201 ; /usr/bin/env /usr/lib/jvm/java-21-openjdk-
amd64/bin/java -XX:+ShowCodeDetailsInExceptionMessages -cp
/home/arkosh/.vscode-
server/data/User/workspaceStorage/c10569d5ef0ee75b2280be6f131d5434/redhat.j
ava/jdt_ws/605_201_1aef2a31/bin mod4.PrintYear
[?2004l
Please enter the year you want to see the calendar for (YYYY): 2024
        January 2024
-----------------------------
 Sun Mon Tue Wed Thu Fri Sat
       1   2   3   4   5   6
   7   8   9  10  11  12  13
  14  15  16  17  18  19  20
  21  22  23  24  25  26  27
  28  29  30  31
        February 2024
-----------------------------
 Sun Mon Tue Wed Thu Fri Sat
                   1   2   3
   4   5   6   7   8   9  10
  11  12  13  14  15  16  17
  18  19  20  21  22  23  24
  25  26  27  28  29
         March 2024
-----------------------------
 Sun Mon Tue Wed Thu Fri Sat
                       1   2
   3   4   5   6   7   8   9
  10  11  12  13  14  15  16
  17  18  19  20  21  22  23
  24  25  26  27  28  29  30
  31
         April 2024
-----------------------------
 Sun Mon Tue Wed Thu Fri Sat
       1   2   3   4   5   6
   7   8   9  10  11  12  13
  14  15  16  17  18  19  20
  21  22  23  24  25  26  27
  28  29  30
          May 2024
-----------------------------
 Sun Mon Tue Wed Thu Fri Sat
               1   2   3   4
   5   6   7   8   9  10  11
  12  13  14  15  16  17  18
  19  20  21  22  23  24  25
  26  27  28  29  30  31
          June 2024
-----------------------------
 Sun Mon Tue Wed Thu Fri Sat
                           1
   2   3   4   5   6   7   8
   9  10  11  12  13  14  15
  16  17  18  19  20  21  22
  23  24  25  26  27  28  29
  30
          July 2024
-----------------------------
 Sun Mon Tue Wed Thu Fri Sat
       1   2   3   4   5   6
   7   8   9  10  11  12  13
  14  15  16  17  18  19  20
  21  22  23  24  25  26  27
  28  29  30  31
         August 2024
-----------------------------
 Sun Mon Tue Wed Thu Fri Sat
                   1   2   3
   4   5   6   7   8   9  10
  11  12  13  14  15  16  17
  18  19  20  21  22  23  24
  25  26  27  28  29  30  31

       September 2024
-----------------------------
 Sun Mon Tue Wed Thu Fri Sat
   1   2   3   4   5   6   7
   8   9  10  11  12  13  14
  15  16  17  18  19  20  21
  22  23  24  25  26  27  28
  29  30
        October 2024
-----------------------------
```

```
 Sun Mon Tue Wed Thu Fri Sat
           1   2   3   4   5
   6   7   8   9  10  11  12
  13  14  15  16  17  18  19
  20  21  22  23  24  25  26
  27  28  29  30  31
        November 2024
-----------------------------
 Sun Mon Tue Wed Thu Fri Sat
                       1   2
   3   4   5   6   7   8   9
  10  11  12  13  14  15  16
  17  18  19  20  21  22  23
  24  25  26  27  28  29  30

        December 2024
-----------------------------
 Sun Mon Tue Wed Thu Fri Sat
   1   2   3   4   5   6   7
   8   9  10  11  12  13  14
  15  16  17  18  19  20  21
  22  23  24  25  26  27  28
  29  30  31
[?2004h]0;arkosh@Mjolnir:
~/dev/605_201[01;32markosh@Mjolnir[00m:[01;34m~/dev/605_201[00m$ exit
[?2004l
exit

Script done on 2025-02-16 16:54:54-05:00 [COMMAND_EXIT_CODE="0"]
```