# Java Programming AP Edition

## U2R2 Review on Unit 2

DESIGN AND IMPLEMENTATION

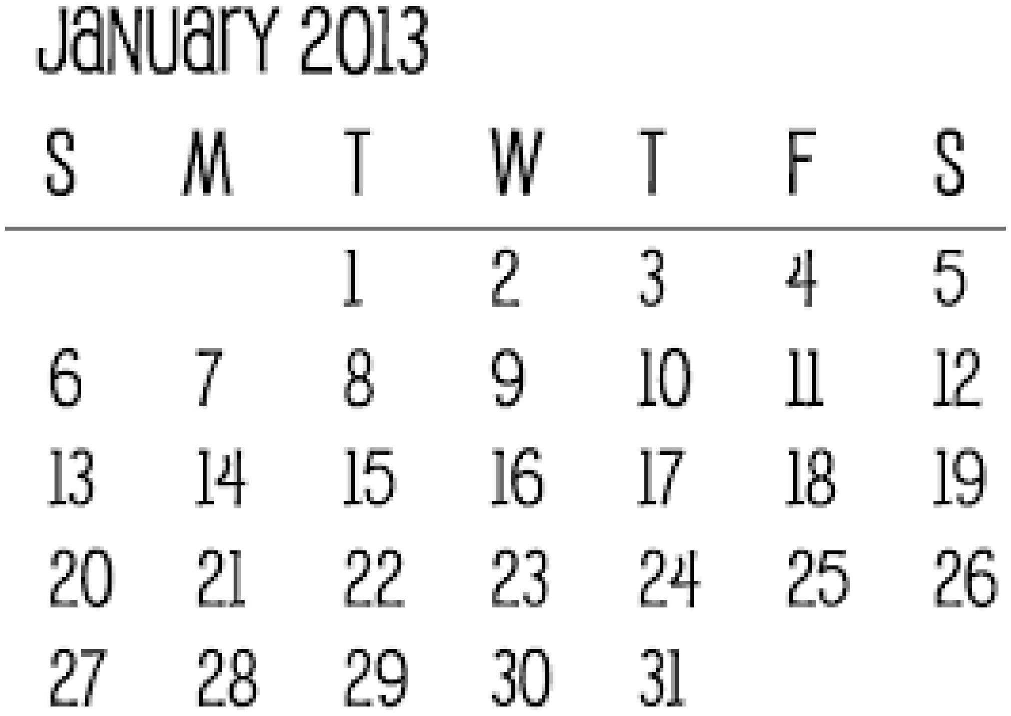
ERIC Y. CHOU, PH.D. IEEE SENIOR MEMBER

# PrintCalender Case Study

## Top-Down Design

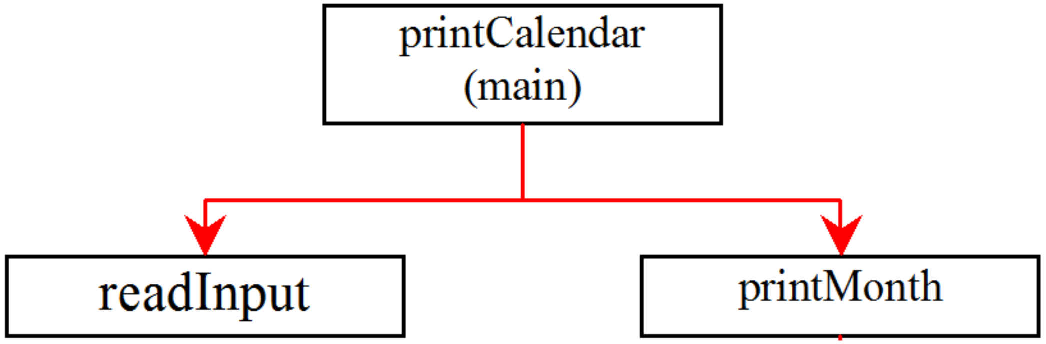
Let us use the PrintCalendar example (Unit Project) to demonstrate the **stepwise refinement** approach (Using methods).

PrintCalendar.java



January 2013

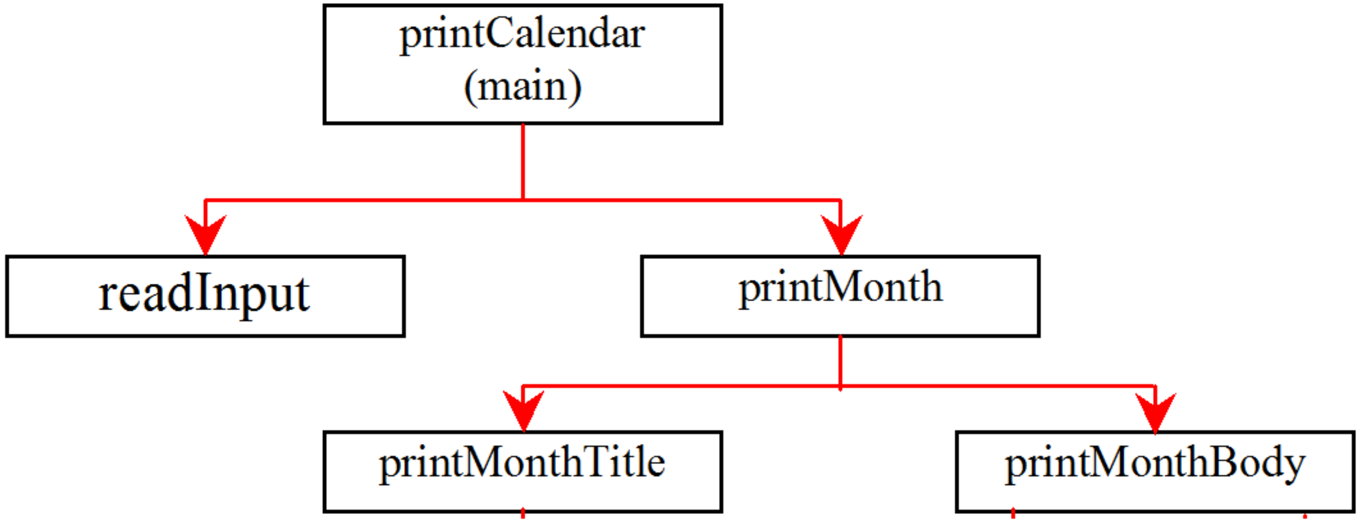
| S | M | T | W | T | F | S |
|---|---|---|---|---|---|---|
|  |  | 1 | 2 | 3 | 4 | 5 |
| 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| 20 | 21 | 22 | 23 | 24 | 25 | 26 |
| 27 | 28 | 29 | 30 | 31 |  |  |



printCalendar (main)

readInput

printMonth



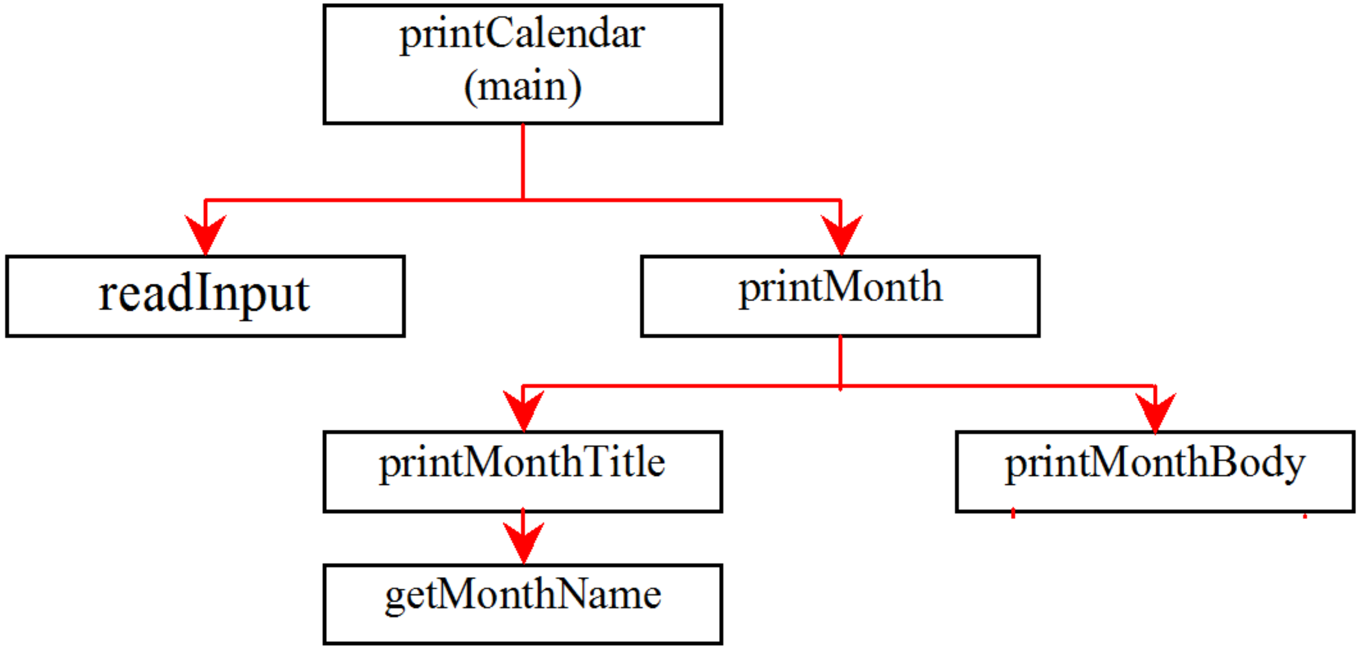
printCalendar
(main)

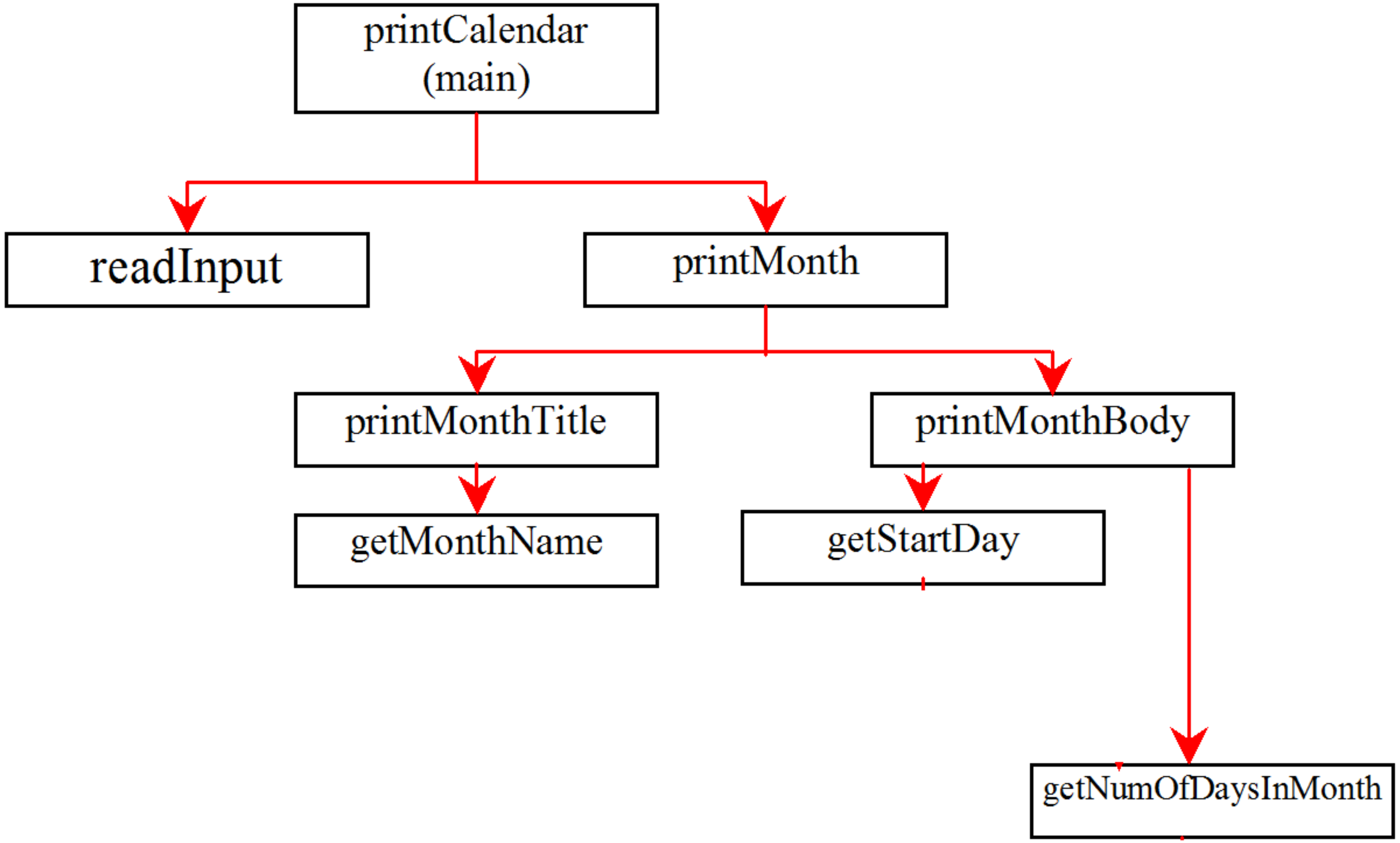
readInput

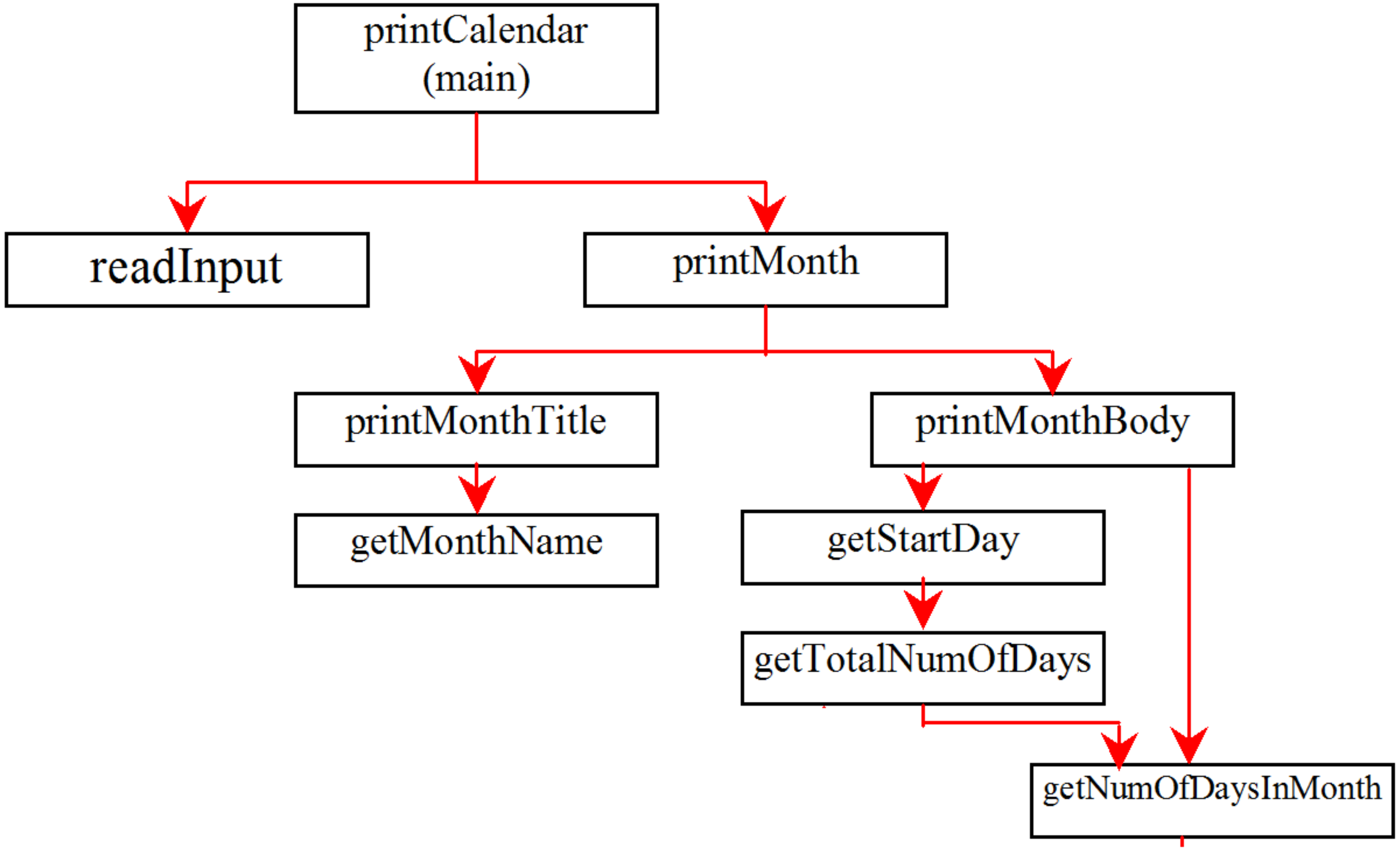
printMonth

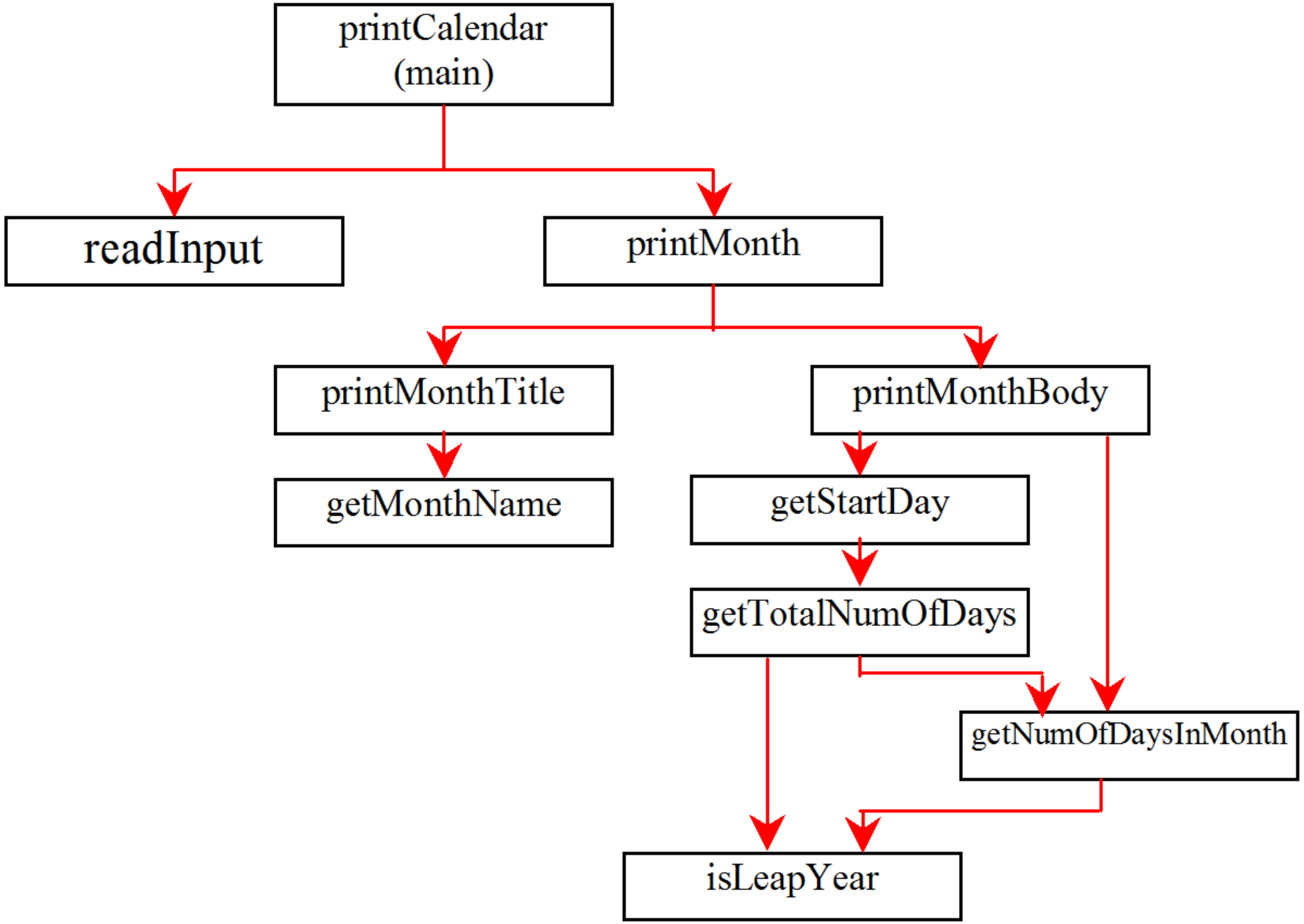
printMonthTitle

printMonthBody


printCalendar
(main)
readInput
printMonth
printMonthTitle
printMonthBody
getMonthName


printCalendar
(main)
readInput
printMonth
printMonthTitle
printMonthBody
getMonthName
getStartDay
getNumOfDaysInMonth


printCalendar
(main)
readInput
printMonth
printMonthTitle
getMonthName
printMonthBody
getStartDay
getTotalNumOfDays
getNumOfDaysInMonth


printCalendar
(main)
readInput
printMonth
printMonthTitle
printMonthBody
getMonthName
getStartDay
getTotalNumOfDays
getNumOfDaysInMonth
isLeapYear

# Implementation: Top-Down

Top-down approach is to implement one method in the structure chart at a time from the top to the bottom. **Stubs** can be used for the methods waiting to be implemented. A **stub** is a simple but incomplete version of a method. The use of stubs enables you to test invoking the method from a caller.

Implement the main method first and then use a stub for the printMonth method. For example, let printMonth display the year and the month in the **stub**. Thus, your program may begin like this:

# Stubs (Main)

```
/** Main method */
public static void main(String[] args) {
  Scanner input = new Scanner(System.in);
  // Prompt the user to enter year
  System.out.print("Enter full year (e.g., 2015): ");
  int year = input.nextInt();
  // Prompt the user to enter month
  System.out.print("Enter month in number between 1 and 12: ");
  int month = input.nextInt();
  // Print calendar for the month of the year
  printMonth(year, month);
}
```

# Stubs (like Header File)

```
/** Print the calendar for a month in a year */
public static void printMonth(int year, int month) {}
/** Print the month title, e.g., May, 2015 */
public static void printMonthTitle(int year, int month) {}
/** Print month body */
public static void printMonthBody(int year, int month) {}
/** Get the English name for the month */
public static String getMonthName(int month) {}
/** Get the start day of month/1/year */
public static int getStartDay(int year, int month) {}
/** Get the total number of days since January 1, 1800 */
public static int getTotalNumberOfDays(int year, int month) {}
/** Get the number of days in a month */
public static int getNumberOfDaysInMonth(int year, int month) {}
/** Determine if it is a leap year */
public static boolean isLeapYear(int year) {}
```

# Implementation: Bottom-Up

Bottom-up approach is to implement one method in the structure chart at a time from the bottom to the top. For each method implemented, write a test program to test it. Both top-down and bottom-up methods are fine. Both approaches implement the methods incrementally and help to isolate programming errors and makes debugging easy. Sometimes, they can be used together.

# Usual Methodology of Design and Implementation

Top down design and bottom up implementation.

Review the Unit 1 Review for Spiral Design, V-Model, Waterfall model. Unit 1 models are for software development life cycle. (SDLC) **(Larger Scale)**

This lecture is for program development methodology. **(More details)**

Think about middle way out as well.

# Benefits of Stepwise Refinement

Stepwise refinement breaks a large problem into smaller manageable subproblems. Each subproblem can be implemented using a method.  This approach makes the program easier to write, reuse, debug, test, modify, and maintain.

- Simpler Program
- Reusing Methods
- Easier Developing, Debugging, and Testing
- Better Facilitating Teamwork