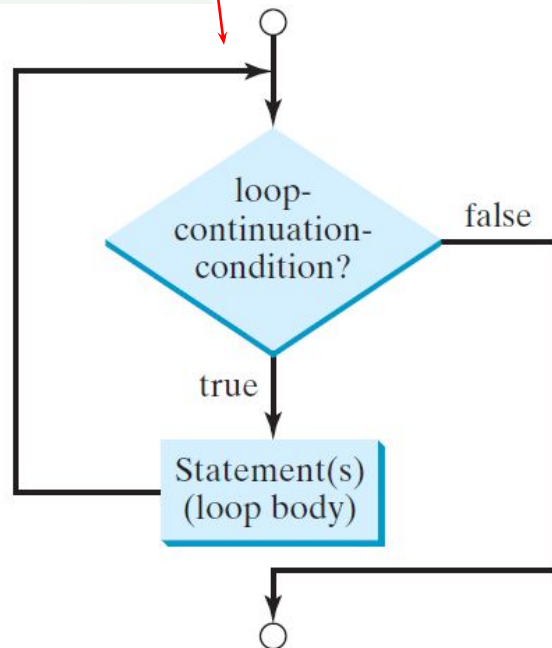# Chapter 5 Loops
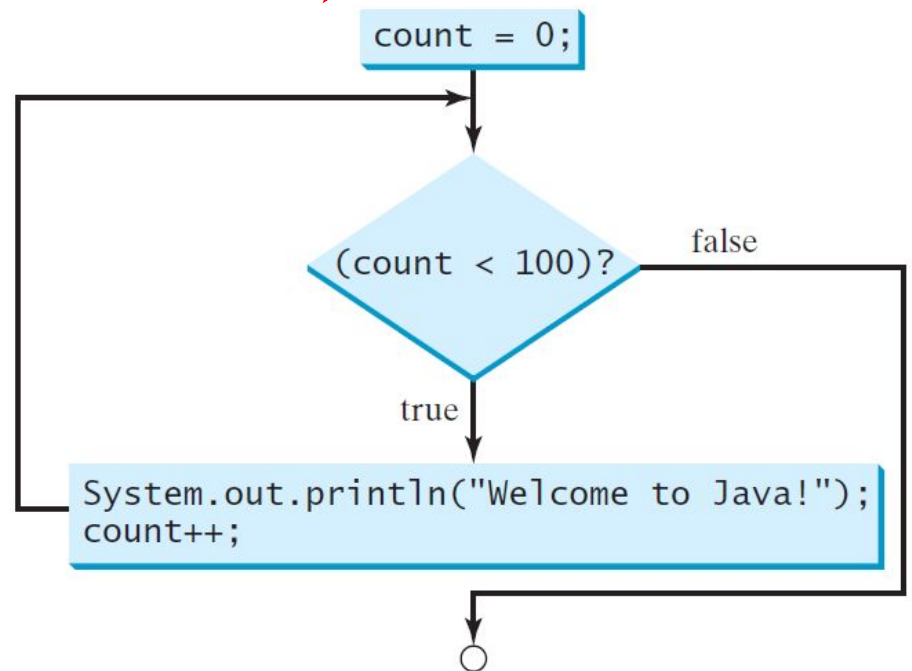# Chapter 6 Methods

1

# while Loop Flow Chart

while (loop-continuation-condition) {

  // loop-body;

  Statement(s);

}

int count = 0;

while (count < 100) {

  System.out.println("Welcome to Java!");

  count++;

}

Liang, Introduction to Java Programming, Eleventh Edition, (c) 2017 Pearson Education, Inc. All rights reserved.

2

# Trace while Loop

Initialize count

```
int count = 0;

while (count < 2) {

    System.out.println("Welcome to Java!");

    count++;

}
```

Uploaded By: Jibreel Bornat

3

# Problem: Repeat Addition Until Correct

Recall that Listing 3.1 AdditionQuiz.java gives a program that prompts the user to enter an answer for a question on addition of two single digits. Using a loop, you can now rewrite the program to let the user enter a new answer until it is correct.

RepeatAdditionQuiz    Run

```java
import java.util.Scanner;
public class RepeatAdditionQuiz {
    public static void main(String[] args) {
    int number1 = (int)(Math.random() * 10);
    int number2 = (int)(Math.random() * 10);

    // Create a Scanner
    Scanner input = new Scanner(System.in);
    System.out.print( "What is " + number1 + " + " + number2 + "? ");
    int answer = input.nextInt();

    while (number1 + number2 != answer) {
        System.out.print("Wrong answer. Try again. What is " +
         number1 + " + " + number2 + "? ");
        answer = input.nextInt();
    }
    System.out.println("You got it!");
    }
}
```

Uploaded By: Jibreel Bornat

5

# Ending a Loop with a Sentinel Value

You may use an input value to signify the end of the loop. Such a value is known as a *sentinel value*.

```java
Scanner input = new Scanner(System.in);
// Read an initial data
System.out.print( "Enter an integer (the input ends if it is 0): ");
int data = input.nextInt();
// Keep reading data until the input is 0 int sum = 0;
while (data != 0) {
    sum += data;
    // Read the next data
    System.out.print( "Enter an integer (the input ends if it is 0): ");
    data = input.nextInt();
}
```

SentinelValue    Run

Uploaded By: Jibreel Bornat    6

# Caution

Don't use floating-point values for equality checking in a loop control. Since floating-point values are approximations for some values, using them could result in imprecise counter values and inaccurate results. Consider the following code for computing $1 + 0.9 + 0.8 + ... + 0.1$:

```java
double item = 1; double sum = 0;
while (item != 0) { // No guarantee item will be 0
  sum += item;
  item -= 0.1;
}
System.out.println(sum);
```
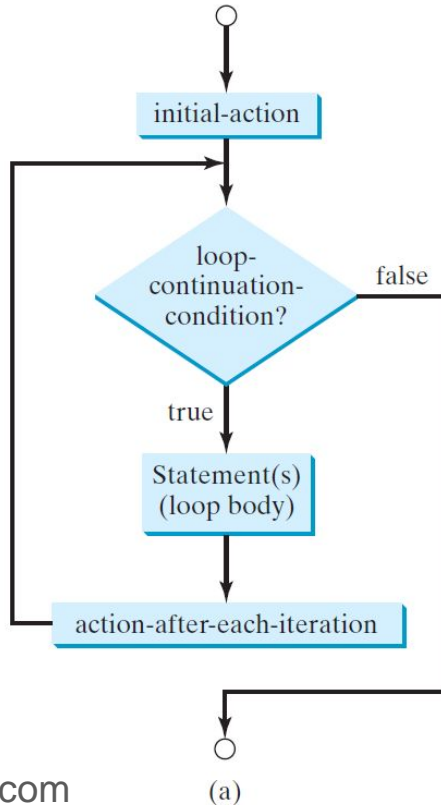
# do-while Loop



```
do {
    // Loop body;
    Statement(s);
} while (loop-continuation-condition);
```
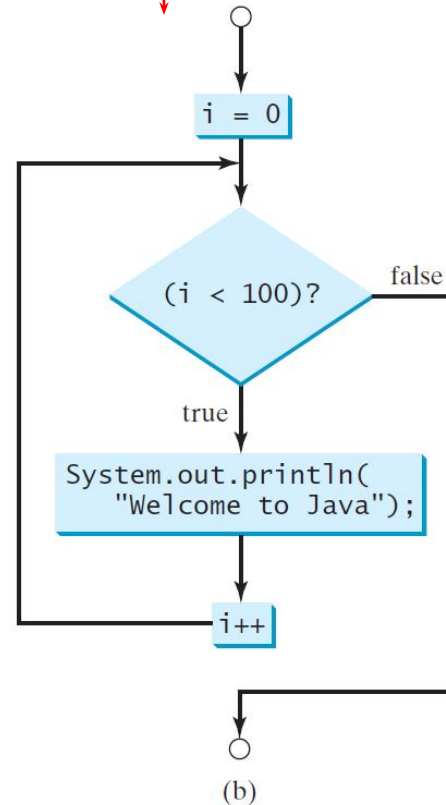
Uploaded By: Jibreel Bornat
8
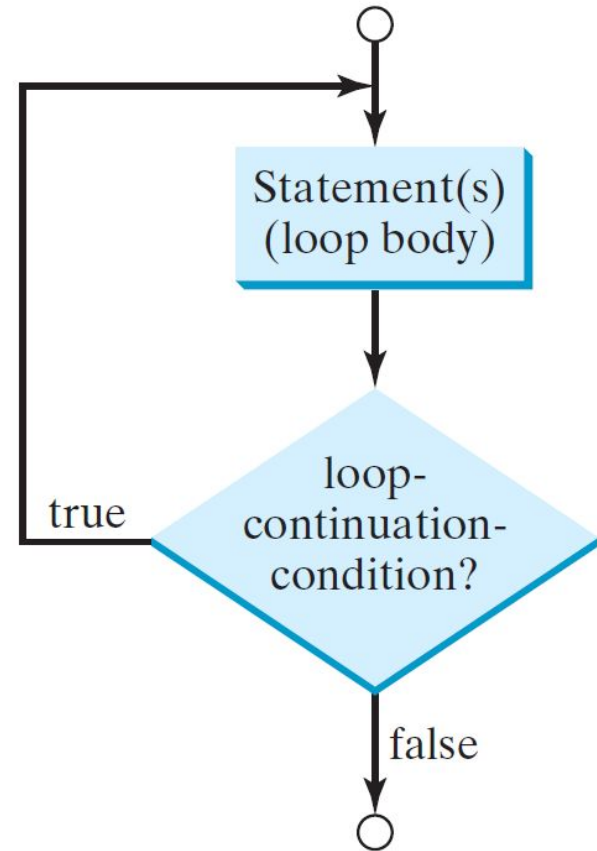
# for Loops

```
for (initial-action;
    loop-continuation-condition;
    action-after-each-iteration) {
    // loop body;
    Statement(s);
}
```

```
int i;
for (i = 0; i < 100; i++) {
    System.out.println(
        "Welcome to Java!");
}
```



(a)



(b)

Inc. All
Uploaded By: Jibreel Bornat 9

# Note

The <u>initial-action</u> in a <u>for</u> loop can be a list of zero or more comma-separated expressions. The <u>action-after-each-iteration</u> in a <u>for</u> loop can be a list of zero or more comma-separated statements. Therefore, the following two <u>for</u> loops are correct. They are rarely used in practice, however.

```java
for (int i = 1; i < 100; System.out.println(i++));


for (int i = 0, j = 0; (i + j < 10); i++, j++) {
  // Do something

}
```

Uploaded By: Jibreel Bornat

# Note

If the <u>loop-continuation-condition</u> in a <u>for</u> loop is omitted, it is implicitly true. Thus the statement given below in (a), which is an infinite loop, is correct. Nevertheless, it is better to use the equivalent loop in (b) to avoid confusion:

```
for ( ; ; ) {
   // Do something
}
```

(a)

Equivalent

```
while (true) {
   // Do something
}
```

(b)

Uploaded By: Jibreel Bornat 11

# Caution

Adding a semicolon at the end of the <u>for</u> clause before the loop body is a common mistake, as shown below:

Logic
Error

```
for (int i=0; i<10; i++);
{
    System.out.println("i is " + i);
}
```

Uploaded By: Jibreel Bornat

# Caution, cont.

Similarly, the following loop is also wrong:

```
int i=0;
while (i < 10);          Logic Error
{
  System.out.println("i is " + i);
  i++;
}
```

In the case of the <u>do</u> loop, the following semicolon is needed to end the loop.

```
int i=0;
do {
  System.out.println("i is " + i);
  i++;
} while (i<10);          Correct
```

Uploaded By: Jibreel Bornat

13

# Problem:
# Finding the Greatest Common Divisor

Problem: Write a program that prompts the user to enter two positive integers and finds their greatest common divisor.

```
// Prompt the user to enter two integers
System.out.print("Enter first integer: ");
int n1 = input.nextInt();
System.out.print("Enter second integer: ");
int n2 = input.nextInt();
int gcd = 1;
int k = 2;
while (k <= n1 && k <= n2) {
    if (n1 % k == 0 && n2 % k == 0)
        gcd = k;
    k++;
}
```

GreatestCommonDivisor      Run

STUDENTS-HUB.com      Uploaded By: Jibreel Bornat      14

# break

```java
public class TestBreak {
  public static void main(String[] args) {
    int sum = 0;
    int number = 0;

    while (number < 20) {
      number++;
      sum += number;
      if (sum >= 100)
        break;
    }

    System.out.println("The number is " + number);
    System.out.println("The sum is " + sum);
  }
}
```

Uploaded By: Jibreel Bornat 15

# continue

```java
public class TestContinue {
  public static void main(String[] args) {
    int sum = 0;
    int number = 0;

    while (number < 20) {
      number++;
      if (number == 10 || number == 11)
        continue;
      sum += number;
    }

    System.out.println("The sum is " + sum);
  }
}
```

Uploaded By: Jibreel Bornat
16

# Problem: Checking Palindrome

A string is a palindrome if it reads the same forward and backward. The words "mom," "dad," and "noon," for instance, are all palindromes.

```java
// Prompt the user to enter a string
System.out.print("Enter a string: ");
String s = input.nextLine();
// The index of the first character in the string
int low = 0;
// The index of the last character in the string
int high = s.length() - 1;
boolean isPalindrome = true;
while (low < high) {
    if (s.charAt(low) != s.charAt(high)) {
        isPalindrome = false;
        break;
    }
    low++;
    high--;
}
```

Palindrome    Run

# Problem: Displaying Prime Numbers

Problem: Write a program that displays the first 50 prime numbers in five lines, each of which contains 10 numbers. An integer greater than 1 is *prime* if its only positive divisor is 1 or itself. For example, 2, 3, 5, and 7 are prime numbers, but 4, 6, 8, and 9 are not.

Solution: The problem can be broken into the following tasks:
- For number = 2, 3, 4, 5, 6, ..., test whether the number is prime.
- Determine whether a given number is prime.
- Count the prime numbers.
- Print each prime number, and print 10 numbers per line.

```
for (int divisor = 2; divisor <= number / 2; divisor++) {
    if (number % divisor == 0) {
        isPrime = false;
        break;
    }
}
```

PrimeNumber    Run

# Problem

```
int sum = 0;
for (int i = 1; i <= 10; i++)
    sum += i;
System.out.println("Sum from 1 to 10 is " + sum);

sum = 0;
for (int i = 20; i <= 30; i++)
    sum += i;
System.out.println("Sum from 20 to 30 is " + sum);

sum = 0;
for (int i = 35; i <= 45; i++)
    sum += i;
System.out.println("Sum from 35 to 45 is " + sum);
```

# Solution

```
public static int sum(int i1, int i2) {
  int sum = 0;
  for (int i = i1; i <= i2; i++)
    sum += i;
  return sum;
}
```

MethodDemo   Run

```
public static void main(String[] args) {
  System.out.println("Sum from 1 to 10 is " + sum(1, 10));
  System.out.println("Sum from 20 to 30 is " + sum(20, 30));
  System.out.println("Sum from 35 to 45 is " + sum(35, 45));
}
```

Uploaded By: Jibreel Bornat

# Defining Methods

A method is a collection of statements that are grouped together to perform an operation.

Define a method

```java
public static int max(int num1, int num2) {

    int result;

    if (num1 > num2)
        result = num1;
    else
        result = num2;

    return result;
}
```
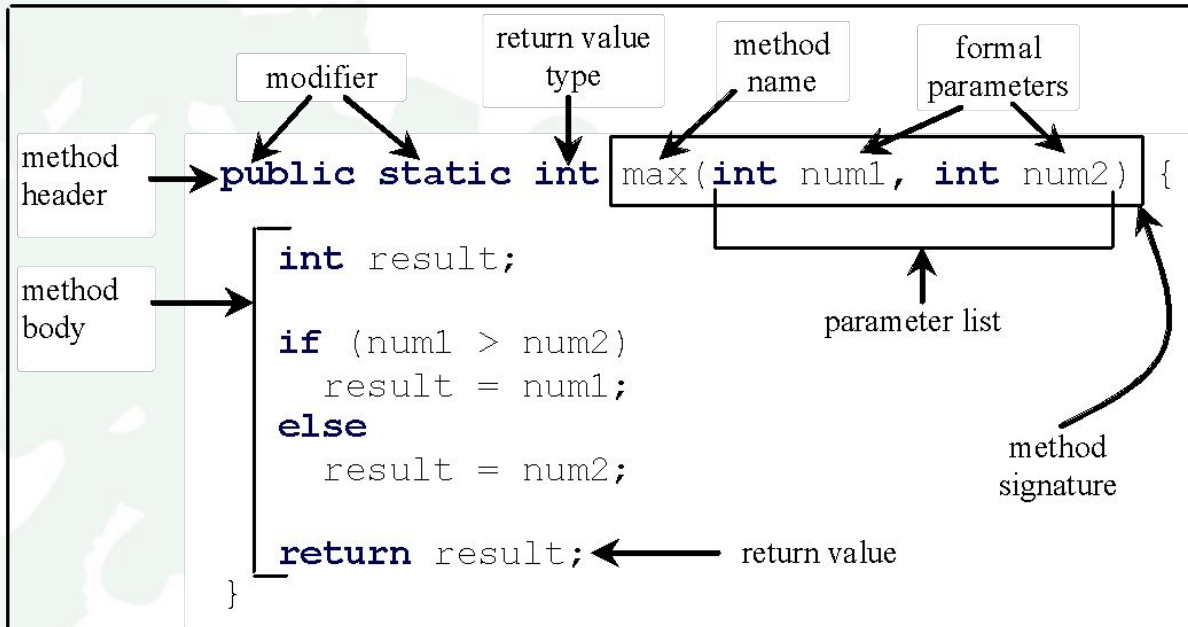
Invoke a method

```java
int z = max(x, y);
```

actual parameters
(arguments)
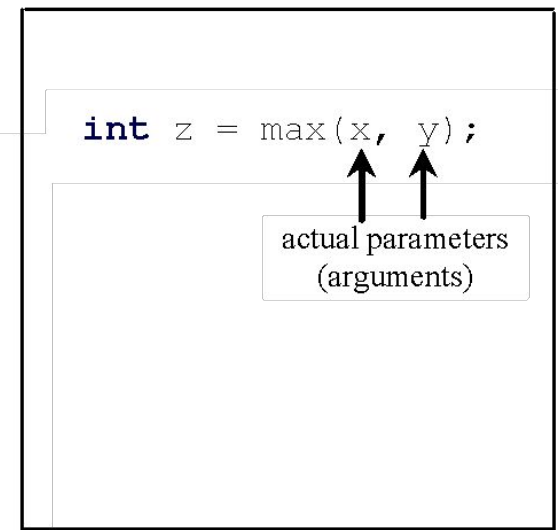
STUDENTS-HUB.com

Uploaded By: Jibreel Bornat

21

# Defining Methods

A method is a collection of statements that are grouped together to perform an operation.



Define a method

Invoke a method

method header → `public static int max(int num1, int num2) {`

modifier, return value type, method name, formal parameters

method body → 
```
    int result;

    if (num1 > num2)
        result = num1;
    else
        result = num2;

    return result;
}
```

parameter list

method signature

return value

`int z = max(x, y);`

actual parameters (arguments)

Uploaded By: Jibreel Bornat

22

# Method Signature

*Method signature* **is the combination of the method name and the parameter list.**

Uploaded By: Jibreel Bornat

23

# Calling Methods, cont.

pass the value of i

pass the value of j

```
public static void main(String[] args) {
  int i = 5;
  int j = 2;
  int k = max(i, j);

  System.out.println(
   "The maximum between " + i +
   " and " + j + " is " + k);
}
```

```
public static int max(int num1, int num2) {
  int result;

  if (num1 > num2)
    result = num1;
  else
    result = num2;

  return result;
}
```
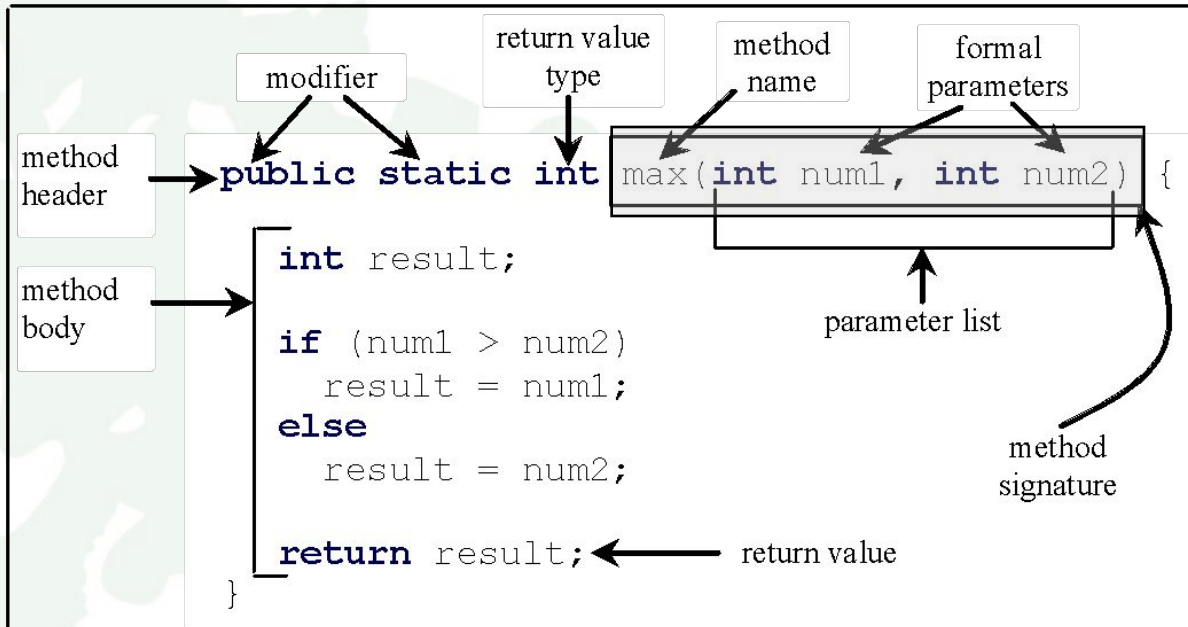
Uploaded By: Jibreel Bornat
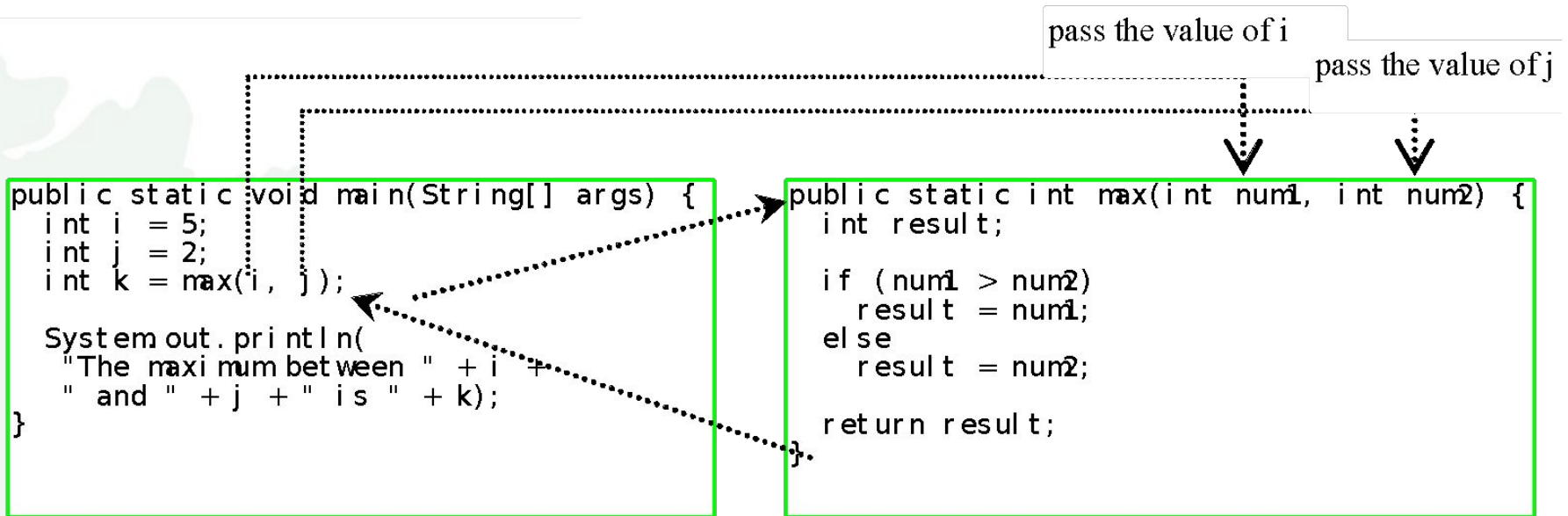
# Trace Method Invocation

> i is now 5

```java
public static void main(String[] args) {
   int i = 5;
   int j = 2;
   int k = max(i, j);

   System.out.println(
     "The maximum between " + i +
     " and " + j + " is " + k);
}
```

```java
public static int max(int num1, int num2) {
   int result;

   if (num1 > num2)
     result = num1;
   else
     result = num2;

   return result;
}
```

Uploaded By: Jibreel Bornat
25

# CAUTION

A <u>return</u> statement is required for a value-returning method. The method shown below in (a) is logically correct, but it has a compilation error because the Java compiler thinks it possible that this method does not return any value.

```java
public static int sign(int n) {
  if (n > 0)
    return 1;
  else if (n == 0)
    return 0;
  else if (n < 0)
    return -1;
}
```
(a)

Should be →

```java
public static int sign(int n) {
  if (n > 0)
    return 1;
  else if (n == 0)
    return 0;
  else
    return -1;
}
```
(b)

To fix this problem, delete *if (n < 0)* in (a), so that the compiler will see a <u>return</u> statement to be reached regardless of how the <u>if</u> statement is evaluated.

STUDENTS-HUB.com

Uploaded By: Jibreel Bornat

# Trace Call Stack

i is declared and initialized

```
public static void main(String[] args) {
  int i = 5;
  int j = 2;
  int k = max(i, j);

  System.out.println(
    "The maximum between " + i +
    " and " + j + " is " + k);
}
```

```
public static int max(int num1, int num2) {
  int result;

  if (num1 > num2)
    result = num1;
  else
    result = num2;

  return result;
}
```

i: 5

The main method
is invoked.

Uploaded By: Jibreel Bornat
27

# Trace Call Stack

pass the values of i and j to num1 and num2

```
public static void main(String[] args) {
  int i = 5;
  int j = 2;
  int k = max(i, j);

  System.out.println(
   "The maximum between " + i +
   " and " + j + " is " + k);
}
```
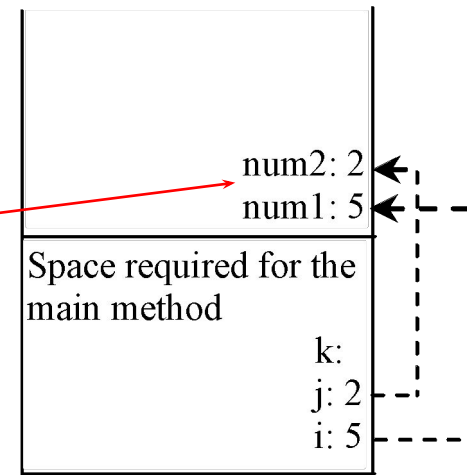
```
public static int max(int num1, int num2) {
  int result;

  if (num1 > num2)
    result = num1;
  else
    result = num2;

  return result;
}
```

num2: 2
num1: 5

Space required for the main method

k:
j: 2
i: 5

The max method is invoked.

Uploaded By: Jibreel Bornat
28

# Trace Call Stack

Return result and assign it to k

```
public static void main(String[] args) {
  int i = 5;
  int j = 2;
  int k = max(i, j);

  System.out.println(
   "The maximum between " + i +
   " and " + j + " is " + k);
}
```

```
public static int max(int num1, int num2) {
  int result;

  if (num1 > num2)
    result = num1;
  else
    result = num2;

  return result;
}
```
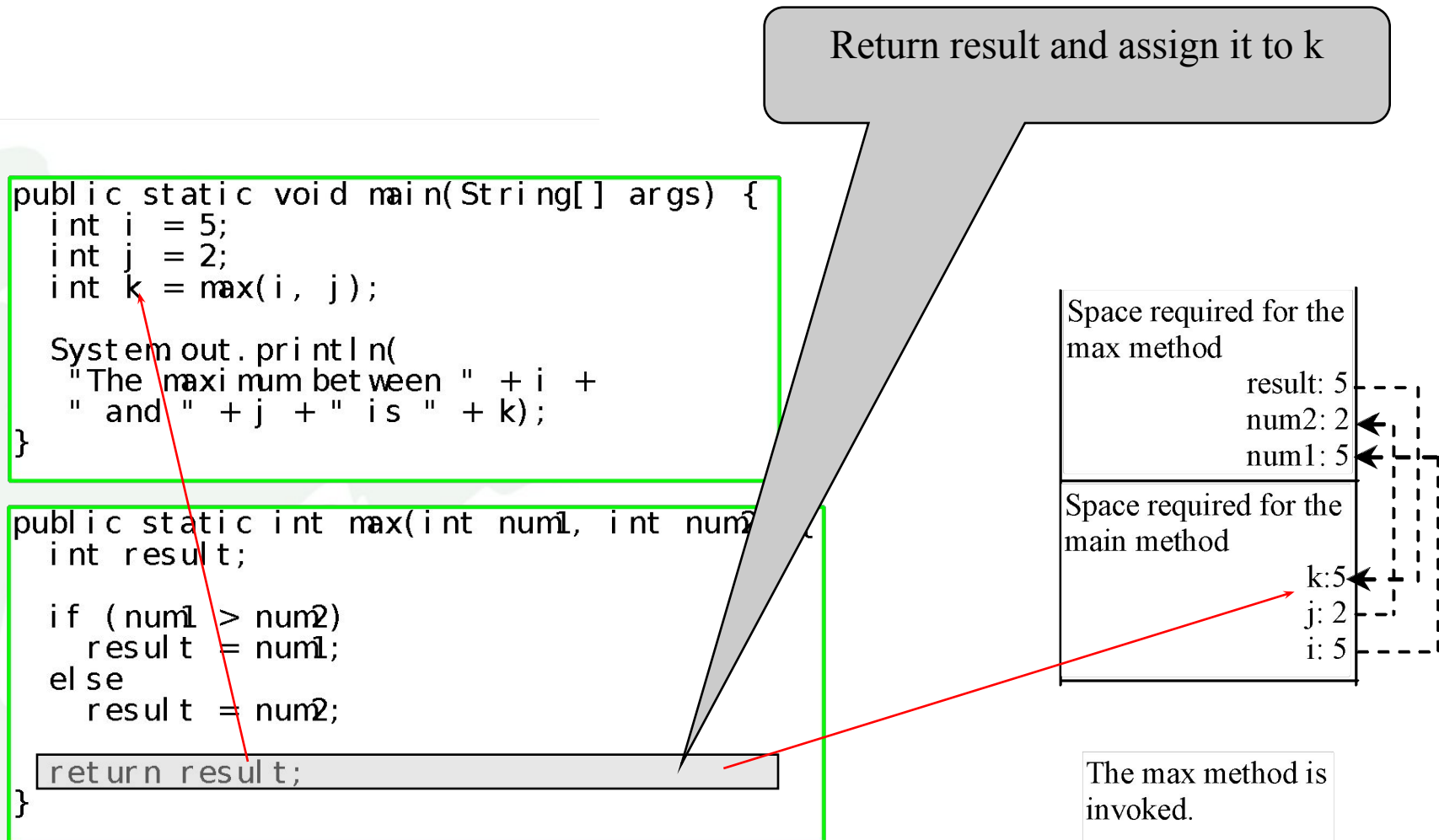
| Space required for the max method | |
| --- | --- |
| | result: 5 |
| | num2: 2 |
| | num1: 5 |

| Space required for the main method | |
| --- | --- |
| | k:5 |
| | j: 2 |
| | i: 5 |

The max method is invoked.

Uploaded By: Jibreel Bornat

# Trace Call Stack

Execute print statement

```
public static void main(String[] args) {
  int i = 5;
  int j = 2;
  int k = max(i, j);

  System.out.println(
  "The maximum between " + i +
  " and " + j + " is " + k);
}
```

```
public static int max(int num1, int num2) {
  int result;

  if (num1 > num2)
    result = num1;
  else
    result = num2;

  return result;
}
```

Space required for the
main method

k:5
j: 2
i: 5

The main method
is invoked.

Uploaded By: Jibreel Bornat

# Passing Parameters

```
public static void nPrintln(String message, int n) {
  for (int i = 0; i < n; i++)
    System.out.println(message);
}
```

Suppose you invoke the method using
    nPrintln("Welcome to Java", 5);
What is the output?

Suppose you invoke the method using
    nPrintln("Computer Science", 15);
What is the output?

Can you invoke the method using
    nPrintln(15, "Computer Science");

# Pass by Value

This program demonstrates passing values to the methods.

```java
public static void swap(int n1, int n2) {
    System.out.println("\tInside the swap method");
    System.out.println("\t\tBefore swapping, n1 is " + n1 +
    " and n2 is " + n2);
    // Swap n1 with n2
    int temp = n1; n1 = n2;
    n2 = temp;
    System.out.println("\t\tAfter swapping, n1 is " + n1 + " and n2 is " + n2);
}
```

Run

TestPassByValue

# Overloading Methods

*Overloading methods enable you to define the methods with the same name as long as their parameter lists are different.*

```java
public static int max(int num1, int num2) {
    if (num1 > num2)
        return num1;
    else
        return num2;
}

public static double max(double num1, double num2) {
    if (num1 > num2)
        return num1;
    else
        return num2;
}

public static double max(double num1, double num2, double num3) {
    return max(max(num1, num2), num3);
}
```

TestMethodOverloading

33

# Ambiguous Invocation: Error

```java
public class AmbiguousOverloading {
  public static void main(String[] args) {
    System.out.println(max(1, 2));
  }

  public static double max(int num1, double num2) {
    if (num1 > num2)
      return num1;
    else
      return num2;
  }

  public static double max(double num1, int num2) {
    if (num1 > num2)
      return num1;
    else
      return num2;
  }
}
```

 Uploaded By: Jibreel Bornat 34

# Scope of Local Variables

A local variable: a variable defined inside a method.

Scope: the part of the program where the variable can be referenced.

The scope of a local variable starts from its declaration and continues to the end of the block that contains the variable. A local variable must be declared before it can be used.

You can declare a local variable with the same name multiple times in different non-nesting blocks in a method, but you cannot declare a local variable twice in nested blocks.

Uploaded By: Jibreel Bornat

# Scope of Local Variables, cont.

It is fine to declare i in two non-nesting blocks

```
public static void method1() {
    int x = 1;
    int y = 1;

    for (int i = 1; i < 10; i++) {
        x += i;
    }

    for (int i = 1; i < 10; i++) {
        y += i;
    }
}
```

It is wrong to declare i in two nesting blocks

```
public static void method2() {

    int i = 1;
    int sum = 0;

    for (int i = 1; i < 10; i++)
        sum += i;
    }
}
```

 Uploaded By: Jibreel Bornat 36

# Scope of Local Variables, cont.

```java
// Fine with no errors
public static void correctMethod() {
    int x = 1;
    int y = 1;
    // i is declared
    for (int i = 1; i < 10; i++) {
        x += i;
    }
    // i is declared again
    for (int i = 1; i < 10; i++) {
        y += i;
    }
}
```
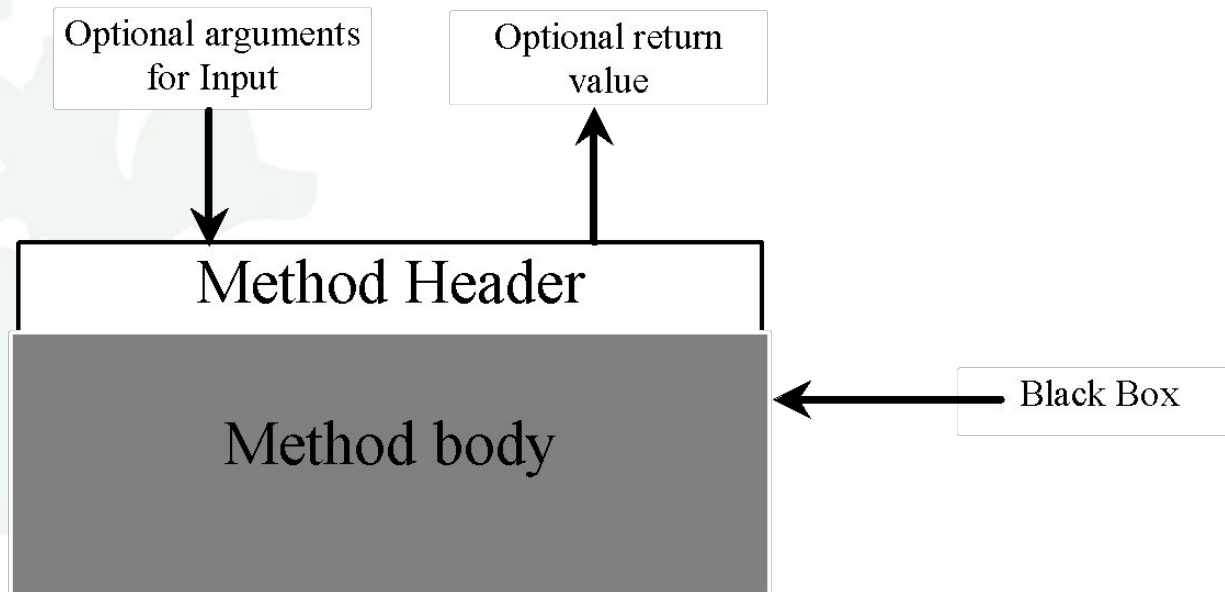
Uploaded By: Jibreel Bornat

# Scope of Local Variables, cont.

```
// With errors
public static void incorrectMethod() {
    int x = 1;
    int y = 1;
    for (int i = 1; i < 10; i++) {
      int x = 0;
        x += i;
    }
}
```

STUDENTS-HUB.com

Uploaded By: Jibreel Bornat

# Method Abstraction

You can think of the method body as a black box that contains the detailed implementation for the method.

Uploaded By: Jibreel Bornat
39

# Case Study: Generating Random Characters, cont.

To generalize the foregoing discussion, a random character between any two characters ch1 and ch2 with ch1 < ch2 can be generated as follows:

(char)(ch1 + Math.random() * (ch2 – ch1 + 1))

Uploaded By: Jibreel Bornat

# The RandomCharacter Class

```java
// RandomCharacter.java: Generate random characters
public class RandomCharacter {
  /** Generate a random character between ch1 and ch2 */
  public static char getRandomCharacter(char ch1, char ch2) {
    return (char)(ch1 + Math.random() * (ch2 - ch1 + 1));
  }

  /** Generate a random lowercase letter */
  public static char getRandomLowerCaseLetter() {
    return getRandomCharacter('a', 'z');
  }

  /** Generate a random uppercase letter */
  public static char getRandomUpperCaseLetter() {
    return getRandomCharacter('A', 'Z');
  }

  /** Generate a random digit character */
  public static char getRandomDigitCharacter() {
    return getRandomCharacter('0', '9');
  }

  /** Generate a random character */
  public static char getRandomCharacter() {
    return getRandomCharacter('\u0000', '\uFFFF');
  }
}
```
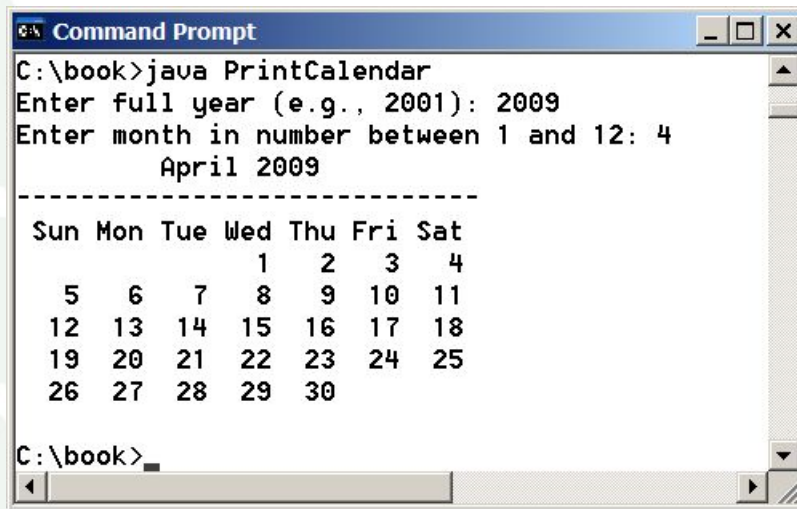
RandomCharacter

TestRandomCharacter

Run

Uploaded By: Jibreel Bornat

# PrintCalender Case Study

Let us use the PrintCalendar example to demonstrate the stepwise refinement approach.



PrintCalendar    Run

Uploaded By: Jibreel Bornat    42

# Design Diagram

Liang, Introduction to Java Programming, Eleventh Edition, (c) 2017 Pearson Education, Inc. All rights reserved.
Uploaded By: Jibreel Bornat
43
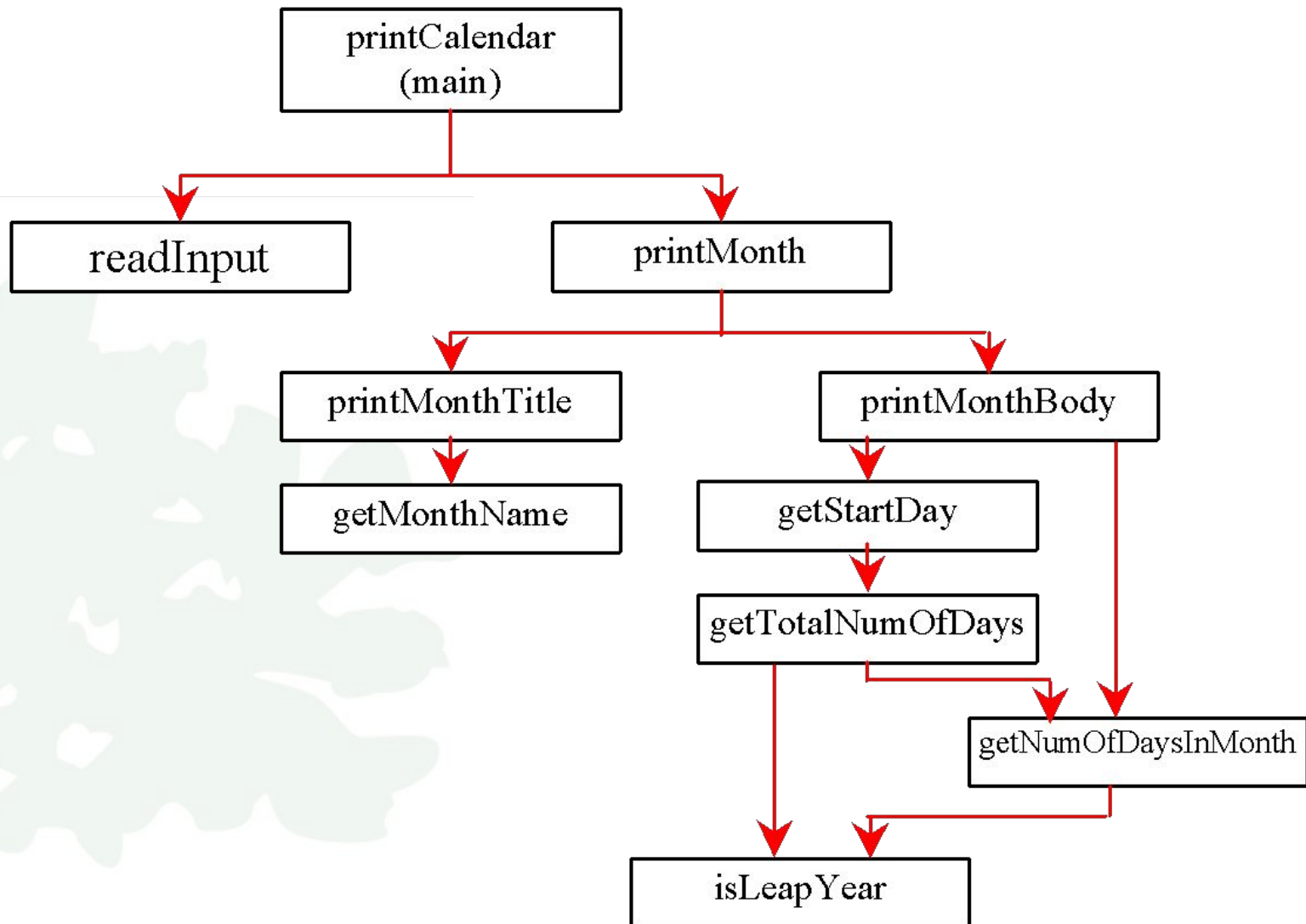
```java
public static boolean isLeapYear(int year) {
    return year % 400 == 0 || (year % 4 == 0 && year % 100 != 0);
}

public static int getNumberOfDaysInMonth(int year, int month) {
    if (month == 1 || month == 3 || month == 5 || month == 7 ||
    month == 8 || month == 10 || month == 12)
        return 31;
    if (month == 4 || month == 6 || month == 9 || month == 11)
        return 30;
    if (month == 2)
        return isLeapYear(year) ? 29 : 28; return 0;
}
```

Uploaded By: Jibreel Bornat

```java
public static int getTotalNumberOfDays(int year, int month) {
    int total = 0; // Get the total days from 1800 to 1/1/year

    for (int i = 1800; i < year; i++)
        if (isLeapYear(i))
            total = total + 366;
        else total = total + 365;

    // Add days from Jan to the month prior to the calendar month
    for (int i = 1; i < month; i++)
        total = total + getNumberOfDaysInMonth(year, i);
    return total;
}
```

Uploaded By: Jibreel Bornat

```java
public static int getStartDay(int year, int month) {
    final int START_DAY_FOR_JAN_1_1800 = 3;

    // Get total number of days from 1/1/1800 to month/1/year
    int totalNumberOfDays = getTotalNumberOfDays(year, month);

    // Return the start day for month/1/year
    return (totalNumberOfDays + START_DAY_FOR_JAN_1_1800) % 7;
}
```

```java
/** Print month body */
public static void printMonthBody(int year, int month) {

    // Get start day of the week for the first date in the month
    int startDay = getStartDay(year, month);

    // Get number of days in the month
    int numberOfDaysInMonth = getNumberOfDaysInMonth(year, month);

    // Pad space before the first day of the month
    int i = 0;
    for (i = 0; i < startDay; i++)
        System.out.print("  ");
    for (i = 1; i <= numberOfDaysInMonth; i++) {
        System.out.printf("%4d", i);
        if ((i + startDay) % 7 == 0) System.out.println();
    }
    System.out.println();
}
```

Uploaded By: Jibreel Bornat 47

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

```java
public static void printMonth(int year, int month) {
    // Print the headings of the calendar
    printMonthTitle(year, month);

    // Print the body of the calendar
    printMonthBody(year, month);
}

/** Print the month title, e.g., May, 1999 */
public static void printMonthTitle(int year, int month) {
    System.out.println("  " + getMonthName(month) + " " + year);
    System.out.println("-----------------------------");
    System.out.println(" Sun Mon Tue Wed Thu Fri Sat");
}
```