Chapter 4: Methods

- Introducing Methods
- Declaring Methods
- Calling Methods
- Passing Parameters
- Pass by Value
- Overloading Methods
- Method Abstraction
- The Math Class
- Recursion (Optional)
Introducing Methods

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

```java
public static int max(int num1, int num2) {
    int result = 0;
    if (num1 > num2) {
        result = num1;
    } else {
        result = num2;
    }
    return result;
}
```
Declaring Methods

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

• A method can be called from another method. It is named as called method. The method call other method is named as calling method.

• Example 4.1 Testing the max method

• This program demonstrates calling a method max to return the largest of the int values
public class TestMax {
// Main method
    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);
    }
}
// A method for finding a max between two numbers
static int max(int num1, int num2)
{
    if (num1 > num2)
        return num1;
    else
        return num2;
}
Passing Parameters

In calling method:

```java
nPrintln("List my name is seven times.", 7);
```

In called method definition

```java
void nPrintln(String message, int n) {
    for (int i=0; i<n; i++)
        System.out.println(message);
}
```
Pass by Value

Example 4.2  Testing Pass by value

This program demonstrates passing values to the methods.
// TestPassByValue.java: Demonstrate passing values to methods
public class TestPassByValue
{
    public static void main(String[] args)
    {
        // Declare and initialize variables
        int num1 = 1;
        int num2 = 2;
        System.out.println("Before invoking the swap method, num1 is "+
            num1 + " and num2 is " + num2);

        // Invoke the swap method to attempt to swap two variables
        swap(num1, num2);

        System.out.println("After invoking the swap method, num1 is "+
            num1 + " and num2 is " + num2);
    }
}
// TestPassByValue.java: Demonstrate passing values to methods

// The method for swapping two variables
static void swap(int n1, int n2)
{
    System.out.println("    Inside the swap method");
    System.out.println("    Before swapping n1 is "+ n1
        + " n2 is "+ n2);
    // Swapping n1 with n2
    int temp = n1;
    n1 = n2;
    n2 = temp;

    System.out.println("    After swapping n1 is "+ n1
        + " n2 is "+ n2);
}
}
Before invoking the swap method, num1 is 1 and num2 is 2
  Inside the swap method
  Before swapping n1 is 1 n2 is 2
  After swapping n1 is 2 n2 is 1
After invoking the swap method, num1 is 1 and num2 is 2
Press any key to continue . . .
Overloading Methods

Multiple method can share with same name

Example 4.3  Overloading three \texttt{max} Methods
// TestMethodOverloading.java: Demonstrate method overloading
public class TestMethodOverloading {
    // Main method
    public static void main(String[] args) {
        // Invoke the max method with int parameters
        System.out.println("The maximum between 3 and 4 is "+ max(3, 4));
        // Invoke the max method with the double parameters
        System.out.println("The maximum between 3.0 and 5.4 is "+ max(3.0, 5.4));

        // Invoke the max method with three double parameters
        System.out.println("The maximum between 3.0, 5.4, and 10.14 is "+ max(3.0, 5.4, 10.14));
    }
}
// Find the max between two int values
static int max(int num1, int num2)
{
    if (num1 > num2)
        return num1;
    else
        return num2;
}

// Find the max between two double values
static double max(double num1, double num2)
{
    if (num1 > num2)
        return num1;
    else
        return num2;
}
// Find the max among three double values
static double max(double num1, double num2, double num3)
{
    return max(max(num1, num2), num3);
}

The maximum between 3 and 4 is 4
The maximum between 3.0 and 5.4 is 5.4
The maximum between 3.0, 5.4, and 10.14 is 10.14
Press any key to continue . . . .
Method Abstraction

You can think of the method body as a black box that contains the detailed implementation for the method. `System.out.println()`, or `MyInput.readDouble()`
The Math Class

Class constants:
- \( \pi \)
- \( E \)

Class methods:
- Trigonometric Methods
- Exponent Methods
- Miscellaneous
Trigonometric Methods

\begin{itemize}
\item $\sin(\text{double } a)$
\item $\cos(\text{double } a)$
\item $\tan(\text{double } a)$
\item $\acos(\text{double } a)$
\item $\asin(\text{double } a)$
\item $\atan(\text{double } a)$
\end{itemize}
Exponent Methods

- **exp(double a)**
  Returns $e$ raised to the power of $a$.

- **log(double a)**
  Returns the natural logarithm of $a$.

- **pow(double a, double b)**
  Returns $a$ raised to the power of $b$.

- **sqrt(double a)**
  Returns the square root of $a$. 
Miscellaneous Methods

- $\text{max}(a, b)$ and $\text{min}(a, b)$
  - Returns the maximum or minimum of two parameters.

- $\text{abs}(a)$
  - Returns the absolute value of the parameter.

- $\text{random}()$
  - Returns a random double value in the range $[0.0, 1.0)$. 
Example 4.4 Computing Mean and Standard Deviation

Generate 10 random numbers and compute the mean and standard deviation.

\[
mean = \frac{\sum_{i=1}^{n} x_i}{n}
\]

\[
deviation = \sqrt{\frac{\sum_{i=1}^{n} x_i^2 - mean}{n - 1}}
\]
// ComputeMeanDeviation.java: Demonstrate using the math methods
public class ComputeMeanDeviation
{
    // Main method
    public static void main(String[] args)
    {
        int number = 0; // Store a random number
        double sum = 0; // Store the sum of the numbers
        double squareSum = 0; // Store the sum of the squares

        // Create 10 numbers, find its sum, and its square sum
        for (int i=1; i<=10; i++)
        {
            // Generate a new random number
            number = (int)Math.round(Math.random()*1000);
            System.out.println(number);
        }
    }
}
// Add the number to sum
    sum += number;
    // Add the square of the number to squareSum
    squareSum += Math.pow(number, 2); // Same as number*number;
}
// Find mean
double mean = sum/10;

// Find standard deviation
double deviation = Math.sqrt((squareSum - mean)/(10 - 1));

// Display result
System.out.println("The mean is " + mean);
System.out.println("The standard deviation is " + deviation);
Case Studies

Example 4.5 Displaying Calendars

The program reads in the month and year and displays the calendar for a given month of the year.
// PrintCalendar.java: Print a calendar for a given month in a year
public class PrintCalendar
{
    // Main method
    public static void main(String[] args)
    {
        // The user enters year and month
        System.out.print("Enter full year: ");
        int year = MyInput.readInt();
        System.out.print("Enter month in number between 1 and 12: ");
        int month = MyInput.readInt();

        // Print calendar for the month of the year
        printMonth(year, month);
    }
}
// Print the calendar for a month in a year
static void printMonth(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 numOfDaysInMonth = getNumOfDaysInMonth(year, month);

    // Print headings
    printMonthTitle(year, month);

    // Print body
    printMonthBody(startDay, numOfDaysInMonth);
}
// Get the start day of the first day in a month
static int getStartDay(int year, int month)
{
    // Get total number of days since 1/1/1800
    int startDay1800 = 3;
    long totalNumOfDays = getTotalNumOfDays(year, month);

    // Return the start day
    return (int)((totalNumOfDays + startDay1800) % 7);
}
// Get the total number of days since Jan 1, 1800
static long getTotalNumOfDays(int year, int month)
{
    long total = 0;

    // Get the total days from 1800 to year -1
    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 + getNumOfDaysInMonth(year, i);
    return total;
}
/ Get the number of days in a month
static int getNumOfDaysInMonth(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)
        if (isLeapYear(year))
            return 29;
        else
            return 28;
    return 0; // If month is incorrect.
}
// Determine if it is a leap year
static boolean isLeapYear(int year) {
    if ((year % 400 == 0) || ((year % 4 == 0) && (year % 100 != 0)))
        return true;
    return false;
}

// Print month body
static void printMonthBody(int startDay, int numOfDaysInMonth) {
    // 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 <= numOfDaysInMonth; i++)
{
    if (i < 10)
        System.out.print("   " + i);
    else
        System.out.print("  " + i);
    
    if ((i + startDay) % 7 == 0)
        System.out.println();
}

System.out.println();
}
// Print the month title, i.e. May, 1999
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");
}

// Get the English name for the month
static String getMonthName(int month)
{
    String monthName = null;
    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;
}
Enter full year: 2002
Enter month in number between 1 and 12: 1
January, 2002

<table>
<thead>
<tr>
<th>Sun</th>
<th>Mon</th>
<th>Tue</th>
<th>Wed</th>
<th>Thu</th>
<th>Fri</th>
<th>Sat</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
</tr>
<tr>
<td>19</td>
<td>20</td>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
</tr>
<tr>
<td>26</td>
<td>27</td>
<td>28</td>
<td>29</td>
<td>30</td>
<td>31</td>
<td></td>
</tr>
</tbody>
</table>

Press any key to continue . . .
Recursion (Optional)

Example 4.6 Computing Factorial

0! = 1
1! = 1
2! = 2 \times 1! = 1
3! = 3 \times 2 \times 1 = 6
4! = 4 \times 3! = 4 \times 3 \times 2 \times 1 = 24
...
N! = N \times (N-1)! = N \times (N-1) \times (N-2)! = N \times (N-1) \times (N-2) \times \ldots \times 2 \times 1

factorial(0) = 1;
factorial(n) = n \times factorial(n-1);
public class ComputeFactorial {
    public static void main(String[] args) {
        // Prompt the user to enter an integer
        System.out.println("Please enter a nonnegative integer");
        int n = MyInput.readInt();

        System.out.println("Factorial of " + n + " is " + factorial(n));
    }
}

// ComputeFactorial.java: Compute factorial of an integer

// Recursive method for computing factorial of n
static long factorial(int n)
{
    if (n == 0) // Stopping condition
        return 1;
    else
        return n*factorial(n-1); // Call factorial recursively
}

Please enter a nonnegative integer
10
Factorial of 10 is 3628800
Press any key to continue . . .
Example 4.7 Computing Fibonacci Numbers

Find Fibonacci numbers using recursion.

\[
\text{fib}(0) = 1; \\
\text{fib}(1) = 1; \\
\text{fib}(2) = \text{fib}(1) + \text{fib}(0); \\
\text{fib}(3) = \text{fib}(2) + \text{fib}(1); \\
\]

\[
\text{fib}(n) = \text{fib}(n-2) + \text{fib}(n-1); \quad n \geq 2
\]
// ComputeFibonacci.java: Find a Fibonacci number for a given index
public class ComputeFibonacci
{
    // Main method
    public static void main(String args[])
    {
        // Read the index
        System.out.println("Enter an index for the Fibonacci number");
        int n = MyInput.readInt();

        // Find and display the Fibonacci number
        System.out.println("Fibonacci number at index "+ n + " is "+fib(n));
    }
}

//fib is a function to compute Fibonacci number
// The method for finding the Fibonacci number
public static long fib(long n)
{
    if ((n==0)||(n==1)) // Stopping condition
        return 1;
    else // Reduction and recursive calls
        return fib(n-1) + fib(n-2);
}
Towers of Hanoi

Example 4.8 Solving the Towers of Hanoi Problem

Solve the towers of Hanoi problem.
public class TowersOfHanoi
{
    // Main method
    public static void main(String[] args)
    {
        // Read number of disks, n
        System.out.println("Enter number of disks");
        int n = MyInput.readInt();

        // Find the solution recursively
        System.out.println("The moves are:");
        moveDisks(n, 'A', 'B', 'C');
    }
}

// TowersOfHanoi.java: Find solutions for
// the Towers of Hanoi problem
public class TowersOfHanoi
{
    // Main method
    public static void main(String[] args)
    {
        // Read number of disks, n
        System.out.println("Enter number of disks");
        int n = MyInput.readInt();

        // Find the solution recursively
        System.out.println("The moves are:");
        moveDisks(n, 'A', 'B', 'C');
    }
}
// The method for finding the solution to move n disks
// from fromTower to toTower with auxTower
public static void moveDisks(int n, char fromTower, char toTower, char auxTower)
{
    if (n==1) // Stopping condition
        System.out.println("Move disk " + n + " from " + fromTower+" to " + toTower);
    else
    {
        moveDisks(n-1, fromTower, auxTower, toTower);
        System.out.println("Move disk " + n + " from " + fromTower + " to " + toTower);
        moveDisks(n-1, auxTower, toTower, fromTower);
    }
}
Enter number of disks
3
The moves are:
Move disk 1 from A to B
Move disk 2 from A to C
Move disk 1 from B to C
Move disk 3 from A to B
Move disk 1 from C to A
Move disk 2 from C to B
Move disk 1 from A to B
Press any key to continue . . .

Enter number of disks
4
The moves are:
Move disk 1 from A to C
Move disk 2 from A to B
Move disk 1 from C to B
Move disk 3 from A to C
Move disk 1 from B to A
Move disk 2 from B to C
Move disk 1 from A to C
Move disk 4 from A to B
Move disk 1 from C to B
Move disk 2 from C to A
Move disk 1 from B to A
Move disk 3 from C to B
Move disk 1 from A to C
Move disk 2 from A to B
Move disk 1 from C to B
Press any key to continue . . .