Lecture 17:
Introduction to Methods
Feb 24 2015
Opening Problem

Find the sum of integers from 1 to 10, from 20 to 30, and from 35 to 45, respectively.

```java
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);
```
Problem

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

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));
}

Method Abstraction

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

*Internal behavior of the code is unknown*
Method Abstraction

Until now we’ve only been using a main method

```java
public static void main(String[] args) {
    //statements
}
```

Now with methods:

```java
//inside class

public static void main(String[] args) {
    //statements
    // methods calls
}
```
Method Abstraction

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

• Write a method once and reuse it anywhere.

• Information hiding. Hide the implementation from the user.

• Reduce complexity.
Defining Methods

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

```
public static int max(int num1, int num2) {
    int result;
    if (num1 > num2)
        result = num1;
    else
        result = num2;
    return result;
}
```
Method Signature

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

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

Define a method

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

Invoke a method
Formal Parameters

The variables defined in the method header are known as formal parameters.
Actual Parameters

When a method is invoked, you pass a value to the parameter. This value is referred to as *actual parameter or argument*.

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

```
int z = max(x, y);
```
Return Value Type

A method may return a value. The `returnValueType` is the data type of the value the method returns. If the method does not return a value, the `returnValueType` is the keyword `void`. For example, the `returnValueType` in the `main` method is `void`.

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

Define a method

Invoke a method

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

Testing the \texttt{max} method

This program demonstrates calling a method \texttt{max} to return the largest of the \texttt{int} values.

```java
public static void main(String[] args) {
    int i = 5;
    int j = 2;
    int k = max(i, j);
    System.out.println("The maximum between \texttt{" + 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;
}
```
public static void main(String[] args) {
    int i = 5;
    int j = 2;
    int k = max(i, j);
    System.out.println("The maximum between "+i+", "+j+" is "+k);
}

public static int max(int num1, int num2) {
    int result;
    if (num1 > num2)
        result = num1;
    else
        result = num2;
    return result;
}
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;
}
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;
}
Trace Method Invocation

```
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;
}
```

invoke max(i, j)
Pass the value of i to num1
Pass the value of j to 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;
}
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;
}

(num1 > num2) is true since num1 is 5 and num2 is 2
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;
}
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;
}
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;
}
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;
}
A return 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.

To fix this problem, delete if \((n < 0)\) in (a), so that the compiler will see a return statement to be reached regardless of how the if statement is evaluated.
NOTE: One of the benefits of methods is for reuse. The \texttt{max} method can be invoked from any class besides \texttt{TestMax}. If you create a new class \texttt{Test}, you can invoke the \texttt{max} method using \texttt{ClassName.methodName} (e.g., \texttt{TestMax.max}).
Next time, we’ll explore
Return types
And several examples…
void Method Example

This type of method does not return a value. The method performs some actions.
Passing Parameters

```java
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");
```