Unit 1, Part 3

Procedural Decomposition

(How to Use Methods to Write Better Programs)

Computer Science S-111
Harvard University

David G. Sullivan, Ph.D.

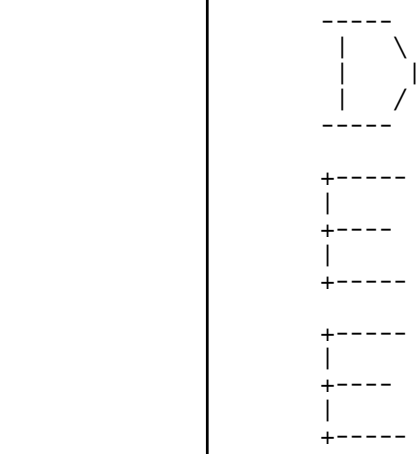
Example Program: Writing Block Letters

- Here's a program that writes the name "DEE" in block letters:

```
public class BlockLetters {
    public static void main(String[] args) {
        System.out.println("    -----");
        System.out.println("     |   \\");
        System.out.println("     |    |");
        System.out.println("     |   /");
        System.out.println("    -----");
        System.out.println();
        System.out.println("    +-----");
        System.out.println("    |");
        System.out.println("    +----");
        System.out.println("    |");
        System.out.println("    +-----");
        System.out.println();
        System.out.println("    +-----");
        System.out.println("    |");
        System.out.println("    +----");
        System.out.println("    |");
        System.out.println("    +-----");
    }
}
```

Example Program: Writing Block Letters

- The output looks like this:



Code Duplication

```
public class BlockLetters {
    public static void main(String[] args) {
        System.out.println("    -----");
        System.out.println("     |   \\");
        System.out.println("     |    |");
        System.out.println("     |   /");
        System.out.println("    -----");
        System.out.println();
        System.out.println("    +-----");
        System.out.println("    |");
        System.out.println("    +----");
        System.out.println("    |");
        System.out.println("    +-----");
        System.out.println();
        System.out.println("    +-----");
        System.out.println("    |");
        System.out.println("    +----");
        System.out.println("    |");
        System.out.println("    +-----");
    }
}
```

- The code that writes an E appears twice – it is duplicated.

Code Duplication (cont.)

- Code duplication is undesirable. Why?

- Also, what if we wanted to create another word containing the letters D or E? What would we need to do?

- A better approach: create a command for writing each letter, and execute that command as needed.

- To create our own command in Java, we define a method.

Defining a Simple Static Method

- We've already seen how to define a `main` method:

```
public static void main(String[] args) {
    <statement>;
    <statement>;
    …
    <statement>;
}
```

- The simple methods that we'll define have a similar syntax:

```
public static void <name>() {
    <statement>;
    <statement>;
    …
    <statement>;
}
```

- This type of method is known as *static method.*

Defining a Simple Static Method (cont.)

- Here's a static method for writing a block letter E:

```
public static void writeE() {
    System.out.println("    +-----");
    System.out.println("    |");
    System.out.println("    +----");
    System.out.println("    |");
    System.out.println("    +-----");
}
```

- It contains the same statements that we used to write an E in our earlier program.
- This method gives us a command for writing an E.
- To use it, we simply include the following statement:

```
writeE();
```

Calling a Method

- The statement

```
writeE();
```

  is known as a *method call.*

- General syntax for a static method call:

```
<method-name>();
```

- Calling a method causes the statements inside the method to be executed.

Using Methods to Eliminate Duplication

- Here's a revised version of our program:

```
public class BlockLetters2 {
    public static void writeE() {
        System.out.println("    +-----");
        System.out.println("    |");
        System.out.println("    +----");
        System.out.println("    |");
        System.out.println("    +-----");
    }

    public static void main(String[] args) {
        System.out.println("    -----");
        System.out.println("     |   \\");
        System.out.println("     |    |");
        System.out.println("     |   /");
        System.out.println("    -----");
        System.out.println();
        writeE();
        System.out.println();
        writeE();
    }
}
```

Methods Can Be Defined In Any Order

- Here's a version in which we put the `main` method first:

```
public class BlockLetters2 {
    public static void main(String[] args) {
        System.out.println("    -----");
        System.out.println("     |   \\");
        System.out.println("     |    |");
        System.out.println("     |   /");
        System.out.println("    -----");
        System.out.println();
        writeE();
        System.out.println();
        writeE();
    }

    public static void writeE() {
        System.out.println("    +-----");
        System.out.println("    |");
        System.out.println("    +----");
        System.out.println("    |");
        System.out.println("    +-----");
    }
}
```

- By convention, the `main` method should appear first or last.

Flow of Control

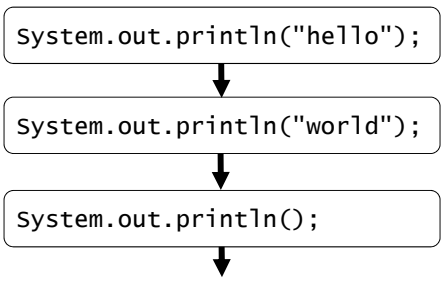
- A program's *flow of control* is the order in which its statements are executed.
- By default, the flow of control:
  - is sequential
  - begins with the first statement in the `main` method

Flow of Control (cont.)

- Example: consider the following program:

```
public class HelloWorldAgain {
    public static void main(String[] args) {
        System.out.println("hello");
        System.out.println("world");
        System.out.println();
        ...
    }
}
```

- We can represent the flow of control using a flow chart:



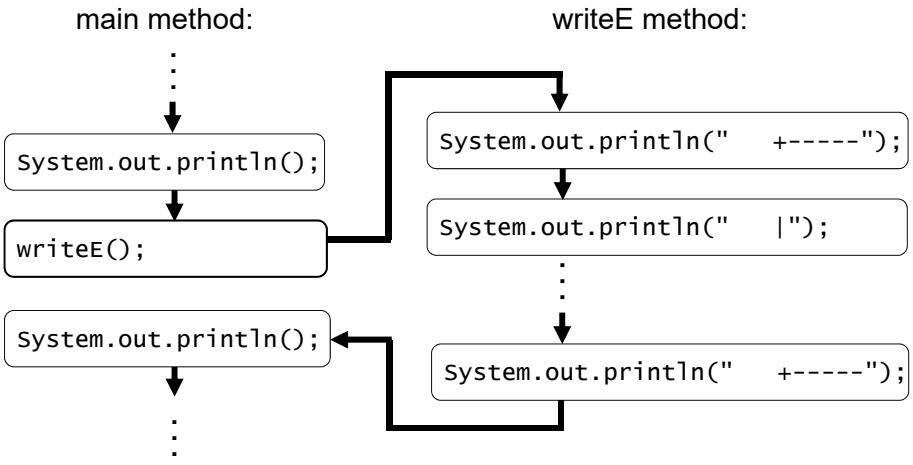
Method Calls and Flow of Control

- When we call a method, the flow of control jumps to the method.
- After the method completes, the flow of control jumps back to the point where the method call was made.

```
public class BlockLetters2 {
    public static void writeE() {
        System.out.println("    +-----");
        System.out.println("    |");
        System.out.println("    +----");
        System.out.println("    |");
        System.out.println("    +-----");
    }

    public static void main(String[] args) {
        System.out.println("    -----");
        System.out.println("     |   \\");
        System.out.println("     |    |");
        System.out.println("     |   /");
        System.out.println("    -----");
        System.out.println();
        writeE();
        System.out.println();
        ...
```

Method Calls and Flow of Control (cont.)

- Here's a portion of the flowchart for our program:



Another Use of a Static Method

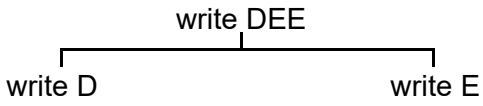
```
public class BlockLetters3 {
    public static void writeD() {
        System.out.println("    -----");
        System.out.println("     |   \\");
        System.out.println("     |    |");
        System.out.println("     |   /");
        System.out.println("    -----");
    }

    public static void writeE() {
        System.out.println("    +-----");
        System.out.println("    |");
        System.out.println("    +----");
        System.out.println("    |");
        System.out.println("    +-----");
    }

    public static void main(String[] args) {
        writeD();
        System.out.println();
        writeE();
        System.out.println();
        writeE();
    }
}
```

Another Use of a Static Method (cont.)

- The code in the `writeD` method is only used once, so it doesn't eliminate code duplication.
- However, using a separate static method still makes the overall program more readable.
- It helps to reveal the *structure* of the program.

Procedural Decomposition

- In general, methods allow us to *decompose* a problem into smaller subproblems that are easier to solve.
  - the resulting code is also easier to understand and maintain
- In our program, we've decomposed the task "write DEE" into two subtasks:
  - write D
  - write E (which we perform twice).
- We can use a *structure diagram* to show the decomposition:



Procedural Decomposition (cont.)

- How could we use procedural decomposition in printing the following lyrics?

Dashing through the snow in a one-horse open sleigh,
O'er the fields we go, laughing all the way.
Bells on bobtail ring, making spirits bright.
What fun it is to ride and sing a sleighing song tonight!

Jingle bells, jingle bells, jingle all the way!
O what fun it is to ride in a one-horse open sleigh!
Jingle bells, jingle bells, jingle all the way!
O what fun it is to ride in a one-horse open sleigh!

A day or two ago, I thought I'd take a ride,
And soon Miss Fanny Bright was seated by my side.
The horse was lean and lank; misfortune seemed his lot;
We got into a drifted bank and then we got upsot.

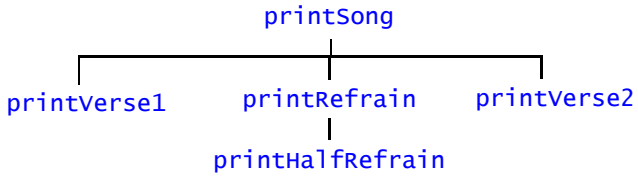
Jingle bells, jingle bells, jingle all the way!
O what fun it is to ride in a one-horse open sleigh!
Jingle bells, jingle bells, jingle all the way!
O what fun it is to ride in a one-horse open sleigh!

Procedural Decomposition (cont.)

Dashing through the snow in a one-horse open sleigh,
O'er the fields we go, laughing all the way.
Bells on bobtail ring, making spirits bright.
What fun it is to ride and sing a sleighing song tonight! — printVerse1

Jingle bells, jingle bells, jingle all the way! — printRefrain
O what fun it is to ride in a one-horse open sleigh!
Jingle bells, jingle bells, jingle all the way! — printHalfRefrain
O what fun it is to ride in a one-horse open sleigh!

A day or two ago, I thought I'd take a ride,
And soon Miss Fanny Bright was seated by my side.
The horse was lean and lank; misfortune seemed his lot;
We got into a drifted bank and then we got upsot. — printVerse2



Code Reuse

- Once we have a set of methods, we can use them to solve other problems.
- Here's a program that writes the name "ED":

```
public class BlockLetters4 {
    // these methods are the same as before
    public static void writeD() {
        ...
    }

    public static void writeE() {
        ...
    }

    public static void main(String[] args) {
        writeE();
        System.out.println();
        writeD();
    }
}
```

Tracing the Flow of Control

- What is the output of the following program?

```
public class FlowControlTest {
    public static void methodA() {
        System.out.println("starting method A");
    }

    public static void methodB() {
        System.out.println("starting method B");
    }

    public static void methodC() {
        System.out.println("starting method C");
    }

    public static void main(String[] args) {
        methodC();
        methodA();
    }
}
```

Methods Calling Methods

- The definition of one method can include calls to other methods.
- We've seen this already in the `main` method:

```
public static void main(String[] args) {
    writeE();
    System.out.println();
    writeD();
}
```

- We can also do this in other methods:

```
public static void foo() {
    System.out.println("This is method foo.");
    bar();
}

public static void bar() {
    System.out.println("This is method bar.");
}
```

Methods Calling Methods (cont.)

- What is the output of the following program?

```
public class FlowControlTest2 {
    public static void methodOne() {
        System.out.println("boo");
        methodThree();
    }
    public static void methodTwo() {
        System.out.println("hoo");
        methodOne();
    }
    public static void methodThree() {
        System.out.println("foo");
    }
    public static void main(String[] args) {
        methodOne();
        methodThree();
        methodTwo();
    }
}
```

Comments

- Comments are text that is ignored by the compiler.
- Used to make programs more readable
- Two types:
  1. line comments: begin with `//`
     - compiler ignores from `//` to the end of the line
     - examples:

```
// this is a comment
System.out.println();  // so is this
```

  2. block comments: begin with `/*` and end with `*/`
     - compiler ignores everything in between
     - typically used at the top of each source file

Comments (cont.)

```
/*
 * DrawTriangle.java
 * Dave Sullivan (dgs@cs.bu.edu)
 * This program draws a triangle.
 */

public class DrawTriangle {
    public static void main(String[] args) {
        System.out.println("Here's my drawing:");

        // Draw the triangle using characters.
        System.out.println("    ^");
        System.out.println("   / \\");
        System.out.println("  /   \\");
        System.out.println(" /     \\");
        System.out.println(" -------");
    }
}
```

block comments

line comments

Comments (cont.)

- Put comments:
  - at the top of each file, naming the author and explaining what the program does
  - at the start of every method other than `main`, describing its behavior
  - inside methods, to explain complex pieces of code (this will be more useful later in the course)
- We will deduct points for failing to include the correct comments and other stylistic problems.