Lesson 1.05 — Static Methods & Method Calls (1/2)

Overview

Objectives — *Students will be able to...*

- **Use** procedural decomposition to plan complex programs using structure diagrams.
- **Manage** complexity by using method calls

Assessments — *Students will...*

- **Complete** Practice problems

Homework — *Students will...*

- **Read** BJP 1.5
- **Complete** Ch.1 Exercises 11, 12, 14, 16

Materials & Prep

- **Projector and computer** (if you are able to/opt to use Eclipse with your students)
- **Whiteboard and marker**
- **Overly complicated diagram** (https://tinyurl.com/y67z3cym)

Pacing Guide

Section	Total Time
Bell-work and attendance	5min
Introduction and note-taking	15min
Practice questions	25min
Students trade work, check, and turn in	5min

Procedure

This class introduces many new, intertwined concepts in one class period. These concepts will be re-taught in the next class, but you should be aware that your students have a lot of information to absorb in a short amount of time. This lesson will be a good litmus test—if students have been doing their reading and homework, the class should move along smoothly. If students are not completing the readings, you will probably only get through ~50% of the material. If needed, use this opportunity to convince students of the pace and commitment level required for the class.

Bell-work and Attendance [5 minutes]

Introduction and Note-Taking [15 minutes]

1. Have the **complicated algorithm** up on the board, or printed out for students to pass around. If you have any confusing furniture assembly manuals, or overly complicated directions, bring those to pass around too.

   Start class off with a whole group discussion about why the instructions or diagrams are confusing, and ask students what strategies could be used to make them easier to understand. The diagram listed in materials is an actual slide from the Pentagon, and illustrates how too much complexity can cause all meaning to be lost. Guide the conversation towards decomposition to begin your lecture:

   - **Decomposition**: dividing a problem into smaller, more manageable pieces.
   - **Procedural decomposition**: dividing a whole program into a series of individual steps or actions to program 1 at a time.

- **Structure diagram**: a way of organizing your approach to building a larger program. Ask students to help you draw a structure diagram for a program with the output shown below:

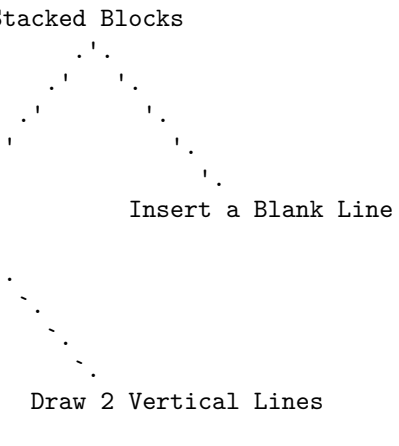
OUTPUT:

```
+-------+
|       |
|       |
+-------+

+-------+
|       |
|       |
+-------+

+-------+
|       |
|       |
+-------+
```

STRUCTURE DIAGRAM:



2. Have students take 3 minutes to write the DrawBoxes program the long way:

```
public class DrawBoxes {
    public static void main (String[] args) {
        System.out.println("+-------+");
        System.out.println("|       |");
        …
```

3. Point out to students that anything they would cut and paste to save time on creating would make a good unit to turn into a "static method"
   - **Static method** — a block of Java statements that is given its own name (ask students to point to a Java statement)
   - Has the same structure we're familiar with from the main method we already wrote, but we give it a different name than "main".
   - A method call interrupts the sequential execution of statements, causing the program to first execute the statements in the method before continuing.
   - Once the last statement in the method or constructor has executed or a return statement is executed, flow of control is returned to the point immediately following where the method or constructor was called.
4. Ask students what simple unit we should build into a static method (have them refer to the structure diagram), and have them suggest a name for the method.
5. Rewrite the method as a class, then show students how to write methods in main that call the new static method. Make sure that students insert the println statements between each method call. It should look something like this:

```
public class DrawBoxes3 {

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

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

```
        drawBox();
        System.out.println();
        drawBox();
    }
}
```

6. Call on a student to come to the board and physically trace the flow of control with the marker. Start them off by pointing out that Java always starts with the main method.

   If the student seems nervous, encourage the rest of the class to call out directions to the student. Make sure students are drawing the flow of control on their own notes as well.

Practice questions [25 minutes]

1. Have students complete the following practice questions:
   1. Self-Check 1.22: Tricky
   2. Self-Check 1.23: Strange
   3. Self-Check 1.26: Confusing
   4. Self-Check 1.29: LotsOfErrors-errors

Copy the practice questions to a worksheet and have students complete the practice problems by writing out the answers and using their error-checking algorithm sheets.

Some students will jump right into this activity, but others will need additional assistance from you.

At this point in the school year, we suggest that you insist on structure diagrams with each program. Structure diagrams encourage algorithmic thinking and the creation of efficient solutions; both of which are vital computational thinking skills.

If need be, work on "Tricky" as a whole group, so you can model the correct steps to approaching a problem. If your class decides on an algorithm for "predict the output" type questions, have a student make that algorithm into a poster for the whole class to refer to.

Students trade work, check, and hand in [5 minutes]

Have students trade work and check each other's responses on Practice-It before submitting.

Accommodation and Differentiation

In ELL classrooms, this lesson should be delivered over the course of 2 days. Extra time should be spent drilling static methods, methods that call other methods, and flow of control. Try adapting some of the examples from the book to include students' native language so they can focus on structuring code instead of translating language. One easy way to introduce familiar, repetitive content would be to have students output the lyrics to a song with a refrain. For a physical activity to demonstrate flow-of-control, check out lesson plan 1.6.

If you have students who are speeding through this lesson, you should encourage them to: - Complete the remaining 1. Self-Check 1.24: Strange2 2. Self-Check 1.25: Strange3 3. Self-Check 1.27: Confusing2 4. Self-Check 1.28: Confusing3 - Have the student write a sample test question with output that can be written using method calls. Be sure they include the answer key with the sample question!

Teacher Prior CS Knowledge

Java has both static and non-static methods. Static methods allow the programmer to call the method without creating an object from the class. Non-static methods covered in the 2nd half of the course requires an object be created from the class before calling the method. Because "Building Java Programs" introduces functions before objects, methods in early lesson plans are static. For a description of the difference between static and non-static methods see http://beginnersbook.com/2013/05/static-vs-non-static-methods/.

Teaching Tips

- One of the big fundamental concepts of problem solving in computer science is the concept abstraction. This lesson has a number of new syntax constructs for students to create methods and it is important to give students the big picture idea of factoring code and reducing redundancy.
- The practice problems have students tracing code where methods call other methods. Giving students a way of tracing code like using a table to keep track of the method calls. If you are using a black/white board, we recommend crossing out the method call when it completes instead of erasing the name so the student can review the entire flow of control from beginning to end after the exercise is complete. Another technique that can be introduced now that can be used later when the flow of control gets more complex is memory diagrams: https://www.youtube.com/watch?v=t-_TeH0dSZs&feature=youtu.be&list=PL0g5FWk3FEqjmrq4ystAvlRyenEF7lUwa

Misconceptions

- When declaring a method, students will sometimes incorrectly add a semicolon to the method header, as in `public void foo();`. Students have a misconception that all statements in Java end in a semicolon. They need to know the distinction between statements that do end in semicolon and statements that begin blocks with curly brackets. The addition of the semicolon to the message header could also be students incorrectly pattern matching the method declaration with the method call where there is a semicolon: `foo();` This overgeneralization could lead to semi-colon being incorrectly placed.
- For methods without parameters, students will sometimes omit the parenthesis `()`. To clarify the difference between variables and methods, always use parenthesis when referring to methods. This will reinforce the notion that methods in Java require parenthesis, even for methods with zero parameters.

Videos

- BJP 1-3, *Programming with Methods* http://media.pearsoncmg.com/aw/aw_reges_bjp_2/videoPlayer.php?id=c1-3
- CSE 142, *Static Methods* (44:12-49:21) https://www.youtube.com/watch?v=i2pQHeW5CeY&start=2652
- CSE 142, *Procedural Decomposition* (20:10–29:35) https://www.youtube.com/watch?v=KZY0S7wpMAg&start=1211
- CSE 142, *Eliminating Redundancy* (29:36–35:49) https://www.youtube.com/watch?v=KZY0S7wpMAg&start=1775

Forum discussion

Lesson 1.05 Static Methods and Method Calls (TEALS Discourse account required)