Lesson 8.02 — Writing Recursive Solutions

Overview

Objectives — *Students will be able to…*

- **Identify** recursive methods.
- **Predict** the output (or return value) of recursive methods.

Assessments — *Students will…*

- **Evaluate** statements and **predict** output during a game of grudgeball

Homework — *Students will…*

- **Read** BJP 12.2
- **Complete** self-check questions #5, 7-9 and exercise #1

Materials & Prep

- **Projector and computer** (optional)
- **White paper and markers**
- **Rules** for grudgeball (see website for details: http://toengagethemall.blogspot.com/2013/02/grudgeball-review-game-where-kids-attack.html)
- **Team assignments** that divide your class into 5 or 6 teams
- **Nerf hoop & ball** (or wastepaper and trash can)
- **Taped 2- and 3-point lines**

Briefly review the rules of Grudgeball if you have forgotten them. If you have removed your 2 and 3 point lines from last time you played, test out your 2 and 3 point lines before class begins. If you do not wish to play Grudgeball in your classroom (or if you are unable to), most Grudgeball questions can be found in the self-check question bank for 12.1.

Pacing Guide

Section	Total Time
Bell-work and attendance	5min
Introduction and note-taking	15min
Activity: Grudgeball	35min

Procedure

To hook your class for today's material, and if space and whiteboard setup allow, set up the grudgeball "court" and scoreboard before class begins. Remind students that lecture content will be tested during the game.

Bell-work and Attendance [5 minutes]

Introduction and note-taking [10 minutes]

1. Begin by asking students what makes a method recursive (it calls itself). Ask students to offer suggestions as to how to write a recursive method. Using pseudocode (or actual code, if they offer it), outline their suggestion on the board. It should look something like this:

```
public static void writeStars (int x) {
    writeStars(x - 1);    // Prints a star.
}
```

2. Don't worry if students don't include a base case! Congratulate your students on remembering that the method calls itself, then ask students how this method is supposed to stop. (It won't! This is called infinite

recursion.) To make sure that you write recursive methods that work, you need to remember 2 key ingredients:

- **Base case**: a case within a recursive solution that is so simple, it can be solved without needing to call the method again (a recursive call).
- **Recursive case**: A case within a recursive solution that involves reducing the overall problem to a simpler problem of the same kind that can be solved by a recursive call.

3. Add these 2 ingredients to the example you currently have on the board. It should look something like this:

```
public static void writeStars (int x) {
    if (x == 0) {
        // This is the base case: "write 0 stars" needs no additional method.
        System.out.println();
    } else {
        // This is the recursive case: write one star, then write however many
        // stars are left.
        System.out.println("*");
        writeStars(x - 1);
    }
}
```

4. Emphasize to students that you can write more than one recursive case, but you must always have at least 1 base case and 1 recursive case, or the code won't work. (Because you need both types of cases, recursive solutions are often written as if/else or nested statements.)
5. Ask students to explain what doesn't work about this code, and ask them to correct the recursive code here:

```
public static void writeStars (int x) {
    System.out.print("*");
    writeStars(n-1);
}
```

   There's a recursive case, which is good, but there is no base case! This causes infinite recursion since it has no way of stopping. Instead of stopping at 0 stars (which is what the base case would be), the code will try to write -1, -2, -3... stars forever!

Activity: Grudgeball [35 minutes]

1. Divide students into their assigned teams.
2. Review the rules for grudgeball, and have the students repeat the rules back to you.
3. Using the problems listed below (and any you may add, depending on your class' needs), play grudgeball until a team wins, or until the class period ends.
   - If a class gets the answer wrong, BRIEFLY pause the game to have students offer corrections before moving to the next team's question.
   - If correction seems to be dragging on, jump in and quickly re-teach using the incorrect answer as your example. It is important to keep the pace going to maintain student interest in the game!

   Gudgeball problems & answers have been grouped assuming that you have 6 teams. If you have fewer teams, each "round" will be shifted accordingly, so you may have rounds where different teams are practicing different concepts. Judge each team's knowledge gaps, and adjust which questions you ask each group accordingly.
4. Questions for your Grudgeball game are listed below:

GRUDGEBALL PROBLEMS

Conceptual Questions

a) What is recursion?

b) How does a recursive method differ from an iterative method?
c) What do you look for in a recursive method? (What parts does it have?)
d) What is a base case?
e) What is a recursive case?
f) Does a recursive method need to have both base and recursive cases?
g) Why does a recursive method need to have a base case and a recursive case?
h) Can a recursive method have more than one base case?
i) Can a recursive method have more than one recursive case?
j) What happens if a recursive solution is missing a base case?

Predict-the-Output Questions Use the following method for questions k – s:

```java
public static void mystery1 (int n) {
    if (n <= 1) {
        System.out.print(n);
    } else {
        mystery1(n / 2);
        System.out.print(", " + n);
    }
}
```

What is the output produced by the method call indicated?

k) `mystery1(1);`
l) `mystery1(2);`
m) `mystery1(3);`
n) `mystery1(4);`
o) `mystery1(16);`
p) `mystery1(30);`
q) `mystery1(100);`
r) `mystery1(-1);`
s) `mystery1(2.2);`

Predict-the-Output Questions (continued) Use the following method for questions t – x:

```java
public static void mystery2 (int n) {
    if (n > 100) {
        System.out.print(n);
    } else {
        mystery2(2*n);
        System.out.print(", " + n);
    }
}
```

What output is produced by the method call indicated?

t) `mystery2(113);`
u) `mystery2(70);`
v) `mystery2(42);`
w) `mystery2(30);`
x) `mystery2(10);`

Accommodation and Differentiation

In ELL classrooms, read the questions aloud in addition to showing the question on the board or projector. Consider distributing a worksheet with the questions on it so students can write down answers during the game.

If students are having difficulty with the "predict the output" questions, a step-by-step explanation of how a recursive method executes can be found here (http://tinyurl.com/lablr3h). You can pattern your explanations/re-teaching to students using the same method.

Misconceptions

- When the recursive call is made, mistakenly thinking the same parameter is being used instead of a distinct copy for the recursive call.
- The only time the base case is called is if the initial call is the base case.

Video

- BJP 12-2, *Implementing a Recursive Function* http://media.pearsoncmg.com/aw/aw_reges_bjp_2/videoPlayer.php?id=c12-2

Forum discussion

Lesson 8.02 Writing Recursive Solutions (TEALS Discourse account required)