Building Java Programs

Chapter 12
Lecture 12-2: recursive programming

reading: 12.2 - 12.3
Hey baby, this pickup line is recursive. The "hey" is short for "hey baby, this pickup line is recursive. The..."

Benoit Mandelbrot: Master of seduction.
Exercise

• Write a recursive method `pow` accepts an integer base and exponent and returns the base raised to that exponent.
  • Example: `pow(3, 4)` returns 81

• Solve the problem recursively and without using loops.
An optimization

- Notice the following mathematical property:
  \[ 3^{12} = 531441 = 9^6 = (3^2)^6 \]
  \[ 531441 = (9^2)^3 = ((3^2)^2)^3 \]

- When does this "trick" work?
- How can we incorporate this optimization into our \texttt{pow} method?
- What is the benefit of this trick if the method already works?
There are only 10 types of people in the world: Those who understand binary and those who don’t
Exercise

- Write a recursive method `printBinary` that accepts an integer and prints that number's representation in binary (base 2).
  
  - Example: `printBinary(7)` prints 111
  - Example: `printBinary(12)` prints 1100
  - Example: `printBinary(42)` prints 101010

- Write the method recursively and without using any loops.
Stutter

• How did we break the number apart?

```java
public static int stutter(int n) {
    if (n < 10) {
        return (10 * n) + n;
    } else {
        int a = mystery(n / 10);
        int b = mystery(n % 10);
        return (100 * a) + b;
    }
}
```
Case analysis

- Recursion is about solving a small piece of a large problem.
  - What is 69743 in binary?
    - Do we know *anything* about its representation in binary?
  
- Case analysis:
  - What is/are easy numbers to print in binary?
  - Can we express a larger number in terms of a smaller number(s)?
printBinary solution

// Prints the given integer's binary representation.
// Precondition: n >= 0
public static void printBinary(int n) {
    if (n < 2) {
        // base case; same as base 10
        System.out.println(n);
    } else {
        // recursive case; break number apart
        printBinary(n / 2);
        printBinary(n % 2);
    }
}

• Can we eliminate the precondition and deal with negatives?
Exercise

- Write a recursive method `isPalindrome` accepts a String and returns `true` if it reads the same forwards as backwards.

  - `isPalindrome("madam")` → `true`
  - `isPalindrome("racecar")` → `true`
  - `isPalindrome("step on no pets")` → `true`
  - `isPalindrome("able was I ere I saw elba")` → `true`
  - `isPalindrome("Java")` → `false`
  - `isPalindrome("rotater")` → `false`
  - `isPalindrome("byebye")` → `false`
  - `isPalindrome("notion")` → `false`
public static boolean isPalindrome(String s) {
    if (s.length() < 2) {
        return true;    // base case
    } else {
        char first = s.charAt(0);
        char last  = s.charAt(s.length() - 1);
        if (first != last) {
            return false;
        } else {      // recursive case
            String middle = s.substring(1, s.length() - 1);
            return isPalindrome(middle);
        }
    }
}
public static boolean isPalindrome(String s) {
    if (s.length() < 2) {
        return true;  // base case
    } else {
        return s.charAt(0) == s.charAt(s.length() - 1)
            && isPalindrome(s.substring(1, s.length() - 1));
    }
}