import stanford.karel.*;

public class UnitedNationsKarel extends SuperKarel {

    public void run() {
        while (frontIsClear()) {
            if (beepersPresent()) {
                pickBeeper();
                backup();
                buildHouse();
            } else if (frontIsClear()) {
                move();
            }
        }
    }

    private void buildHouse() {
        turnLeft();
        putThreeBeepers();
        move();
        turnRight();
        move();
        turnRight();
        putThreeBeepers();
        turnAround();
        move();
        turnRight();
        move();
        turnRight();
        putThreeBeepers();
        turnLeft();
    }

    // CONTINUED...
*/
* File: UnitedNationsKarel.java
* ----------------------------
* The UnitedNationsKarel program builds houses at corners
* marked by rubble.
*/
Try downloading the Eclipse project for this section from the course website and running this program on your own computer. What would you do if you wanted the houses to be taller? How would you make them 5 beepers high? Is there any way you could improve the style of the solutions?
/ * File: ChadKarel.java  
 *  
 * ChadKarel is a program in which Karel cleans up hanging chads  
 * from a ballot.  
 */

import stanford.karel.*;

class ChadKarel extends SuperKarel {

    public void run() {
        while (frontIsClear()) {
            processColumn();
            move();
        }
        processColumn();
    }

    private void processColumn() {
        // If there is chad to clear, clear that chad from the ballot.
        if (noBeepersPresent()) {
            removeAllChad();
        }
    }

    private void removeAllChad() {
        turnLeft();  // clean the upper corner
        cleanChad();
        turnAround();  // Clean the lower corner.
        cleanChad();
        turnLeft();  // Face East

        // CONTINUED...
private void cleanChad() {
    move();
    while (beepersPresent()) {
        pickBeeper();
    }
    backup();
}

private void backup() {
    turnAround();
    move();
    turnAround();
}

Try downloading the Eclipse project for this section from the course website and running this program on your own computer. What would you do if you wanted Karel to repair ballots that were broken (that is, the center of the rectangle is not punched out, but some other square in the rectangle is)?

Style Focus for Section 1:

Comments: Make sure to comment every method you write, and describe what the method does, and what the assumptions are before and after it is called. Write your comments so that your program could easily be understood by another person.

Good Method Names: Part of good style is good naming. You want your method name to succinctly describe what it does. Never call a method doStuff, give it a good specific name like backup. Be consistent in how you name your methods. In our solutions, we will use lower camel case naming conventions.

Short Methods: We could have written our whole program in the run method, but it is not good style and is difficult to follow. Try and break it down into methods that are small, understandable pieces of code, and that accomplish one main task. Establish milestones as you work so that you can write some code for one simpler task, test it, and fix it before moving on to the next task in solving a complex problem.

See the CS 106A Style Guide on the course website (linked to from the sidebar on the homepage) for more important style tips!