The first thing you probably noticed on this example is that it is significantly more code than the single-class implementation. And you're absolutely right – for this example, it makes much more sense to just use a single class. However, as your programs get bigger, you will need to divide them up like this to be able to keep track of what's going on. This example illustrates how to break things into several classes so that you'll know how when you do get a big project.

Example: Connect Four

In this example, we will use the Model-View-Controller architecture to implement a Connect Four game. Again, we will first divide this game into model, view, and controller components, and then implement each piece:

`IO` (View component)

`Scanner` instance variable/constructor to initialize

```
public void printBoard(String board)
public int getMove(char piece)
public void printResults(String msg)
```

`Board` (Model component)

`char[][]` instance variable/constructor to initialize

```
public boolean move(int column, char piece)
public boolean full()
public String toString()
public boolean winner(char piece)
```

`ConnectFour` (Controller component)

Single `main` method – call back and forth between `IO` and `Board.`

Now, here's the implementation of each class:

```
import java.util.*;
public class IO {
     private Scanner s;

     public IO() {
          s = new Scanner(System.in);
     }

     public void printBoard(String board) {
          System.out.println("\nCurrent board:\n");
          System.out.println(board);
          System.out.println();
     }

     public int getMove(char piece) {
          System.out.print("User "+piece+", enter a column: ");
```