```java
package Validator;

/** This class checks to see if a sodoku puzzle selected by
 *  the user is a valid solution
 *
 * @author Matt
 *
 * @version 1.0 (March 2013)                              */

import BasicIO.*;

public class SolutionValidator {

    private ASCIIDisplayer d;
    private ASCIIDataFile f;

    private int[][] p = new int[9][9]; // array to map puzzle

    public SolutionValidator ( ) {

        d = new ASCIIDisplayer (15, 50);
        f = new ASCIIDataFile ();

    } // constructor

    private void isValid ( ) {

        mapP();
        for (int i = 0 ; i < 9 ; i++) {
            for (int j = 0 ; j < 9 ; j++) {
                d.writeInt(p[j][i]);
            } // for
            d.writeLine("");
        } // for
        d.writeLine("");
        if (valid()) {
            d.writeLine("Puzzle is valid.");
        } else {
            d.writeLine("Puzzle is invalid.");
        } // else

    } // isValid

    private void mapP ( ) { // maps sudoku puzzle to array

        for (int i = 0 ; i < 9 ; i++) {
            for (int j = 0 ; j < 9 ; j++) {
                p[j][i] = f.readInt(); // read in board to 2-dimensional array
            } // for
        } // for
        f.close();

    } // mapP

    private boolean valid ( ) { // checks to see if puzzle is valid

        boolean[] numCheck = new boolean[9];
    boolean isValid = true;

    for (int i = 0 ; i < 9 ; i++) { // initialize numCheck (false = unused, true = used)
        numCheck[i] = false;
    } // for

    for (int i = 0 ; i < 9 ; i++) { // check if solved vertically
        for (int j = 0 ; j < 9 ; j++) {
            if (numCheck[p[i][j]-1] == true) {
                isValid = false;
            } else {
                numCheck[p[i][j]-1] = true;
            } // else
        } // for
        for (int j = 0 ; j < 9 ; j++) { // reinitializes numCheck
            numCheck[j] = false;
        } // for
    } // for

    if (isValid == true) { // only checks if valid
        for (int i = 0 ; i < 9 ; i++) { // check if solved horizontally
            for (int j = 0 ; j< 9 ; j++) {
                if (numCheck[p[j][i]-1] == true) {
                    isValid = false;
                } else {
                    numCheck[p[j][i]-1] = true;
                } // else
            } // for
            for (int j = 0 ; j < 9 ; j++) { // reinitializes numCheck
                numCheck[j] = false;
            } // for
        } // for
    } // if

    if (isValid == true) { // only checks if valid
        for (int i = 0 ; i < 3 ; i++) {
            for (int j = 0 ; j < 3 ; j++) {
                for (int k = 0 ; k < 3 ; k++) {
                    for (int l = 0 ; l < 3 ; l++) {
                        if (numCheck[p[k+(3*i)][l+(3*j)]-1] == true) {
                            isValid = false;
                        } else {
                            numCheck[p[k+(3*i)][l+(3*j)]-1] = true;
                        } // else
                    } // for
                } // for
                for (int k = 0 ; k < 9 ; k++) { // reinitializes numCheck
                    numCheck[k] = false;
                } // for
            } // for
        } // for
    } // if
    return isValid;

} // valid

public static void main ( String[] args ) {new SolutionValidator().isValid(); };
```

```
} // SolutionValidator
```