

Refactoring (Tetris #4 example)

Make the first test in FallingPiecesTest to compile.

```

33     fallingBlock = test;
34 }
35 }
36
37 private boolean conflictsWithBoard(Block block) {
38     return outsideBoard(block) || hitsAnotherBlock(block);
39 }
40
41 private boolean outsideBoard(Block block) {
42     return block.row() >= rows();
43 }
44
45 private boolean hitsAnotherBlock(Block block) {
46     return blocks[block.row()][block.col()] != EMPTY;
47 }
48
49 X public void drop(Tetrominoe tetrominoe) {
50 }
51
52 public void drop(Block block) {
53     if (hasFalling()) {
54         throw new IllegalStateException("Another block may not be dropped when one is already falling");
55     }
56     fallingBlock = block.moveTo(0, columns() / 2);
57 }
58
59 public boolean hasFalling() {
60     return fallingBlock != null;
61 }
62
63 private void stopFallingBlock() {
64     assert hasFalling();
65     copyToBoard(fallingBlock);
66     fallingBlock = null;
67 }
68
69 private void copyToBoard(Block block) {
70     for (int row = 0; row < rows(); row++) {
71         for (int col = 0; col < columns(); col++) {
72             if (block.isOccupied(row, col)) {
73                 blocks[row][col] = block.getTetrominoe().getSymbol();
74             }
75         }
76     }
77 }
78
79 public static class When_a_piece_is_dropped extends TestCase {
80     private Board board;
81
82     protected void setUp() throws Exception {
83         board = new Board(6, 8);
84         board.drop(Tetrominoe.T_SHAPE);
85     }
86
87     public void test_It_starts_from_top_middle() {
88         assertEquals(" +
89             "....T...\n" +
90             "...TTT...\n" +
91             ".....\n" +
92             ".....\n" +
93             ".....\n" +
94             ".....\n", board.toString());
95     }
96 }

```

Copy

Make Block implement the same interface as Tetrominoe.

```
10
11 /**
12  * @author Esko Luontola
13  */
14 public class Block implements Grid {
15
16     private final int row;
17     private final int col;
18     private final char style;
19
20     public Block(char style) {
21         this(0, 0, style);
22     }
23
24     private Block(int row, int col, char style) {
25         this.row = row;
26         this.col = col;
27         this.style = style;
28     }
29
30     public int row() {
31         return row;
32     }
33
34     public int col() {
35         return col;
36     }
37
38     public char style() {
39         return style;
40     }
41
42     public int rows() {
43         return 1;
44     }
45
46     public int columns() {
47         return 1;
48     }
49
50     public char cellAt(int row, int col) {
51         return style;
52     }
53
54     public boolean isAt(int row, int col) {
55         return row == this.row && col == this.col;
56     }
57
58     public Block moveTo(int row, int col) {
59         return new Block(row, col, style);
60     }
61 }
```

Use Block through the same interface as Tetrominoe.

public boolean hasFalling() { return fallingBlock != null ; }	59	62	
	60	63	
private void stopFallingBlock() { assert hasFalling(); copyToBoard(fallingBlock); fallingBlock = null ; }	61	64	private void stopFallingBlock() { assert hasFalling(); copyToBoard(fallingBlock); fallingBlock = null ; }
	62	65	
	63	66	
	64	67	
	65	68	
	66	69	private void copyToBoard(Block block) { for (int row = 0; row < blocks.length; row++) { for (int col = 0; col < blocks[row].length; col++) { if (block.isAt(row, col)) { blocks[row][col] = block.cellAt(row, col); } } }
private void copyToBoard(Block block) { blocks[block.row()][block.col()] = block.style(); }	67	70	
	68	71	
	69	72	
	70	73	
	71	74	
	72	75	
public int rows() { return blocks.length; }	73	76	
	74	77	
	75	78	
	76	79	
public int columns() { return blocks[0].length; }	77	80	public int rows() { return blocks.length; }
	78	81	
	79	82	
	80	83	public int columns() { return blocks[0].length; }
public char cellAt(int row, int col) { if (fallingBlock != null && fallingBlock.isAt(row, col)) { return fallingBlock.style(); } else { return blocks[row][col]; } }	81	84	
	82	85	
	83	86	public char cellAt(int row, int col) { if (fallingBlock != null && fallingBlock.isAt(row, col)) { return fallingBlock.cellAt(row, col); } else { return blocks[row][col]; } }
	84	87	
	85	88	
	86	89	
	87	90	
	88	91	
public String toString() { return new GridAsciiView(this).toString(); }	89	92	
	90	93	
	91	94	
	92	95	public String toString() { return new GridAsciiView(this).toString(); }
	93	96	
		97	
		98	
		99	

Make Block contain a Grid.

```
10
11 /**
12  * @author Esko Luontola
13  */
14 public class Block implements Grid {
15
16     private final int row;
17     private final int col;
18     private final char style;
19 x private final Grid inner;
20
21     public Block(char style) {
22         this(0, 0, style, new Piece(style + "\n"));
23     }
24
25 x public Block(Grid inner) {
26     this(0, 0, 'z', inner);
27 }
28
29 private Block(int row, int col, char style, Grid inner) {
30     this.row = row;
31     this.col = col;
32     this.style = style;
33 x this.inner = inner;
34 }
35
36 public int row() {
37     return row;
38 }
39
55
56 a↑ public char cellAt(int row, int col) {
57     return style;
58 }
59
60 public boolean isAt(int row, int col) {
61     return row == this.row && col == this.col;
62 }
63
64 public Block moveTo(int row, int col) {
65     return new Block(row, col, style, inner);
66 }
67
68 public Block moveDown() {
69     return new Block(row + 1, col, style, inner);
70 }
71 }
72
```

Change Block to delegate its Grid methods to the contained Grid.

<pre> public char style() { return style; } public int rows() { return 1; } public int columns() { return 1; } public char cellAt(int row, int col) { return style; } public boolean isAt(int row, int col) { return row == this.row && col == this.col; } public Block moveTo(int row, int col) { return new Block(row, col, style, inner); } public Block moveDown() { </pre>	<pre> 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 </pre>	<pre> 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 </pre>	<pre> public char style() { return style; } public int rows() { return inner.rows(); } public int columns() { return inner.columns(); } public char cellAt(int row, int col) { return inner.cellAt(row, col); } public boolean isAt(int row, int col) { return row == this.row && col == this.col; } public Block moveTo(int row, int col) { return new Block(row, col, style, inner); } public Block moveDown() { </pre>
---	--	--	---

Run All tests

Done: 58 of 58 Failed: 14(0,047 s)

Output Statistics

```

java.lang.ArrayIndexOutOfBoundsException: 1
    at tetris.Piece.cellAt(Piece.java:89)
    at tetris.Block.cellAt(Block.java:57)
    at tetris.Board.cellAt(Board.java:89)
    at tetris.GridAsciiView.toString(GridAsciiView.java:26)
    at tetris.Board.toString(Board.java:96)
    at tetris.FallingBlocksTest$When_a_block_is_dropped.te
    at sun.reflect.NativeMethodAccessorImpl.invoke0(Native
    at sun.reflect.NativeMethodAccessorImpl.invoke(Native

```

Fix the row/column coordinate mismatch in Block.cellAt().

```

}
public int columns() {
    return inner.columns();
}

public char cellAt(int row, int col) {
    return inner.cellAt(row, col);
}

public boolean isAt(int row, int col) {
    return row == this.row && col == this.col;
}

public Block moveTo(int row, int col) {
    return new Block(row, col, style, inner);
}

public Block moveDown() {
    return new Block(row + 1, col, style, inner);
}
}

50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72

51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74

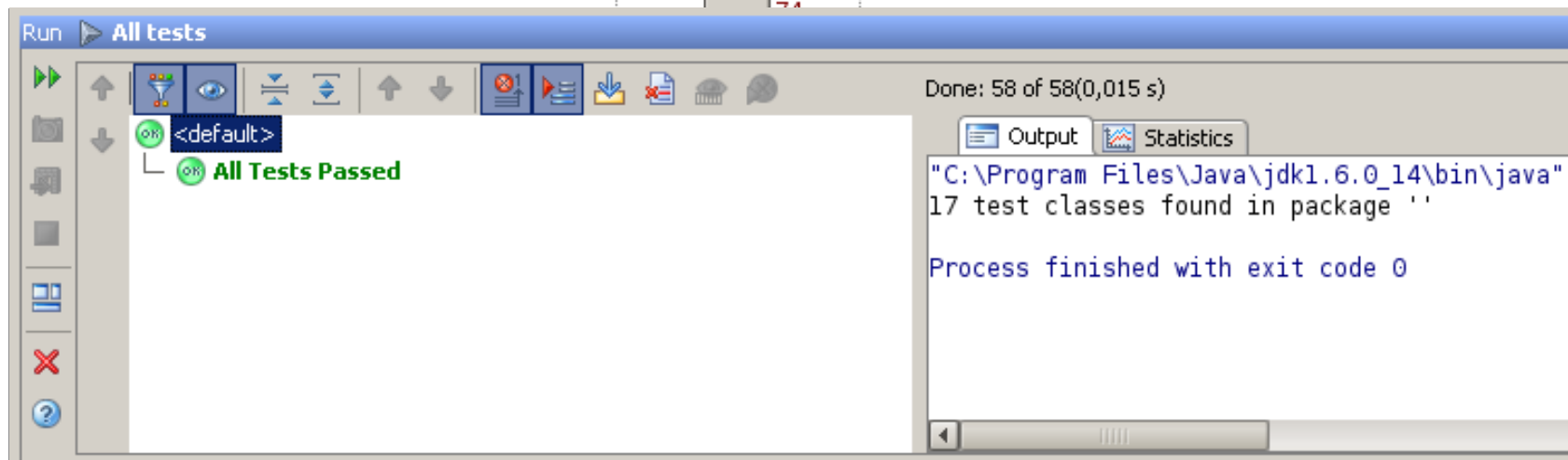
public int columns() {
    return inner.columns();
}

public char cellAt(int row, int col) {
    int innerRow = row - rowOffset;
    int innerCol = col - colOffset;
    return inner.cellAt(innerRow, innerCol);
}

public boolean isAt(int row, int col) {
    return row == this.rowOffset && col == this.colOffset;
}

public Block moveTo(int row, int col) {
    return new Block(row, col, style, inner);
}

public Block moveDown() {
    return new Block(rowOffset + 1, colOffset, style, inner);
}
}
```



Change Board.drop() to accept any Grid.
Block contains the movement logic, so we wrap the Grid in it.

private boolean hitsAnotherBlock(Block block) { return blocks[block.row()][block.col()] != EMPT }	44 45 46 47 48	44 45 .6 47 48	private boolean hitsAnotherBlock(Block block) { return blocks[block.row()][block.col()] != EMPTY ; }
public void drop(Tetrominoe tetrominoe) { }	» 49 50 51	49 .0 *1	public void drop(Grid block) { if (hasFalling()) { throw new IllegalStateException("Another block may not b
public void drop(Block block) { if (hasFalling()) { throw new IllegalStateException("Another bl } fallingBlock = block.moveTo(0, columns() / 2); }	52 53 54 55 » 56 57 58 59 60	52 .3 54 55 56 .7 58 59 60	} fallingBlock = new Block(block).moveTo(0, columns() / 2); } public boolean hasFalling() { return fallingBlock != null ; } private void stopFallingBlock() {

Try running FallingPiecesTest. It fails.

```
21
22 public static class When_a_piece_is_dropped extends TestCase {
23
24     private Board board;
25
26     protected void setUp() throws Exception {
27         board = new Board(6, 8);
28         board.drop(Tetrominoe.T_SHAPE);
29     }
30
31     public void test_It_starts_from_top_middle() {
32         assertEquals("'''' +
33             \"....T...\\n\" +
34             \"...TTT...\\n\" +
35             \".....\\n\" +
36
37
38
39     }
40 }
41
42
```

assertEquals(String, String) failed

Ignore whitespace:

Expected(Read-only)		Actual(Read-only)	
1T...	1
2	...TTT...	2
3	3
4	4
5	5
6	6
7	7

2 differences

Deleted Changed Inserted

Done: 58 of 58 Failed: 2(0,031 s)

Output Statistics

junit.framework.ComparisonFailure: null [Click to see difference](#)
at tetris.FallingPiecesTest\$When_a_piece_is_dropped.test_It_starts_from_top_middle([FallingPiecesTest.java](#))
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
at sun.reflect.NativeMethodAccessorImpl.invoke([NativeMethodAccessorImpl.java:39](#))
at sun.reflect.DelegatingMethodAccessorImpl.invoke([DelegatingMethodAccessorImpl.java:25](#))
at com.intellij.rt.execution.junit.JUnit4TestRunner.main(JUnit4TestRunner.java:40)
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
at sun.reflect.NativeMethodAccessorImpl.invoke([NativeMethodAccessorImpl.java:39](#))
at sun.reflect.DelegatingMethodAccessorImpl.invoke([DelegatingMethodAccessorImpl.java:25](#))

Fix the row/column coordinate mismatch in Block.isAt().

Now FallingPiecesTest *almost* passes.

```

public char cellAt(int row, int col) {
    int innerRow = row - rowOffset;
    int innerCol = col - colOffset;
    return inner.cellAt(innerRow, innerCol);
}

public boolean isAt(int row, int col) {
    return row == this.rowOffset && col == this.colOffset;
}

public Block moveTo(int row, int col) {
    return new Block(row, col, style, inner);
}

public Block moveDown() {
    return new Block(rowOffset + 1, colOffset, style, inner);
}

```

56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74

```

public char cellAt(int row, int col) {
    int innerRow = row - rowOffset;
    int innerCol = col - colOffset;
    return inner.cellAt(innerRow, innerCol);
}

public boolean isAt(int row, int col) {
    int innerRow = row - rowOffset;
    int innerCol = col - colOffset;
    return innerRow >= 0 && innerRow < inner.rows() &&
        innerCol >= 0 && innerCol < inner.columns();
}

public Block moveTo(int row, int col) {
    return new Block(row, col, style, inner);
}

public Block moveDown() {
    return new Block(rowOffset + 1, colOffset, style, inner);
}

```

assertEquals(String, String) failed

Ignore whitespace: Do not ignore

Expected(Read-only)		Actual(Read-only)	
1T..	1T..
2	...TTT.	2	...TTT.
3	3
4	4
5	5
6	6
7	7

2 differences

Deleted Changed Inserted

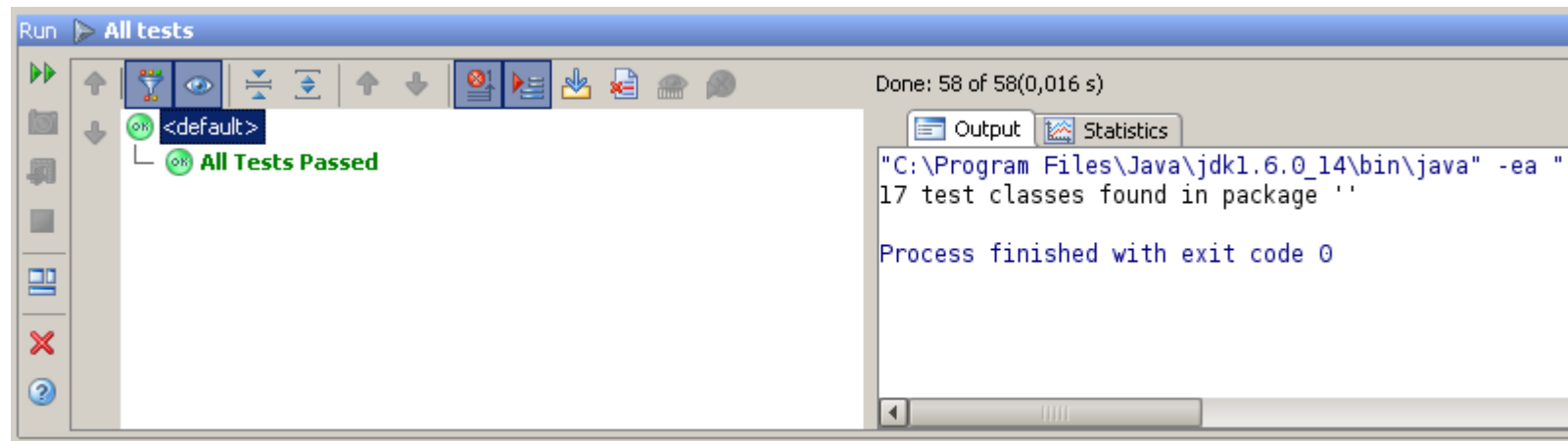
Done: 58 of 58 Failed: 2(0,047 s)

Output Statistics

junit.framework.ComparisonFailure: null [<Click to see difference>](#)
 at tetris.FallingPiecesTest\$When_a_piece_is_dropped.test_It_starts_from_top_middle(FallingPiecesTest.java:32)
 at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
 at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39)
 at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)

Fix the falling Grid's initial position in Board.drop().
Now FallingPiecesTest passes. (Wohoo!)

<pre> lock(Block block) {][block.col()] != EMPTY; { teException("Another block may not (block).moveTo(0, columns() / 2); </pre>	<pre> 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 </pre>	<pre> private boolean hitsAnotherBlock(Block block) { return blocks[block.row()][block.col()] != EMPTY; } public void drop(Grid block) { if (hasFalling()) { throw new IllegalStateException("Another block may not be dropped when one is alr } fallingBlock = new Block(block).moveTo(0, columns() / 2 - block.columns() / 2); } public boolean hasFalling() { return fallingBlock != null; } private void stopFallingBlock() { assert hasFalling(); copyToBoard(fallingBlock); fallingBlock = null; } </pre>
--	--	--



```

13 */
14 public class Block implements Grid {
15
16     private final int rowOffset;
17     private final int colOffset;
18     private final char style;
19     private final Grid inner;
20
21     public Block(char style) {
22         this(0, 0, style, new Piece(style + "\n"));
23     }
24
25     public Block(Grid inner) {
26         this(0, 0, 'z', inner);
27     }
28
29     private Block(int rowOffset, int colOffset, char style, Grid inner) {
30         this.rowOffset = rowOffset;
31         this.colOffset = colOffset;
32         this.style = style;
33         this.inner = inner;
34     }
35
36     public int row() {
37         return rowOffset;
38     }
39
40     public int col() {
41         return colOffset;
42     }
43
44     public char style() {
45         return style;
46     }
47
48     public int rows() {
49         return inner.rows();
50     }
51
52     public int columns() {
53         return inner.columns();
54     }
55
56     public char cellAt(int row, int col) {
57         int innerRow = row - rowOffset;
58         int innerCol = col - colOffset;
59         return inner.cellAt(innerRow, innerCol);
60     }
61
62     public boolean isAt(int row, int col) {
63         int innerRow = row - rowOffset;
64         int innerCol = col - colOffset;
65         return innerRow >= 0 && innerRow < inner.rows() &&
66             innerCol >= 0 && innerCol < inner.columns();
67     }
68
69     public Block moveTo(int row, int col) {
70         return new Block(row, col, style, inner);
71     }
72
73     public Block moveDown() {
74         return new Block(rowOffset + 1, colOffset, style, inner);
75     }
76 }

```

Block is a mess.

Remove the obvious code duplication.

<pre> public int rows() { return inner.rows(); } public int columns() { return inner.columns(); } public char cellAt(int row, int col) { int innerRow = row - rowOffset; int innerCol = col - colOffset; return inner.cellAt(innerRow, innerCol); } public boolean isAt(int row, int col) { int innerRow = row - rowOffset; int innerCol = col - colOffset; return innerRow >= 0 && innerRow < inner.rows() && innerCol >= 0 && innerCol < inner.columns(); } public Block moveTo(int row, int col) { return new Block(row, col, style, inner); } public Block moveDown() { return new Block(rowOffset + 1, colOffset, style, inner); } </pre>	<div>47</div> <div>48</div> <div>49</div> <div>50</div> <div>51</div> <div>52</div> <div>53</div> <div>54</div> <div>55</div> <div>56</div> <div>57</div> <div>58</div> <div>59</div> <div>60</div> <div>61</div> <div>62</div> <div>63</div> <div>64</div> <div>65</div> <div>66</div> <div>67</div> <div>68</div> <div>69</div> <div>70</div> <div>71</div> <div>72</div> <div>73</div> <div>74</div> <div>75</div> <div>76</div> <div>77</div>	<pre> 51 52 ↗ public int columns() { 53 return inner.columns(); 54 } 55 56 ↗ public char cellAt(int row, int col) { 57 int innerRow = toInnerRow(row); 58 int innerCol = toInnerCol(col); 59 return inner.cellAt(innerRow, innerCol); 60 } 61 62 public boolean isAt(int row, int col) { 63 int innerRow = toInnerRow(row); 64 int innerCol = toInnerCol(col); 65 return innerRow >= 0 && innerRow < inner.rows() && 66 innerCol >= 0 && innerCol < inner.columns(); 67 } 68 69 × private int toInnerCol(int col) { 70 return col - colOffset; 71 } 72 73 private int toInnerRow(int row) { 74 return row - rowOffset; 75 } 76 77 public Block moveTo(int row, int col) { 78 return new Block(row, col, style, inner); 79 } 80 81 public Block moveDown() { 82 return new Block(rowOffset + 1, colOffset, style, inner); 83 } 84 } </pre>
---	---	---

Block has two responsibilities: moving the Grid and being a 1x1 Grid itself.
Copy Block into MovableGrid and remove everything having to do with the 1x1 Grid.

<i>* @author Esko Luontola</i>	12	11	<i>/**</i>
<i>*/</i>	13	12	<i>* @author Esko Luontola</i>
public class Block implements Grid {	14	13	<i>*/</i>
private final int rowOffset;	15	14	public class MovableGrid implements Grid {
private final int colOffset;	16	15	
private final char style;	17	16	private final int rowOffset;
private final Grid inner;	18	17	private final int colOffset;
	19	18	private final Grid inner;
public Block(char style) {	20	19	
this (0, 0, style, new Piece(style + "\n"));	21	20	public MovableGrid(Grid inner) {
}	22	21	this (0, 0, inner);
	23	22	}
public Block(Grid inner) {	24	23	
this (0, 0, 'z', inner);	25	24	private MovableGrid(int rowOffset, int colOffset, Grid inner) {
}	26	25	this .rowOffset = rowOffset;
	27	26	this .colOffset = colOffset;
private Block(int rowOffset, int colOffset, char style, Grid inner) {	28	27	this .inner = inner;
this .rowOffset = rowOffset;	29	28	}
this .colOffset = colOffset;	30	29	
this .style = style;	31	30	public int row() {
this .inner = inner;	32	31	return rowOffset;
}	33	32	}
	34	33	
public int row() {	35	34	public int col() {
return rowOffset;	36	35	return colOffset;
}	37	36	}
	38	37	
public int col() {	39	38	public int rows() {
return colOffset;	40	39	return inner.rows();
}	41	40	}
	42	41	
public char style() {	43	42	public int columns() {
return style;	44	43	return inner.columns();
}	45	44	}
	46	45	
public int rows() {	47	46	public char cellAt(int row, int col) {
return inner.rows();	48	47	int innerRow = toInnerRow(row);
}	49	48	int innerCol = toInnerCol(col);
	50	49	return inner.cellAt(innerRow, innerCol);
	51	50	}

Also make the row/column coordinates explicit.

public int columns() { return inner.columns(); }	41 42 43 44 45	41 42 43 44 45	public int columns() { return inner.columns(); }
public char cellAt(int row, int col) { int innerRow = toInnerRow(row); int innerCol = toInnerCol(col); return inner.cellAt(innerRow, innerCol); }	>> 46 >> 47 >> 48 49 50 51	46 47 48 49 50 51	public char cellAt(int outerRow, int outerCol) { int innerRow = toInnerRow(outerRow); int innerCol = toInnerCol(outerCol); return inner.cellAt(innerRow, innerCol); }
public boolean isAt(int row, int col) { int innerRow = toInnerRow(row); int innerCol = toInnerCol(col); return innerRow >= 0 && innerRow < inner.rows() && innerCol >= 0 && innerCol < inner.columns(); }	>> 52 >> 53 >> 54 55 56 57 58	52 53 54 55 56 57 58	public boolean isAt(int outerRow, int outerCol) { int innerRow = toInnerRow(outerRow); int innerCol = toInnerCol(outerCol); return innerRow >= 0 && innerRow < inner.rows() && innerCol >= 0 && innerCol < inner.columns(); }
private int toInnerCol(int col) { return col - colOffset; }	>> 59 >> 60 61 62	59 60 61 62	private int toInnerCol(int outerCol) { return outerCol - colOffset; }
private int toInnerRow(int row) { return row - rowOffset; }	>> 63 >> 64 65 66	63 64 65 66	private int toInnerRow(int outerRow) { return outerRow - rowOffset; }
public MovableGrid moveTo(int row, int col) { return new MovableGrid(row, col, inner); }	>> 67 >> 68 69 70	67 68 69 70	public MovableGrid moveTo(int rowOffset, int colOffset) { return new MovableGrid(rowOffset, colOffset, inner); }
public MovableGrid moveDown() { return new MovableGrid(rowOffset + 1, colOffset, inner); }	71 72 73 74	71 72 73 74	public MovableGrid moveDown() { return new MovableGrid(rowOffset + 1, colOffset, inner); }


```

13
14  * @auth Change Board to use MovableGrid instead of Block.
15  */
16  public class Board implements Grid {
17
18      private MovableGrid fallingBlock;
19      private char[][] blocks;
20
21      public Board(int rows, int columns) {
22          blocks = new char[rows][columns];
23          for (char[] tmp : blocks) {
24              Arrays.fill(tmp, EMPTY);
25          }
26      }
27
28      public void tick() {
29          MovableGrid test = fallingBlock.moveDown();
30          if (conflictsWithBoard(test)) {
31              stopFallingBlock();
32          } else {
33              fallingBlock = test;
34          }
35      }
36
37      private boolean conflictsWithBoard(MovableGrid block) {
38          return outsideBoard(block) || hitsAnotherBlock(block);
39      }
40
41      private boolean outsideBoard(MovableGrid block) {
42          return block.row() >= rows();
43      }
44
45      private boolean hitsAnotherBlock(MovableGrid block) {
46          return blocks[block.row()][block.col()] != EMPTY;
47      }
48
49      public void drop(Grid block) {
50          if (hasFalling()) {
51              throw new IllegalStateException("Another block may not be dropped when one is already falling");
52          }
53          fallingBlock = new MovableGrid(block).moveTo(0, columns() / 2 - block.columns() / 2);
54      }
55
56      public boolean hasFalling() {
57          return fallingBlock != null;
58      }
59
60      private void stopFallingBlock() {
61          assert hasFalling();
62          copyToBoard(fallingBlock);
63          fallingBlock = null;
64      }
65
66      private void copyToBoard(MovableGrid block) {
67          for (int row = 0; row < blocks.length; row++) {

```



```

14  * @author Esko Luontola
15  */
16  public class Board implements Grid {
17
18      private MovableGrid falling;
19      private char[][] blocks;
20
21      public Board(int rows, int columns) {
22          blocks = new char[rows][columns];
23          for (char[] tmp : blocks) {
24              Arrays.fill(tmp, EMPTY);
25          }
26      }
27
28      public void tick() {
29          MovableGrid test = falling.moveDown();
30          if (conflictsWithBoard(test)) {
31              stopFalling();
32          } else {
33              falling = test;
34          }
35      }
36
37      private boolean conflictsWithBoard(MovableGrid piece) {
38          return outsideBoard(piece) || hitsStationaryBlock(piece);
39      }
40
41      private boolean outsideBoard(MovableGrid piece) {
42          return piece.row() >= rows();
43      }
44
45      private boolean hitsStationaryBlock(MovableGrid piece) {
46          return blocks[piece.row()][piece.col()] != EMPTY;
47      }
48
49      public void drop(Grid piece) {
50          if (hasFalling()) {
51              throw new IllegalStateException("Another piece may not be dropped when one is already falling");
52          }
53          falling = new MovableGrid(piece).moveTo(0, columns() / 2 - piece.columns() / 2);
54      }
55
56      public boolean hasFalling() {
57          return falling != null;
58      }
59
60      private void stopFalling() {
61          assert hasFalling();
62          copyToBoard(falling);
63          falling = null;
64      }
65
66      private void copyToBoard(MovableGrid block) {
67          for (int row = 0; row < blocks.length; row++) {

```

Remove from Block everything that is already in MovableGrid.
Now Block is used only in FallingBlocksTest, so it can be moved to the test source folder.

```
10
11  /**
12   * @author Esko Luontola
13   */
14  public class Block implements Grid {
15
16      private final char style;
17
18      public Block(char style) {
19          this.style = style;
20      }
21
22      public int rows() {
23          return 1;
24      }
25
26      public int columns() {
27          return 1;
28      }
29
30      public char cellAt(int row, int col) {
31          return style;
32      }
33  }
34
```