Step 2: The Board class

The interesting modeling of the game will be in `Board.java` and other class files. So start reading code at `Board.java`.

WE'LL COVER THE FOLLOWING

To focus on learning Java, you may reasonably ignore the Tetrominos class. It will be the file that you run to start the program, but the `Board` class can be thought of as the core of your Tetrominos program, and it's where you'll write the most code. Go to Eclipse and start reading the code now.

The keyword `extends` and some provided code in the constructor set the Board up as a user interface component that can be added to a window, and you may safely ignore this code.

Each of the other existing methods in the Board class will be called in response to certain actions, either by the user, or due to time passing. Specifically:

- `paintComponent` will be called any time you need to draw the blocks on the screen.
- `nextTurn` will be called every 300 milliseconds, when it is time to move the active piece downwards on the screen.
- `slide`, `rotateLeft`, and `rotateRight` will be called when the user presses control keys to move the game piece. Calls to `repaint()` in these methods request that `paintComponent` be called so that you can see the results of the action.

Exercise: test user actions

Add `System.out.println` calls to `nextTurn`, `slide`, `rotateLeft`, and `rotateRight`, and run the program. Observe how methods are called in response to the timer and in response to key presses

Drawing to the screen with Graphics objects

Swing drawing is done using objects of the `JComponent` class. The `extends` keyword in the definition of `Board` indicates that a Board can be used as a JComponent, and has several standard methods available to it. Once a JComponent has been added to the user interface (in this case by Tetrominos.java), the special method `paintComponent` will be called whenever the user interface needs to draw that component.

So, if you put drawing code in `paintComponent` of the `Board` class, that drawing code will be called at least once, when the main window is initially drawn. When `paintComponent` is drawn, it is passed a parameter, a reference to a Graphics object. You can draw on the component using methods of this Graphics object.

The starter code I've provided draws two rectangles:

```
	public void paintComponent(Graphics g) {
		// clear the screen with black
		g.setColor(Color.black);
		g.fillRect(0, 0, getWidth(), getHeight());

		// demonstrate drawing a rectangle
		g.setColor(Color.blue);
		g.fillRect(120, 120, 42, 19);

	}
```

The `setColor` method takes as a parameter a `Color` object. Several references to Color objects are available in the `Color` class, which must be imported. In Eclipse, verify that `java.awt.Color` has been imported. (`awt` is a reference to the Abstract Windowing Toolkit, which was user interface library that shipped with Java before Swing was developed.)

The `fillRect()` method has parameters that indicate the x and y coordinates of the upper left of the rectangle, and width in height. Go to Exlipse and experiment with drawing a few other rectangles. Don't forget to run the

experiment with drawing a few other rectangles. Don't forget to run the program using Tetrominos.java.

When you are done, delete all of the code in paintComponent except the first two calls that draw a black rectangle as the background for your game.

You might notice the calls to `getWidth()` and `getHeight()`. These calls are methods of the JComponent class, and therefore are available to call on the `this` object passed into `paintComponent`.