Tracing a Method Using SwapRedBlue

Objectives:

The objectives of this lesson are two fold:

- To introduce the idea and process of tracing a method.
- To reinforce the concept of typing and in particular of object reference types.

I. Problem
a. Illustrate by running the BadSwapTest program.
b. The problem is that the colors are wrong. Why?
c. To see why we need to trace the code. That means we need to hand-execute the method.

II. Consider the following method code.

```java
public void swapRedBlue4() {
    // get a one dimensional array of the pixels in the picture
    Pixel [] pixels = this.getPixels();

    // swap the red and blue for each pixel in the array
    for (Pixel curPixel: pixels) {
        // get a copy of the curPixel
        Pixel tempPix = curPixel;

        // swap the curPixel's red and blue
        curPixel.setRed(tempPix.getBlue());
        curPixel.setBlue(tempPix.getRed());
    }
}
```

III. In order to figure out what is wrong we can trace the code. We start by supposing we have a small picture object and then execute the code from there. Consider the diagram on the next page. It shows the state of objects in memory after the execution of the first statement. Add the curPixel variable and the tempPixel. Introduce the idea of scope by emphasizing how these variables are know only within the body of the loop.
IV. Fixing the problem. The obvious fix is to replace the tempPixel variable with an int variable, oldRedValue and use it in the swap.

```java
public void swapRedBlue4() {
    // get a one dimensional array of the pixels in the picture
    Pixel [] pixels = this.getPixels();

    // swap the red and blue for each pixel in the array
    for (Pixel curPixel: pixels) {
        // get a copy of the curPixel
        int oldRedValue = curPixel.getRed();

        // swap the curPixel's red and blue
        curPixel.setRed(curPixel.getBlue());
        curPixel.setBlue(oldRedValue);
    }
}
```