CMSC131

Basic 2D Graphics: Part II

Pixel class (for P4)

While we could use the **Color** class provided in the Java libraries, we will use a locally written class called **`Pixel`** in Project 4 to get a little more practice thinking about simple objects.

Pixel class: Creation

When you create a new **`Pixel`** object you will specify an amount of red, green, and blue to the constructor. These channels cannot be altered once the object is created. You can, however, get each channel's value back out:

- **`getRed()`**: returns an **`int`** representing red in pixel
- **`getGreen()`**: returns an **`int`** representing green in pixel
- **`getBlue()`**: returns an **`int`** representing blue in pixel

Some Example Colors

```
Pixel myRed = new Pixel(255,0,0);
Pixel myGreen = new Pixel(0,255,0);
Pixel myDarkerGreen = new Pixel(0,191,0);
Pixel myBlue = new Pixel(0,0,255);
Pixel myWhite = new Pixel(255,255,255);
Pixel myBlack = new Pixel(0,0,0);
Pixel myOrange = new Pixel(255,128,0);
```

Photograph class (for P4)

When you create a new **`Photograph`** object in the next project, you will specify a width and height to the constructor. These dimensions cannot be altered once the object is created.

Photograph class: functionality

The following instance methods are available:

- **`getHt()`**: returns an **`int`** representing its pixel height
- **`getWd()`**: returns an **`int`** representing its pixel width
- **`getPixel(int x, int y)`**: returns a reference to the **`Pixel`** in the specified x,y-coord position
- **`setPixel(int x, int y, Pixel p)`**: sets the **`Pixel`** in the specified x,y-coord position

Example: solid color

A helper method that is send a reference to a **Photograph** object and needs to paint it white:

```
public static void helperWhite(Photograph theImage) {
  for (int x=0; x<theImage.getWd(); x++) {
    for (int y=0; y<theImage.getHt(); y++) {
      theImage.setPixel(x, y, new Pixel(255,255,255));
    }
  }
}
```

Note that "x" and "y" are meaningful variable names in this context!

Example: simple pattern

A helper method that is sent a reference to a **Photograph** object and paints the left half red and the right half blue:

```
public static void helperLeftAndRight(Photograph theImage) {
   for (int y=0; y<theImage.getHt(); y++) {
      for (int x=0; x<theImage.getWd()/2; x++) {
         theImage.setPixel(x, y, new Pixel(255,0,0));
      }
      for (int x=theImage.getWd()/2; x<theImage.getWd(); x++) {
         theImage.setPixel(x, y, new Pixel(0,0,255));
      }
   }
}
```

Example: pattern with parameters

Imagine needing to write a similar helper method that not only gets a reference to a **Photograph** object but also to two **Pixel** objects and paints the top half using the first **Pixel** and the bottom half using the second.

The method signature could be:

```
public static void helperLeftAndRight(
    Photograph theImage,
    Pixel firstPixel, Pixel secondPixel
)
```

Half and Half

```
public static void helperLeftAndRight(
    Photograph theImage,
    Pixel firstPixel, Pixel secondPixel
) {
 for (int y=0; y<theImage.getHt(); y++) {
      for (int x=0; x<theImage.getWd()/2; x++) {
         theImage.setPixel(x, y, firstPixel);
      }
      for (int x=theImage.getWd()/2; x<theImage.getWd(); x++) {
         theImage.setPixel(x, y, secondPixel);
      }
   }
}
```