## Pixlab 4:

1.

```
public int getCount(int number)
  {
    int number_found = 0;
    for(int i_row=0; i_row < matrix.length; i_row++)
    {
        for(int i_col=0; i_col < matrix[0].length; i_col++)
        {
            if(matrix[i_row][i_col] == number) {
                number_found++;
            }
        }
    }
    return number_found;
  }
```

2.

```
public int getLargest()
  {
    int largest = 0;
    for(int i_row=0; i_row < matrix.length; i_row++)
    {
        for(int i_col=0; i_col < matrix[0].length; i_col++)
        {
            if(matrix[i_row][i_col] > largest) {
                largest = matrix[i_row][i_col];
            }
        }
    }
    return largest;
  }
```

3.

```
public int getColTotal(int index)
  {
    int col_total = 0;
    for(int i_row=0; i_row < matrix.length; i_row++)
    {
        col_total = col_total + matrix[i_row][index];
    }
    return col_total;
  }
```

## Pixlab 5:

1. No
2. Yes
3. No the compiler returns an error “`classes.DigitalPicture is abstract; cannot be instantiated.`
4. Yes, the code compiles without error

5. Yes, assuming there is a no-argument constructor
6. yes
7. No the compiler returns an error “`incompatible types`”

1. yes it worked.
2. works.
3.

```
public void KeepOnlyBlue()
  {
    Pixel[][] pixels = this.getPixels2D();
    for (Pixel[] rowArray : pixels)
    {
      for (Pixel pixelObj : rowArray)
      {
        pixelObj.setRed(0);
        pixelObj.setGreen(0);
      }
    }
  }
```

4.

```
public void Negate()
  {
    Pixel[][] pixels = this.getPixels2D();
    for (Pixel[] rowArray : pixels)
    {
      for (Pixel pixelObj : rowArray)
      {
          pixelObj.setRed(255 - pixelObj.getRed());
          pixelObj.setGreen(255 - pixelObj.getGreen());
          pixelObj.setBlue(255 - pixelObj.getBlue());
      }
    }
  }
```

5.

```
public void Grayscale()
  {
    Pixel[][] pixels = this.getPixels2D();
    for (Pixel[] rowArray : pixels)
    {
      for (Pixel pixelObj : rowArray)
      {
          pixelObj.setRed((pixelObj.getRed() + pixelObj.getGreen() + pixelObj.getBlue()) /
3);
          pixelObj.setGreen((pixelObj.getRed() + pixelObj.getGreen() + pixelObj.getBlue()) /
3);
          pixelObj.setBlue((pixelObj.getRed() + pixelObj.getGreen() + pixelObj.getBlue()) /
3);
      }
    }
  }
```

6.

```java
public void fixUnderwater()
  {
    Pixel[][] pixels = this.getPixels2D();
    for (Pixel[] rowArray : pixels)
    {
      for (Pixel pixelObj : rowArray)
      {
         pixelObj.setRed(70 + pixelObj.getRed());
         pixelObj.setGreen(pixelObj.getBlue() - 50);
         pixelObj.setBlue(pixelObj.getGreen() + 50);

      }
    }
  }
```

## Pixlab 6:

1.

```java
public void mirrorVerticalRightToLeft()
  {
    Pixel[][] pixels = this.getPixels2D();
    Pixel leftPixel = null;
    Pixel rightPixel = null;
    int width = pixels[0].length;
    for (int row = 0; row < pixels.length; row++)
    {
      for (int col = 0; col < width / 2; col++)
      {
        leftPixel = pixels[row][col];
        rightPixel = pixels[row][width - 1 - col];
        leftPixel.setColor(rightPixel.getColor());
      }
    }
  }
```

2.

```java
public void mirrorHorizontal()
  {
    Pixel[][] pixels = this.getPixels2D();
    Pixel topPixel = null;
    Pixel bottomPixel = null;
    int width = pixels[0].length;
    int length = pixels.length;
    for (int row = 0; row < length / 2; row++)
    {
      for (int col = 0; col < width; col++)
      {
        topPixel = pixels[row][col];
        bottomPixel = pixels[(length - 1) - row][col];
        bottomPixel.setColor(topPixel.getColor());
        System.out.println(topPixel.getColor());
      }
    }
  }
```

3.

```java
public void mirrorDiagnol()
  {
    Pixel[][] pixels = this.getPixels2D();
    Pixel topPixel = null;
    Pixel bottomPixel = null;
    int width = pixels[0].length;
    int length = pixels.length;
    int col_offset = 0;
    for (int row = 0; row < length; row++)
    {
      for (int col = 0; col < width - col_offset; col++)
      {
        topPixel = pixels[row][col];
        bottomPixel = pixels[(length - 1) - row][(width - 1) - col];
        bottomPixel.setColor(topPixel.getColor());
      }
      col_offset = col_offset + width / length;
    }
  }
```

## **Pixlab 7:**

1. The loop would execute 72 times
2. The loop would execute 90 times

1.

```java
public void mirrorTemple()
  {
    int mirrorPoint = 276;
    Pixel leftPixel = null;
    Pixel rightPixel = null;
    int count = 0;
    Pixel[][] pixels = this.getPixels2D();

    // loop through the rows
    for (int row = 27; row < 97; row++)
    {
      // loop from 13 to just before the mirror point
      for (int col = 13; col < mirrorPoint; col++)
      {

        leftPixel = pixels[row][col];
        rightPixel = pixels[row]
                         [mirrorPoint - col + mirrorPoint];
        rightPixel.setColor(leftPixel.getColor());
        count++;
      }
    }
    System.out.println(count);
  }
```

2.

```
public void mirrorArms()
  {
    int mirrorPoint = 195;
    Pixel leftPixel = null;
    Pixel rightPixel = null;
    int count = 0;
    Pixel[][] pixels = this.getPixels2D();

    // loop through the rows
    for (int col = 95; col < 301; col++)
    {
      // loop from 13 to just before the mirror point
      for (int row = 155; row < mirrorPoint; row++)
      {

        leftPixel = pixels[row][col];
        rightPixel = pixels[mirrorPoint - row + mirrorPoint][col];
        rightPixel.setColor(leftPixel.getColor());
        count++;
      }
    }

  }

```

3.

```
public void mirrorGull()
  {
    int mirrorPoint = 357;
    Pixel leftPixel = null;
    Pixel rightPixel = null;
    int count = 0;
    Pixel[][] pixels = this.getPixels2D();

    // loop through the rows
    for (int row = 233; row < 319; row++)
    {
      // loop from 13 to just before the mirror point
      for (int col = 237; col < mirrorPoint; col++)
      {

        leftPixel = pixels[row][col];
        rightPixel = pixels[row]
                         [mirrorPoint - col];
        rightPixel.setColor(leftPixel.getColor());
        count++;
      }
    }

  }
```

## Pixlab 8:

1.

```
public void copy2(Picture fromPic,
                  int startRow, int startCol, int endRow, int endCol)
  {
    Pixel fromPixel = null;
    Pixel toPixel = null;
    Pixel[][] toPixels = this.getPixels2D();
    Pixel[][] fromPixels = fromPic.getPixels2D();
    for (int fromRow = endRow, toRow = startRow;
         fromRow < fromPixels.length &&
         toRow < toPixels.length;
         fromRow++, toRow++)
    {
      for (int fromCol = endCol, toCol = startCol;
           fromCol < fromPixels[0].length &&
           toCol < toPixels[0].length;
           fromCol++, toCol++)
      {
        fromPixel = fromPixels[fromRow][fromCol];
        toPixel = toPixels[toRow][toCol];
        toPixel.setColor(fromPixel.getColor());
      }
    }
  }
```

## Pixlab 9:

1.

```
public void edgeDetection(int edgeDist)
  {
    Pixel leftPixel = null;
    Pixel rightPixel = null;
    Pixel[][] pixels = this.getPixels2D();
    Color rightColor = null;
    for (int row = 0; row < pixels.length; row++)
    {
      for (int col = 0;
           col < pixels[0].length-1; col++)
      {
        leftPixel = pixels[row][col];
        rightPixel = pixels[row][col+1];
        rightColor = rightPixel.getColor();
        if (leftPixel.colorDistance(rightColor) >
            edgeDist)
          leftPixel.setColor(Color.BLACK);
        else
          leftPixel.setColor(Color.WHITE);
      }
    }
  }
```