**Package** [dipProject.src.main.java.src](package-summary.html)

**Class transformationOperations**

* java.lang.Object
  + dipProject.src.main.java.src.transformationOperations
* public class **transformationOperations**

extends java.lang.Object

**Author:**

Ellen, Marcus

* + ***Constructor Summary***

|  |  |
| --- | --- |
| **Constructors** | |
| **Constructor** | **Description** |
| [**transformationOperations**](#%3Cinit%3E())() |  |

* + ***Method Summary***

|  |  |  |
| --- | --- | --- |
| **All MethodsStatic MethodsConcrete Methods** | | |
| **Modifier and Type** | **Method** | **Description** |
| static void | [**imageAddition**](#imageAddition(java.awt.image.BufferedIm)​(java.awt.image.BufferedImage fileImage, java.awt.image.BufferedImage fileImage2) | Arithmetic Operations are passed two separate BufferedImages from two different Image Objects. |
| static void | [**imageDivision**](#imageDivision(java.awt.image.BufferedIm)​(java.awt.image.BufferedImage fileImage, java.awt.image.BufferedImage fileImage2) | Arithmetic Operations are passed two separate BufferedImages from two different Image Objects. |
| static void | [**imageMultiplication**](#imageMultiplication(java.awt.image.Buff)​(java.awt.image.BufferedImage fileImage, java.awt.image.BufferedImage fileImage2) | Arithmetic Operations are passed two separate BufferedImages from two different Image Objects. |
| static void | [**imageNegative**](#imageNegative(java.awt.image.BufferedIm)​(java.awt.image.BufferedImage fileImage) | Takes the image object's BufferedImage data member and applies a negative image transformation, only utilizing the length and width of the photo The image has its color values adjusted |
| static void | [**imageSubtraction**](#imageSubtraction(java.awt.image.Buffere)​(java.awt.image.BufferedImage fileImage, java.awt.image.BufferedImage fileImage2) | Arithmetic Operations are passed two separate BufferedImages from two different Image Objects. |
| static void | [**logTransformation**](#logTransformation(java.awt.image.Buffer)​(java.awt.image.BufferedImage fileImage) | The image object's BufferedImage data member is passed to the logTransfarmation and the log transformation is performed on it A log table is initialized and a c value is obtained from the user using JOptionPane |
| static void | [**powerLawTransform**](#powerLawTransform(java.awt.image.Buffer)​(java.awt.image.BufferedImage fileImage) | Applies a power law transformation to the provided Buffered Image Uses image dimensions, c-value, and gamma to produce a new edited image values |

* + - **Methods inherited from class java.lang.Object**

clone, equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

* + ***Constructor Detail***
    - **transformationOperations[Show source in BlueJ]**

public transformationOperations()

* + ***Method Detail***
    - **powerLawTransform[Show source in BlueJ]**

public static void powerLawTransform​(java.awt.image.BufferedImage fileImage) throws java.io.IOException

Applies a power law transformation to the provided Buffered Image Uses image dimensions, c-value, and gamma to produce a new edited image values

**Parameters:**

fileImage -

**Throws:**

java.io.IOException

* + - **imageNegative[Show source in BlueJ]**

public static void imageNegative​(java.awt.image.BufferedImage fileImage) throws java.io.IOException

Takes the image object's BufferedImage data member and applies a negative image transformation, only utilizing the length and width of the photo The image has its color values adjusted

**Parameters:**

fileImage -

**Throws:**

java.io.IOException

* + - **logTransformation[Show source in BlueJ]**

public static void logTransformation​(java.awt.image.BufferedImage fileImage) throws java.io.IOException

The image object's BufferedImage data member is passed to the logTransfarmation and the log transformation is performed on it A log table is initialized and a c value is obtained from the user using JOptionPane

**Parameters:**

fileImage -

**Throws:**

java.io.IOException

* + - **imageSubtraction[Show source in BlueJ]**

public static void imageSubtraction​(java.awt.image.BufferedImage fileImage, java.awt.image.BufferedImage fileImage2) throws java.io.IOException

Arithmetic Operations are passed two separate BufferedImages from two different Image Objects. The primary (first) image's dimensions are used and the operation is applied to that first image passed.

**Parameters:**

fileImage -

fileImage2 -

**Throws:**

java.io.IOException

* + - **imageAddition[Show source in BlueJ]**

public static void imageAddition​(java.awt.image.BufferedImage fileImage, java.awt.image.BufferedImage fileImage2) throws java.io.IOException

Arithmetic Operations are passed two separate BufferedImages from two different Image Objects. The primary (first) image's dimensions are used and the operation is applied to that first image passed.

**Parameters:**

fileImage -

fileImage2 -

**Throws:**

java.io.IOException

* + - **imageMultiplication[Show source in BlueJ]**

public static void imageMultiplication​(java.awt.image.BufferedImage fileImage, java.awt.image.BufferedImage fileImage2) throws java.io.IOException

Arithmetic Operations are passed two separate BufferedImages from two different Image Objects. The primary (first) image's dimensions are used and the operation is applied to that first image passed.

**Parameters:**

fileImage -

fileImage2 -

**Throws:**

java.io.IOException

* + - **imageDivision[Show source in BlueJ]**

public static void imageDivision​(java.awt.image.BufferedImage fileImage, java.awt.image.BufferedImage fileImage2) throws java.io.IOException

Arithmetic Operations are passed two separate BufferedImages from two different Image Objects. The primary (first) image's dimensions are used and the operation is applied to that first image passed.

**Parameters:**

fileImage -

fileImage2 -

**Throws:**

java.io.IOException