Exercice 1

```
/**
 * Exercice 1
 * @param a
 * @param b
 * @return
 * @throws Exception
 */
 public Image globalMultiplyAdd(double a, double b) throws Exception {
   Image ret = new Image(this);
   int tmp;
   for(int i=0; i<ret.pixels.length; i++) {
     tmp = (int) (a*ret.pixels[i]+b);
     if(tmp > 255) tmp=255;
     if(tmp < 0) tmp=0;
     ret.pixels[i]= tmp;
   }
   return ret;
 }
```

Exercice 2

```
 /**
  * Exercice 2
  * @param a
  * @param b
  * @param c
  * @param d
  * @return
  * @throws Exception
  */
 public Image spatialMultiplyAdd(Image a, Image b, double c) throws Exception {
   Image ret = new Image(this);
//    if(a == null) throw new Exception("Image a null");
//    if(b == null) throw new Exception("Image b null");
//    if(c<0) throw new Exception("valeur c < 0");

   int tmp;
   b = b.globalMultiplyAdd(c,0);
   for(int i=0; i<ret.pixels.length; i++) {
     tmp = (a.pixels[i]*ret.pixels[i]+b.pixels[i]);
     if(tmp > 255) tmp=255;
     if(tmp < 0) tmp=0;
     ret.pixels[i]= tmp;
   }
   return ret;
 }
```

Exercice 3

```
 /**
  * Exercice 3
  * @param alpha
  * @param image1
  * @param image2
  * @return
  * @throws Exception
  */
 public Image alphaBlending(double alpha, Image image1, Image image2) throws Ex-
ception {
   Image ret = new Image(this);
   if(image1 == null) throw new Exception("image1 null");
   if(image2 == null) throw new Exception("image2 null");
   int tmp;
   for(int i=0; i<ret.pixels.length; i++) {
     tmp = (int) (alpha*image1.pixels[i]+(1-alpha)*image2.pixels[i]);
```

```
      if(tmp > 255) tmp=255;
      if(tmp < 0) tmp=0;
      ret.pixels[i]= tmp;
    }
    return ret;
  }

  /**
   * Exercice 3
   * @param alpha
   * @param image1
   * @param image2
   * @return
   * @throws Exception
   */
  public Image spatialAlphaBlending(Image alpha, Image image1, Image image2) throws Exception {
    if(image1 == null) throw new Exception("image1 null");
    if(image2 == null) throw new Exception("image2 null");
    int tmp;
    int minWidth, minHeight;
    minWidth = image1.width;
    if(minWidth > image2.width)minWidth = image2.width;
    if(minWidth > alpha.width)minWidth = alpha.width;
    minHeight = image1.height;
    if(minHeight > image2.height)minHeight = image2.height;
    if(minHeight > alpha.height)minHeight = alpha.height;

    Image ret = new Image(minWidth, minHeight);
    for(int i=0; i<ret.pixels.length; i++) {
      tmp = (int) ((alpha.pixels[i]/255.0)*image1.pixels[i]+(1-(alpha.pixels[i]/255.0))*image2.pixels[i]);
      if(tmp > 255) tmp=255;
      if(tmp < 0) tmp=0;
      ret.pixels[i]= tmp;
    }
    return ret;
  }
```

Exercice 4

```
  /**
   * Exercice 4
   */
  public Image dynamicExpansion() {
    double val;
    int min = 255;
    int max = 0;
    for(int i=0; i<this.pixels.length; i++) {
      if(this.pixels[i] < min) min = this.pixels[i];
      if(this.pixels[i] > max) max = this.pixels[i];
    }
    int[] value = new int[256];
    for(int k=0; k<256; k++) {
      val = k-min;
      val = (val/(max-min))*255;
      value[k] = (int) val;
    }
    for(int j=0; j<this.pixels.length; j++) {
      this.pixels[j] = value[this.pixels[j]];
    }
    return this;
  }

  /**
```

```
 * Exercice 5
 * @return
 */
public int[] histogram() {
  int[] ret = new int[256];
  for(int i=0; i<this.pixels.length; i++) {
    int index = this.pixels[i];
    ret[index]++;
  }
  return ret;
}
```

5 Exercice 5

```
/**
 * Exercice 5
 * @return
 */
public double[] normalizedHistogram() {
  double[] ret = new double[256];
  int[] histo = this.histogram();
  double val=0.0;
  for (int i=0; i<256;i++){
    val = histo[i];
    ret[i] = val / this.pixels.length;
  }
  return ret;
}

/**
 * Exercice 5
 * @param image
 * @param histogramme
 * @return
 */
public Image dynamicExpansion(Image image, int[] histogramme) {
  double val;
  int min = 255;
  int max = 0;
  for(int i=0; i<256; i++) {
    if(i < min && histogramme[i] != 0) min = i;
    if(i > max && histogramme[i] != 0) max = i;
  }
  int[] value = new int[256];
  for(int k=0; k<256; k++) {
    val = k-min;
    val = (val/(max-min))*255;
    value[k] = (int) val;
  }
  for(int j=0; j<this.pixels.length; j++) {
    this.pixels[j] = value[this.pixels[j]];
  }
  return this;
}
```

Exercice 6

```
/**
 * Exercice 6
 * @param histogramme
 * @return
 */
public double[] cumule(double[] histogramme) {
  double[] ret = new double[256];
  double cumule =0.0;
  for(int i=0; i<256; i++) {
    cumule = cumule + histogramme[i];
    ret[i] = cumule;
  }
  return ret;
}

/**
 * Exercice 6
 * @param histogramme
 * @return
 */
public Image dynamicEgalization(double[] histogramme) {
  histogramme = cumule(histogramme);
  for(int i=0; i<histogramme.length; i++) {
    histogramme[i] = histogramme[i]*255.0;
  }
  for(int j=0; j<this.pixels.length; j++) {
    this.pixels[j] = (int) histogramme[this.pixels[j]];
  }
  return this;
}
```