Exercice 1

```
/**
 * TP3_1
 * @return
 */
public Image segmentation() {
   Image ret = new Image(this);
   ret = ret.otsu();
   ret = ret.etiquetage();
   return ret;
}
```

Exercice 2 & 3

```
/**
 * TP3_2
 * @return
 */
/* public Image segmentationLPE() {
   Image ret = new Image(this);
   ArrayList<Integer> fifo = new ArrayList<Integer>();
   int curLab = 0;
   int[] lab, dist;
   int hMin = this.minGrey();
   int hMax = this.maxGrey();
   lab = new int[this.pixels.length];
   dist = new int[this.pixels.length];
   for (int i = 0; i < this.pixels.length; i++) {
      lab[i] = -1;
      dist[i] = 0;
   }
   for (int h = hMin; h <= hMax; h++) {
      for (int i = 0; i < this.pixels.length; i++) {
         if (this.pixels[i] == h) {
            lab[i] = -2;

         }
      }
   }
   return ret;
}

public int maxGrey() {
   int ret = 0;
   for (int i = 0; i < this.pixels.length; i++) {
      if (ret < this.pixels[i]) {
         ret = this.pixels[i];
      }
   }
   return ret;
}

public int minGrey() {
   int ret = 255;
   for (int i = 0; i < this.pixels.length; i++) {
      if (ret > this.pixels[i]) {
         ret = this.pixels[i];
      }
   }
   return ret;
} */
```

Exercice 4

```
/**
 * TP3_4
```

```
     * @return
     */
    public Image transformeeDistance1() {
        Image ret = new Image(this.width, this.height);
        for (int x = 0; x < this.width; x++) {
            for (int y = 0; y < this.height; y++) {
                ret.setValue(x, y, this.calculDist(x, y, ret));
            }
        }
        return ret;
    }

    public int calculDist(int x, int y, Image img) {
        int ret = 0;
        return ret;
    }
```

TP 4 Exercice 1

```
    /**
     * TP4_1
     * @param masque
     * @return
     */
    public Image convolution(int[][] masque) {
        Image ret = new Image(this.width, this.height);
        int lengthMas = masque.length;
        System.out.println("length : " + lengthMas);
        int val = (lengthMas - 1) / 2;
        System.out.println("val : " + val);
        int parcoursX, parcoursY, sum;
        for (int x = 0; x < this.width; x++) {
            for (int y = 0; y < this.height; y++) {
                sum = 0;
                parcoursX = 0;
                for (int i = x - val; i < x + val; i++) {
                    if (i >= 0 && i < this.width && parcoursX <= lengthMas) {
                        parcoursY = 0;
                        for (int j = y - val; j < y + val; j++) {
                            if (j >= 0 && j < this.height && parcoursY <= lengthMas) {
                                sum += this.getValue(i, j) * masque[parcoursX][parcoursY];
                            }
                            parcoursY++;
                        }
                    }
                    parcoursX++;
                }
                ret.setValue(x, y, sum);
            }
        }
        return ret;
    }
```

5 TP 4 Exercice 2

```
    /**
     * TP4_2
     * @param masque1
     * @param masque2
     * @return
     */
    public Image convolution(int[] masque1, int[] masque2) {
        Image ret = this;
        if (masque1.length == masque2.length) {
            int lengthMas = masque1.length;
            System.out.println("length : " + lengthMas);
        int val = (lengthMas - 1) / 2;
        System.out.println("val : " + val);
        int parcoursX, parcoursY, sum;
        for (int x = 0; x < this.width; x++) {
            for (int y = 0; y < this.height; y++) {
                sum = 0;
                parcoursX = 0;
                for (int i = x - val; i < x + val; i++) {
                    if (i >= 0 && i < this.width && parcoursX <= lengthMas) {
                        parcoursY = 0;
                        for (int j = y - val; j < y + val; j++) {
                            if (j >= 0 && j < this.height && parcoursY <= lengthMas) {
                                sum += this.getValue(i, j) * masque1[parcoursX] * masque2[par-
coursY];
                            }
                            parcoursY++;
                        }
                    }
                    parcoursX++;
                }
                ret.setValue(x, y, sum);
            }
        }
    }
    return ret;
}

/**
 * TP4_2
 * @param masque
 * @return
 */
```

```
public Image convolution(int[] masque) {
    Image ret = new Image(this.width, this.height);
    int lengthMas = (int) Math.sqrt(masque.length);
    System.out.println("length : " + lengthMas);
    int val = (lengthMas - 1) / 2;
    System.out.println("val : " + val);
    int parcours, sum;
    for (int x = 0; x < this.width; x++) {
        for (int y = 0; y < this.height; y++) {
            sum = 0;
            parcours = 0;
            for (int i = x - val; i < x + val; i++) {
                if (i >= 0 && i < this.width && parcours <= masque.length) {
                    for (int j = y - val; j < y + val; j++) {
                        if (j >= 0 && j < this.height && parcours <= masque.length) {
                            sum += this.getValue(i, j) * masque[parcours];
                        }
                        parcours++;
                    }
                }
                parcours++;
            }
            ret.setValue(x, y, sum);
        }
    }
    return ret;
}
```