



Haute école d'ingénierie et d'architecture Fribourg  
Hochschule für Technik und Architektur Freiburg

---

# Algorithmique et structures de données

S13 Générateur de nombre pseudo aléatoire

---



*Auteurs :*  
*Marc Roten*

*Professeur :*  
*Rudolph SCHEURER*

16 mai 2018

## Table des matières

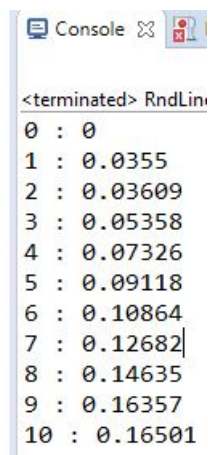
1	<code>rndTriangleAvgArea</code>	2
2	<code>RndFlipCoin</code>	3
3	<code>RndLinear</code>	4
4	<code>hasMajority</code>	5
5	<code>rndWalkMirrorAvgLength</code>	6
5.1	5a . . . . .	6
5.2	5b . . . . .	6
5.3	5c . . . . .	7

## Table des figures

1	Résultat à la console . . . . .	2
2	Résultat à la console . . . . .	3
3	Résultat à la console . . . . .	4
4	Résultat à la console . . . . .	5
5	Résultat à la console . . . . .	6
6	Résultat à la console . . . . .	6
7	Résultat à la console . . . . .	7

# 1 rndTriangleAvgArea

```
public class EX1_RndTriangle {
    public static void main(String [] args) {
        int nbOfExperiments = 100000;
        Random r = new Random();
        if (args.length>0) nbOfExperiments = Integer.parseInt(args[0]);
        System.out.println(rndTriangleAvgArea(r, nbOfExperiments));
    }
    //=====
    public static double rndTriangleAvgArea(Random r, int nbOfExperiments) {
        double x1, x2, x3,y1,y2,y3;
        double res = 0;
        for (int i = 0; i <nbOfExperiments; i++) {
            x1 = r.nextDouble(); x2= r.nextDouble(); x3= r.nextDouble();
            y1 = r.nextDouble(); y2= r.nextDouble(); y3= r.nextDouble();
            res+=0.5*Math.abs((x1*(y2-y3)) + (x2*(y3-y1)) + (x3*(y1-y2)));
        }
        return res/nbOfExperiments;
    }
}
```

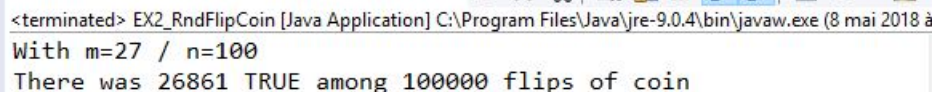


```
<terminated> RndLin
0 : 0
1 : 0.0355
2 : 0.03609
3 : 0.05358
4 : 0.07326
5 : 0.09118
6 : 0.10864
7 : 0.12682
8 : 0.14635
9 : 0.16357
10 : 0.16501
```

FIGURE 1 – Résultat à la console

## 2 RndFlipCoin

```
public class EX2_RndFlipCoin {
    public static void main(String [] args) {
        int nbOfExperiments = 100000;
        int n=100, m=27;
        Random r = new Random();
        testFlipCoin(r, m, n, nbOfExperiments);
    }
    // -----
    public static boolean flipCoin(Random r, int m, int n) {
        return (double)m/(double)n >=r.nextDouble();
    }
    //=====
    static void testFlipCoin(Random r, int m, int n, int nbOfTests) {
        System.out.printf("With m=%d / n=%d \n", m, n);
        int nbOfTrue = 0;
        for (int i = 0; i < nbOfTests; i++) {
            if (flipCoin(r, m, n))
                nbOfTrue++;
        }
        System.out.println("There was " + nbOfTrue + " TRUE among " + nbOfTests +
                           " flips of coin");
    }
}
```



```
<terminated> EX2_RndFlipCoin [Java Application] C:\Program Files\Java\jre-9.0.4\bin\javaw.exe (8 mai 2018 à ...)
With m=27 / n=100
There was 26861 TRUE among 100000 flips of coin
```

FIGURE 2 – Résultat à la console

### 3 RndLinear

```
public class EX3_RndLinear {
    public static void main(String [] args) {
        int nbOfExperiments = 100000;
        int n=10;
        Random r = new Random();
        testLinear(r, n, nbOfExperiments);
    }
    //=====
    public static int rndLinear(Random r, int n) {
        int sommeIndex = (n*(n+1))/2;
        int i = r.nextInt(sommeIndex);
        int res = 1;
        while(((res*(res+1))/2<i)){
            res++;
        }
        return res;
    }
    //=====
    static void testLinear(Random r, int n, int nbOfExperiments) {
        int[] t = new int[n + 1];
        for (int i = 0; i < nbOfExperiments; i++)
            t[rndLinear(r, n)]++;
        System.out.println(0 + " : " + t[0]);
        for (int i = 1; i < n + 1; i++)
            System.out.println(i + " : " + (double) t[i] / nbOfExperiments);
    }
}
```

```
<terminated> EX3_Rndl
0 : 0
1 : 0.03706
2 : 0.03691
3 : 0.05529
4 : 0.07233
5 : 0.09131
6 : 0.10898
7 : 0.12733
8 : 0.14547
9 : 0.16315
10 : 0.16217
```

FIGURE 3 – Résultat à la console

## 4 hasMajority

```
public static boolean hasMajority(Random r, int [] t, double risk) {
    /*
     * pour qu'un element soit majoritaire dans un tableau de par exemple de 10
     * lments
     * il y aura forcement une occurrence
     * 0 1 2 3 4 5 6 7 8 9
     * 2 0 2 1 2 9 2 8 2 2
     * X X X X X X
     * notre 2 apparait 6 fois, c'est donc lui qui nous intresse.
     * s'il n'y a pas d'occurrence, pas de majorit.
     * et ce n'est pas parcequ'il y a une occurrence que c'est forcement le
     * nombre dit qui
     * est majoritaire. il faudra donc vrifier cette condition
     */
    int nbrMajoritaire =0,tempo=0;
    for(int i=1;i<t.length;i++) {
        if(t[i]==t[i-1]) {
            nbrMajoritaire=t[i];
            break;
        }
    }
    for(int i=0;i<t.length;i++) {
        if(t[i]==nbrMajoritaire) {
            tempo++;
        }
    }
    if(tempo>(t.length/2)) {
        return true;
    }else return false;
}
```

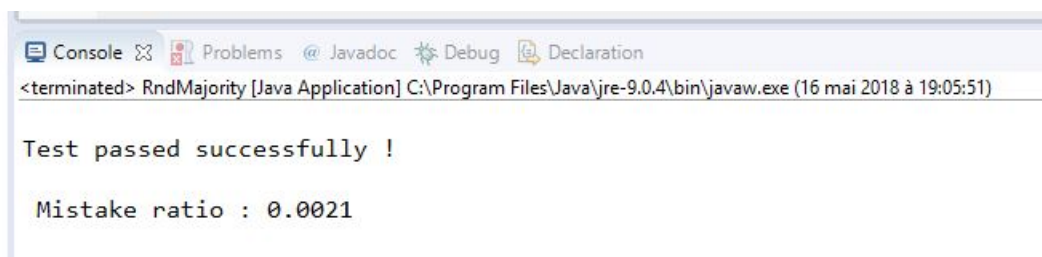


FIGURE 4 – Résultat à la console

## 5 rndWalkMirrorAvgLength

```
static double rndWalkMirrorAvgLength
(Random r, int pointToReach, int leftChoicePercentage, int nbOfExperiments) {
    int x, nbOfSteps = 0;
    int total=0;
    for (int i=0; i<nbOfExperiments; i++) {
        x=0; nbOfSteps=0;
        while(x!=pointToReach) {
            double d = r.nextDouble()*100;
            if (d<leftChoicePercentage) {
                if (x != 0) {
                    x-- ;
                }
            } else {
                x++;
            }
            nbOfSteps++;
        }
        total+= nbOfSteps;
    }
    return total/((double)nbOfExperiments;
}
```

### 5.1 5a



FIGURE 5 – Résultat à la console

### 5.2 5b

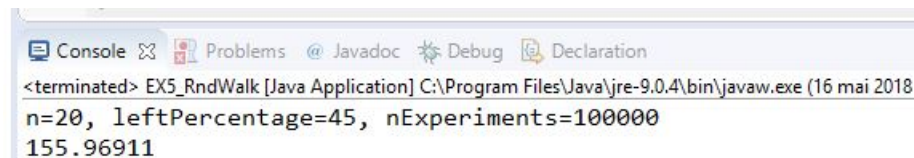
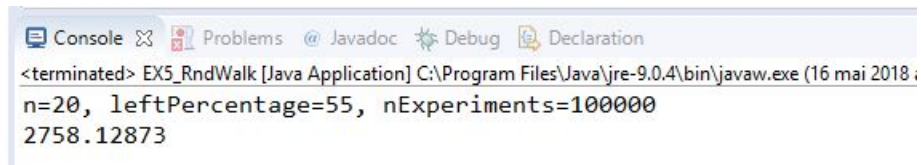


FIGURE 6 – Résultat à la console

## 5.3 5c



```
<terminated> EX5_RndWalk [Java Application] C:\Program Files\Java\jre-9.0.4\bin\javaw.exe (16 mai 2018 ;  
n=20, leftPercentage=55, nExperiments=100000  
2758.12873
```

FIGURE 7 – Résultat à la console