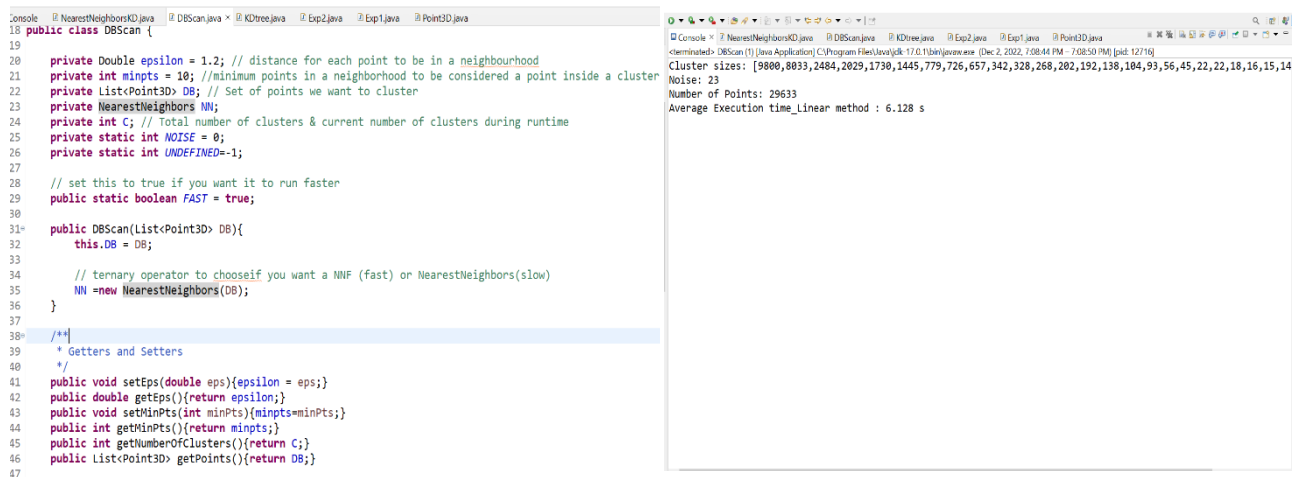


Experience 3

1) Pour le fichier Point_Cloud_1.csv

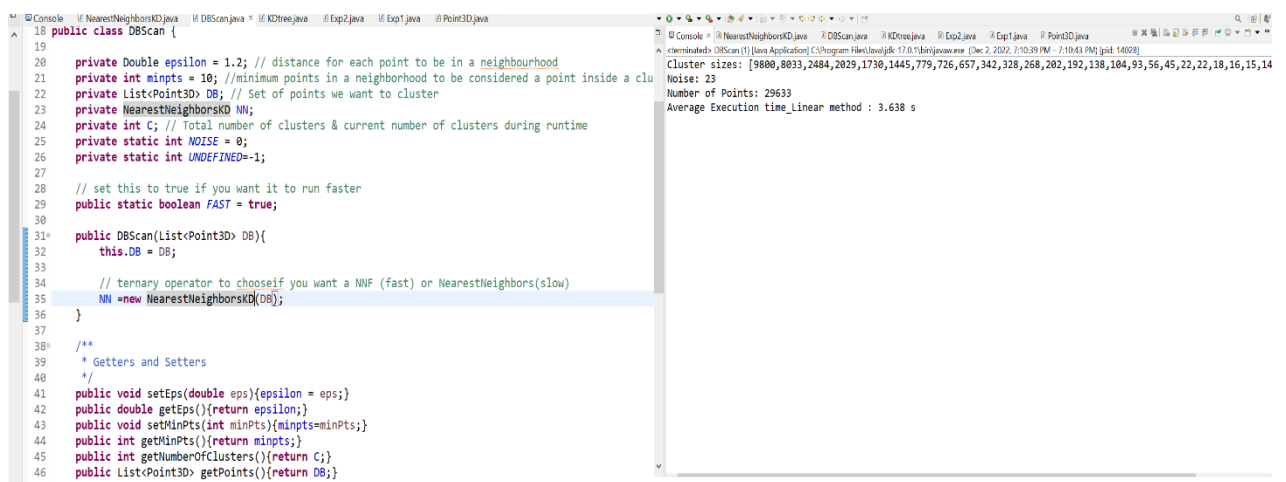
- 1- Le temps d'exécution pour la méthode linéaire de NearestNeighbors est de : 6.128 secondes ou 6128 ms.



```
18 public class DBScan {
19
20     private Double epsilon = 1.2; // distance for each point to be in a neighbourhood
21     private int minpts = 10; // minimum points in a neighborhood to be considered a point inside a cluster
22     private List<Point3D> DB; // Set of points we want to cluster
23     private NearestNeighbors NN;
24     private int C; // Total number of clusters & current number of clusters during runtime
25     private static int NOISE = 0;
26     private static int UNDEFINED=-1;
27
28     // set this to true if you want it to run faster
29     public static boolean FAST = true;
30
31     public DBScan(List<Point3D> DB){
32         this.DB = DB;
33
34         // ternary operator to choose if you want a NNF (fast) or NearestNeighbors(slow)
35         NN = new NearestNeighbors(DB);
36     }
37
38     /**
39      * Getters and Setters
40      */
41     public void setEps(double eps){epsilon = eps;}
42     public double getEps(){return epsilon;}
43     public void setMinPts(int minPts){minpts=minPts;}
44     public int getMinPts(){return minpts;}
45     public int getNumberOfClusters(){return C;}
46     public List<Point3D> getPoints(){return DB;}
47 }
```

```
<terminated> DBScan (1) [Java Application] C:\Program Files\Java\jdk-17.0.1\bin\javaw.exe (Dec 2, 2022, 7:08:44 PM - 7:08:50 PM) [pid: 12716]
Cluster sizes: [9800,8033,2484,2029,1730,1445,779,726,657,342,328,268,202,192,138,104,93,56,45,22,22,18,16,15,14
Noise: 23
Number of Points: 29633
Average Execution time_Linear method : 6.128 s
```

- 2- Le temps d'exécution pour la méthode KDTree de NearestNeighbors est de : 3.638 secondes ou 3638 ms



```
18 public class DBScan {
19
20     private Double epsilon = 1.2; // distance for each point to be in a neighbourhood
21     private int minpts = 10; // minimum points in a neighborhood to be considered a point inside a cluster
22     private List<Point3D> DB; // Set of points we want to cluster
23     private NearestNeighborsKD NN;
24     private int C; // Total number of clusters & current number of clusters during runtime
25     private static int NOISE = 0;
26     private static int UNDEFINED=-1;
27
28     // set this to true if you want it to run faster
29     public static boolean FAST = true;
30
31     public DBScan(List<Point3D> DB){
32         this.DB = DB;
33
34         // ternary operator to choose if you want a NNF (fast) or NearestNeighbors(slow)
35         NN = new NearestNeighborsKD(DB);
36     }
37
38     /**
39      * Getters and Setters
40      */
41     public void setEps(double eps){epsilon = eps;}
42     public double getEps(){return epsilon;}
43     public void setMinPts(int minPts){minpts=minPts;}
44     public int getMinPts(){return minpts;}
45     public int getNumberOfClusters(){return C;}
46     public List<Point3D> getPoints(){return DB;}
47 }
```

```
<terminated> DBScan (1) [Java Application] C:\Program Files\Java\jdk-17.0.1\bin\javaw.exe (Dec 2, 2022, 7:10:39 PM - 7:10:43 PM) [pid: 14008]
Cluster sizes: [9800,8033,2484,2029,1730,1445,779,726,657,342,328,268,202,192,138,104,93,56,45,22,22,18,16,15,14
Noise: 23
Number of Points: 29633
Average Execution time_Linear method : 3.638 s
```

2) Pour le fichier Point_Cloud_2.csv

- 1- Le temps d'exécution pour la méthode linéaire de NearestNeighbors est de : 15.221 secondes ou 15221 ms.

```
17
18 public class DBSCAN {
19
20     private Double epsilon = 1.2; // distance for each point to be in a neighbourhood
21     private int minpts = 10; // minimum points in a neighborhood to be considered a point inside a cluster
22     private List<Point3D> DB; // Set of points we want to cluster
23     private NearestNeighbors NN;
24     private int C; // Total number of clusters & current number of clusters during runtime
25     private static int NOISE = 0;
26     private static int UNDEFINED=-1;
27
28     // set this to true if you want it to run faster
29     public static boolean FAST = true;
30
31     public DBSCAN(List<Point3D> DB){
32         this.DB = DB;
33
34         // ternary operator to choose if you want a NNF (fast) or NearestNeighbors(slow)
35         NN = new NearestNeighbors(FAST);
36     }
37
38     /**
39      * Getters and Setters
40      */
41     public void setEps(double eps){epsilon = eps;}
42     public double getEps(){return epsilon;}
43     public void setMinPts(int minPts){minpts=minPts;}
44     public int getMinPts(){return minpts;}
45     public int getNumberOfClusters(){return C;}
46     public List<Point3D> getPoints(){return DB;}
47 }
```

```
terminated: DBSCAN (1) [Java Application] C:\Program Files\Java\jre-17.0.11\bin\java.exe (Dec 2, 2022, 7:19:51 PM - 7:20:05 PM) [pid: 13260]
Cluster sizes: [28960,13695,2949,1691,860,698,672,355,199,125,82,77,57,55,53,30,27,25,24,20,19,18,17,17,16]
Noise: 12
Number of Points: 50821
Average Execution time_Linear method : 15.221 s
```

- 2- Le temps d'exécution pour la méthode KDTree de NearestNeighbors est de : 9.172 secondes ou 9172 ms.

```
17
18 public class DBSCAN {
19
20     private Double epsilon = 1.2; // distance for each point to be in a neighbourhood
21     private int minpts = 10; // minimum points in a neighborhood to be considered a point inside a cluster
22     private List<Point3D> DB; // Set of points we want to cluster
23     private NearestNeighborsKD NN;
24     private int C; // Total number of clusters & current number of clusters during runtime
25     private static int NOISE = 0;
26     private static int UNDEFINED=-1;
27
28     // set this to true if you want it to run faster
29     public static boolean FAST = true;
30
31     public DBSCAN(List<Point3D> DB){
32         this.DB = DB;
33
34         // ternary operator to choose if you want a NNF (fast) or NearestNeighbors(slow)
35         NN = new NearestNeighborsKD(FAST);
36     }
37
38     /**
39      * Getters and Setters
40      */
41     public void setEps(double eps){epsilon = eps;}
42     public double getEps(){return epsilon;}
43     public void setMinPts(int minPts){minpts=minPts;}
44     public int getMinPts(){return minpts;}
45     public int getNumberOfClusters(){return C;}
46     public List<Point3D> getPoints(){return DB;}
47 }
```

```
terminated: DBSCAN (1) [Java Application] C:\Program Files\Java\jre-17.0.11\bin\java.exe (Dec 2, 2022, 7:23:00 PM - 7:33:00 PM) [pid: 12150]
Cluster sizes: [28960,13695,2949,1691,860,698,672,355,199,125,82,77,57,55,53,30,27,25,24,20,19,18,17,17,16]
Noise: 12
Number of Points: 50821
Average Execution time_Linear method : 9.172 s
```

3) Pour le fichier Point_Cloud_3.csv

- 1- Le temps d'exécution pour la méthode linéaire de NearestNeighbors est de : 11.507 secondes ou 11507 ms.

```
17
18 public class DBSCAN {
19
20     private Double epsilon = 1.2; // distance for each point to be in a neighbourhood
21     private int minpts = 10; // minimum points in a neighborhood to be considered a point inside a cluster
22     private List<Point3D> DB; // Set of points we want to cluster
23     private NearestNeighbors NN;
24     private int C; // Total number of clusters & current number of clusters during runtime
25     private static int NOISE = 0;
26     private static int UNDEFINED=-1;
27
28     // set this to true if you want it to run faster
29     public static boolean FAST = true;
30
31     public DBSCAN(List<Point3D> DB){
32         this.DB = DB;
33
34         // ternary operator to choose if you want a NNF (fast) or NearestNeighbors(slow)
35         NN = new NearestNeighbors(FAST);
36     }
37
38     /**
39      * Getters and Setters
40      */
41     public void setEps(double eps){epsilon = eps;}
42     public double getEps(){return epsilon;}
43     public void setMinPts(int minPts){minpts=minPts;}
44     public int getMinPts(){return minpts;}
45     public int getNumberOfClusters(){return C;}
46     public List<Point3D> getPoints(){return DB;}
47 }
```

```
terminated: DBSCAN (1) [Java Application] C:\Program Files\Java\jre-17.0.11\bin\java.exe (Dec 2, 2022, 7:27:02 PM - 7:27:15 PM) [pid: 12648]
Cluster sizes: [13481,10525,8086,6354,3170,862,363,312,297,280,239,209,137,93,92,75,65,62,61,39,39,32,31,28,28,2]
Noise: 7
Number of Points: 45141
Average Execution time_Linear method : 11.507 s
```

2- Le temps d'exécution pour la méthode KDTree de NearestNeighbors est de :
5.712 secondes ou 5712ms.

```
17
18 public class DBScan {
19
20     private Double epsilon = 1.2; // distance for each point to be in a neighbourhood
21     private int minpts = 10; //minimum points in a neighborhood to be considered a point inside a cluster
22     private List<Point3D> DB; // Set of points we want to cluster
23     private NearestNeighborsKD NN;
24     private int C; // Total number of clusters & current number of clusters during runtime
25     private static int NOISE = 0;
26     private static int UNDEFINED=-1;
27
28     // set this to true if you want it to run faster
29     public static boolean FAST = true;
30
31     public DBScan(List<Point3D> DB){
32         this.DB = DB;
33
34         // ternary operator to choose if you want a MNF (fast) or NearestNeighbors(slow)
35         NN = new NearestNeighborsKD(DB);
36     }
37
38     /**
39      * Getters and Setters
40      */
41     public void setEps(double eps){epsilon = eps;}
42     public double getEps(){return epsilon;}
43     public void setMinPts(int minPts){minpts=minPts;}
44     public int getMinPts(){return minpts;}
45     public int getNumberOfClusters(){return C;}
46     public List<Point3D> getPoints(){return DB;}
```

```
Console x NearestNeighborsKD.java DBScan.java KDTree.java Exp2.java Exp1.java Point3D.java
<terminated> DBScan (1) [Java Application] C:\Program Files\Java\jdk-17.0.1\bin\java.exe (Dec 2, 2022, 7:28:09 PM - 7:28:15 PM) [pid: 14112]
Cluster sizes: [13481,18525,8886,6354,3170,862,363,312,297,280,239,209,137,93,92,75,65,62,61,39,39,32,31,28,28,2
Noise: 7
Number of Points: 45141
Average Execution time_linear method : 5.712 s
```