```java
import java.util.Random;
//sÃ©quencement
public class EtatSeq {

    public int dimEtat;
    public int[] idAvions;
    private static Random generateur = new Random(123);
    private int oldIndexI;
    private int oldIndexJ;
    /*****************************************************************************************************/
    /* Methodes locales */
    /*****************************************************************************************************/
    public void exchange(int I, int J) {
        int buffer;
        buffer = idAvions[I];
        idAvions[I] = idAvions[J];
        idAvions[J] = buffer;
    }
    /*****************************************************************************************************/
    /* Constructeur */
    /*****************************************************************************************************/
    public EtatSeq(int dimEtat) {
        this.dimEtat = dimEtat;
        idAvions = new int[dimEtat];
    }
    /*****************************************************************************************************/
    /* Initialisation aleatoire de l'etat */
    /*****************************************************************************************************/
    public void initAleatEtat() {
        for (int i = 0; i < dimEtat; ++i) {
            idAvions[i] = i;
        }
    }
    /*****************************************************************************************************/
    /* Affichage sequence avions
     * */
    /*****************************************************************************************************/
    public void afficherEtat() {
        System.out.println("rang d'atterrissage ; id avion");
        for (int i = 0; i < idAvions.length; i++) {
            System.out.println(i+1 + " ; "+idAvions[i]);
        }
    }
    /*****************************************************************************************************/
    /* GenererVoisin */
    /*****************************************************************************************************/
//  public void genererVoisin2() {
//      int indexI = generateur.nextInt(dimEtat); // on prends deux index au hasard entre 0 et 99
//      int indexJ = generateur.nextInt(dimEtat);
//        // on permute les avions i.e leur ordre d'atterrissage compris entre ces deux index de maniere symetrique
```

```
51 //      for (int i = 0; i < Math.ceil(Math.abs(indexI - indexJ) / 2); ++i) {
52 //          exchange(Math.min(indexI, indexJ) + i, Math.max(indexI, indexJ) - i);
53 //      }
54 //      oldIndexI = indexI;
55 //      oldIndexJ = indexJ;
56 //
57 //      for (int i = 0; i < dimEtat - 1; ++i) {
58 //          if (idAvions[i] <= i - 4 || idAvions[i] >= i + 4) {
59 //              comeBack();
60 //              break;
61 //          }
62 //      }
63 //  }
64
65      public void genererVoisin() {
66          int indexI = generateur.nextInt(dimEtat); // on prends un index au hasard entre 0
    et 99 (dimEtat=100=DIMENSION)
67          int indexJ = indexI - 4 + generateur.nextInt(9); // on gÃ©nÃ¨re un indexJ dans
    indexI +/- 4 pour tenir compte de la
68          //contrainte de vitesse ; un avion ne peut changer de rang au dÃ©lÃ  de +/- 4/
69          if (indexJ<0) indexJ = 0;
70          if(indexJ>=dimEtat) indexJ=dimEtat-1;
71          // on permute les avions i.e leur ordre d'atterrissage compris entre ces deux index
    de maniere symetrique
72          for (int i = 0; i < Math.ceil(Math.abs(indexI - indexJ) / 2); ++i) {
73              exchange(Math.min(indexI, indexJ) + i, Math.max(indexI, indexJ) - i);
74          }
75          oldIndexI = indexI;
76          oldIndexJ = indexJ;
77      }
78
    /*************************************************************************************
    *************************/
79      /* Retour a la solution prÃ©cÃ©dente */
80      /*************************************************************************************
    ****************************/
81      public void comeBack() {
82          for (int i = 0; i < Math.ceil(Math.abs(oldIndexI - oldIndexJ) / 2); ++i) {
83              exchange(Math.min(oldIndexI, oldIndexJ) + i, Math.max(oldIndexI, oldIndexJ) -
    i);
84          }
85      }
86      /*************************************************************************************
    ****************************/
87      /* Evaluation des objectifs */
88      /*************************************************************************************
    ****************************/
89      public double calculCritere() {
90          double cost = 0;
91          // tabClasseAvions contient les classes des 100 avions (0 : Heavy, 1 : Medium, 2 :
    Light),
92          // l'index de tabAvions correspond Ã  l'identifiant avion de 0 Ã  99
93          //idAvions contient les avions identifiÃ©s par un chiffre de 0 Ã  99
94          // les index du tableau idAvions donne l'ordre d'atterrissage : idAvion[0] donne
    l'id de l'avion qui atterrit en premier.
95          for (int i = 0; i < dimEtat - 1; ++i) {
96              cost += Data.separation[Data.tabClasseAvions[idAvions[i]]]
    [Data.tabClasseAvions[idAvions[i + 1]]];
97          }
98          return cost;
99      }
100 }
```

101