```java

public class RecuitKP {
    // recuit pour le problÃ¨me du sac Ã  dos
    /*******************************************************/
      /* Parametres du recuit */
    private static final int nbTransitions = 2000;
    private static final double alpha = 0.995;
    private static final boolean minimisation = false;

      /*******************************************************/
      /* dimension du probleme */
    private static final int DIMENSION = 100;

     // principe d'acceptation
        private boolean accept(double yi, double yj, double T, boolean min) {
            boolean isAccepted = false;
            double dE = yj-yi;
            double proba = Math.exp(-Math.abs(dE)/T);
            double tirage = Math.random();

            if(min) {
                if(dE<0) isAccepted = true;
                else if (tirage <= proba) isAccepted=true;//acceptation frequente si T grand car alors proba proche de 1
            } else {// opt en maximisation
                if(dE>0) isAccepted = true;
                else if (tirage <= proba) isAccepted=true;
            }
            return isAccepted;
        }
     // ************************************
      // Chauffage
      // ************************************
      public double chauffage() {
          int nbAccept = 0;
          double yi; // critere courant
          double yj; //critere voisin
          double T=0.01;
          double tauxAccept=0.0;
          EtatKP xi = new EtatKP(DIMENSION);

          do {
              nbAccept=0;
              for (int i = 0; i < nbTransitions; i++) {
                  // generation d'un point de l'espace d'etat
                  xi.initAleatEtat();
                  yi = xi.calculCritere();
                  // generation d'un voisin
                  xi.genererVoisin();
                  yj = xi.calculCritere();

                  if(accept(yi,yj,T,minimisation)) nbAccept++;

                  tauxAccept= (double)nbAccept / (double)nbTransitions;
                 }
              T=1.1*T;
            System.out.println("T = "+ T + " Taux acceptation "+ tauxAccept);

          }while (tauxAccept <0.8);
          return T;
      }//fin chauffage

```

```java
    // ****************************************
      // Refroidissement
      // ****************************************
      public EtatKP refroidissement(double Tinit) {
        double yi = 0.0, yj = 0.0; // criteres courant yi et critere voisin mis e 0
        double T = Tinit;
        EtatKP xi = new EtatKP(DIMENSION);

        xi.initAleatEtat();
        yi = xi.calculCritere();

        do {

        for(int i =0; i< nbTransitions;i++) {
            xi.genererVoisin();
            yj = xi.calculCritere();
            if(accept(yi, yj, T, minimisation)) {
                yi = yj;
            }else {
                xi.comeBack();
            }
        }
        T = T * alpha;
        System.out.println("T = "+ T + " valeur critere " + yi);
      //  System.out.println(xi.afficherEtat());
        xi.afficherEtat();

        } while (T > 0.0001 * Tinit || xi.getP()>2000 || xi.getP()<1990);
        xi.afficherEtat();
//      for(int i = 0; i<DIMENSION;i++)
//          System.out.println(i + " : "+xi.X[i]+ " ");
        return xi;
      }//fin refroidt

      // ****************************************
      // MAIN
      // ****************************************
      public static void main(String args[]) {
          double temperature;
          RecuitKP monRecuit = new RecuitKP();
          // generation des donnees
            System.out.println("**********************Generation des donnees
***************");
            Data.genererObjets(DIMENSION);

            System.out.println("**************************Chauffage ***************");
            temperature = monRecuit.chauffage();  // on recupere la temperature apres
chauffage
            // i.e. T lorsque le taux d'acceptation est de 0.8

            System.out.println("======================Refroidissement ============");
            monRecuit.refroidissement(temperature);
      }//fin main
}//fin class Recuit2

```

```java
import java.util.Random;
// recuit pour le problÃ¨me du sac Ã  dos
public class EtatKP {

    public int dimEtat;
    public int[] X;

    private static Random generateur = new Random(999);
    private int oldIndexI;//, oldIndexJ;
    //private double oldPoids;
    private double capacite = 2000;
    private double P, V;


/*************************************************************************************************/
      /* Methodes locales */

/*************************************************************************************************/
    //echange deux valeurs du vecteur d'etat X
    public void swap(int a, int b) {
          int tmp;
          tmp = X[a];
          X[a]=X[b];
          X[b]=tmp;
      }

    public double getP()
    { return P;}
    /*************************************************************************************************/
      /* Constructeur */

/*************************************************************************************************/
      public EtatKP(int dimEtat) {
          this.dimEtat = dimEtat;
          X = new int[dimEtat];
      }

/*************************************************************************************************/
      /* Initialisation aleatoire de l'etat */

/*************************************************************************************************/
      public void initAleatEtat() {
          for(int i= 0 ; i<dimEtat;i++)
              X[i]=generateur.nextInt(2);
      }

/*************************************************************************************************/
      /* Evaluation des objectifs */

/*************************************************************************************************/
      public double calculCritere() {
          double valeur = 0;
          double poids = 0;

```

```java
          for (int i=0; i<dimEtat; ++i){
                valeur += X[i] * Data.tabValeurs[i];
                poids  += X[i] * Data.tabPoids[i];
            }
          V = valeur;
          P = poids;
          double delta = poids - capacite;
          double epsilon = delta / capacite;
            if (delta <= 0){
                return valeur;
            }
            else{
                //return valeur/(1+epsilon*epsilon);
                return valeur -10*delta;
                //return 0;
            }
      }

  /*************************************************************************************************************/
      /* GenererVoisin */

  /*************************************************************************************************************/
      public void genererVoisin() {
          int indexI = generateur.nextInt(dimEtat); // on choisit un indice au hasard entre 0 et dimEtat-1
          X[indexI]=(X[indexI] + 1) % 2;// si X(i)= 0, 0+1=1%2 donne 1, si X[i]= 1, 1+1=2%2 donne 0
          oldIndexI=indexI; //on memorise l'index pour pouvoir faire un retour arriere le cas echeant
      }

  /*************************************************************************************************************/
      /* Retour a la solution precedente */

  /*************************************************************************************************************/
      public void comeBack() {
          X[oldIndexI] = (X[oldIndexI] + 1) % 2;
      }

  /*************************************************************************************************************/
      /* Affichage */

  /*************************************************************************************************************/
      public void afficherEtat() {
          System.out.println("poids sac = " + P +" valeur sac =  " + V);
      }
}

```

```
1 import java.util.*;
2
3 public class Data {
4
5     public static double tabPoids[];
6     public static double tabValeurs[];
7     public static double[][] tabVilles;
8     public static int tabAvions[];
9     private static Random generateur=new Random(123);
10
11
12    /*************************************************************/
13    /*   parametres de generation                                */
14    private static double RAYON=100.0;
15    private static int POIDS=100;
16    private static int VALEURS=100;
17    /*************************************************************/
18
19
20    public static int separation[][]={{4,5,6},
21                                      {3,3,4},
22                                      {3,3,3}};
23
24
25    // generation de n objets pour le probleme du sac a dos
26    public static void genererObjets(int n) {
27        tabPoids=new double[n];
28        tabValeurs=new double[n];
29        for (int i=0;i<n;i++){
30            tabPoids[i]=(double)(1+generateur.nextInt(POIDS));
31            tabValeurs[i]=(double)(1+generateur.nextInt(VALEURS));
32        }
33    }
34
35    // affichage des objets
36    public static void afficherObjets() {
37    System.out.println("*****************************
  Objets*****************************");
38        for (int i=0;i<tabPoids.length;i++){
39            System.out.println("Objet " + i + " poids= "+ tabPoids[i] + " valeur=" +
  tabValeurs[i]);
40        }
41    }
42
43
44    // generation de n villes sur le cercle (TSP)
45    public static void genererVillesCercle(int n) {
46    double theta,x,y;
47    tabVilles=new double[n][2];
48        for (int i=0;i<n;i++){
49            theta=2*Math.PI*generateur.nextDouble();
50            x=RAYON*Math.cos(theta);
51            y=RAYON*Math.sin(theta);
52
53            tabVilles[i][0]=x;
54            tabVilles[i][1]=y;
55        }
56    }
57
58
59    // generation de n villes dans un carre RAYONxRAYON (TSP)
60    public static void genererVilles(int n) {
```

```
61      double x,y;
62
63      tabVilles = new double[n][2];
64          for (int i=0;i<n;i++){
65              x=RAYON*generateur.nextDouble();
66              y=RAYON*generateur.nextDouble();
67              tabVilles[i][0]=x;
68              tabVilles[i][1]=y;
69          }
70      }
71
72      // affichage des villes
73      public static void afficherVilles() {
74          System.out.println("***************************** Villes
    ******************************");
75          for (int i=0;i<tabVilles.length;i++){
76              System.out.println("Ville " + i + " x= "+ tabVilles[i][0]+ " y= " +
    tabVilles[i][1]);
77          }
78      }
79
80      // generation de n avions (classe heavy(0), Medium(1), Small(2))
81      public static void genererAvions(int n) {
82      tabAvions=new int[n];
83      for (int i=0;i<n;i++){
84              tabAvions[i]=generateur.nextInt(3);
85          }
86      }
87
88      // affichage sequence avions
89      public static void afficherAvions() {
90      System.out.println("***************************** Avions
    ******************************");
91      for (int i=0;i<tabAvions.length;i++){
92          System.out.println("Avion " + i + " classe= "+ tabAvions[i]);
93      }
94      }
95
96
97      public static void main(String[] args){
98          int dim=100;
99          genererObjets(dim);
100         afficherObjets();
101         genererVillesCercle(dim);
102         afficherVilles();
103         genererVilles(dim);
104         afficherVilles();
105         genererAvions(dim);
106         afficherAvions();
107     }
108 }
```

```java
// sequencement
public class RecuitSeq {
    /*******************************************************/
    /* Parametres du recuit */
    private static final int nbTransitions = 2000;
    private static final double alpha = 0.995;
    private static final boolean minimisation = true;
    /*******************************************************/
    /* dimension du probleme */
    private static final int DIMENSION = 100;

     // principe d'acceptation
    private boolean accept(double yi, double yj, double T, boolean min) {
        boolean isAccepted = false;
        double dE = yj-yi;
        double proba = Math.exp(-Math.abs(dE)/T);
        double tirage = Math.random();

        if(min) {
            if(dE<0) isAccepted = true;
            else if (tirage <= proba) isAccepted=true;//acceptation frequente si T grand car alors proba proche de 1
        } else {// opt en maximisation
            if(dE>0) isAccepted = true;
            else if (tirage <= proba) isAccepted=true;
        }
        return isAccepted;
    }
    // ************************************
    // Heat Up
    // ************************************
    public double heatUpLoop() { // HeatUp heat = new HeatUp();
        int acceptCount = 0;
        double yi = 0, yj;
        double T = 0.01, tauxAccept = 0.0;
        EtatSeq xi = new EtatSeq(DIMENSION);

        do {
            acceptCount = 0;
            for (int i = 0; i < nbTransitions; i++) {

                // generation d'un point de l'espace d'etat
                xi.initAleatEtat();
                yi = xi.calculCritere();

                // generation d'un voisin
                xi.genererVoisin();
                yj = xi.calculCritere();

                if (accept(yi, yj, T, minimisation))
                    acceptCount++;
                tauxAccept = (double) acceptCount / (double) nbTransitions;
            }
            T = T * 1.1;
            System.out.println("T= " + T + " tauxAccept= " + tauxAccept + " currentCost= " + yi);
        } while (tauxAccept < 0.8);
        return T;
    }
    // ****************************************
    // COOLING
    // ****************************************
```

```
61      public EtatSeq coolingLoop(double Tinit) { // HeatUp heat = new HeatUp();
62          double yi = 0.0, yj = 0.0;
63          double T = Tinit;
64          EtatSeq xi = new EtatSeq(DIMENSION);
65
66          xi.initAleatEtat();
67          yi = xi.calculCritere();
68          do {
69              for (int i = 0; i < nbTransitions; i++) {
70                  xi.genererVoisin();
71                  yj = xi.calculCritere();
72                  if (accept(yi, yj, T, minimisation)) {
73                      yi = yj;
74                  } else {
75                      xi.comeBack();
76                  }
77              }
78              T = T * alpha;
79              System.out.println("T= " + T + " valeur critere " + yi);
80          } while (T > 0.0001 * Tinit);
81          xi.afficherEtat();
82          //Data.afficherAvions();
83          return xi;
84      }
85      // ****************************************
86      // MAIN
87      // ****************************************
88      public static void main(String args[]) {
89          double temperature;
90          RecuitSeq monRecuit = new RecuitSeq();
91          // generation des donnees
92          System.out.println("**********************Generation des donnee
    ***************");
93          Data.genererAvions(DIMENSION);
94
95          System.out.println("**************************Chauffage ***************");
96          temperature = monRecuit.heatUpLoop();
97          System.out.println("======================Refroidissement ============");
98          monRecuit.coolingLoop(temperature);
99      }// end main
100 }// End class HeatUp2
101
```

```java
import java.util.Random;

public class Data {

    public static int tabClasseAvions[];
    private static Random generateur = new Random(123);

    public static int separation[][] = { { 4, 5, 6 }, { 3, 3, 4 }, { 3, 3, 3 } };

    // generation de n avions (classe heavy(0), Medium(1), Small(2))
    public static void genererAvions(int n) {
        tabClasseAvions = new int[n];
        for (int i = 0; i < n; i++) {
            tabClasseAvions[i] = generateur.nextInt(3);
        }
    }

    // affichage des classes des avions
    public static void afficherClassesAvions() {
        System.out.println("***************************** Avions *****************************");
        for (int i = 0; i < tabClasseAvions.length; i++) {
            System.out.println("Avion " + i + " classe= " + tabClasseAvions[i]);
        }
    }
}
```

```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

```java
// TSP
public class RecuitTSP {
        /*******************************************************/
        /*  Parametres du recuit                               */
        private static final int nbTransitions=2000;
        private static final double alpha=0.995;
        private static final boolean minimisation=true;
        /*******************************************************/
        /* dimension du probleme                               */
        private static final int DIMENSION=100;

        // principe d'acceptation
        private boolean accept(double yi, double yj, double T, boolean min) {
            boolean isAccepted = false;
            double dE = yj-yi;
            double proba = Math.exp(-Math.abs(dE)/T);
            double tirage = Math.random();

            if(min) {
                if(dE<0) isAccepted = true;
                else if (tirage <= proba) isAccepted=true;//acceptation frequente si T grand car alors proba proche de 1
            } else {// opt en maximisation
                if(dE>0) isAccepted = true;
                else if (tirage <= proba) isAccepted=true;
            }
            return isAccepted;
        }
    //*************************************
    //        Heat Up
    //*************************************
    public double heatUpLoop()
    {   //HeatUp heat = new HeatUp();
        int acceptCount = 0;
        double yi=0,yj;
        double T=0.01, tauxAccept=0.0;
        EtatTSP xi=new EtatTSP(DIMENSION);
        do {
            acceptCount=0;
            for (int i=0; i< nbTransitions;i++)
            {
                //generation d'un point de l'espace d'etat
                xi.initAleatEtat();
                yi=xi.calculCritere();

                //generation d'un voisin
                xi.genererVoisin();
                yj=xi.calculCritere();

                if (accept(yi,yj,T,minimisation)) acceptCount++;
                tauxAccept=(double)acceptCount/(double)nbTransitions;
            }
            T=T*1.1;
            System.out.println("T= " + T + " tauxAccept= " + tauxAccept + " currentCost= " + yi);
        } while(tauxAccept<0.8);
    return T;
    }
    //*************************************
    //              COOLING
    //*************************************
    public EtatTSP coolingLoop(double Tinit)
```

```
61      {   //HeatUp heat = new HeatUp();
62          double yi=0.0,yj=0.0;
63          double T=Tinit;
64          EtatTSP xi=new EtatTSP(DIMENSION);
65
66          xi.initAleatEtat();
67          yi=xi.calculCritere();
68          do {
69              for (int i=0; i< nbTransitions;i++)
70              {
71                  xi.genererVoisin();
72                  yj=xi.calculCritere();
73                  if (accept(yi,yj,T,minimisation)){
74                      yi=yj;
75                  }
76                  else
77                  {
78                      xi.comeBack();
79                  }
80              }
81              T=T*alpha;
82              System.out.println("T= " + T + " valeur critere " + yi);
83              System.out.println(xi.afficherEtat());
84          } while(T>0.0001*Tinit);
85          return xi;
86      }
87      //*****************************************
88      //           MAIN
89      //*****************************************
90      public static void main( String args[] )
91      {
92          double temperature;
93          RecuitTSP monRecuit = new RecuitTSP();
94          //generation des donnees
95          System.out.println("**********************Generation des donnee
    ***************");
96          Data.genererVillesCercle(DIMENSION);
97          //Data.genererVilles(DIMENSION);
98
99          System.out.println("*************************Chauffage **************");
100         temperature = monRecuit.heatUpLoop();
101         System.out.println("======================Refroidissement ============");
102         monRecuit.coolingLoop(temperature);
103     }//end main
104 }//End class HeatUp2
105
```

```java
import java.util.*;
// TSP
public class EtatTSP {
    public int dimEtat;
    public int[] vecteur;

    private static Random generateur = new Random(123);
    private int oldIndexI;
    private int oldIndexJ;
    private static final int methode =2;

/*************************************************************************************************************/
    /*                                  Methodes locales
*/

/*************************************************************************************************************/
    public void exchange(int I, int J){
        int buffer;
        buffer = vecteur[I];
        vecteur[I] = vecteur[J];
        vecteur[J] = buffer;
    }

/*************************************************************************************************************/
    /*                                  Constructeur
*/

/*************************************************************************************************************/
    public EtatTSP(int dimEtat)
    {
        this.dimEtat=dimEtat;
        vecteur = new int[dimEtat];
    }

/*************************************************************************************************************/
    /*                              Initialisation aleatoire de l'etat
*/

/*************************************************************************************************************/
    public void initAleatEtat()
    {
        for (int i=0; i<dimEtat; ++i){
            vecteur[i] = i;
        }
    }

/*************************************************************************************************************/
    /*                                  Affichage
*/

/*************************************************************************************************************/
    public String afficherEtat()
    {
        return "";
```

```
43      }
44
/*************************************************************************************************************/
45      /*                                  GenererVoisin                                                    */
/*************************************************************************************************************/
46
47      public void genererVoisin()
48      {
49          int indexI = generateur.nextInt(dimEtat);
50          int indexJ = generateur.nextInt(dimEtat);
51
52          if(methode ==1)
53              exchange(indexI, indexJ); //***1er methode pour changer la position aleatoirement
54          if (methode ==2){
55              for (int i=0; i<Math.ceil(Math.abs(indexI-indexJ)/2); ++i){
56                  exchange(Math.min(indexI, indexJ)+i, Math.max(indexI, indexJ)-i);
57                  //System.out.println(i+ " " + (Math.min(indexI, indexJ)+i) + " " + (Math.max(indexI, indexJ)-i) );
58              }
59          }
60          oldIndexI = indexI;
61          oldIndexJ = indexJ;
62      }
63
/*************************************************************************************************************/
64      /*                                  Retour a la solution prÃ©cÃ©dente                                  */
/*************************************************************************************************************/
65
66      public void comeBack()
67      {
68          if(methode ==1) exchange(oldIndexI, oldIndexJ); //**1er methode pour changer la position aleatoirement
69          if (methode ==2){
70              for (int i=0; i<Math.ceil(Math.abs(oldIndexI-oldIndexJ)/2); ++i){
71                  exchange(Math.min(oldIndexI, oldIndexJ)+i, Math.max(oldIndexI, oldIndexJ)-i);
72              }
73          }
74      }
75
/*************************************************************************************************************/
76      /*                                  Evaluation des objectifs                                         */
/*************************************************************************************************************/
77
/*************************************************************************************************************/
78      public double calculCritere(){
79          double cost=0;
80          double dx,dy;
81          for (int i=0; i<dimEtat-1; ++i){
82              dx = Data.tabVilles[vecteur[i+1]][0]-Data.tabVilles[vecteur[i]][0];
83              dy = Data.tabVilles[vecteur[i+1]][1]-Data.tabVilles[vecteur[i]][1];
84              cost += Math.sqrt(dx*dx+dy*dy);
85          }
```

```
86          dx = Data.tabVilles[vecteur[dimEtat-1]][0]-Data.tabVilles[vecteur[0]][0];
87          dy = Data.tabVilles[vecteur[dimEtat-1]][1]-Data.tabVilles[vecteur[0]][1];
88          cost += Math.sqrt(dx*dx+dy*dy);
89          return cost;
90      }
91  }//fin EtatTSP
92
93
94
```

```
1 import java.util.*;
2
3 public class Data {
4
5     public static double tabPoids[];
6     public static double tabValeurs[];
7     public static double[][] tabVilles;
8     public static int tabAvions[];
9     private static Random generateur=new Random(123);
10
11
12    /*************************************************************/
13    /*   parametres de generation                                */
14    private static double RAYON=100.0;
15    private static int POIDS=100;
16    private static int VALEURS=100;
17    /*************************************************************/
18
19
20    public static int separation[][]={{4,5,6},
21                                      {3,3,4},
22                                      {3,3,3}};
23
24
25    // generation de n objets pour le probleme du sac a dos
26    public static void genererObjets(int n) {
27        tabPoids=new double[n];
28        tabValeurs=new double[n];
29        for (int i=0;i<n;i++){
30            tabPoids[i]=(double)(1+generateur.nextInt(POIDS));
31            tabValeurs[i]=(double)(1+generateur.nextInt(VALEURS));
32        }
33    }
34
35    // affichage des objets
36    public static void afficherObjets() {
37    System.out.println("*****************************
  Objets*****************************");
38        for (int i=0;i<tabPoids.length;i++){
39            System.out.println("Objet " + i + " poids= "+ tabPoids[i] + " valeur=" +
  tabValeurs[i]);
40        }
41    }
42
43
44    // generation de n villes sur le cercle (TSP)
45    public static void genererVillesCercle(int n) {
46    double theta,x,y;
47    tabVilles=new double[n][2];
48        for (int i=0;i<n;i++){
49            theta=2*Math.PI*generateur.nextDouble();
50            x=RAYON*Math.cos(theta);
51            y=RAYON*Math.sin(theta);
52
53            tabVilles[i][0]=x;
54            tabVilles[i][1]=y;
55        }
56    }
57
58
59    // generation de n villes dans un carre RAYONxRAYON (TSP)
60    public static void genererVilles(int n) {
```

```
61      double x,y;
62
63      tabVilles = new double[n][2];
64          for (int i=0;i<n;i++){
65              x=RAYON*generateur.nextDouble();
66              y=RAYON*generateur.nextDouble();
67              tabVilles[i][0]=x;
68              tabVilles[i][1]=y;
69          }
70      }
71
72      // affichage des villes
73      public static void afficherVilles() {
74          System.out.println("***************************** Villes
    ******************************");
75          for (int i=0;i<tabVilles.length;i++){
76              System.out.println("Ville " + i + " x= "+ tabVilles[i][0]+ " y= " +
    tabVilles[i][1]);
77          }
78      }
79
80      // generation de n avions (classe heavy(0), Medium(1), Small(2))
81      public static void genererAvions(int n) {
82      tabAvions=new int[n];
83      for (int i=0;i<n;i++){
84              tabAvions[i]=generateur.nextInt(3);
85          }
86      }
87
88      // affichage sequence avions
89      public static void afficherAvions() {
90      System.out.println("***************************** Avions
    ******************************");
91      for (int i=0;i<tabAvions.length;i++){
92          System.out.println("Avion " + i + " classe= "+ tabAvions[i]);
93      }
94      }
95
96
97      public static void main(String[] args){
98          int dim=100;
99          genererObjets(dim);
100         afficherObjets();
101         genererVillesCercle(dim);
102         afficherVilles();
103         genererVilles(dim);
104         afficherVilles();
105         genererAvions(dim);
106         afficherAvions();
107     }
108 }
```