

```

1 package backtracking;
2
3 import java.util.Random;
4
5 public class Solucio {
6     // atributs donats de la classe Solucio
7     private Article[] articles; // articles
8     private Motxilla[] motxilles; // motxilles a emplenar
9
10    // Atenció: a la classe motxilla s'han afegit els atributs: pesActual,
    volumActual i quantsEssencials per controlar la solució actual
11
12    // atributs de la solució actual
13    private int[] solucioActual; // a l'índex article assigno l'índex motxilla o -1
    (es queda a fora)
14    private int quantsArticlesActual; // Núm. articles assignats a motxilles
15    private int sumaUtilitatsActual; // utilitat de la solució actual
16
17    // atributs de la millor solució trobada
18    private int[] millorSolucio;
19    private int quantsArticlesMillor;
20    private int sumaUtilitatsMillor;
21
22    private static Article[] obtenirDadesArticles(){
23        //crea i retorna els articles que es volen portar
24        Random rand = new Random();
25        Article[] articles = new Article[rand.nextInt(10,15)];
26        for(int i=0; i<articles.length; i++) {
27            int pes = rand.nextInt(1, 10);
28            int volum = rand.nextInt(1, 10);
29            int utilitat = rand.nextInt(1, 6);
30            boolean esEssencial = rand.nextBoolean();
31            articles[i] = new Article(pes, volum, utilitat, i, esEssencial);
32        }
33        return articles;
34    }
35    private static Motxilla[] obtenirDadesMotxilles(){
36        // crea i retorna les motxilles que s'han d'emplenar
37        Random rand = new Random();
38        Motxilla[] motxilles = new Motxilla[rand.nextInt(2,4)];
39        for(int i=0; i<motxilles.length; i++) {
40            int pesMaxim = rand.nextInt(15, 25);
41            int volumMaxim = rand.nextInt(15, 25);
42            motxilles[i] = new Motxilla(pesMaxim, volumMaxim);
43        }
44        return motxilles;
45    }
46    /* Exercici 2 - Part 2 (0.5 punts) Escriu el mètode constructor, has de carregar
    les dades dels
47    magatzems amb els articles i motxilles disponibles. Crear i inicialitzar els
    atributs de la classe segons
48    el que m'has explicat a l'apartat c anterior. El constructor no tindrà cap
    paràmetre.*/
49    public Solucio() {
50        articles = obtenirDadesArticles();
51        motxilles = obtenirDadesMotxilles();
52
53        solucioActual = new int[articles.length];
54        for (int i = 0; i < solucioActual.length; i++)
55            solucioActual[i] = -1; //inicialment tots els articles fora
56        quantsArticlesActual = 0;
57        sumaUtilitatsActual = 0;
58        millorSolucio = new int[articles.length];
59        quantsArticlesMillor = Integer.MIN_VALUE; // volem maximitzar
60        sumaUtilitatsMillor = Integer.MIN_VALUE; // en cas d'empat, maximitzar
        utilitat
61    }
62
63    /* Exercici 2 - Part 3 (0.25 punts) Escriu el mètode main que ubicaràs també a la
    classe Solucio.
64    Aquest invocarà al backtracking i visualitzarà la solució trobada.*/
65    public static void main(String[] args) {
66        Solucio s = new Solucio();

```

```

67     s.backMillor(0); // crida al mètode de backtracking
68     System.out.println(s); // mostra la solució trobada
69 }
70 /* Exercici 2 - Part 4 (6.25 punts) Implementa el mètode que aplica la tècnica
del backtracking
71 i que ubicaràs a la classe Solucio public void backMillor(???). Aquest mètode
no pot ser estàtic. Has
72 de determinar el(s) paràmetre(s) necessaris, minimitzant-los. Es valorarà la
descomposició funcional
73 aplicada, fes mètodes privats per les diferents parts de l'esquema:
esAcceptable(???), esMillor(???),
74 ... Es valorarà l'eficiència. Quan sigui possible cal podar l'arbre
de cara a millorar l'eficiència de la
75 implementació.*/
76 public void backMillor(int idxArticle) {
77     // IMPORTANT: amplada de l'arbre = num. motxilles + 1 (deixar fora)
78     for (int idxMotxilla = 0; idxMotxilla <= motxilles.length; idxMotxilla++) {
79         if (acceptable(idxArticle, idxMotxilla)) {
80             anotar(idxArticle, idxMotxilla);
81             if (esSolucio(idxArticle)) {
82                 if (esMillor()) {
83                     guardarMillorSolucio();
84                 }
85             } else {
86                 int quantsRestants = articles.length - (idxArticle + 1);
87                 if( quantsRestants > 0 && // no és fulla
88                     (quantsArticlesActual + quantsRestants) >=
89                     quantsArticlesMillor ) // poda
90                     backMillor(idxArticle + 1); // crida recursiva
91             }
92             desanotar(idxArticle, idxMotxilla); // desfem l'assignació
93         }
94     }
95 private boolean acceptable(int idxArticle, int idxMotxilla) {
96     if (idxMotxilla == motxilles.length)
97         return true; // deixar fora sempre és acceptable
98     Motxilla m = motxilles[idxMotxilla];
99     Article a = articles[idxArticle];
100    return m.hiCap(a.getPes(), a.getVolum());
101 }
102 private void anotar(int idxArticle, int idxMotxilla) {
103     if( idxMotxilla == motxilles.length )
104         solucioActual[idxArticle] = -1; // deixar fora
105     else {
106         Motxilla m = motxilles[idxMotxilla];
107         Article a = articles[idxArticle];
108         m.afegirArticle(a.getPes(), a.getVolum(), a.isEssencial());
109         quantsArticlesActual++;
110         sumaUtilitatsActual += a.getUtilitat();
111         solucioActual[idxArticle] = idxMotxilla;
112     }
113 }
114 private void desanotar(int idxArticle, int idxMotxilla) {
115     if( idxMotxilla < motxilles.length ){
116         Motxilla m = motxilles[idxMotxilla];
117         Article a = articles[idxArticle];
118         m.treureArticle(a.getPes(), a.getVolum(), a.isEssencial());
119         quantsArticlesActual--;
120         sumaUtilitatsActual -= a.getUtilitat();
121     }
122     solucioActual[idxArticle] = -1;
123 }
124 private boolean esSolucio(int idxArticle) {
125     if (idxArticle < articles.length-1) // cal trobar una fulla de l'arbre
126         return false;
127     for( Motxilla m : motxilles ){ // Cal mirar totes les motxilles tenen un
essencial
128         if( !m.conteEssencials() )
129             return false;
130     }
131     return true;
132 }

```

```

133 private boolean esMillor() {
134     //comprova si la solució actual és millor que la millor trobada fins ara
135     if (quantsArticlesActual > quantsArticlesMillor)
136         return true;
137     if (quantsArticlesActual == quantsArticlesMillor) // empatat en num.
138         articles, mirem utilitat
139         return (sumaUtilitatsActual > sumaUtilitatsMillor);
140     return false;
141 }
142 private void guardarMillorSolucio() {
143     // guarda la solució actual com la millor trobada fins ara
144     quantsArticlesMillor = quantsArticlesActual;
145     sumaUtilitatsMillor = sumaUtilitatsActual;
146     for (int i = 0; i < solucioActual.length; i++)
147         millorSolucio[i] = solucioActual[i];
148 }
149
150 /* Exercici 2 - Part 5 (0.5 punts) Redefineix el mètode toString() a la classe
151 Solucio per mostrar
152 la solució trobada. Ha de retornar una cadena que mostri l'identificador dels
153 articles carregats en cada
154 motxilla i per a cadascuna de les motxilla. Les motxilles les pots enumerar amb
155 1, 2, 3...*/
156 public String toString(){
157     // visualitza la solució trobada, si hi ha
158     if( quantsArticlesMillor<=0)
159         return "No hi ha solució vàlida\n";
160
161     StringBuilder sb = new StringBuilder();
162     sb.append("Millor solució trobada porta:"+quantsArticlesMillor+"\n");
163     for(int m=0; m<motxilles.length; m++){
164         sb.append("Motxilla " + (m+1) + ": ");
165         for(int a=0; a<articles.length; a++){
166             if(millorSolucio[a]==m){
167                 sb.append(articles[a].toString());
168             }
169         }
170         sb.append("\n");
171     }
172     sb.append("Total articles carregats: " + quantsArticlesMillor + "\n");
173     sb.append("Utilitat total: " + sumaUtilitatsMillor + "\n");
174     return sb.toString();
175 }
176 } //fi classe Solució
177

```