

```

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,
11    volumActual i quantsEssencials per controlar la solució actual
12
13    // atributs de la solució actual
14    private int[] solucioActual; // a l'índex article assigno l'índex motxilla o -1
15    (es queda a fora)
16    private int quantsArticlesActual; // Núm. articles assignats a motxilles
17    private int sumaUtilitatsActual; // utilitat de la solució actual
18
19    // atributs de la millor solució trobada
20    private int[] millorSolucio;
21    private int quantsArticlesMillor;
22    private int sumaUtilitatsMillor;
23
24    private static Article[] obtenirDadesArticles(){
25        //crea i retorna els articles que es volen portar
26        Random rand = new Random();
27        Article[] articles = new Article[rand.nextInt(10,15)];
28        for(int i=0; i<articles.length; i++) {
29            int pes = rand.nextInt(1, 10);
30            int volum = rand.nextInt(1, 10);
31            int utilitat = rand.nextInt(1, 6);
32            boolean esEssencial = rand.nextBoolean();
33            articles[i] = new Article(pes, volum, utilitat, i, esEssencial);
34        }
35        return articles;
36    }
37    private static Motxilla[] obtenirDadesMotxilles(){
38        // crea i retorna les motxilles que s'han d'emplenar
39        Random rand = new Random();
40        Motxilla[] motxilles = new Motxilla[rand.nextInt(2,4)];
41        for(int i=0; i<motxilles.length; i++) {
42            int pesMaxim = rand.nextInt(15, 25);
43            int volumMaxim = rand.nextInt(15, 25);
44            motxilles[i] = new Motxilla(pesMaxim, volumMaxim);
45        }
46        return motxilles;
47    }
48    /* Exercici 2 - Part 2 (0.5 punts) Escriu el mètode constructor, has de carregar
49    les dades dels
50    magatzems amb els articles i motxilles disponibles. Crear i inicialitzar els
51    atributs de la classe segons
52    el que m'has explicat a l'apartat c anterior. El constructor no tindrà cap
53    paràmetre.*/
54    public Solucio() {
55        articles = obtenirDadesArticles();
56        motxilles = obtenirDadesMotxilles();
57
58        solucioActual = new int[articles.length];
59        for (int i = 0; i < solucioActual.length; i++)
60            solucioActual[i] = -1; //inicialment tots els articles fora
61        quantsArticlesActual = 0;
62        sumaUtilitatsActual = 0;
63        millorSolucio = new int[articles.length];
64        quantsArticlesMillor = Integer.MIN_VALUE; // volem maximitzar
65        sumaUtilitatsMillor = Integer.MIN_VALUE; // en cas d'empat, maximitzar
66        utilitat
67    }
68    /* Exercici 2 - Part 3 (0.25 punts) Escriu el mètode main que ubicaràs també a la
69    classe Solucio.
70        Aquest invocarà al backtracking i visualitzarà la solució trobada.*/
71    public static void main(String[] args) {
72        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
71 del backtracking
72     i que ubicaràs a la classe Solucio public void backMillor(????). Aquest mètode
73 no pot ser estàtic. Has
74 de determinar el(s) paràmetre(s) necessaris, minimitzant-los. Es valorarà la
75 descomposició funcional
76 aplicada, fes mètodes privats per les diferents parts de l'esquema:
77 esAcceptable(????), esMillor(????),
78         ... Es valorarà l'eficiència. Quan sigui possible cal podar l'arbre
79         de cara a millorar l'eficiència de la
80         implementació.*/
81 public void backMillor(int idxArticle) {
82     // IMPORTANT: amplada de l'arbre = num. motxilles + 1 (deixar fora)
83     for (int idxMotxilla = 0; idxMotxilla <= motxilles.length; idxMotxilla++) {
84         if (acceptable(idxArticle, idxMotxilla)) {
85             anotar(idxArticle, idxMotxilla);
86             if (esSolucio(idxArticle)) {
87                 if (esMillor()) {
88                     guardarMillorSolucio();
89                 }
90             } else{
91                 int quantsRestants = articles.length - (idxArticle + 1);
92                 if( quantsRestants > 0 && // no és fulla
93                     (quantsArticlesActual + quantsRestants) >=
94                     quantsArticlesMillor ) // poda
95                     backMillor(idxArticle + 1); // crida recursiva
96                 }
97                 desanotar(idxArticle, idxMotxilla); // desfem l'assignació
98             }
99         }
100    }
101 }
102 private boolean acceptable(int idxArticle, int idxMotxilla) {
103     if (idxMotxilla == motxilles.length)
104         return true; // deixar fora sempre és acceptable
105     Motxilla m = motxilles[idxMotxilla];
106     Article a = articles[idxArticle];
107     return m.hiCap(a.getPes(), a.getVolum());
108 }
109 private void anotar(int idxArticle, int idxMotxilla) {
110     if( idxMotxilla == motxilles.length )
111         solucioActual[idxArticle] = -1; // deixar fora
112     else {
113         Motxilla m = motxilles[idxMotxilla];
114         Article a = articles[idxArticle];
115         m.afegirArticle(a.getPes(), a.getVolum(), a.isEssencial());
116         quantsArticlesActual++;
117         sumaUtilitatsActual += a.getUtilitat();
118         solucioActual[idxArticle] = idxMotxilla;
119     }
120 }
121 private void desanotar(int idxArticle, int idxMotxilla) {
122     if( idxMotxilla < motxilles.length ){
123         Motxilla m = motxilles[idxMotxilla];
124         Article a = articles[idxArticle];
125         m.treureArticle(a.getPes(), a.getVolum(), a.isEssencial());
126         quantsArticlesActual--;
127         sumaUtilitatsActual -= a.getUtilitat();
128     }
129     solucioActual[idxArticle] = -1;
130 }
131 private boolean esSolucio(int idxArticle) {
132     if (idxArticle < articles.length-1) // cal trobar una fulla de l'arbre
133         return false;
134     for( Motxilla m : motxilles ){ // Cal mirar totes les motxilles tenen un
135         essencial
136         if( !m.conteEssentials() )
137             return false;
138     }
139     return true;
140 }
```

```

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     /* Exercici 2 - Part 5 (0.5 punts) Redefineix el mètode toString() a la classe
150     Solucio per mostrar
151     la solució trobada. Ha de retornar una cadena que mostri l'identificador dels
152     articles carregats en cada
153     motxilla i per a cadascuna de les motxilla. Les motxilles les pots enumerar amb
154     1, 2, 3...
155     */
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

```