

TASK 1 (AlberiBinari)

```
1. class NodoBin {
2.     public String info;
3.     public NodoBin sinistro;
4.     public NodoBin destro;
5. }
6.
7. public class AlberiBinari {
8.
9.     //Visite dell'albero
10.
11.     //Visita in Preordine
12.     public static void visitaPreordine(NodoBin a) {
13.         if (a == null) //albero vuoto
14.             return;
15.         else {
16.             System.out.print(a.info + " "); //opera su nodo corrente
17.             visitaPreordine(a.sinistro); //visita sottoalb. sin.
18.             visitaPreordine(a.destro); //visita sottoalb. des.
19.         }
20.     }
21.
22.
23.     //Visita in Postordine
24.     public static void visitaPostordine(NodoBin a) {
25.         if (a == null) //albero vuoto
26.             return;
27.         else {
28.             visitaPostordine(a.sinistro);
29.             visitaPostordine(a.destro);
30.             System.out.print(a.info + " ");
31.         }
32.     }
33.
34.
35.     //Visita in Simmetrica
36.     public static void visitaSimmetrica(NodoBin a) {
37.         if (a == null) //albero vuoto
38.             return;
39.         else {
40.             visitaSimmetrica(a.sinistro);
41.             System.out.print(a.info + " ");
42.             visitaSimmetrica(a.destro);
43.         }
44.     }
45.
46.
47.     //Esempi di uso delle visite per realizzare funzionalita'
48.     // calcolo della profondita' di un albero (con prof. foglia == 0)
49.     public static int calcolaProfondita(NodoBin a) {
50.         if (a == null) //albero vuoto
51.             return -1;
52.         else {
53.             int profsx = calcolaProfondita(a.sinistro);
54.             int profdx = calcolaProfondita(a.destro);
55.             if (profsx > profdx) return profsx + 1;
56.             else return profdx + 1;
57.         }
58.     }
59.
60.     // verifica della presenza di un dato elemento nell'albero
61.     public static boolean presente(NodoBin a, String x) {
62.         if (a == null) // albero vuoto
63.             return false;
64.         else if (a.info.equals(x)) return true;
65.         else return presente(a.sinistro,x) ||
66.             presente(a.destro,x);
67.     }
68.
69.     // conta i nodi dell'albero
70.     public static int contaNodi(NodoBin a) {
71.         if (a == null) //albero vuoto
72.             return 0;
73.         else {
74.             int nodisx = contaNodi(a.sinistro);
75.             int nodidx = contaNodi(a.destro);
76.             return 1 + nodisx + nodidx;
77.         }
78.     }
79.
80.
81.     // conta le foglie dell'albero
```

```

82. public static int contaFoglie(NodoBin a) {
83.     if (a == null) //albero vuoto
84.         return 0;
85.     else if (a.sinistro == null && a.destro == null)
86.         return 1;
87.     else {
88.         int fogliesx = contaFoglie(a.sinistro);
89.         int fogliedx = contaFoglie(a.destro);
90.         return fogliesx + fogliedx;
91.     }
92. }
93.
94. // restituire una lista contenente le foglie dell'albero
95. public static NodoLista listaFoglie(NodoBin a) {
96.     if (a == null)
97.         return null;
98.     else if(a.sinistro == null && a.destro == null) {
99.         NodoLista ris = new NodoLista();
100.        ris.info = a.info;
101.        ris.next = null;
102.        return ris;
103.    }
104.    else {
105.        NodoLista listasx = listaFoglie(a.sinistro); //visita sottoalb. sin.
106.        NodoLista listadx = listaFoglie(a.destro); //visita sottoalb. des.
107.        return append(listasx, listadx); //opera su nodo corrente
108.    }
109. }
110.
111. // restituire un array contenente le foglie dell'albero
112. public static String[] arrayFoglie(NodoBin a) {
113.     Env e = new Env();
114.     e.array = new String[contaFoglie(a)];
115.     e.i = 0;
116.     aggiornaEnv(a,e);
117.     return e.array;
118. }
119.
120. private static void aggiornaEnv(NodoBin a, Env e) {
121.     if (a == null)
122.         return;
123.     else if(a.sinistro == null && a.destro == null) {
124.         e.array[e.i] = a.info;
125.         (e.i)++;
126.     }
127.     else {
128.         aggiornaEnv(a.sinistro,e); //visita sottoalb. sin.
129.         aggiornaEnv(a.destro,e); //visita sottoalb. des.
130.         //sul nodo corrente non si deve fare altro
131.     }
132. }
133.
134. public static void stampa(NodoBin a) {
135.     if (a == null) System.out.print("()");
136.     else {
137.         System.out.print("(");
138.         System.out.print(a.info + " ");
139.         stampa(a.sinistro);
140.         System.out.print(" ");
141.         stampa(a.destro);
142.         System.out.print(")");
143.     }
144. }
145.
146. // metodi ausiliari non su alberi
147. // append gia' realizzato precedentemente
148. private static NodoLista append(NodoLista p1, NodoLista p2) {
149.     if (p1 == null)
150.         return p2;
151.     else {
152.         NodoLista ris = append(p1.next,p2);
153.         p1.next = ris;
154.         return p1;
155.     }
156. }
157.
158. class Env {
159.     public String[] array; //array di stringhe
160.     public int i; //posizione corrente
161. }
162.
163. class NodoLista {
164.     public String info;
165.     public NodoLista next;

```

