```java
/*

PUC Minas - Ciencia da Computacao     Nome: ArvoreBinariaDeContatos

Autor: Axell Brendow Batista Moreira  Matricula: 631822

Versao:  1.0                          Data: 09/10/2018

*/

class Node
{
    Object element;
    Node left;
    Node right;
    Lista contactsList;

    public Node(){}

    public Node(Object element)
    {
        this.element = element;
    }

    public Node(Node left, Node right)
    {
        this.left = left;
        this.right = right;
    }

    public Node(Object element, Node left, Node right)
    {
        this(left, right);

        this.element = element;
    }

    public Node(Object element, Node left, Node right, Lista contactsList)
    {
        this(element, left, right);

        this.contactsList = contactsList;
    }

    public Object getElement()
    {
        return element;
    }

    public void setElement(Object element)
    {
        this.element = element;
    }

    public Node getLeft()
    {
        return left;
    }

    public void setLeft(Node left)
    {
        this.left = left;
    }

    public Node getRight()
    {
        return right;
    }

    public void setRight(Node right)
    {
        this.right = right;
    }
```

```
74
75      public Lista getContactsList()
76      {
77          return contactsList;
78      }
79
80      public void setContactsList(Lista contactsList)
81      {
82          this.contactsList = contactsList;
83      }
84  }
85
86  class BinarySearchTree
87  {
88      Node root;
89      TreeType treeType;
90
91      public enum TreeType
92      {
93          Strings,
94          Numbers
95      };
96
97      public BinarySearchTree(TreeType treeType)
98      {
99          this.treeType = treeType;
100     }
101
102     public Node getRoot()
103     {
104         return root;
105     }
106
107     public void setRoot(Node root)
108     {
109         this.root = root;
110     }
111
112     // retorna o proprio no' atual ou entao um novo no' que sera' filho do no'
        anterior
113     private Node insert(String str, Node current)
114     {
115         if (current == null)
116         {
117             current = new Node();
118             current.setElement(str);
119             current.setContactsList( new Lista() );
120         }
121
122         else
123         {
124             String currentStr = (String) current.getElement();
125
126             if (currentStr == null) {}
127
128             else if (str.compareTo(currentStr) < 0)
129             {
130                 current.setLeft( insert(str, current.getLeft()) );
131             }
132
133             else if (str.compareTo(currentStr) > 0)
134             {
135                 current.setRight( insert(str, current.getRight()) );
136             }
137         }
138
139         return current;
140     }
141
142     public void insert(Object element)
143     {
144         switch (treeType)
145         {
```

```
146                 case Strings:
147                     insert((String) element, getRoot());
148                     break;
149 
150                 /*case Numbers:
151                     insert((Double) element, getRoot());
152                     break;*/
153 
154                 default:
155                     break;
156             }
157         }
158 
159         public Node search(Double num, Node current)
160         {
161             Node result = null;
162 
163             if (current != null)
164             {
165                 Double currentNum = (Double) current.getElement();
166 
167                 if ( num.equals(currentNum) )
168                 {
169                     result = current;
170                 }
171 
172                 else if ( num.compareTo(currentNum) < 0 )
173                 {
174                     result = search(num, current.getLeft());
175                 }
176 
177                 else
178                 {
179                     result = search(num, current.getRight());
180                 }
181             }
182 
183             return result;
184         }
185 
186         public Node search(String str, Node current)
187         {
188             Node result = null;
189 
190             if (current != null)
191             {
192                 String currentStr = (String) current.getElement();
193 
194                 if (currentStr == null) {}
195 
196                 else if ( str.equals(currentStr) )
197                 {
198                     result = current;
199                 }
200 
201                 else if ( str.compareTo(currentStr) < 0 )
202                 {
203                     result = search(str, current.getLeft());
204                 }
205 
206                 else
207                 {
208                     result = search(str, current.getRight());
209                 }
210             }
211 
212             return result;
213         }
214 
215         public Node search(Object element)
216         {
217             switch (treeType)
218             {
```

```
219                 case Strings:
220                     return search((String) element, getRoot());
221 
222                 case Numbers:
223                     return search((Double) element, getRoot());
224 
225                 default:
226                     return null;
227             }
228         }
229     }
230 
231     class Contato
232     {
233         public String nome;
234         public String telefone;
235         public String email;
236         public int cpf;
237 
238         public Contato(){}
239 
240         public Contato(String nome, String telefone, String email, int cpf)
241         {
242             this.nome = nome;
243             this.telefone = telefone;
244             this.email = email;
245             this.cpf = cpf;
246         }
247     }
248 
249     class Agenda
250     {
251         BinarySearchTree agenda;
252 
253         public Agenda()
254         {
255             agenda = new BinarySearchTree(BinarySearchTree.TreeType.Strings);
256 
257             agenda.setRoot( new Node("M", null, null, new Lista()) );
258 
259             for (char letter = 'A'; letter <= 'Z'; letter++)
260             {
261                 agenda.insert("" + letter);
262             }
263         }
264 
265         public void insert(Contato contato)
266         {
267             String firstLetter = contato.nome.substring(0, 1);
268 
269             Node contatoNode = agenda.search(firstLetter);
270 
271             if (contatoNode != null)
272             {
273                 contatoNode.getContactsList().add(contato);
274             }
275         }
276 
277         public boolean search(String nome)
278         {
279             boolean found = false;
280 
281             String firstLetter = nome.substring(0, 1);
282 
283             Node contatoNode = agenda.search(firstLetter);
284 
285             if (contatoNode != null)
286             {
287                 Lista contactsList = contatoNode.getContactsList();
288                 Contato currentContact;
289                 int size = contactsList.size();
290 
291                 for (int i = 0; i < size && !found; i++)
```

```
292                 {
293                     currentContact = (Contato) contactsList.get(i);
294 
295                     found = currentContact.nome.equals(nome);
296                 }
297             }
298 
299             return found;
300         }
301 
302         public boolean search(int cpf)
303         {
304             boolean found = false;
305 
306             for (char firstLetter = 'A'; firstLetter <= 'Z' && !found; firstLetter++)
307             {
308                 String firstLetterStr = "" + firstLetter;
309                 Node contatoNode = agenda.search(firstLetterStr);
310 
311                 if (contatoNode != null)
312                 {
313                     Lista contactsList = contatoNode.getContactsList();
314                     Contato currentContact;
315                     int size = contactsList.size();
316 
317                     for (int i = 0; i < size && !found; i++)
318                     {
319                         currentContact = (Contato) contactsList.get(i);
320 
321                         found = currentContact.cpf == cpf;
322                     }
323                 }
324             }
325 
326             return found;
327         }
328     }
```