```java
/*

PUC Minas - Ciencia da Computacao     Nome: SimpleList

Autor: Axell Brendow Batista Moreira  Matricula: 631822

Versao:  1.0                          Data: 24/09/2018

*/

class SimpleList
{
    // e' importante ressaltar que essa nao e' verdadeiramente a primeira celula
    // da fila, a primeira celula fica sempre depois dessa. Caso nao exista uma
    // celula depois, a fila esta' vazia.
    Cell _head;
    Cell _last;

    public SimpleList()
    {
        _head = new Cell();
        _last = _head;
    }

    private Cell getHead()
    {
        return _head;
    }

    private void setHead(Cell first)
    {
        _head = first;
    }

    private Cell getFirst()
    {
        return getHead().getNext();
    }

    private Cell getLast()
    {
        return _last;
    }

    private void setLast(Cell last)
    {
        _last = last;
    }

    public boolean isEmpty()
    {
        return getHead() == getLast();
    }

    public int getNumberOfElements()
    {
        int numberOfElements = 0;
        Cell currentCell = getHead();
        Cell lastCell = getLast();

        while (currentCell != lastCell)
        {
            numberOfElements++;
            currentCell = currentCell.getNext();
        }

        currentCell = null;
        lastCell = null;

        return numberOfElements;
    }

    public Cell getCellOnIndex(int index)
```

```
74        {
75            int numberOfElements = getNumberOfElements();
76            Cell cell = null;
77
78            if (index < 0 || index >= numberOfElements)
79            {
80                System.out.println("[SimpleList]: Indice invalido. (index = " + index +
                  ") - funcao getCellOnIndex(int)");
81            }
82
83            else if (numberOfElements > 0)
84            {
85                int currentIndex = 0;
86                cell = getFirst();
87
88                while (currentIndex < index)
89                {
90                    currentIndex++;
91                    cell = cell.getNext();
92                }
93            }
94
95            return cell;
96        }
97
98        public Object removeOnEnd()
99        {
100           Object removedElement = null;
101
102           if (!isEmpty())
103           {
104               int numberOfElements = getNumberOfElements();
105
106               Cell last = getLast();
107               Cell previous = getCellOnIndex(numberOfElements - 2);
108
109               removedElement = last.getElement();
110
111               last.setNext(null);
112               previous.setNext(null);
113               setLast(previous);
114
115               previous = null;
116               last = null;
117           }
118
119           return removedElement;
120       }
121
122       public Object removeOnStart()
123       {
124           Object removedElement = null;
125
126           if (!isEmpty())
127           {
128               Cell head = getHead();
129               Cell first = getFirst();
130
131               removedElement = first.getElement();
132
133               head.setNext(null);
134
135               setHead(first);
136
137               first = null;
138               head = null;
139           }
140
141           return removedElement;
142       }
143
144       public Object remove(int index)
145       {
```

```
146             int numberOfElements = getNumberOfElements();
147             Object removedElement = null;
148 
149             if (index == numberOfElements - 1)
150             {
151                 removedElement = removeOnEnd();
152             }
153 
154             else if (index == 0)
155             {
156                 removedElement = removeOnStart();
157             }
158 
159             else
160             {
161                 Cell previousOfRemovedCell = getCellOnIndex(index - 1);
162 
163                 if (previousOfRemovedCell != null)
164                 {
165                     Cell removedCell = previousOfRemovedCell.getNext();
166                     removedElement = removedCell.getElement();
167 
168                     previousOfRemovedCell.setNext(removedCell.getNext());
169 
170                     removedCell.setNext(null);
171                     removedCell = null;
172                 }
173 
174                 previousOfRemovedCell = null;
175             }
176 
177             return removedElement;
178         }
179 
180         public Object removeSecondPosition()
181         {
182             return remove(1);
183         }
184 
185         public void addOnEnd(Object element)
186         {
187             Cell newCell = new Cell(element, null);
188 
189             getLast().setNext(newCell);
190             setLast(newCell);
191 
192             newCell = null;
193         }
194 
195         public void add(Object element, int index)
196         {
197             int numberOfElements = getNumberOfElements();
198 
199             if (index > numberOfElements || index < 0)
200             {
201                 System.out.println("[SimpleList]: Indice invalido. (index =" + index +
                    ") - funcao add(Object, int)");
202             }
203 
204             else if (index == numberOfElements)
205             {
206                 addOnEnd(element);
207             }
208 
209             else
210             {
211                 Cell previousOfCellOnIndex = index == 0 ? getHead() :
                    getCellOnIndex(index - 1);
212                 Cell cellOnIndex = previousOfCellOnIndex.getNext();
213                 Cell newCell = new Cell(element, cellOnIndex);
214 
215                 previousOfCellOnIndex.setNext(newCell);
216 
```

```
217                 newCell = null;
218                 cellOnIndex = null;
219                 previousOfCellOnIndex = null;
220             }
221         }
222 
223         public void add(Object element)
224         {
225             addOnEnd(element);
226         }
227 
228         public void printSimpleList()
229         {
230             if (!isEmpty())
231             {
232                 Cell currentCell = getFirst();
233                 Cell lastCell = getLast();
234 
235                 System.out.print("[ " + currentCell.getElement());
236 
237                 while (currentCell != lastCell)
238                 {
239                     currentCell = currentCell.getNext();
240 
241                     System.out.print(", " + currentCell.getElement());
242                 }
243 
244                 System.out.println(" ]");
245             }
246         }
247     }
```