

```

1  /*
2
3  PUC Minas - Ciencia da Computacao      Nome: Stack
4
5  Autor: Axell Brendow Batista Moreira   Matricula: 631822
6
7  Versao: 1.0                            Data: 19/09/2018
8
9  */
10
11 class Stack
12 {
13     Cell _top;
14
15     public Stack()
16     {
17         _top = null; // redundante em Java
18     }
19
20     private Cell getTop()
21     {
22         return _top;
23     }
24
25     private void setTop(Cell top)
26     {
27         _top = top;
28     }
29
30     public int sumElements()
31     {
32         int sum = 0;
33         Cell currentCell = getTop();
34
35         while (currentCell != null)
36         {
37             sum += (int) currentCell.getElement();
38
39             currentCell = currentCell.getNext();
40         }
41
42         return sum;
43     }
44
45     public int sumElementsRecursively(Cell currentCell)
46     {
47         int sum = 0;
48
49         if (currentCell != null)
50         {
51             sum += (int) currentCell.getElement();
52
53             sum += sumElementsRecursively(currentCell.getNext());
54         }
55
56         return sum;
57     }
58
59     public int getGreatestElement()
60     {
61         int greatestElement = Integer.MIN_VALUE;
62         int currentElement;
63         Cell currentCell = getTop();
64
65         while (currentCell != null)
66         {
67             currentElement = (int) currentCell.getElement();
68
69             if (currentElement > greatestElement)
70             {
71                 greatestElement = currentElement;
72             }
73

```

```

74         currentCell = currentCell.getNext();
75     }
76
77     return greatestElement;
78 }
79
80 public int getGreatestBetweenPreviousAndCurrent(Cell currentCell, int
previousGreatestElement)
81 {
82     int greatestElement = previousGreatestElement;
83
84     if (currentCell != null)
85     {
86         int currentElement = (int) currentCell.getElement();
87
88         if (currentElement > greatestElement)
89         {
90             greatestElement = currentElement;
91         }
92
93         greatestElement =
getGreatestBetweenPreviousAndCurrent(currentCell.getNext(),
greatestElement);
94     }
95
96     return greatestElement;
97 }
98
99 public int getGreatestElementRecursively()
100 {
101     return getGreatestBetweenPreviousAndCurrent(getTop(), Integer.MIN_VALUE);
102 }
103
104 public void printGoingToBase(Cell currentCell)
105 {
106     if (currentCell != null)
107     {
108         System.out.println(currentCell.getElement().toString());
109
110         printGoingToBase(currentCell.getNext());
111     }
112 }
113
114 public void printFromTopToBase()
115 {
116     printGoingToBase(getTop());
117 }
118
119 public void printFromBaseToCell(Cell currentCell)
120 {
121     if (currentCell != null)
122     {
123         printFromBaseToCell(currentCell.getNext());
124
125         System.out.println(currentCell.getElement().toString());
126     }
127 }
128
129 public void printFromBaseToTop()
130 {
131     printFromBaseToCell(getTop());
132 }
133
134 public int getNumberOfElements()
135 {
136     int numberOfElements = 0;
137     Cell currentCell = getTop();
138
139     while (currentCell != null)
140     {
141         numberOfElements++;
142         currentCell = currentCell.getNext();
143     }

```

```
144         return numberOfElements;
145     }
146
147     public Cell getCellOnIndex(int index)
148     {
149         int currentIndex = 0;
150         Cell currentCell = getTop();
151
152         while (currentIndex < index)
153         {
154             currentIndex++;
155             currentCell = currentCell.getNext();
156         }
157
158         return currentCell;
159     }
160
161     public int getCellIndex(Cell cellToFind, int numberOfElements)
162     {
163         int cellIndex = -1;
164         Cell currentCell = getTop();
165
166         for (int i = 0; i < numberOfElements; i++)
167         {
168             if (currentCell == cellToFind)
169             {
170                 cellIndex = i;
171                 i = numberOfElements;
172             }
173
174             else
175             {
176                 currentCell = currentCell.getNext();
177             }
178         }
179
180         return cellIndex;
181     }
182
183     public void printFromBaseToCellIteratively(Cell cell)
184     {
185         int numberOfElements = getNumberOfElements();
186         int cellIndex = getCellIndex(cell, numberOfElements);
187
188         for (int i = numberOfElements - 1; i >= cellIndex; i--)
189         {
190             System.out.println(getCellOnIndex(i).getElement().toString());
191         }
192     }
193
194     public void printFromBaseToTopIteratively()
195     {
196         printFromBaseToCellIteratively(getTop());
197     }
198 }
199
```