```
/*

PUC Minas - Ciencia da Computacao     Nome: Queue

Autor: Axell Brendow Batista Moreira  Matricula: 631822

Versao:  1.0                          Data: 24/09/2018

*/

class Queue
{
    // 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 Queue()
    {
        _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("[Queue]: 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 removeOnStart()
99      {
100         Object removedElement = null;
101
102         if (!isEmpty())
103         {
104             Cell head = getHead();
105             Cell first = getFirst();
106
107             removedElement = first.getElement();
108
109             head.setNext(null);
110
111             setHead(first);
112
113             first = null;
114             head = null;
115         }
116
117         return removedElement;
118     }
119
120     public Object remove()
121     {
122         return removeOnStart();
123     }
124
125     public void addOnEnd(Object element)
126     {
127         Cell newCell = new Cell(element, null);
128
129         getLast().setNext(newCell);
130         setLast(newCell);
131
132         newCell = null;
133     }
134
135     public void add(Object element)
136     {
137         addOnEnd(element);
138     }
139
140     public void printQueue()
141     {
142         if (!isEmpty())
143         {
144             Cell currentCell = getFirst();
145             Cell lastCell = getLast();
```

```
146
147             System.out.print("[ " + currentCell.getElement());
148
149             while (currentCell != lastCell)
150             {
151                 currentCell = currentCell.getNext();
152
153                 System.out.print(", " + currentCell.getElement());
154             }
155
156             System.out.println(" ]");
157         }
158     }
159
160     public int getGreatestElement()
161     {
162         int greatestElement = Integer.MIN_VALUE;
163
164         if (!isEmpty())
165         {
166             Cell lastCell = getLast();
167             Cell currentCell = getFirst();
168             greatestElement = (int) currentCell.getElement();
169             int currentElement;
170
171             while (currentCell != lastCell)
172             {
173                 currentCell = currentCell.getNext();
174
175                 currentElement = (int) currentCell.getElement();
176
177                 if (currentElement > greatestElement)
178                 {
179                     greatestElement = currentElement;
180                 }
181             }
182         }
183
184         return greatestElement;
185     }
186
187     public int getThirdElement()
188     {
189         return (int) getFirst().getNext().getNext().getElement();
190     }
191
192     public void printThirdElement()
193     {
194         System.out.println("3rd element = " + getThirdElement());
195     }
196
197     public int sumElements()
198     {
199         int sum = 0;
200
201         if (!isEmpty())
202         {
203             Cell currentCell = getFirst();
204             Cell lastCell = getLast();
205
206             sum += (int) currentCell.getElement();
207
208             while (currentCell != lastCell)
209             {
210                 currentCell = currentCell.getNext();
211
212                 sum += (int) currentCell.getElement();
213             }
214         }
215
216         return sum;
217     }
218
```

```
219         public void invertQueue()
220         {
221             if (!isEmpty())
222             {
223                 Cell headCell = getHead();
224                 Cell lastCell = getLast();
225                 Cell currentCell = getFirst();
226                 Cell previousOfCurrent;
227                 Cell nextOfCurrent = currentCell.getNext();
228 
229                 setLast(currentCell);
230                 currentCell.setNext(null);
231 
232                 while (currentCell != lastCell)
233                 {
234                     previousOfCurrent = currentCell;
235                     currentCell = nextOfCurrent;
236                     nextOfCurrent = currentCell.getNext();
237 
238                     currentCell.setNext(previousOfCurrent);
239                 }
240 
241                 headCell.setNext(currentCell);
242 
243                 headCell = null;
244                 lastCell = null;
245                 currentCell = null;
246                 previousOfCurrent = null;
247                 nextOfCurrent = null;
248             }
249         }
250 
251         public int oddAndMultiplesOf5R(Cell currentCell, int previousCount)
252         {
253             int count = previousCount;
254 
255             if (currentCell != getLast())
256             {
257                 Object obElement = currentCell.getElement();
258                 int element = obElement == null ? 1 : (int) obElement;
259 
260                 if (element % 2 == 0 && element % 5 == 0)
261                 {
262                     count++;
263                 }
264 
265                 count = oddAndMultiplesOf5R(currentCell.getNext(), count);
266             }
267 
268             else
269             {
270                 Object obElement = currentCell.getElement();
271                 int element = obElement == null ? 1 : (int) obElement;
272 
273                 if (element % 2 == 0 && element % 5 == 0)
274                 {
275                     count++;
276                 }
277             }
278 
279             return count;
280         }
281 
282         public int getOddAndMultiplesOf5()
283         {
284             if (!isEmpty())
285             {
286                 return oddAndMultiplesOf5R(getFirst(), 0);
287             }
288 
289             else
290             {
291                 return 0;
```

```
292             }
293         }
294     }
```