```java
package flobee.pageradapterex;

import android.support.v4.view.PagerAdapter;

import java.util.ArrayList;

public class SkywalkerAdapter{
 private static Integer DISPLAYED    = 1;
 private static Integer NOT_DISPLAYED = 2;
 private static Integer NOT_IN_USE   = 3;
 private ArrayList<ArrayList<Character>> characters    = new ArrayList<>();
 private ArrayList<Integer> displayInfos = new ArrayList<>();
 int root = 0;
 int currLineIndex;

 public void addCharacters(ArrayList<Character> additionalCharacters){
  this.characters.add(additionalCharacters);
  if(displayInfos.isEmpty()){
   for(int ii=0; ii<additionalCharacters.size(); ii++){
    displayInfos.add(NOT_DISPLAYED);
   }
  }
 }

 public Character getCharacterAt(int position){
  int correctedPosition = position + root;
  if(displayInfos.size()>correctedPosition){
   return getCharacter(currLineIndex, correctedPosition);
  }
  return null;
 }

 public void gotCharacterAtPosition(int position){
  int correctedPosition = position + root;
  displayInfos.set(correctedPosition, DISPLAYED);
 }

 public int getCount(){
  int count = 0;
  for(int ii=0; ii<displayInfos.size(); ii++){
   if(displayInfos.get(ii)!= 3)
    count++;
  }
  return count;
 }
```

```
46
47   // Returns one of these: PagerAdapter.POSITION_UNCHANGED,
48   // PagerAdapter.POSITION_NONE, or new position.
49   public int getItemPosition(Character character){
50    int indexOfCharacter = findIndexOfCharacter(currLineIndex, character);
51    if(indexOfCharacter == -1)
52     return PagerAdapter.POSITION_NONE;
53    if(displayInfos.get(indexOfCharacter) == NOT_IN_USE)
54     return PagerAdapter.POSITION_NONE;
55    if(displayInfos.get(indexOfCharacter) == DISPLAYED)
56     return PagerAdapter.POSITION_UNCHANGED;
57    int correctedPosition = indexOfCharacter - root;
58    return correctedPosition;
59   }
60
61   public void changeChildTo(String parent, String child){
62    if(parent == null){
63     changeChildToRoot(child);
64    }
65    else{
66      moveToNewLine(child);
67    }
68   }
69
70   // if root is not zero, make it zero.
71   public void refreshCurrLine(String child){
72    if(findIndexOfCharacter(currLineIndex, child) == -1){
73     // TODO throw an exception.
74    }
75    else{
76     root = 0;
77     charactersAfterAndIncludingHaveNotBeenDisplayed(root);
78    }
79   }
80
81   private void changeChildToRoot(String child){
82    int indexOfChildInCurrLine = findIndexOfCharacter(currLineIndex, child);
83    if(indexOfChildInCurrLine != -1){
84     // child is in current line.
85     root = indexOfChildInCurrLine;
86     clearAboveRoot();
87     charactersAfterAndIncludingHaveNotBeenDisplayed(root);
88    }
89    else{
90     // move to new line.
```

```
91      currLineIndex = findLineWithCharacter(child);
92      updateDisplayInfoSize();
93      root = findIndexOfCharacter(currLineIndex, child);
94      clearAboveRoot();
95      charactersAfterAndIncludingHaveNotBeenDisplayed(root);
96     }
97    }
98
99    private void moveToNewLine (String child){
100    int indexOfChildInCurrLine = findIndexOfCharacter(currLineIndex, child);
101    // Actually, if child is in current line, I dont have to move to a new line.
102    // Only if indexOfChildInCurrLine is -1 do I move to new line.
103    if (indexOfChildInCurrLine == -1){
104     int oldLineIndex = currLineIndex;
105     currLineIndex = findLineWithCharacter(child);
106     updateDisplayInfoSize();
107     updateDisplayInfoWithOldLine(oldLineIndex);
108     int characterIndex = findIndexOfCharacter(currLineIndex, child);
109     charactersAfterAndIncludingHaveNotBeenDisplayed(characterIndex);
110    }
111   }
112
113   private void updateDisplayInfoSize (){
114    int correctInfoSize = characters.get(currLineIndex).size();
115    while (displayInfos.size()< correctInfoSize){
116     displayInfos.add(NOT_DISPLAYED);
117    }
118    while (displayInfos.size()> correctInfoSize){
119     displayInfos.remove(displayInfos.size()-1);
120    }
121   }
122
123   private void updateDisplayInfoWithOldLine (int oldLineIndex){
124    ArrayList<Character> currentLine = characters.get(currLineIndex);
125    ArrayList<Character> oldLine = characters.get(oldLineIndex);
126    for (int ii=0; ii<currentLine.size(); ii++){
127     if (oldLine.size()> ii){
128      Character oldCharacter = oldLine.get(ii);
129      Character newCharacter = currentLine.get(ii);
130      if (!oldCharacter.equals(newCharacter))
131       displayInfos.set(ii, NOT_DISPLAYED);
132     }
133     else{
134      if (displayInfos.size()> ii)
135       displayInfos.set(ii, NOT_DISPLAYED);
```

```
136       else
137         displayInfos.add(NOT_DISPLAYED);
138       }
139
140     }
141   }
142
143   private void clearAboveRoot(){
144     for(int ii=0; ii<root; ii++){
145       displayInfos.set(ii, NOT_IN_USE);
146     }
147   }
148
149   private Character getCharacter(int line, int index){
150     return characters.get(line).get(index);
151   }
152
153   private int findIndexOfCharacter(int lineIdx, Character character){
154     ArrayList<Character> line = characters.get(lineIdx);
155     for(int ii = 0; ii< line.size(); ii++){
156       if(line.get(ii).equals(character))
157         return ii;
158     }
159     return -1;
160   }
161
162   private int findIndexOfCharacter(int lineIdx, String characterName){
163     ArrayList<Character> line = characters.get(lineIdx);
164     for(int ii = 0; ii< line.size(); ii++){
165       if(line.get(ii).getCharacterName().equals(characterName))
166         return ii;
167     }
168     return -1;
169   }
170
171   // look in every line.
172   private int findLineWithCharacter(String characterName){
173     for(int ii=0; ii<characters.size(); ii++){
174       int indexFromCurrLine = findIndexOfCharacter(ii, characterName);
175       if(indexFromCurrLine != -1)
176         return ii;
177     }
178     return -1;
179   }
180
```

```
181    private void charactersAfterAndIncludingHaveNotBeenDisplayed(int index){
182     for (int ii = index; ii< displayInfos.size(); ii++){
183      displayInfos.set(ii, NOT_DISPLAYED);
184     }
185    }
186  }
187
```