Програма със свързани списъци

В този урок ще имплементираме всички методи, които подготвихме в миналият урок: 1. printMe() - принтира даден списък в конзолата 2. removeStuff() - премаха елементи от списък 3. reverseMe() - принтира дписък отзад-напред

Имплементация на printMe()

Единственият параметър, който ще се подава на този метод е списъкът, който искаме да бъде принтиран.

```
private static void printMe(List<String> list){
  //описание на метода
}
```

За принтирането отново ще използваме цикъл, който да преминава през всеки елемент от списъка и да изкарва съдържанието му на конзолата.

```
private static void printMe(List<String> list){
  for(String b: list){
    System.out.printf("%s ", b);
  }
  System.out.println();
}
```

Имплементация на removeStuff()

При този метод имаме 3 параметъра: 1. Списък, който да бъде редактиран 2. Индекс на началният елемент 3. Индекс на крайният елемент (няма да бъде премахнат)

```
private static void removeStuff(List<String> list, int from, int to){
  //съдържание на метода
}
```

Понеже работим със свързани списъци, можем да използваме техните методи: * **sublList(from, to)** взима част от списъка * **clear()** изтрива тази част

```
private static void removeStuff(List<String> list, int from, int to){
  list.subList(from, to).clear();
}
```

Имплементация на reverseMe()

Тук отново имаме само един параметър - списъкът, който трябва да бъде принтиран.

```
private static void reverseMe(List<String> list){
  //описание на метода
}
```

В метода ще използваме отново итератори, но този път трябва да започнем обхождането на списъка от последният елемент.

```
// инициализиране на итератор,
// но със селектиране на последният list.size() елемент
ListIterator<String> iterator = list.listIterator(list.size());
```

След това отново с цикъл, но вървейки от последният елемент до първия, принтираме в конзолата:

```
// докато има предходен елемент
while(iterator.hasPrevious()){
  // принтирай го в конзолата
  System.out.printf("%s ", iterator.previous());
}
```

След това поставяме един празен ред след цикъла за по-подредено извеждане в конзолата и методът е готов:

```
private static void reverseMe(List<String> list){
  ListIterator<String> iterator = list.listIterator(list.size());
  while(iterator.hasPrevious()){
    System.out.printf("%s ", iterator.previous());
  }
  System.out.println();
}
```

С това е готова и цялата програма

```
import java.util.*;
class Demo{
  public static void main(String[] args){

    String[] things = {"apples", "noobs", "pwnge", "bacon", "goATS"};
    List<String> list1 = new LinkedList<String>();

    for(String x: things){
      list1.add(x);
    }

    String[] things2 = {"sausage", "bacon", "goats", "harrypotter"};
    List<String> list2 = new LinkedList<String>();

    for(String y: things2){
```

```
      list2.add(y);
    }

    list1.addAll(list2);
    list2 = null;

    printMe(list1);
    removeStuff(list1, 2, 5);
    printMe(list1);
    reverseMe(list1);
  }

  //printMe method
  private static void printMe(List<String> list){
    for(String b: list){
      System.out.printf("%s ", b);
    }
    System.out.println();
  }

  //removeStuff method
  private static void removeStuff(List<String> list, int from, int to){
    list.subList(from, to).clear();
  }

  //reverseMe method
  private static void reverseMe(List<String> list){
    ListIterator<String> iterator = list.listIterator(list.size());
    while(iterator.hasPrevious()){
      System.out.printf("%s ", iterator.previous());
    }
    System.out.println();
  }
}
```