Laboratório 21 - Coleções III (30/10/2014)

1.

```
package lab21;
import java.util.Comparator;

public class IntegerComparator implements Comparator<Integer>{
/**
 * Compares its two arguments for order. Returns a negative integer, zero, or a positive
 * integer as the first argument is less than, equal to, or greater than the second.
 *
 * @return one of -1, 0, or 1 according to whether the value of expression is negative,
.* zero or positive.
 */
      @Override
      public int compare(Integer i1, Integer i2) {
      return (i1 < i2) ? -1 : (i1==i2) ? 0 : 1;
      }
}
```

```
package lab21;

import java.util.ArrayList;
import java.util.Collection;
import java.util.Comparator;
import java.util.Iterator;
import java.util.LinkedList;
import java.util.List;

public class Remove {
      /**
       * This method removes, from the iterated sequence "iter", all integers which are greater
       * than the value given by CMP comparator.
       * This method returns a collection of all integers removed from the original collection.
       * @param iter
       * @param cmp
       * @param value
       * @return list collection of all integers removed from the original collection.
       */
      public static Collection<Integer> removeGreaterThanAndGetLastNRemoved(
            Iterator<Integer> iter, Comparator<Integer> cmp, Integer value)
      {
         LinkedList<Integer> list = new LinkedList<Integer>(); //afetação da list

         while (iter.hasNext()) {

            Integer num = iter.next();

//comparar cada elemento do iter com o value, retorna 0 se são iguais, um nº menor que 0
// se este Integer é numericamente inferior ao argumento value e um nº maior que 0 se
// este Integer é numericamente superior ao argumento value.

            if (cmp.compare(num,value)>0){
                     // queremos remover apenas os valores superiores ao argumento dado

               list.add(num);     // adicionar o elemento à nova lista
               iter.remove();     //remover o elemento do iterador
            }
         }
         return list;
      }
```

```
/**
 * This method removes from the iterated sequence by "it" the different elements of null that are
 * multiples of 5. The elements removed from the iterated sequence by "it" are placed in a new
 * collection.
 * In the new collection odd values must precede peers. Returns a new collection.
 * @param it
 * @return multiple5List a new collection where the odd elements are first, than appears the peers
```

```
 * elements
 */
       public static Collection<Integer> multipleOfFiveOddBeforeEven( Iterator<Integer> it ){
          LinkedList<Integer> multiple5List = new LinkedList<Integer>();
                                                                    //afetação da list
          while (it.hasNext()) {
              Integer num = it.next();

              if (num%5==0 && num%10==0){ // números multiplos de 10
                 multiple5List.addLast(num);
                             // adicionar o elemento à nova lista (ainda não está ordenada)
                 it.remove();                  // remove estes elementos da lista inicial
              }
              else
                 if (num%5==0 && num%10!=0){
                                       // números multiplos de 5 que não são multiplos de 10
                     multiple5List.addFirst(num);
                     it.remove();
                 }
          }
          return multiple5List;
       }
```

```
       public static void main (String[] args){    // para testar o método
              List<Integer> list = new ArrayList<>();
              list.add(3);
              list.add(5);
              list.add(10);
              list.add(4);
              list.add(80);
              list.add(2);
              list.add(17);     // adicionar elementos à lista que vai ser iterada
              list.add(100);
              list.add(50);
              list.add(25);
              list.add(98);
              list.add(35);

              //Exercício 1
              Iterator<Integer> elements1 = list.iterator();
                                                 //afetação da variável elements - iterador
              Comparator<Integer> cmp = new IntegerComparator();
                                       //afetação da variável cmp do tipo IntegerComparator
              LinkedList<Integer> removeList = (LinkedList<Integer>)
removeGreaterThanAndGetLastNRemoved(elements1, cmp, 20);
              System.out.print("Lista dos elementos removidos (elementos superiores a 20): ");

              for (int idx=0; idx<removeList.size(); idx++){
                     // imprimir os elementos que foram removidos e que são superiores a 2
                 System.out.print((idx==removeList.size()-1)? removeList.get(idx) :
removeList.get(idx)+ ", ");
              }

              System.out.print("\nLista inicial modificada: ");
              for (int i=0; i<list.size(); i++){
                  // imprimir os elementos que ficaram na lista original
                 System.out.print((i==list.size()-1)? list.get(i) : list.get(i)+ ", ");
              }

              //Exercício 2
              List<Integer> list1 = new ArrayList<>();
              list1.add(3);
              list1.add(5);
              list1.add(10);
              list1.add(4);
              list1.add(80);
```

```
            list1.add(2);
            list1.add(17);    // adicionar elementos à lista que vai ser iterada
            list1.add(100);
            list1.add(50);
            list1.add(25);
            list1.add(98);
            list1.add(35);

            Iterator<Integer> elements2 = list1.iterator();
            System.out.print("\nLista inicial com os elementos: ");
            for (int i=0; i<list1.size(); i++){
               System.out.print((i==list1.size()-1)? list1.get(i): list1.get(i)+ ", ");
            }

            LinkedList<Integer> multiple5List = (LinkedList<Integer>)
multipleOfFiveOddBeforeEven(elements2);
            System.out.print("\nLista resultante (Múltiplos de 5): ");

            for (int idx=0; idx<multiple5List.size(); idx++){
                                                  // imprimir os elementos multiplos de 5
               System.out.print((idx==multiple5List.size()-1)? multiple5List.get(idx):
multiple5List.get(idx)+ ", ");
            }

            System.out.print("\nLista inicial modificada: ");
            for (int i=0; i<list1.size(); i++){
                                 // imprimir os elementos que ficaram na lista original
               System.out.print((i==list1.size()-1)? list1.get(i) : list1.get(i)+ ", ");
            }
      }
}
```

Output

```
Lista dos elementos removidos (elementos superiores a 20): 80, 100, 50, 25, 98, 35
Lista inicial modificada: 3, 5, 10, 4, 2, 17
Lista inicial com os elementos: 3, 5, 10, 4, 80, 2, 17, 100, 50, 25, 98, 35
Lista resultante (Múltiplos de 5): 35, 25, 5, 10, 80, 100, 50
Lista inicial modificada: 3, 4, 2, 17, 98
```