1. **Compare performance of ArrayList and LinkedList in difference cases**

   To compare the performance follow the next steps:
   a. Create ArrayList object and add 1.000.000 elements there (add Integers objects in the loop)
   b. Create LinkedList object and add 1.000.000 elements there (add Integers objects in the loop)
   c. Create method that takes list as an argument and adds specific amount of integers into the beginning of list

      Method declaration:

      public static void addElementsToBeginning(List<Integer> list, int numberOfElementsToAdd)

   d. Create method that takes list as an argument and adds specific amount of integers into the middle of list

      Method declaration:

      public static void addElementsToMiddle(List<Integer> list, int numberOfElementsToAdd)

   e. Create method that takes list as an argument and adds specific amount of integers into the end of list

      Method declaration:

      public static void addElementsToEnd(List<Integer> list, int numberOfElementsToAdd)

   f. Create method that takes list as an argument and removes specific amount of integers from the beginning of the list

      public static void removeElementsFromBeginning(List<Integer> list, int numberOfElementsToRemove)

   g. Create method that takes list as an argument and removes specific amount of integers from the middle of the list

      public static void removeElementsFromMiddle(List<Integer> list, int numberOfElementsToRemove)

   h. Create method that takes list as an argument and removes specific amount of integers from the end of the list

public static void removeElementsFromEnd(List<Integer> list, int numberOfElementsToRemove)

i. Perform next operations for both: LinkedList and ArrayList:
   i. add 100 elements into the end of the List
   ii. add 10.000 elements into the end of the List
   iii. add 100.000 elements into the end of the List

   iv. add 100 elements into the middle of the List
   v. add 10.000 elements into the middle of the List
   vi. add 100.000 elements into the middle of the List

   vii. add 100 elements into the beginning of the List
   viii. add 10.000 elements into the beginning of the List
   ix. add 100.000 elements into the beginning of the List

   x. remove 100 elements from the end of the List
   xi. remove 10.000 elements from the end of the List
   xii. remove 100.000 elements from the end of the List

   xiii. remove 100 elements from the middle of the List
   xiv. remove 10.000 elements from the middle of the List
   xv. remove 100.000 elements from the middle of the List

   xvi. remove 100 elements from the beginning of the List
   xvii. remove 10.000 elements from the beginning of the List
   xviii. remove 100.000 elements from the beginning of the List

j. Fill out the next tables:

INSERTION

	beginning			middle			end		
	100	10 000	100 000	100	10 000	100 000	100	10 000	100 000
ArrayList									
LinkedList									

DELETION

	beginning			middle			end		
	100	10 000	100 000	100	10 000	100 000	100	10 000	100 000
ArrayList									
LinkedList									

k. Technical note: to measure time you can capture the time at the specific moment, after that execute specific method, and after that calculate delta in time like this:

long mill = System.nanoTime();
removeElementsFromEnd(list, 100);
long delta = (System.nanoTime() - mill) / 10000;

And one more note: it is allowed to insert some constant value like Integer.MAX_VALUE

l. Do not forget to reset state of elements to 1_000_000 each time before testing new scenario

m. After you completed all steps you can compare your results with the results of your tutor. Pay attention that results and numbers will be different. But the pattern will be the same.

n. Optionally, you may investigate performance during the ‘get’ operation.

INSERTION

	beginning			middle			end		
	100	10 000	100 000	100	10 000	100 000	100	10 000	100 000
ArrayList	13947	267291	1759892	**2290**	**67520**	**706177**	**2**	81	449
LinkedList	**4**	**178**	**281**	22908	2146884	36050328	3	**47**	**297**

DELETION

	beginning			middle			end		
	100	10 000	100 000	100	10 000	100 000	100	10 000	100 000
ArrayList	8608	211366	1748510	**2278**	**67952**	**672554**	**7**	**101**	**292**
LinkedList	**7**	**48**	**269**	30275	2242008	22345702	8	151	310