Laborator 5 – Liste

Tema de realizat

1. Pe pagina cursului, deschideti materialul Exemplu lista, tip generic, Iterator, Iterable(PDF)
2. Creati un proiect Lab5 in care sa includeti intregul pachet *listadiamant*
3. **(1p)** In clasa **ListaGenerica** introduceti o functie noua size() care sa dea numarul de elemente din lista. Complexitatea acestei metode trebuie sa fie O(1).
   In functia main() din clasa **AplicatieLista** adaugati cod care sa produca afisarea

```
Orasele din lista:Suceava Bucuresti Iasi
Numerele din lsta:19 17 15 13 11 9 7 5 3 1 0 2 4 6 8 10 12 14 16 18 20
Intregii din lsta x10: 190, 170, 150, 130, 110, 90, 70, 50, 30, 10, 0, 20, 40, 60,
80, 100, 120, 140, 160, 180, 200,
Lungimea listei este 21
```

4. **(1.5p)** Adaugati o clasa noua **ListaGenerica2<T>** care sa extinda **ListaGenerica<T>** si care sa implementeze metoda public T eliminaPrimul();
   In functia main() din clasa AplicatieLista adaugati cod care sa produca afisarea urmatoare

```
Orasele din lista:Suceava Bucuresti Iasi
Numerele din lsta:19 17 15 13 11 9 7 5 3 1 0 2 4 6 8 10 12 14 16 18 20
Intregii din lsta x10: 190, 170, 150, 130, 110, 90, 70, 50, 30, 10, 0, 20,
40, 60, 80, 100, 120, 140, 160, 180, 200,
Lungimea listei este 21

================ Demo eliminaPrimul() ==================
Numerele din lista(21): 19 17 15 13 11 9 7 5 3 1 0 2 4 6 8 10 12 14 16 18 20
Numerele din lista(20): 17 15 13 11 9 7 5 3 1 0 2 4 6 8 10 12 14 16 18 20
Numerele din lista(19): 15 13 11 9 7 5 3 1 0 2 4 6 8 10 12 14 16 18 20
Numerele din lista(18): 13 11 9 7 5 3 1 0 2 4 6 8 10 12 14 16 18 20
Numerele din lista(17): 11 9 7 5 3 1 0 2 4 6 8 10 12 14 16 18 20
Numerele din lista(16): 9 7 5 3 1 0 2 4 6 8 10 12 14 16 18 20
Numerele din lista(15): 7 5 3 1 0 2 4 6 8 10 12 14 16 18 20
Numerele din lista(14): 5 3 1 0 2 4 6 8 10 12 14 16 18 20
Numerele din lista(13): 3 1 0 2 4 6 8 10 12 14 16 18 20
Numerele din lista(12): 1 0 2 4 6 8 10 12 14 16 18 20
Numerele din lista(11): 0 2 4 6 8 10 12 14 16 18 20
Numerele din lista(10): 2 4 6 8 10 12 14 16 18 20
Numerele din lista(9): 4 6 8 10 12 14 16 18 20
Numerele din lista(8): 6 8 10 12 14 16 18 20
Numerele din lista(7): 8 10 12 14 16 18 20
Numerele din lista(6): 10 12 14 16 18 20
Numerele din lista(5): 12 14 16 18 20
Numerele din lista(4): 14 16 18 20
Numerele din lista(3): 16 18 20
Numerele din lista(2): 18 20
Numerele din lista(1): 20
Numerele din lista(0):
```

5. **(1.5p)** In clasa **ListaGenerica2<T>** includeti metoda public T eliminaUltimul(); Inserati cod in main() astfel incat sa se produca dupa afisarea de mai sus urmatorul rezultat:

```
================ Demo eliminaUltimul() ==================
Numerele din lista(21): 19 17 15 13 11 9 7 5 3 1 0 2 4 6 8 10 12 14 16 18 20
```

```
Numerele din lista(20): 19 17 15 13 11 9 7 5 3 1 0 2 4 6 8 10 12 14 16 18
Numerele din lista(19): 19 17 15 13 11 9 7 5 3 1 0 2 4 6 8 10 12 14 16
Numerele din lista(18): 19 17 15 13 11 9 7 5 3 1 0 2 4 6 8 10 12 14
Numerele din lista(17): 19 17 15 13 11 9 7 5 3 1 0 2 4 6 8 10 12
Numerele din lista(16): 19 17 15 13 11 9 7 5 3 1 0 2 4 6 8 10
Numerele din lista(15): 19 17 15 13 11 9 7 5 3 1 0 2 4 6 8
Numerele din lista(14): 19 17 15 13 11 9 7 5 3 1 0 2 4 6
Numerele din lista(13): 19 17 15 13 11 9 7 5 3 1 0 2 4
Numerele din lista(12): 19 17 15 13 11 9 7 5 3 1 0 2
Numerele din lista(11): 19 17 15 13 11 9 7 5 3 1 0
Numerele din lista(10): 19 17 15 13 11 9 7 5 3 1
Numerele din lista(9): 19 17 15 13 11 9 7 5 3
Numerele din lista(8): 19 17 15 13 11 9 7 5
Numerele din lista(7): 19 17 15 13 11 9 7
Numerele din lista(6): 19 17 15 13 11 9
Numerele din lista(5): 19 17 15 13 11
Numerele din lista(4): 19 17 15 13
Numerele din lista(3): 19 17 15
Numerele din lista(2): 19 17
Numerele din lista(1): 19
Numerele din lista(0):
```

6. **(1p)** Adaugati cod in functia main() din clasa **AplicatieLista** cod care sa produca in continuare si afisarea urmatoare:

```
================ Demo elimina Primul si Ultimul alternativ ======================
Numerele din lista(21) <init> 19 17 15 13 11 9 7 5 3 1 0 2 4 6 8 10 12 14 16 18 20
Numerele din lista(20) <19>  :17 15 13 11 9 7 5 3 1 0 2 4 6 8 10 12 14 16 18 20
Numerele din lista(19) <20>  :17 15 13 11 9 7 5 3 1 0 2 4 6 8 10 12 14 16 18
Numerele din lista(18) <17>  :15 13 11 9 7 5 3 1 0 2 4 6 8 10 12 14 16 18
Numerele din lista(17) <18>  :15 13 11 9 7 5 3 1 0 2 4 6 8 10 12 14 16
Numerele din lista(16) <15>  :13 11 9 7 5 3 1 0 2 4 6 8 10 12 14 16
Numerele din lista(15) <16>  :13 11 9 7 5 3 1 0 2 4 6 8 10 12 14
Numerele din lista(14) <13>  :11 9 7 5 3 1 0 2 4 6 8 10 12 14
Numerele din lista(13) <14>  :11 9 7 5 3 1 0 2 4 6 8 10 12
Numerele din lista(12) <11>  :9 7 5 3 1 0 2 4 6 8 10 12
Numerele din lista(11) <12>  :9 7 5 3 1 0 2 4 6 8 10
Numerele din lista(10) <9>  :7 5 3 1 0 2 4 6 8 10
Numerele din lista(9) <10>  :7 5 3 1 0 2 4 6 8
Numerele din lista(8) <7>  :5 3 1 0 2 4 6 8
Numerele din lista(7) <8>  :5 3 1 0 2 4 6
Numerele din lista(6) <5>  :3 1 0 2 4 6
Numerele din lista(5) <6>  :3 1 0 2 4
Numerele din lista(4) <3>  :1 0 2 4
Numerele din lista(3) <4>  :1 0 2
Numerele din lista(2) <1>  :0 2
Numerele din lista(1) <2>  :0
Numerele din lista(0) <0>  :
```

7. **(3p)** Creați o nouă clasă **ListGenerica<T> care să implementeze interfața java.util.List**

Veți implementa acele metode din interfața List<T> care fac posibilă execuția metodelor din clasa TestList. Veți crea un pachet nou `testlist` în care veți introduce clasa **TestList**.

Veti evalua cu clasa **TestList** comparativ durata de execuție a celor 3 implementări ale interfeței List<T>

- **ListGenerica<T>**
- **LinkedList<T>**
- **ArrayList<T>**

Clasa TestList

```
package testlist;

import java.util.ArrayList;
import java.util.LinkedList;
import java.util.List;
import listadiamant.ListGenerica;
/**
 * @author St.Gh. PENTIUC
 */
public class TestList {

    private static String FORMAT="%d";
    static int N = 100000, Nafisate=10;
    public static void main(String[] args) {
        List<Integer> lst1 = new ArrayList<>();
        List<Integer> lst2 = new LinkedList<>();
        List<Integer> lst3 = new ListGenerica<>();
        List liste[] = new List[]{lst3, lst2, lst1};
        for(List lst:liste){
            System.out.println("\n================= Demo "
                               + lst.getClass().getSimpleName()
                               + " ====================");
            System.out.printf("%s\n", lst.toString());
            testInserari(lst);
            testEliminari(lst, true);  // sau false fara if(..contains(x)..
        }
    }
     static void  testInserari(List<Integer> lst) {
      int nrerori = 0;
        int x, y;
        long startTime = System.currentTimeMillis();
        System.out.printf( "Start inserari (i- in fata, I - la urma):\n");
        for (x  = 0; x<N ;    x += 1){
            if (x > N / 2) {
                    lst.add(0, x);
                if ((y = lst.get(0)) != x) {
                   System.out.printf("\n***Eroare. S-a inserat "+FORMAT
                                     +" s-a gasit "+FORMAT, x, y);
                   nrerori++;
                }
            } else {
                lst.add(x);  //inserare la urma
                if ((y = lst.get( lst.size()-1 )) != x) {
                    System.out.printf("\n***Eroare. S-a inserat "+FORMAT
                                      +" s-a gasit "+FORMAT, x, y);
                    nrerori++;
                }
            }
            if (N > Nafisate)

                if ((int) x % 1000 == 0) {
                    System.out.printf("%c", x > N / 2 ? 'i' : 'I');
                }
            if (N <= Nafisate)
                 System.out.printf("\ni=%d %s\n", x, lst);
        }

        if(lst.size()!=N)
            System.out.printf("***Eroare. In lista ar trebui sa fie"
                              +" %d elemente",N);
```

```
        System.out.printf("\nLista are %d elemente.", lst.size());
        System.out.printf("\nTest inserare incheiat cu %d erori\n", nrerori);
        afisDurataExecutie(startTime);

    }
        static void afisDurataExecutie(long startTime){
            long durata = System.currentTimeMillis() - startTime;
            System.out.printf("Durata test %.3f s\nDurata per element"
                              + " %.6f ms\n\n",
                             (float)(durata/1000.), (float)(durata/(double)N));
        }
        static void testEliminari(List<Integer> lst,
                                  boolean  verificaSiContinut){
            int nrerori=0;
            int x=N;
            long startTime = System.currentTimeMillis();
            System.out.printf("\nStart eliminari (d- primul, D -"
                              + " ultimul element):\n");
            while(!lst.isEmpty()) {
               x--;
               if (N<=Nafisate)
                   System.out.printf("\nlst(%d elemente): ", lst.size(), lst);

            if(verificaSiContinut && lst.contains(x)==false){
               System.out.printf( "\n*** Eroare inserare info="+FORMAT
                                 + "In lista %s sunt %d elemente",x,
                                          ((Object)lst), lst.size());
               nrerori++;
            }
            if(x>N/2)
                lst.remove(0);
            else
                lst.remove( lst.size()-1 );
            if(verificaSiContinut && lst.contains(x)!=false) {
                System.out.printf("\n*** Eroare stergere info="+FORMAT
                                  + "In lista %s sunt %d elemente",
                                    x, lst, lst.size());
                nrerori++;
            } else {
               if (N<=Nafisate)
                  System.out.printf(" S-a eliminat "+FORMAT+" (lst a ramas %s)",
                                    x, lst);
               else
                  if((int)x%1000==0)
                      System.out.printf("%c", x>N/2 ? 'd':'D');
            }
           }
           System.out.printf("\nTest eliminare incheiat cu %d erori\n",
                                                          nrerori);
           afisDurataExecutie(startTime);

        }
    }
```

Comparati performantele clasei realizate la curs si laborator cu performantele celor 2 clase din pachetul java.util. Veti face 2 execuții

- una în care testele de remove() au verificat cu contains() conținutul listelor înainte și după remove()

- altă execuție în care se vor efcetua doar operații de remove() fără verificarea cu contains(); în acest scop, în metoda main() din clasa **TestList** veți apela `testEliminari(lst, false);`

Un exemplu de execuție în care testele de remove() au verificat cu contains() listele este urmatorul:

```
================= Demo ListGenerica =====================

Start inserari (i- in fata, I - la urma):
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIiiiiiiiiiiiiiiiiiiiiiiiiiiiii
iiiiiiiiiiiiiiiiiiii
Lista are 100000 elemente.
Test inserare incheiat cu 0 erori
Durata test 0,028 s
Durata per element 0,000280 ms


Start eliminari (d- primul, D - ultimul element):
dddddddddddddddddddddddddddddddddddddddddddddddddddDDDDDDDDDDDDDDDDDDDDDDDDDDDDD
DDDDDDDDDDDDDDDDDDDD
Test eliminare incheiat cu 0 erori
Durata test 15,298 s
Durata per element 0,152980 ms


================= Demo LinkedList =====================
[]
Start inserari (i- in fata, I - la urma):
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIiiiiiiiiiiiiiiiiiiiiiiiiiiiii
iiiiiiiiiiiiiiiiiiii
Lista are 100000 elemente.
Test inserare incheiat cu 0 erori
Durata test 0,015 s
Durata per element 0,000150 ms


Start eliminari (d- primul, D - ultimul element):
dddddddddddddddddddddddddddddddddddddddddddddddddddDDDDDDDDDDDDDDDDDDDDDDDDDDDDD
DDDDDDDDDDDDDDDDDDDD
Test eliminare incheiat cu 0 erori
Durata test 13,148 s
Durata per element 0,131480 ms


================= Demo ArrayList =====================
[]
Start inserari (i- in fata, I - la urma):
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIiiiiiiiiiiiiiiiiiiiiiiiiiiiii
iiiiiiiiiiiiiiiiiiii
Lista are 100000 elemente.
Test inserare incheiat cu 0 erori
Durata test 0,350 s
Durata per element 0,003500 ms


Start eliminari (d- primul, D - ultimul element):
dddddddddddddddddddddddddddddddddddddddddddddddddddDDDDDDDDDDDDDDDDDDDDDDDDDDDDD
DDDDDDDDDDDDDDDDDDDD
Test eliminare incheiat cu 0 erori
Durata test 4,863 s
Durata per element 0,048630 ms
```

8. (2p) Cautati o solutie care sa obtina timpi mai mici de executie la functia de eliminare a elementelor din lista. Pentru a fi mai putin influentati de platforma concreta pe care se executa

programul veti imparti timpul de executie al fiecarei clase la timpul obtinut cu **LinkedList**. Veti completa un tabel ca cel care urmeaza:

	ListGenerica	LinkedList	ArrayList	Clasa dv.
Timpi test remove (cu contains)	15,298	13,148	4,863	
Raportat la *LinkedList*	1,164	1	0,370	
Timpi test remove (fara contains)	2,360	0,009	0,337	
Raportat la *LinkedList*	262,222	1	37,444	

S-au folosit pentru primele clase datele din exemplul anterior. Dv. veti trece timpii obtinuti pe calculatorul dv.

Explicati

- de ce durata testelor de remove() care utilizeaza contains() sunt apropiate in cazul claselor **ListGenerica** si **LinkedList**
- diferenta foarte mare în cazul testelor de remove() efectate fără contains(x) între ListGenerica si **LinkedList** și între **ArrayList** si **LinkedList**

Temă acasă

1. Reproiectați aplicatia astfel încât **ListGenerica** să aibă durate appropriate cu **LinkedList** la testele de remove() și insert() fără a utiliza contains() în cadrul testelor (***5p***).
2. Implementați complet interfața **List** și scrieți un program de test în care să utilizați toate funcțiile (***10p***).