ArrayList<>

Класа `ArrayList` (`java.util.ArrayList`) представља проширив низ.
За чување референци на елементе интерно користи низ.
Имплементира интерфејс `Iterable<>`, па је могуће коришћење collection-based for-петље за обилазак свих елемената.

Еквивалентна је класи `Vector<>` (такође из пакета `java.util`), с тим да је `Vector<>` thread-safe (безбедан за коришћење од стране већег броја нити), а `ArrayList<>` није.

ако желите само да Ваш кôд ради, а не занима Вас да знате баш све детаље, сиве делове прескочите:

Операције `size()`, `isEmpty()`, `get()`, `set()`, `iterator()` и `listIterator()` извршавају се у константном времену. За додавање `n` елемената потребно је време `O(n)`. Све остале операције извршавају се у линеарном времену (грубо речено).

Сваки `ArrayList<>` има капацитет. Капацитет представља величину интерног низа који се користи за чување елемената листе. Капацитет је увек већи или једнак од броја елемената који се тренутно налазе у листи.
Капацитет се аутоматски повећава по потреби. Пре додавања великог броја елемената у листу, могуће је методом `ensureCapacity()` повећати капацитет, чиме би се смањила количина инкременталне реалокације.

Под структуралном модификацијом се подразумева додавање или брисање једног или већег броја елемената или експлицитно мењање величине интерног низа. Просто мењање вредности елемента не спада у структуралну модификацију.

Итератори које враћају методи `iterator()` и `listIterator()` су *„fail-fast“*: уколико се врши структурална модификација листе након креирања итератора, на било који други начин осим помоћу метода `add()` и `remove()` самог итератора, итератор избацује `ConcurrentModificationException`.

Следи попис метода ове класе (нису побројани сви, али сви нама важни јесу):

конструктори

ArrayList()
без аргумената, креира празну листу иницијалног капацитета 10.

```
ArrayList<String> imena = new ArrayList<>();  // zagradice <> su obavezne!
                        // prazna lista stringova (elementi su tipa String)
```

ArrayList(Collection<? extends E> c)
креира листу која садржи елементе дате колекције `c`, у поретку којим их враћа итератор за колекцију `c`.
Избацује изузетак типа NullPointerException ако је колекција null.

Collection<> је интерфејс из Јавине библиотеке, а имплементирају га (од оних које радимо на вежбама) класе: ArrayList<>, LinkedList<>, Stack<> и Vector<>, па се референца на објекат произвољне од њих може проследити као аргумент.

```
(пример је наведен ниже, у делу: "Конверзија низа у листу")
```

ArrayList(int initialCapacity)
креира празну листу задатог иницијалног капацитета.
Избацује изузетак типа IllegalArgumentException ако је задати капацитет негативан.

```
ArrayList<Integer> listaCelihBrojeva = new ArrayList<>(50);
                  // ako se zele cuvati vrednosti primitivnog tipa, za tip elemenata
                  // navodi se odgovarajuca omotac-klasa.
                  // Prazna lista celih brojeva, inicijalnog kapaciteta 50.
```

Методи за смештање објеката у листу

boolean add(E e)
додаје задати елемент на крај листе. Враћа true. Величина листе се увећава за 1.

```
imena.add("Milica");
```

void add(int index, E element)
на задату позицију у листи умеће задати елемент.
Елемент који је тренутно на тој позицији и сви иза њега (ако их има) шифтују се за једно место удесно (индекси им се увећавају за 1).
Избацује изузетак типа IndexOutOfBoundsException ако индекс није из одговарајућег опсега: index < 0 || index > size().

```
imena.add(0, "Pavle");
```

E set(int index, E element)
замењује елемент на задатој позицији у листи датим елементом. Враћа референцу на елемент који је претходно био на тој позицији.
Избацује изузетак типа IndexOutOfBoundsException ако индекс није из одговарајућег опсега: index < 0 || index >= size().

```
imena.set(0, "Sava");
```

boolean addAll(Collection<? extends E> c)
додаје све елементе задате колекције на крај листе, у поретку којим их враћа итератор за колекцију c.
Враћа true ако је листа промењена овим позивом. Избацује изузетак типа NullPointerException ако је задата колекција null.

boolean addAll(int index, Collection<? extends E> c)
почев од задате позиције у листи умеће све елементе задате колекције. Елемент који се претходно налазио на задатој позицији и сви иза њега (ако их има) шифтују се за одговарајући број места удесно (њихови индекси се повећавају). Нови елементи ће се појавити у листи оним редом којим их враћа итератор колекције c. Враћа true ако је листа промењена овим позивом.

Приступ елементима листе

E get(int index)
враћа елемент са задате позиције у листи.
Избацује изузетак типа IndexOutOfBoundsException ако индекс није из одговарајућег опсега: index < 0 || index >= size().

```
String ime = imena.get(1); // nakon gornjih naredbi, to je: "Milica"
```

Iterator<E> iterator()
враћа итератор за садржај листе

```
Iterator<String> iter = imena.iterator();
```

ListIterator<E> listIterator()
враћа лист итератор за садржај листе

```
ListIterator<String> listIter = imena.listIterator();
```

ListIterator<E> listIterator(int index)
враћа лист итератор за садржај листе, почев од задате позиције у листи. Задати индекс одређује први елемент који би био враћен иницијалним позивом метода next(). Иницијални позив метода previous() вратио би елемент чији је индекс за 1 мањи од задатог.
Избацује изузетак типа IndexOutOfBoundsException ако индекс није из одговарајућег опсега: index < 0 || index > size().

```
ListIterator<String> listIter1 = imena.listIterator(1);
```

List<E> subList(int fromIndex, int toIndex)
враћа део листе између задатих позиција, укључујући први и не укључујући другу. Ако су аргументи једнаки, враћена листа је празна.
Враћен је „поглед“ на текућу листу, па се неструктуралне измене над враћеном листом одражавају на полазну, и обрнуто.
Избацује изузетак типа IndexOutOfBoundsException ако индекси нису из одговарајућег опсега: fromIndex < 0 || toIndex > size(),
као и изузетак типа IllegalArgumentException ако индекси нису у одговарајућем поретку: fromIndex > toIndex.
List<> је интерфејс из Јавине библиотеке, а имплементирају га (од оних које радимо на вежбама) класе:
ArrayList<>, LinkedList<>, Stack<> и Vector<>.

```
imena.subList(from, to).clear();
      // uklanjanje elemenata sa indeksima iz opsega [from, to-1] iz liste "imena"

ListIterator<String> listIter2 = imena.subList(5, 15).listIterator(2);
      // dobijanje list iteratora za iteriranje kroz podlistu liste imena
      // koja se sastoji od elemenata sa pozicija [5, 14]
      // inicijalni poziv metoda next() iteratora vratice element sa pozicije 2 u
      // ovoj podlisti (tj. sa pozicije 7 u originalnoj listi)
```

Конверзија листе у низ

Object[] toArray()
враћа низ који садржи све елементе листе. Метод алоцира нови низ. Позивајући метод може модификовати враћени низ што неће утицати на оригиналну листу.

<T> T[] toArray(T[] a)
враћа низ који садржи све елементе листе. Тип враћеног низа биће тип низа који се приликом позива зада као аргумент. Уколико је низ прослеђен као аргумент одговарајуће величине (веће или једнаке од величине листе), он се враћа, а иначе се алоцира нови низ типа истог као задати и величине која одговара величини листе.
Уколико је низ прослеђен као аргумент веће дужине од листе, елемент који непосредно следи за елементима листе поставља се на null (то је корисно за одређивање дужине листе једино ако позивалац зна да листа не садржи null елементе).
Избацује изузетак типа ArrayStoreException ако тип низа који се зада као аргумент метода није исти као тип елемената листе или није његова натклаcа. Такође, избацује изузетак типа NullPointerException ако је задати низ null.

```
String[] podaci = imena.toArray(new String[imena.size()]);
```

Конверзија низа у листу

класа **java.util.Arrays** дефинише статички генерички метод **asList()**, који конвертује низ датог типа T у колекцију List<T>.

```
String[] ljudi = { "Pera", "Mika", "Laza", "Steva" };
List<String> imenaLista = java.util.Arrays.asList(ljudi);

// primer za konstruktor koji kao argument ocekuje postojecu kolekciju
ArrayList<String> imena1 = new ArrayList<>(java.util.Arrays.asList(ljudi));
```

Уклањање објеката из листе

E remove(int index)
уклања елемент са задате позиције у листи. Елементи иза се померају за по једно место улево (индекси им се смањују за 1).
Враћа референцу на уклоњени објекат. Избацује изузетак типа IndexOutOfBoundsException ако индекс није из одговарајућег опсега: index < 0 || index >= size().

```
String ime = imena1.remove(3);
```

boolean remove(Object o)
уклања прву појаву задатог елемента из листе, уколико се елемент појављује у листи. Уколико листа не садржи елемент, остаје непромењена.
Формалније, уклања се елемент са најмањим индексом i таквим да важи:
`o==null ? get(i)==null : o.equals(get(i))`
(ако је o null, уклања се први null елемент из листе, а иначе први за који метод equals() приликом поређења са o, врати true).
Неопходно је у класи елемената листе на погодан начин предефинисати метод equals()!
Враћа true ако листа садржи задати елемент (или, што је еквивалентно, ако је листа промењена овим позивом).

boolean removeAll(Collection<?> c)
уклања из листе све елементе који су садржани у задатој колекцији.
Враћа true ако је листа измењена овим позивом (бар 1 елемент је уклоњен).
Неопходно је у класи елемената листе на погодан начин предефинисати метод equals()!
Избацује изузетак типа ClassCastException ако тип елемента листе није компатибилан са типом задате колекције (опционо).
Такође, избацује изузетак типа NullPointerException ако листа садржи null елемент, а задата колекција не допушта null елементе (опционо) или је задата колекција null.

```
imena.removeAll(imena.subList(5,15));
      // uklonice elemente sa pozicija od 5 do 14, ukljucujuci, iz ArrayList<String>
      // objekta "imena", plus eventualne duplikate tih elemenata, ako postoje u listi
```

boolean retainAll(Collection<?> c)
задржава у листи само елементе који су садржани у задатој колекцији. Другим речима, уклања из листе све елементе који нису садржани у задатој колекцији.
Враћа true ако је листа измењена овим позивом (бар 1 елемент је уклоњен).
Неопходно је у класи елемената листе на погодан начин предефинисати метод equals()!
Избацује изузетак типа ClassCastException ако тип елемента листе није компатибилан са типом задате колекције (опционо).
Такође, избацује изузетак типа NullPointerException ако листа садржи null елемент, а задата колекција не допушта null елементе (опционо) или је задата колекција null.

clear()
уклања све елементе из листе. Величина се поставља на 0. Листа постаје празна.

boolean isEmpty()
враћа true ако листа не садржи елементе.
Листа може садржати само null референце, али то не значи да ће метод size() вратити 0 или метод isEmpty() вратити true.
Да бисмо испразнили листу, морамо заиста уклонити све елементе, а не само поставити их све на null.

Претрага листе

int indexOf(Object o)
враћа индекс прве појаве датог елемента у листи или -1 ако листа не садржи дати елемент.
// Формалније, враћа најмањи индекс i такав да важи: o==null ? get(i)==null : o.equals(get(i)) или -1 ако такав индекс не постоји.
Неопходно је у класи елемената листе на погодан начин предефинисати метод equals()!

int lastIndexOf(Object o)
враћа индекс последње појаве датог елемента у листи или -1 ако листа не садржи дати елемент.
// Формалније, враћа највећи индекс i такав да важи: o==null ? get(i)==null : o.equals(get(i)) или -1 ако такав индекс не постоји.
Неопходно је у класи елемената листе на погодан начин предефинисати метод equals()!

boolean contains(Object o)
враћа true ако листа садржи дати елемент.
// Формалније, враћа true акко листа садржи бар један елемент е такав да је: o==null ? e==null : o.equals(e).
Неопходно је у класи елемената листе на погодан начин предефинисати метод equals()!

boolean containsAll(Collection<?> c)
враћа true ако листа садржи све елементе дате колекције.
Неопходно је у класи елемената листе на погодан начин предефинисати метод equals()!

Капацитет и величина листе

void ensureCapacity(int minCapacity)
повећава капацитет текуће листе, ако је потребно, тако да се у њу може сместити бар minCapacity елемената.

int size()
враћа број елемената у листи. Јасно, број елемената је увек мањи или једнак од тренутног капацитета.

void trimToSize()
капацитет се поставља на тренутну величину листе.
Може се користити како би се минимизовала меморија коју листа заузима.

Стринг-репрезентација листе

toString()
враћа String-репрезентацију текуће листе у истом облику као и метод Arrays.toString().

Не очекује се да се било шта од овога учи напамет, већ да се користи помоћ коју Eclipse свесрдно пружа након што се уради неко од подешавања из упутства **Attach Source... Source Attachment... eclipse.ini.pdf**