

# Napredno programiranje i programski jezici

08 Java

Fakultet tehničkih nauka, Novi Sad

23-24/Z

Dunja Vrbaški

```
public interface Logger {  
    void log();  
    void logShort();  
    void logExtended();  
    void logCrypto();  
}
```

### Imenovanje:

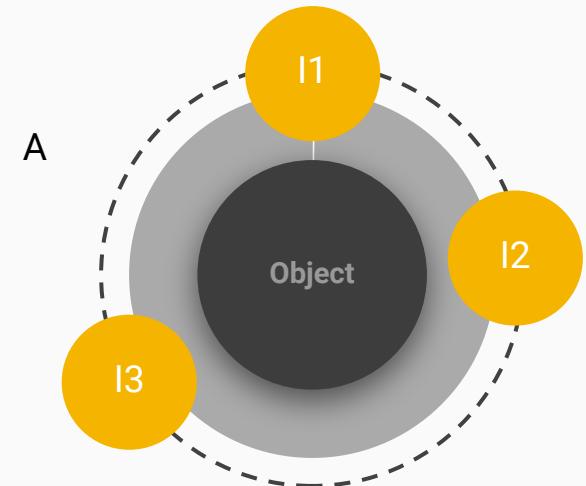
- jezik (eng, srp) get, set, print...
- Logger vs ILogger (StudentClass vs Student)
- velika vs mala slova
- konvencije (opšte > kompanija > projekat > personalne?)

```
public interface Interfjes1 {  
    void metod11(...);  
    int metod12(...);  
    ...  
    void metod1N(...);  
}
```

```
public interface Interfjes2 {  
    void metod21(...);  
    void metod22(...);  
    ...  
    double metod2M(...);  
}
```

```
public interface Interfjes3 {  
    void metod31(...);  
    Object metod32(...);  
    ...  
    String metod3K(...);  
}
```

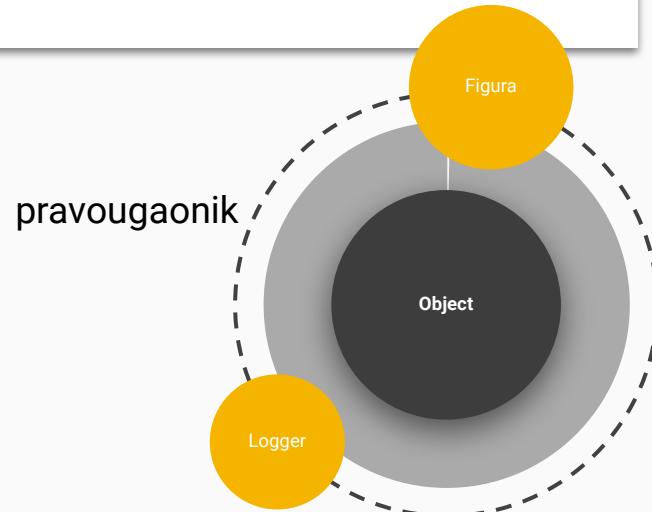
```
public class A implements Interfejs1, Interfejs2, Interfejs3 {  
    public void metod11() {...}  
    public int metod12() {...}  
    ...  
    String void metod3K() {...}  
}
```



```
public interface Logger {  
    void log();  
}
```

```
public interface Figura {  
    double getO();  
    double getP();  
}
```

```
public class Pravougaonik implements Figura, Logger {  
    // polja, ctors  
  
    public double getO {...}  
    public double getP() {...}  
    public void log() {...}  
    ...  
    public String toString()  
}
```



```
public interface Logger {  
    void log();  
}
```

```
public interface Figura {  
    double getO();  
    double getP();  
}
```

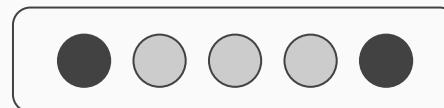
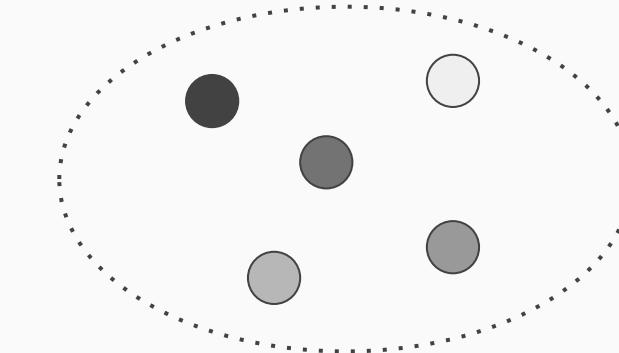
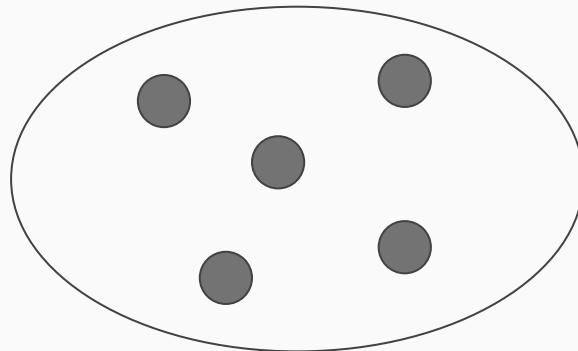
```
public class Pravougaonik implements Figura, Logger {  
    // polja, ctors  
    @Override  
    public String toString() {  
        String str = "";  
        str += "a = " + a + " b = " + b;  
        str += " P = " + getP() + " O = " + getO();  
        return str;  
    }  
    @Override  
    public double getO() {  
        return 2 * (a + b);  
    }  
    @Override  
    public double getP() {  
        return a * b;  
    }  
    @Override  
    public void log() {  
        String str = "LOG: ";  
        str += LocalDateTime.now().getDayOfMonth() + ".";  
        str += LocalDateTime.now().getMonthValue() + ".";  
        str += LocalDateTime.now().getYear() + " ";  
        str += "Pravougaonik: "; str += this;  
        System.out.println(str);  
    }  
}
```

```
public interface Logger {  
    void log();  
}
```

```
public interface Figura {  
    double getO();  
    double getP();  
}
```

```
public class Pravougaonik implements Figura, Logger {  
    // polja, ctors  
    @Override  
    public String toString() {  
        String str = "";  
        str += "a = " + a + " b = " + b;  
        str += " P = " + getP() + " O = " + getO();  
        return str;  
    }  
    @Override  
    public double getO() {  
        return 2 * (a + b);  
    }  
    @Override  
    public double getP() {  
        return a * b;  
    }  
    @Override  
    public void log() {  
        String str = "LOG: ";  
        str += LocalDateTime.now().getDayOfMonth() + ".";  
        str += LocalDateTime.now().getMonthValue() + ".";  
        str += LocalDateTime.now().getYear() + " ";  
        str += "Pravougaonik: "; str += this;  
        System.out.println(str);  
    }  
}
```

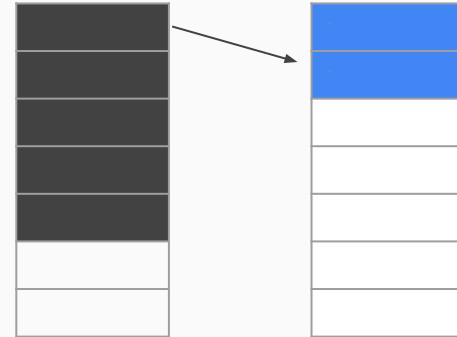
## Kolekcije objekata



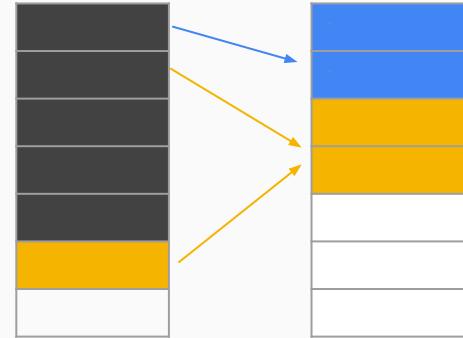
...

```
public static void main(String[] args) {  
    Pravougaonik[] nizP = new Pravougaonik[5];  
    nizP[0] = new Pravougaonik(0, 0);  
    System.out.println(nizP[0]);  
}
```

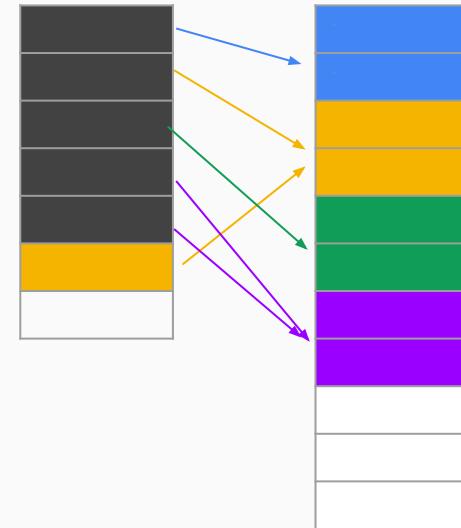
```
public static void main(String[] args) {  
  
    Pravougaonik[] nizP = new Pravougaonik[5];  
  
    nizP[0] = new Pravougaonik(0, 0);  
    System.out.println(nizP[0]);  
  
}
```



```
public static void main(String[] args) {  
  
    Pravougaonik[] nizP = new Pravougaonik[5];  
  
    nizP[0] = new Pravougaonik(0, 0);  
  
    Pravougaonik p = new Pravougaonik(1, 1);  
    nizP[1] = p;  
}
```



```
public static void main(String[] args) {  
  
    Pravougaonik[] nizP = new Pravougaonik[5];  
  
    nizP[0] = new Pravougaonik(0, 0);  
  
    Pravougaonik p = new Pravougaonik(1, 1);  
    nizP[1] = p;  
  
    nizP[2] = new Pravougaonik(2, 2);  
    nizP[3] = new Pravougaonik(3, 3);  
  
    nizP[4] = nizP[3];  
}
```



```
public static void main(String[] args) {  
    Pravougaonik[] nizP = new Pravougaonik[5];  
    nizP[0] = new Pravougaonik(0, 0);  
  
    Pravougaonik p = new Pravougaonik(1, 1);  
    nizP[1] = p;  
  
    nizP[2] = new Pravougaonik(2, 2);  
    nizP[3] = new Pravougaonik(3, 3);  
  
    nizP[4] = nizP[3];  
  
    printNiz(nizP);  
}
```

```
private static void printNiz(Pravougaonik[] niz) {  
    for(int i = 0; i < niz.length; i++) {  
        System.out.println(niz[i]);  
    }  
}
```

```
a = 0.0 b = 0.0 P = 0.0 O = 0.0  
a = 1.0 b = 1.0 P = 1.0 O = 4.0  
a = 2.0 b = 2.0 P = 4.0 O = 8.0  
a = 3.0 b = 3.0 P = 9.0 O = 12.0  
a = 3.0 b = 3.0 P = 9.0 O = 12.0
```

```
public static void main(String[] args) {  
    Pravougaonik[] nizP = new Pravougaonik[5];  
    nizP[0] = new Pravougaonik(0, 0);  
  
    Pravougaonik p = new Pravougaonik(1, 1);  
    nizP[1] = p;  
  
    nizP[2] = new Pravougaonik(2, 2);  
    nizP[3] = new Pravougaonik(3, 3);  
    nizP[4] = nizP[3];  
  
    printNiz(nizP);  
}
```

```
private static void printNiz(Pravougaonik[] niz) {  
    for(int i = 0; i < niz.length; i++) {  
        System.out.println(niz[i]);  
    }  
}
```

```
private static void printNiz(Pravougaonik[] niz) {  
    if (niz == null)  
    {  
        System.out.println("Prazan niz");  
        return;  
    }  
  
    for(int i = 0; i < niz.length; i++) {  
        if (niz[i] == null)  
            System.out.println(" ");  
        else  
            System.out.println(niz[i]);  
    }  
}
```

```
public static void main(String[] args) {
    Pravougaonik[] nizP = new Pravougaonik[5];

    nizP[0] = new Pravougaonik(0, 0);
    nizP[1] = new Pravougaonik(1, 1);
    nizP[2] = new Pravougaonik(2, 2);
    nizP[3] = new Pravougaonik(3, 3);
    nizP[4] = new Pravougaonik(4, 4);
    printNiz(nizP);
}
```

```
public static void main(String[] args) {
    Pravougaonik[] nizP = new Pravougaonik[5];

    nizP[0] = new Pravougaonik(0, 0);
    nizP[1] = new Pravougaonik(1, 1);
    nizP[2] = new Pravougaonik(2, 2);
    nizP[3] = new Pravougaonik(3, 3);
    nizP[4] = new Pravougaonik(4, 4);

    printNiz(nizP);
}

private static void promeniNiz(Pravougaonik[] niz, int index) {
    // TODO provere

    niz[index].setA(niz[index].getA() + 100);
}
```

```
public static void main(String[] args) {
    Pravougaonik[] nizP = new Pravougaonik[5];

    nizP[0] = new Pravougaonik(0, 0);
    nizP[1] = new Pravougaonik(1, 1);
    nizP[2] = new Pravougaonik(2, 2);
    nizP[3] = new Pravougaonik(3, 3);
    nizP[4] = new Pravougaonik(4, 4);

    promeniNiz(nizP, 1);
    printNiz(nizP);
}

private static void promeniNiz(Pravougaonik[] niz, int index) {
    // TODO provere

    niz[index].setA(niz[index].getA() + 100);
}
```

```

public static void main(String[] args) {
    Pravougaonik[] nizP = new Pravougaonik[5];

    nizP[0] = new Pravougaonik(0, 0);

    Pravougaonik p = new Pravougaonik(1, 1);
    nizP[1] = p;

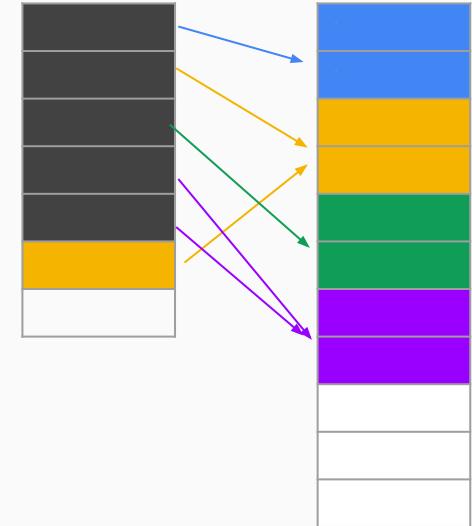
    nizP[2] = new Pravougaonik(2, 2);
    nizP[3] = new Pravougaonik(3, 3);

    nizP[4] = nizP[3];

    promeniNiz(nizP, 1);
    printNiz(nizP);
}

private static void promeniNiz(Pravougaonik[] niz, int index) {
    // TODO provere
    niz[index].setA(niz[index].getA() + 100);
}

```



```

public static void main(String[] args) {
    Pravougaonik[] nizP = new Pravougaonik[5];

    nizP[0] = new Pravougaonik(0, 0);

    Pravougaonik p = new Pravougaonik(1, 1);
    nizP[1] = p;

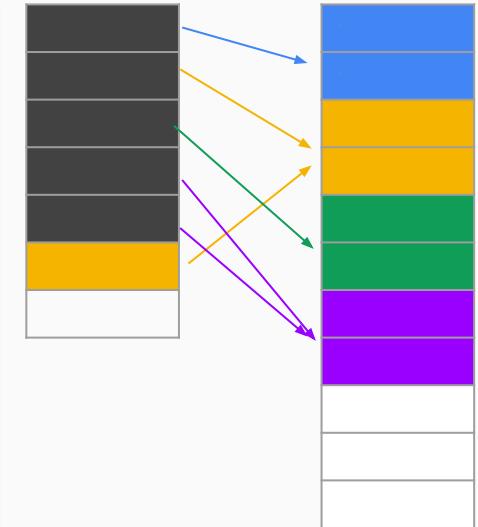
    nizP[2] = new Pravougaonik(2, 2);
    nizP[3] = new Pravougaonik(3, 3);

    nizP[4] = nizP[3];

    promeniNiz(nizP, 3);
    printNiz(nizP);
}

private static void promeniNiz(Pravougaonik[] niz, int index) {
    // TODO provere
    niz[index].setA(niz[index].getA() + 100);
}

```



```
public static void main(String[] args) {  
    Pravougaonik[] nizP = new Pravougaonik[5];  
  
    nizP[0] = new Pravougaonik(0, 0);  
  
    Pravougaonik p = new Pravougaonik(1, 1);  
    nizP[1] = p;  
  
    nizP[2] = new Pravougaonik(2, 2);  
    nizP[3] = new Pravougaonik(3, 3);  
  
    nizP[4] = nizP[3];  
  
    printNiz(nizP);  
    logNiz(nizP);  
}
```

```
private static void printNiz(Pravougaonik[] niz) {  
    ...  
    System.out.println(niz[i]);  
}
```

```
private static void logNiz(Pravougaonik[] niz) {  
    ...  
    niz[i].log();  
}
```

```
public static void main(String[] args) {
    Pravougaonik[] nizP = new Pravougaonik[5];

    nizP[0] = new Pravougaonik(0, 0);

    Pravougaonik p = new Pravougaonik(1, 1);
    nizP[1] = p;

    nizP[2] = new Pravougaonik(2, 2);
    nizP[3] = new Pravougaonik(3, 3);

    nizP[4] = nizP[3];

    printNiz(nizP);
    logNiz(nizP);
}
```

```
private static void printNiz(Pravougaonik[] niz) {
    ...
    System.out.println(niz[i]);
}

private static void logNiz(Pravougaonik[] niz) {
    System.out.println("Trenutno stanje niza ");
    System.out.println(LocalDateTime.now());
    if (niz == null)
    {
        System.out.println("Prazan niz");
        return;
    }

    for(int i = 0; i < niz.length; i++) {
        if (niz[i] == null)
            System.out.println(i + " - bez vrednosti ");
        else
            niz[i].log();
    }
}
```

```
public static void main(String[] args) {  
    Pravougaonik[] nizP = new Pravougaonik[5];  
  
    nizP[0] = new Pravougaonik(0, 0);  
  
    Pravougaonik p = new Pravougaonik(1, 1);  
    nizP[1] = p;  
  
    nizP[2] = new Pravougaonik(2, 2);  
    nizP[3] = new Pravougaonik(3, 3);  
  
    nizP[4] = nizP[3];  
  
    printNiz(nizP);  
    logNiz(nizP);  
}
```

```
private static void logNiz(Pravougaonik[] niz) {  
  
    private static void logNiz(Logger[] niz) {  
        System.out.println("Trenutno stanje niza ");  
        System.out.println(LocalDateTime.now());  
        if (niz == null)  
        {  
            System.out.println("Prazan niz");  
            return;  
        }  
  
        for(int i = 0; i < niz.length; i++) {  
            if (niz[i] == null)  
                System.out.println(i + " - bez vrednosti ");  
            else  
                niz[i].log();  
        }  
    }  
}
```

```
public static void main(String[] args) {  
    Pravougaonik[] nizP = new Pravougaonik[5];  
  
    nizP[0] = new Pravougaonik(0, 0);  
  
    Pravougaonik p = new Pravougaonik(1, 1);  
    nizP[1] = p;  
  
    nizP[2] = new Pravougaonik(2, 2);  
    nizP[3] = new Pravougaonik(3, 3);  
  
    nizP[4] = nizP[3];  
  
    printNiz(nizP);  
    logNiz(nizP);  
  
    Logger[] nizL = new Logger[5];  
    nizL[0] = nizP[2];  
}  
}
```

```
private static void logNiz(Pravougaonik[] niz) {  
  
    private static void logNiz(Logger[] niz) {  
        System.out.println("Trenutno stanje niza ");  
        System.out.println(LocalDateTime.now());  
        if (niz == null)  
        {  
            System.out.println("Prazan niz");  
            return;  
        }  
  
        for(int i = 0; i < niz.length; i++) {  
            if (niz[i] == null)  
                System.out.println(i + " - bez vrednosti ");  
            else  
                niz[i].log();  
        }  
    }  
}
```

```
public class Pravougaonik implements Figura, Logger {  
...}  
  
public class Krug implements Figura, Logger {  
...}  
  
public class Trougao implements Figura, Logger {  
...}
```

```
public static void main(String[] args) {  
  
    Logger[] nizL = new Logger[5];  
    nizL[0] = new Pravougaonik(0, 0);  
    nizL[1] = new Krug(1);  
    logNiz(nizL);  
}
```

```
public interface Logger {  
    void log();  
}
```

```
public interface Figura extends Logger {  
    double getO();  
    double getP();  
}
```

```
public class Pravougaonik implements Figura {  
    ...}  
    ...}
```

```
public class Krug implements Figura {  
    ...}  
    ...}
```

```
public class Trougao implements Figura {  
    ...}  
    ...}
```

```
public interface Logger {  
    void log();  
}
```

```
public abstract class Figura implements Logger  
{  
    public String naziv;  
  
    Figura() {  
        log();  
    }  
  
    abstract double getO();  
    abstract double getP();  
}
```

```
public class Pravougaonik extends Figura {  
...}  
  
public class Krug extends Figura {  
...}  
  
public class Trougao extends Figura {  
...}
```

```
public interface Logger {  
    void log();  
}
```

```
public abstract class Figura implements Logger  
{  
    public String naziv;  
  
    Figura(){  
        log();  
    }  
  
    abstract double getO();  
    abstract double getP();  
}
```

```
public class Pravougaonik extends Figura {  
...}  
...}
```

```
public class Krug extends Figura {  
...}
```

```
public class Trougao extends Figura {  
...}
```

```
public static void main(String[] args) {  
  
    Logger[] nizL = new Logger[5];  
    nizL[0] = new Pravougaonik(0, 0);  
    nizL[1] = new Krug(1);  
    logNiz(nizL);  
  
    Figura[] nizF = ...  
}
```

?

I jedno i drugo se može navesti kao tip

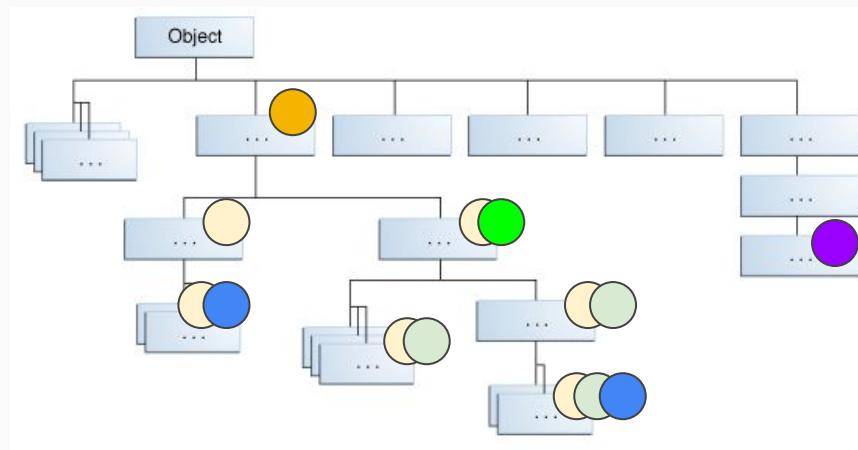
Ni jedno ni drugo se ne može instancirati

Zahtevaju konkretnе implementacije kao instance

```
public static void main(String[] args) {  
    Logger l...  
    Figura f...  
    Logger[] nizL...  
    Figura[] nizF..  
    ...  
}  
  
void metod(Logger l)...  
void metod(Figura f)...  
void metod(Logger[] l)...  
void metod(Figura[] f)...
```

interfejs - ponašanja

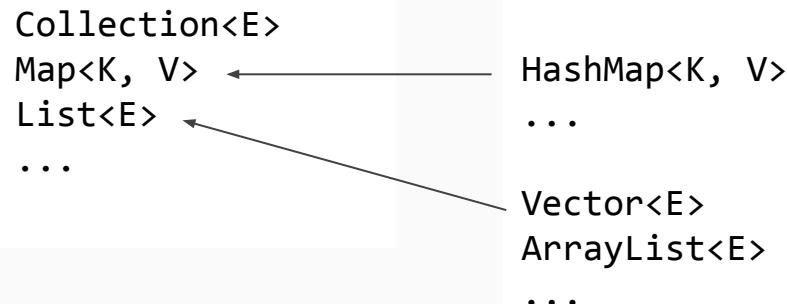
abstract class - struktura, hijerarhija, koncept



## Interfejs

- suština: sve metode public abstract
- još neke elemente može da ima

Clonable  
Readable  
Comparable  
Observable  
...



```
package nppj;

import java.util.ArrayList;

public class KolekcijeTest {

    public static void main(String[] args) {
        ArrayList<Osoba> osobe = new ArrayList<Osoba>();
    }
}
```

**Module** java.base

**Package** java.util

## **Class ArrayList<E>**

```
java.lang.Object
    java.util.AbstractCollection<E>
        java.util.AbstractList<E>
            java.util.ArrayList<E>
```

### **Type Parameters:**

E - the type of elements in this list

### **All Implemented Interfaces:**

Serializable, Cloneable, Iterable<E>, Collection<E>, List<E>, RandomAccess

### **Direct Known Subclasses:**

AttributeList, RoleList, RoleUnresolvedList

---

```
public class ArrayList<E>
extends AbstractList<E>
implements List<E>, RandomAccess, Cloneable, Serializable
```

<https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/ArrayList.html>

**Module** java.base

**Package** java.util

## **Class ArrayList<E>**

```
java.lang.Object
    java.util.AbstractCollection<E>
        java.util.AbstractList<E>
            java.util.ArrayList<E>
```

### **Type Parameters:**

E - the type of elements in this list

### **All Implemented Interfaces:**

Serializable, Cloneable, Iterable<E>, Collection<E>, List<E>, RandomAccess

### **Direct Known Subclasses:**

AttributeList, RoleList, RoleUnresolvedList

---

```
public class ArrayList<E>
extends AbstractList<E>
implements List<E>, RandomAccess, Cloneable, Serializable
```

**Module** java.base  
**Package** java.util

## Interface List<E>

**Type Parameters:**

E - the type of elements in this list

**All Superinterfaces:**

Collection<E>, Iterable<E>

**All Known Implementing Classes:**

AbstractList, AbstractSequentialList, ArrayList, AttributeList, CopyOnWriteArrayList, LinkedList, RoleList, RoleUnresolvedList, Stack, Vector



```
package nppj;

import java.util.ArrayList;

public class KolekcijeTest {

    public static void main(String[] args) {

        ArrayList<Osoba> osobe = new ArrayList<Osoba>();
        ...
    }
}
```



Niz je ograničena struktura  
Koliko elemenata može da stane u ovakvu listu?



`size()`

`capacity  
(private)`

`void ensureCapacity(...)`

```
template <class T>
class List {
private:
...
public:
    List()...
    List(const List<T>&)...

    virtual ~List()...

    List<T>& operator=(const List<T>&)...

    int size() const ...
    bool empty() const ...

    bool add(int, const T&) ...
    bool remove(int) ...
    bool read(int, T&)const ...
    void clear() ...
};

template <class T>
ostream& operator<<(ostream& out, const List<T>& rl)...
```

```
class ArrayList<E>

public ArrayList()...
public ArrayList(int initialCapacity)...
// jos jedan ctor
// nema destruktora

// nema preklapanja, = je dodela referenci

int size() ...
boolean isEmpty() ...

bool add(int index, E element) ... // dodatne add metode
bool remove(int index) ... // dodatne remove metode
E get(int index) ... // exception za pogresan indeks
void clear() ...

// nema preklapanja
```

```
int size() ...
boolean isEmpty() ...

bool add(int index, E element) ... // dodatne add metode
bool remove(int index) ... // dodatne remove metode
E get(int index) ... // exception za pogresan indeks
void clear() ...
```

Metode deklarisane još u interfejsu List<E>

Sve implementacije ovog interfejsa implementiraju metode na svoj, odgovarajući način

Ako implementiraš interfejs List<E> znači da sigurno imaš ove metode

Mogu da te posmatram kao listu

ArrayList nasleđuje klasu AbstractList<E> i implementira interfejs List<E>

## C++ (naša lista)

```
int size() const {
    return noEl;
}

bool empty() const {
    return head == NULL ? 1 : 0;
}

template <class T>
void List<T>::clear() {
    while(!empty())
        remove(1);
}
```

## Java ArrayList

```
public int size() {
    return size;
}

public boolean isEmpty() {
    return size == 0;
}

public void clear() {
    modCount++;
    final Object[] es = elementData;
    for (int to = size, i = size = 0; i < to; i++)
        es[i] = null;
}
```

Ilustracija, nije neophodno ni potrebno da znate implementaciju klase koje ćemo koristiti u Javi

<https://github.com/openjdk/jdk/tree/master/src/java/util/src/share/classes/java/util>  
ili src.zip na disku

```
public static void main(String[] args) {  
  
    ArrayList<Osoba> osobe = new ArrayList<Osoba>();  
  
    osobe.add(new Osoba("Hermione", "Granger", "111111"));  
    osobe.add(new Osoba("Harry", "Potter", "222222"));  
    osobe.add(new Osoba("Ron", "Weasley", "333333"));  
  
    for (int i = 0; i < osobe.size(); i++) {  
        System.out.println(osobe.get(i));  
    }  
}
```

```
public static void main(String[] args) {  
  
    ArrayList<Osoba> osobe = new ArrayList<Osoba>();  
  
    osobe.add(new Osoba("Hermione", "Granger", "111111"));  
    osobe.add(new Osoba("Harry", "Potter", "222222"));  
    osobe.add(new Osoba("Ron", "Weasley", "333333"));  
  
    for (int i = 0; i < osobe.size(); i++) {  
        System.out.println(osobe.get(i));  
    }  
  
    for (Osoba x: osobe) {  
        System.out.println(x);  
    }  
}
```

```
for (Osoba osoba: osobe) {  
    System.out.println(osoba);  
}
```

```
public static void main(String[] args) {  
  
    ArrayList<Osoba> osobe = new ArrayList<Osoba>();  
  
    osobe.add(new Osoba("Hermione", "Granger", "111111"));  
    osobe.add(new Osoba("Harry", "Potter", "222222"));  
    osobe.add(new Osoba("Ron", "Weasley", "333333"));  
  
    for (int i = 0; i < osobe.size(); i++) {  
        System.out.println(osobe.get(i));  
    }  
  
    for (Osoba x: osobe) {  
        System.out.println(x);  
    }  
  
    System.out.println(osobe);  
}  
  
Hermione Granger 111111  
Harry Potter 222222  
Ron Weasley 333333  
Hermione Granger 111111  
Harry Potter 222222  
Ron Weasley 333333  
[Hermione Granger 111111 , Harry Potter 222222 , Ron Weasley 333333 ]
```

```
ArrayList<Osoba> osobe = new ArrayList<Osoba>();  
System.out.println(osobe);
```

```
[Hermione Granger 111111 , Harry Potter 222222 , Ron Weasley 333333 ]
```

Koristi `toString` metodu kolekcije koja za svaki element poziva `toString` metodu Osobe

```
public static void main(String[] args) {  
  
    ArrayList<Osoba> osobe = new ArrayList<Osoba>();  
  
    osobe.add(new Osoba("Hermione", "Granger", "111111"));  
    osobe.add(new Osoba("Harry", "Potter", "222222"));  
    osobe.add(new Osoba("Ron", "Weasley", "333333"));  
    System.out.println(osobe);  
}
```

```
public static void main(String[] args) {  
  
    ArrayList<Osoba> osobe = new ArrayList<Osoba>();  
  
    osobe.add(new Osoba("Hermione", "Granger", "111111"));  
    osobe.add(new Osoba("Harry", "Potter", "222222"));  
    osobe.add(new Osoba("Ron", "Weasley", "333333"));  
    System.out.println(osobe);  
  
    osobe.add(2, new Osoba("Draco", "Malfoy", "444444"));  
    System.out.println(osobe);  
}
```

```
public static void main(String[] args) {  
  
    ArrayList<Osoba> osobe = new ArrayList<Osoba>();  
  
    osobe.add(new Osoba("Hermione", "Granger", "111111"));  
    osobe.add(new Osoba("Harry", "Potter", "222222"));  
    osobe.add(new Osoba("Ron", "Weasley", "333333"));  
    System.out.println(osobe);  
  
    osobe.add(2, new Osoba("Draco", "Malfoy", "444444"));  
    System.out.println(osobe);  
  
    osobe.remove(2);  
    System.out.println(osobe);  
}
```

```
public static void main(String[] args) {  
  
    ArrayList<Osoba> osobe = new ArrayList<Osoba>();  
  
    osobe.add(new Osoba("Hermione", "Granger", "111111"));  
    osobe.add(new Osoba("Harry", "Potter", "222222"));  
    osobe.add(new Osoba("Ron", "Weasley", "333333"));  
    System.out.println(osobe);  
  
    osobe.add(2, new Osoba("Draco", "Malfoy", "444444"));  
    System.out.println(osobe);  
  
    osobe.remove(2);  
    System.out.println(osobe);  
  
    Osoba osoba0 = osobe.get(0);  
    System.out.println(osoba0);  
  
    osobe.remove(osoba0);  
    System.out.println(osobe);  
}
```