(3)

Suprascriere → redefinirea unei metode care este deja definită într-una din superclase.
(și reimplementare)

```
public class Instrument {
    public void canta() {
        System.out.printf("
            Instrumentul cântă!");
    }
}
```

```
public class Chitara extends Instrument {
    @Override
    public void canta() {
        System.out.printf("Chitara cant
            în loc de ");
        super.canta();
    }
}
```

```
Instrument i = new Instrument();
i.canta();
```

```
Chitara c = new Chitara();
c.canta();
Instrument i2 = new Chitara();
i2.canta();
```

```
char c;
c = System.in.read();
Instrument i = null;
if(c == 'i') {
    i = new Instrument();
} else {
    i = new Chitara();
}
i.canta();
```

// exista canta() in clasa Instrument
↳ compile time

↳ runtime → Chitara cântă

i = new Chitara();

this → referință la obiectul curent pentru accesul la membrii definiți în clasa curentă (this.make);
→ apel de constructor al clasei curente din alt constructor (el trebuie să fie pe prima linie din constructor)

```
public Car(){
        this("Dacia", "Logan", 1400);
}
```

---

super → referință la obiectul curent pentru accesul la membrii definiți în superclasă (super.toString())
→ apel de constructor al superclasei dintr-un constructor al clasei curente (și el trebuie să fie pe prima linie)

```
public SportsCar(String make, — ) {
        super(make, — );
```

```
class Elipsa {
    float rm;
    float rM;

void setrm(float f){
    rm = f;
}

- - - - -
float getrm(){
    return rm;
}
- - - - -
}
```

```
class Cerc extends Elipsa {
    @Override
    void setrm(float f){
        rm = f;
        rM = f;
    }
    @Override
    void setrM(float f){
        rm = f;
        rM = f;
    }
}
```

Nu respectă principiul substituției.

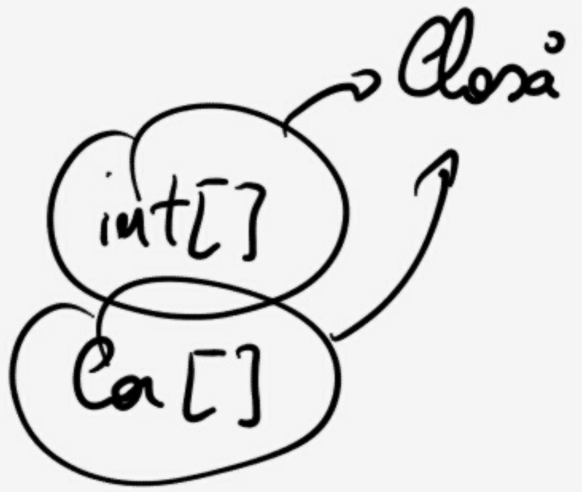
Vectori

- vectorii sunt obiecte

```
int[]  v; // v == null
v = new int[10];
```

↳ nu este un constructor

```
Car[]  carArray = new Car[20];
carArray[0] = new Car();
for(int i=1; i<20; i++){
        carArray[i] = new Car();
}
```



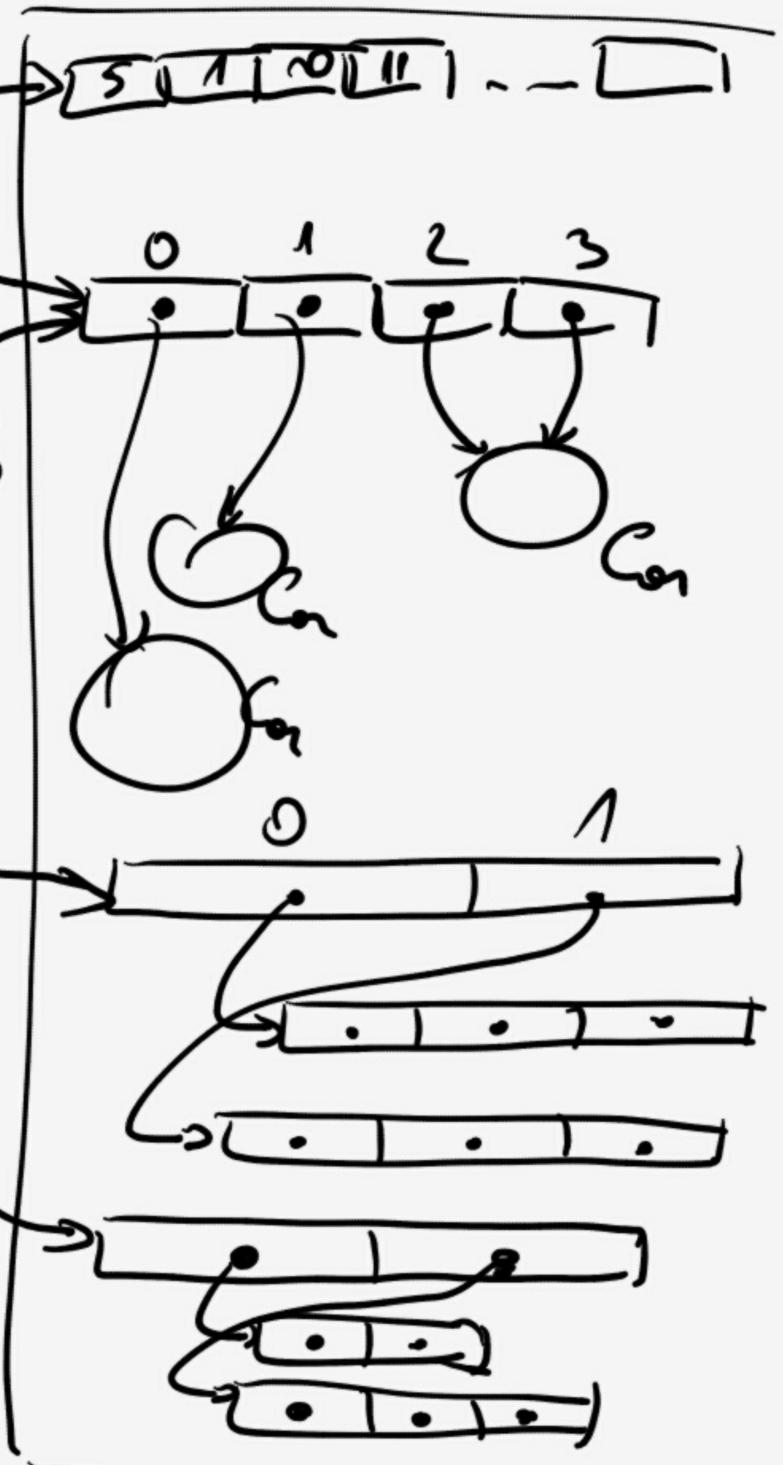
carArray.length == 20

In clasele de tip vector (tip[]) există un câmp read-only, numit „length" care memorează numărul de elemente din vector.

```
int[] myArray = - - - ;
for(i=0; i< myArray.length; i++) System.out.printf("%d, ",
                                                   myArray[i]);
for (int valoare : myArray) System.out.printf("%d, ", valoare);
```

int[][]

```
int[] v = new int[10];

v[0] = 5;
v[1] = 1;
- - -

Con[] c = new Con[10];
c[2] = new Con();
c[3] = c[2];

Con[][] cm = new Con[2][3];

Con[][] cm2 = new Con[2][];
cm2[0] = new Con[2];
cm2[1] = new Con[3];

Con[] c2 = c;
```



```
Con[][] cm2 = new Con[2][];
cm2[0] = new Con[2];
cm2[1] = new Con[3];
```

cm2.length == 2

cm2[0].length == 2

cm2[1].length == 3

~~cm2[2].length~~

↳ Exceptie