②

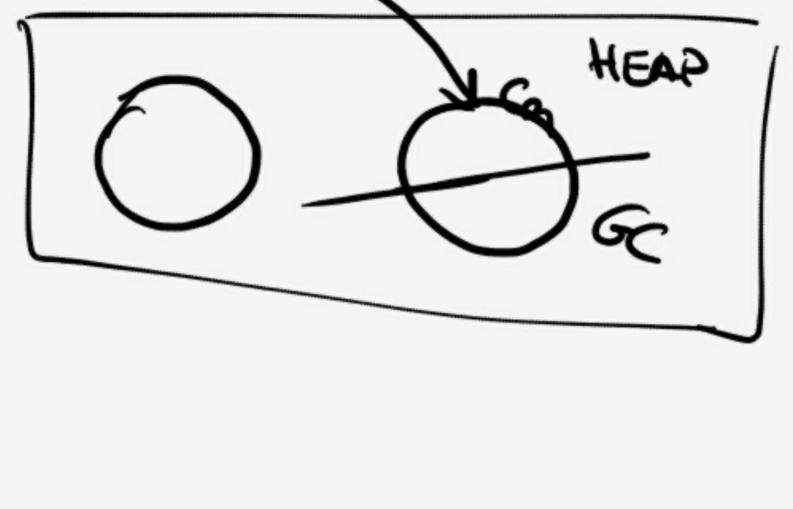
```
public class Car {
    private String make;
    private String model;
    private int cc;

    public Car(String _make,
               String _model,
               int cc) {
        make = _make;
        model = _model;
        this.cc = cc;
    }

    public String getMake(){
        return make;
    }
}
```

compilatorul → super();

Obiectele se creează folosind operatorul **new**.

```
Car c;
c = new Car();
c.make = "Dacia";
c.model = "Logan";
c.cc = 1400;
```

```
Car c2;
c2 = new Car("Dacia",
             "Logan", 1400);
```



Referință la obiectul curent.

```
Car(String make, String model){
    this.super();
    this.make = make;
    this.model = model;
    cc = -1;
}
```

//

```
this(make, model, -1);
     String String int
```

Regula: nu pot exista 2 sau mai multi constructori cu acelasi tip de argumente in aceeasi ordine.

Supraîncărcare → mai multe metode cu același nume, dar argumente diferite

Cuvântul cheie "this" poate fi folosit în 2 moduri:

① referință la obiectul curent

② apel de constructor din alt constructor

Modificatori de acces

public → accesibil de oriunde

protected → pt. câmpuri și metode → doar în clasa curentă, în clasele derivate, în interiorul pachetului

<def.> → aplicabil oricărui element → în clasa curentă și în interiorul pachetului

private → pt. câmpuri și metode → doar în clasa curentă

String

```
String a = "Gigel";
String b = " a lecat la piață și a cumpărat";
String c = a + b + 20 + " kg de mere";
System.out.printf("%s\n", c);
String d = String.valueOf(20) + a + b + --;
```

Mostenire

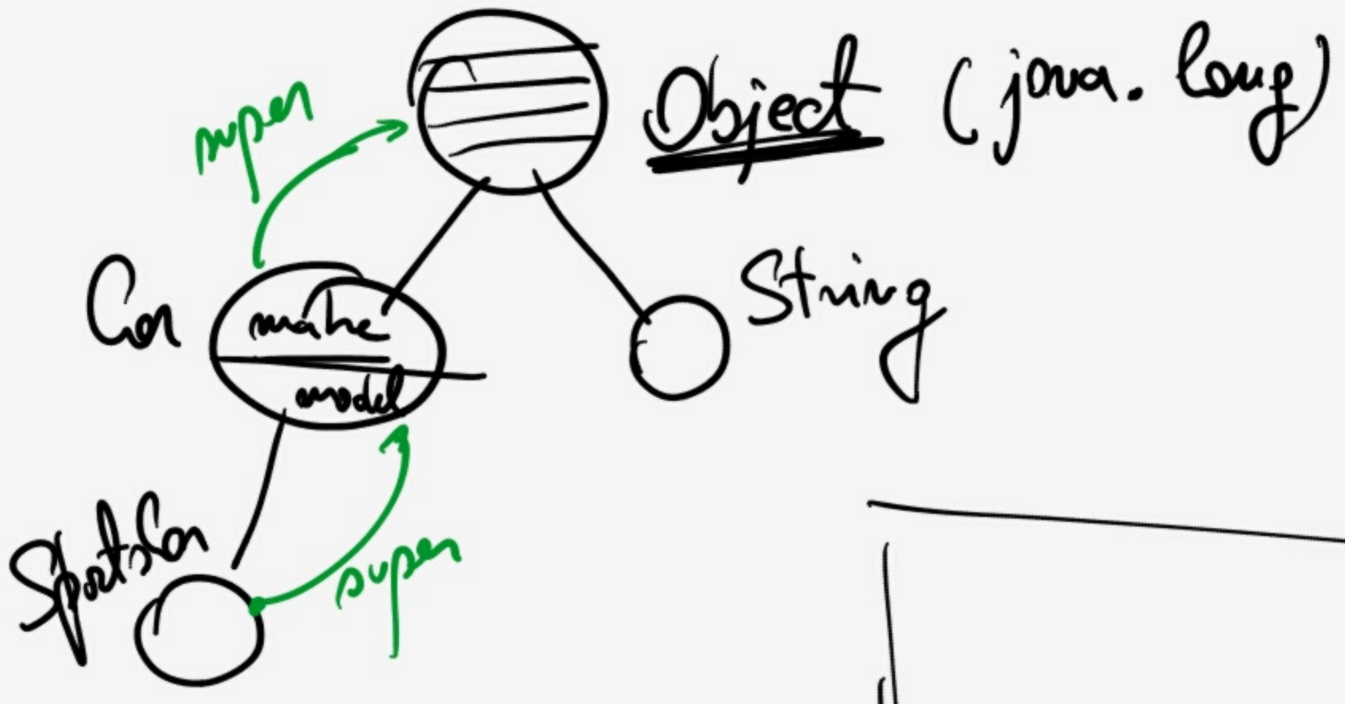
```
public class SportsCar extends Car {
    public SportsCar(String make,
                     String model, int cc) {
        super(make, model, cc);
    }
}
```

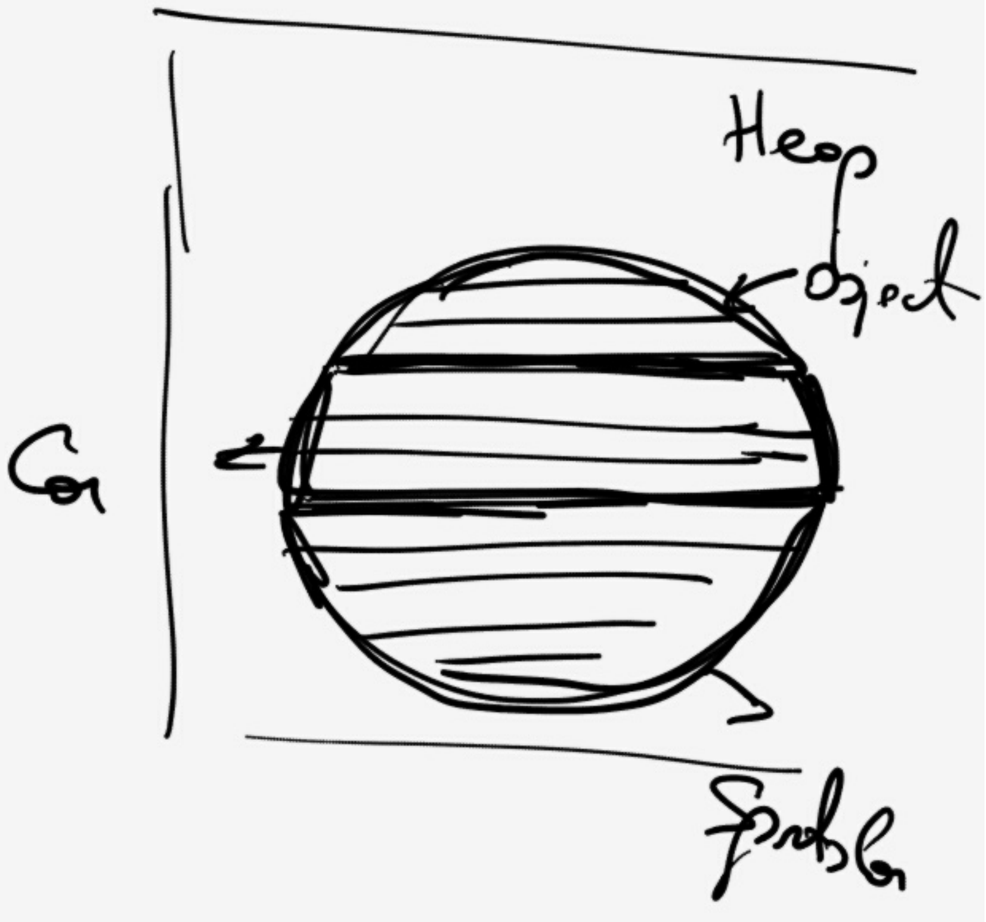
Primul lucru din-un constructor, este obligatoriu un apel de alt constructor

În procesul de extindere, se mostenesc toți membrii clasei de bază, dar sunt accesibili doar membrii publici, protejați, exclusiv constructorii

Java suportă doar mostenirea simplă. În Java, o clasă obligatoriu extinde o singură altă clasă.

Implicit, clasele sunt extinse din Object.





```
public class SportsCar extends Car {
    private int maxSpeed;

    public SportsCar(String make,
                     String model, int cc) {
        super(make, model, cc);   // this(make, model, cc, -1);
        maxSpeed = -1;
    }
    public SportsCar(String make, String model, int cc, int maxSpeed) {
        super(make, model, cc);
        this.maxSpeed = maxSpeed;
    }
}
```

cel puțin un constructor dintr-o clasă trebuie să apeleze un constructor din superclasă.