Lezione 9-03

giovedì 9 marzo 2023 11:22

Quando non è specificata la visibilità per una classe, la visibilità è quella package, ovvero quella di default. Ciò significa che la classe può essere utilizzata da qualsiasi file Java all'interno della cartella dove sta il file della classe.

La chiamata costruttore new crea la struttura dell'oggetto nell'heap ma si possono creare all'interno della classe dei costruttori espliciti con la quale assegniamo dei valori di default dati dal programmatore.

```
class Animal{

    public String nome;
    public int eta;
    public int peso;

    public Animal(){
        nome = "nessun nome";
        eta = 0;
        peso = 0;
    }

    public Animal(String n, int e, int p){ //*
        nome = n;
        eta = e;
        peso = p;
    }

    public Animal(String no, int et, int pe){ //**
        nome = no;
        eta = et;
        peso = pe;
    }

}
```

I costruttori sono metodi che si chiamano nello stesso modo della classe e vengono chiamati quando si inizializza un'oggetto.
Volendo si può creare un metodo (quello segnato da *) da cui si possono prendere come parametri i valori che si trovano all'interno del main.
Il limite di Java è che non si possono creare più costruttori con lo stesso numero di parametri e dello stesso tipo (per esempio non posso avere un altro metodo come quello segnato da **)
Dentro la classe Animal si possono creare dei metodi binari, ovvero ha i parametri dello stesso tipo Animal (this è un parametro e ha tipo Animal, lo stesso vale per il parametro esplicitato nel metodo)

```
public void assegna(Animal altroAnimale){
    nome = altroAnimale.nome;
    eta = altroAnimale.eta;
    peso = altroAnimale.peso
}
```

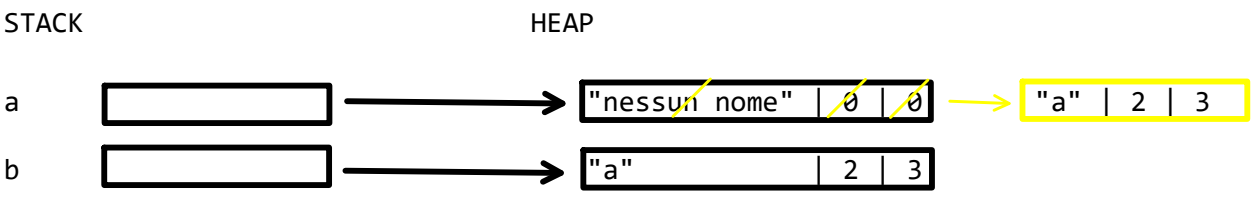
Un altro metodo notevole è il metodo equals: è un metodo boolean e binario e confronta this e il parametro esplicitato

```
public boolean equals(Animal altroAnimale){
    return
    (this.nome.equalsIgnoreCase(altroAnimale.nome))
    &&
    (this.eta == altroAnimale.eta)
    &&
```

```
    (this.peso == altroAnimale.peso)
}
```

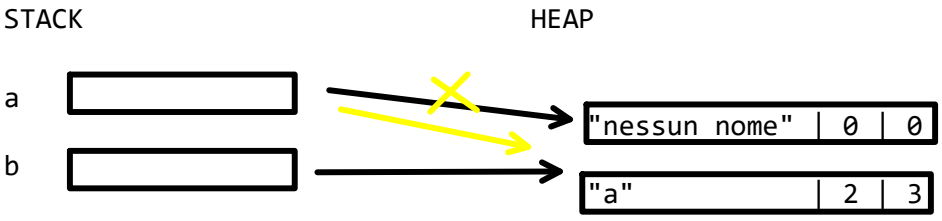
Guardiamo cosa succede nella memoria

```
Animal a = new Animal();
Animal b = new Animal("a", 2, 3);

if(a == b){
    System.out.println("true");
}else{
    System.out.println("false");  //verrà stampato false perché si farà il confronto tra
                                   //gli indirizzi della memoria di a e b
}

if(a.equals(b)){
    System.out.println("true");
}else{
    System.out.println("false");  //verrà stampato false perché i valori dei campi
                                   //sono diversi
}

a.assegna(b);

if(a == b){
    System.out.println("true");
}else{
    System.out.println("false");  //verrà stampato false perché si farà il confronto tra
                                   //gli indirizzi della memoria di a e b
}

if(a.equals(b)){
    System.out.println("true");
}else{
    System.out.println("false");  //verrà stampato true perché i valori dei campi
                                   //sono uguali tra di loro
}
```




```
a = b;
if(a == b){
    System.out.println("true");
}else{
    System.out.println("false");  //verrà stampato true perché a e b puntano nello stesso
                                   //oggetto
}

if(a.equals(b)){
    System.out.println("true");
}else{
    System.out.println("false");  //verrà stampato true perché si fa un confronto per lo
                                   //oggetto
```

```
}
```


STACK
HEAP
a
b
"nessun nome"
0
0
"a"
2
3