Fondamenti di Java

Uguali o identici?

Class P

```
class P {
  int x; int y;
  public String toString() {
     return ("x="+x+" ; y="+y);
  }
}
```

Main di test

```
public class Test {
  public static void main(String []a){new Test();}

 Test() {
      P p1=new P();
      p1.x=1;
      p1.y=2;
      System.out.println(p1);
      P p2=p1;
      p2.x=3;
      System.out.println(p1);
   }
}
```

```
class P {
  int x; int y;
  public String toString() {
      return ("x="+x+" ; y="+y);
  }
}
```

Main di test

```
public class Test {
  public static void main(String []a){new Test();}

 Test() {
      P p1=new P();
      p1.x=1;
      p1.y=2;
      System.out.println(p1);
      P p2=p1;
      p2.x=3;
      System.out.println(p1);
   }
}
```

```
class P {
  int x; int y;
  public String toString() {
      return ("x="+x+" ; y="+y);
  }
}
```

```
x=1 ; y=2
x=3 ; y=2
```

p1 and p2 refer to te same object!

```
public class Test {
  public static void main(String a[]) {new Test();}
  Test() {
     P p1=new P();
     p1.x=1; p1.y=2;
     System.out.println(p1);
     P p2=new P();
     p2.x=1; p2.y=2;
     System.out.println(p2);
     p1.x=3;
     System.out.println(p1);
     System.out.println(p2);
 }
}
```

```
x=1 ; y=2


x=1 ; y=2

x=3 ; y=2
x=1 ; y=2
```

Come testare l'eguaglianza?

```
public class Test {
  public static void main(String a[]){new Test();}
  Test() {
     P p1=new P();
     p1.x=1; p1.y=2;
     P p2=new P();
     p2.x=1; p2.y=2;
     // come testare l'uguaglianza di p1 e p2?
  }
}
```

Operatore ==

```
public class Test {
  public static void main(String[] a){new Test();}
  Test() {
      P p1=new P();
      p1.x=1; p1.y=2;
      P p2=new P();
      p2.x=1; p2.y=2;
      System.out.println(p1==p2);
  }
}
```

```
false
```

Metodo equals()

```
public class Test {
  public static void main(String[] a){new Test();}
  Test() {
      P p1=new P();
      p1.x=1; p1.y=2;
      P p2=new P();
      p2.x=1; p2.y=2;
      System.out.println(p1.equals(p2));
  }
}
```

```
false
```

Metodo equals()

The equals method for class Object implements the most discriminating possible equivalence relation on objects;

that is, for any reference values x and y, this method returns true if and only if x and y refer to the same object (x==y has the value true).

Ma allora a che serve?

equals per la classe P

Equals di Object è la base per implementare il vostro equals

```
class P {
  int x; int y;
  public String toString() {
     return ("x="+x+" ; y="+y);
  }
  public boolean equals(P var){
    return (x==var.x && y==var.y)
  }
}
```

equals() e ==

```
public class Test {
  public static void main(String[] a){new Test();}
  Test() {
      P p1=new P();
      p1.x=1; p1.y=2;
      P p2=new P();
      p2.x=1; p2.y=2;
      System.out.println(p1.equals(p2));
      System.out.println(p1==p2);
 }
}
```

```
true
false
```

Problema 1...

```
public class Test {
  public static void main(String[] a)}new Test();}
  Test() {
      P p1=new P();
      p1.x=1; p1.y=2;
      P p2=null;
      System.out.println(p1.equals(p2));
      System.out.println(p1==p2);
 }
}
```

Error!

equals per la classe P, v.2

```
class P {
  int x; int y;
  public String toString() {
      return ("x="+x+" ; y="+y);
  }
  public boolean equals(P var){
    if(var==null) return false;
    return (x==var.x && y=var.y)
  }
}
```

Problema 2...

Equals deve comparare due Objects!

```
public class Test {
  public static void main(String[] a)}new Test();}
  Test() {
      P p1=new P();
      p1.x=1; P1.y=2;
      Integer p2=new Integer(3);
      System.out.println(p1.equals(p2));
      System.out.println(p1==p2);
 }
}
```

```
false
false
```

equals per la classe P, v.3

```
class P {
  int x; int y;
  public String toString() {
     return ("x="+x+" ; y="+y);
  }
  public boolean equals(Object var){
       if(var==null) return false;
       if (!(var instanceof P)) return false;
       return (x==((P)var).x && y==((P)var).y)
  }
}
```

Problema 3...

```
public class Test {
  public static void main(String[] a)}new Test();}
  Test() {
      P p1=new P();
      p1.x=1; p1.y=2;
      Q p2=new Q();
      p2.x=1; p2.y=2;
      System.out.println(p1.equals(p2));
      System.out.println(p1==p2);
 }
}
```

```
Class Q extends P {
    int z;
}
```

```
true
false
```

equals per la classe P, v.3b

```
class P {
  int x; int y;
  public String toString() {
     return ("x="+x+" ; y="+y);
  }
  public boolean equals(Object var){
    if(var==null) return false;
    if (var.getClass() != this.getClass())
            return false;
    return (x==((P)var).x && y=((P)var).y)
  }
}
```

e ora...

```
public class Test {
  public static void main(String[] a)}new Test();}
  Test() {
      P z1=new P();
      p1.x=1; P1.y=2;
      Q p2=new Q();
      p2.x=1; p2.y=2;
      System.out.println(p1.equals(p2));
      System.out.println(p1==p2);
 }
}
```

```
Class Q extends P {
    int z;
}
```

```
false
false
```

Quale soluzione scegliere?

```
if (o.getClass() != this.getClass())
   return false;

oppure

if (!(var instanceof P)) return false;

?

Dipende...
```

Proprietà richieste ad equals

The equals method implements an equivalence relation:

- It is reflexive: for any reference value x, x.equals(x) should return true.
- It is symmetric: for any reference values x and y, x.equals(y) should return true if and only if y.equals(x) returns true.
- It is transitive: for any reference values x, y, and z, if x.equals(y) returns true and y.equals(z) returns true, then x.equals(z) should return true.

Proprietà richieste ad equals

Additional properties:

- It is consistent: for any reference values x and y, multiple invocations of x.equals(y) consistently return true or consistently return false, provided no information used in equals comparisons on the object is modified.

- For any non-null reference value x, x.equals(null) should return false.

equals e hashCode

Programmers should take note that

any class that overrides the Object.equals method must also override the Object.hashCode method

in order to satisfy the general contract for the Object.hashCode method.

In particular, c1.equals(c2) implies that c1.hashCode()==c2.hashCode()

(the vice versa need not be true)

Le implicazioni

equals dà true => hascode é uguale

equals dà false *non implica nulla su hashcode*

hashcode é lo stesso *non implica nulla su equals*

lo hashcode é diverso => equals dà false

Implementazione di hashCode

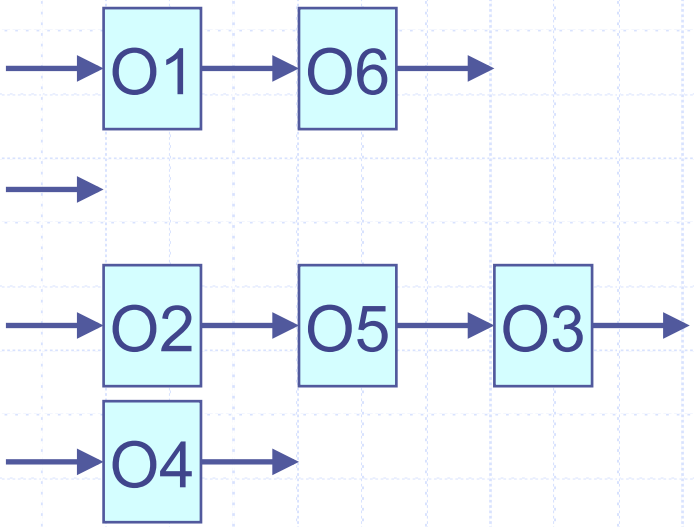
```
public int hashCode() {
      int hash = 0;
    /* a typical implementation:
    for (int i = 0; i < len; i++) {
         hash = 31* hash + val[i];
    }
    */
    hash = x*31+y;
    return hash;
}
```

A che serve hashCode?

Tabelle associative (**Map**)

Dato un oggetto O1 è possibile calcolarne la chiave C1

chiave1	coda1
chiave2	coda2
chiave3	coda3
…	…



Importante anche per i Set