Uguaglianza e Identità

(no, non avete sbagliato corso...)

Fondamenti di Java

Che vuol dire "uguaglianza"?

Che vuol dire "Identità"?

Class P

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

Uguaglianza

```
public class Test {
  public static void main(String[] a){new Test();}
  Test() {
      int k1 = 1;
      int k2 = 1;
      System.out.println(k1==k2);
 }
}
```

```
true
```

Uguaglianza

```
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;
}
}
```

P1 e p2 sono uguali?

Uguaglianza

```
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);
 }
}
```

Uguaglianza

```
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
```

Uguaglianza

```
public class Test {
  public static void main(String[] a){new Test();}
  Test() {
      int k1 = 1;
      int k2 = k1;
      System.out.println(k1==k2);
 }
}
```

```
true
```

Uguaglianza

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

Uguaglianza

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

```
true
```

Uguaglianza

```
int k1=1;
int k2=1;

   k1==k2 ?     TRUE
```

```
int k1=1;
int k2=k1;

   k1==k2 ?     TRUE
```

```
P p1=new P();
p1.x=1; p1.y=2;
P p2=new P();
p2.x=1; p2.y=2;

   p1==p2 ?     FALSE
```

```
P p1=new P();
p1.x=1; p1.y=2;
P p2= p1;


   p1==p2 ?     TRUE
```

PERCHE' ? (ricordiamoci del'allocazione di memoria…)

```
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
```

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!

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
```

java.lang

Class Object

java.lang.Object

`public class Object`

Class `Object` is the root of the class hierarchy. Every class has `Object` as a superclass. All objects, including arrays, implement the methods of this class.

Since:

JDK1.0

See Also:

`Class`

Constructor Summary

Constructors

Constructor and Description
`Object()`

Method Summary

Methods

Modifier and Type	Method and Description
`protected Object`	`clone()` Creates and returns a copy of this object.
`boolean`	`equals(Object obj)` Indicates whether some other object is "equal to" this one.

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));
  }
}
```

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)
  }
}
```

java.lang

Class Object

java.lang.Object

public class Obje

Class Object is the root

Since:

JDK1.0

See Also:

Class

Ma abbiamo fatto overriding o overloading?

```
class P {
   public boolean equals(P var) …
…}
```

Constructor Su

Constructors

Constructor and Description
Object()

Method Summary

Methods

Modifier and Type	Method and Description
protected Object	clone() Creates and returns a copy of this object.
boolean	equals(Object obj) Indicates whether some other object is "equal to" this one.

java.lang

Class Object

java.lang.Object

public class Obje

Class Object is the root

Since:

JDK1.0

See Also:

Class

Constructor Su

Constructors	
Constructor and De	
Object()	

Method Summa

Methods	
Modifier and Type	
protected Object	Creates and returns a copy of this object.
boolean	equals(Object obj) Indicates whether some other object is "equal to" this one.

Ma abbiamo fatto overriding o overloading?

```
class P {
   public boolean equals(P var) …
…}
```

che succede se

```
P p1=new P();
p1.x=1; p1.y=2;
Integer p2=new Integer(3);
System.out.println(p1.equals(p2));
```

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.