Programmazione 2
Esercitazione 4

Livio Pompianu

Contatti e consegna esercizi

- Consegna su Moodle

Oppure, SOLO se Moodle non è disponibile, scrivete una mail indicando, all'inizio dell'oggetto: **[PR2]**

- Livio Pompianu: pompianu.livio@gmail.com

Inviare i testi degli esercizi in un archivio con nome

- `matricola_cognome_nome.zip`

Package and import

Package

I package sono un meccanismo per organizzare le classi e disambiguare classi con lo stesso nome.

Un progetto può essere composto da centinaia o migliaia di classi, per cui ha senso organizzarle in directory e sotto directory.

Il modificatore di default (ovvero quando non si specifica né private né public) lavora utilizzando i package.

Ricapitolando:

- organizzazione delle classi
- evitare conflitti tra i nomi
- controllo degli accessi

Package

Un package crea un nuovo **spazio dei nomi**.

La classe `Persona` dentro il package `anagrafica` è diversa dalla classe `Persona` dentro il package `dipendenti`.

Il nome delle due classi per esteso (*fully qualified name*) sarà

- `anagrafica.Persona`
- `dipendenti.Persona`

Unnamed package

Quando si omette la dichiarazione del package (come abbiamo fatto finora) la classe viene messa in un package senza nome.

- solo per test veloci e temporanei

Package

Scegliere un nome

```
ID ::= [a-zA-Z_][a-zA-Z0-9_]*

PACKAGE_NAME ::= ID (.ID)*
```

Esempi

- `mario.rossi.pr2` (OK)
- `java.awt.event` (OK)
- `un.package.davvero_simpatico` (OK)
- `123.5stella` (NO!)
- `_123.stella` (OK)

Ovviamente non potete usare keyword del linguaggio (if, then, public, switch, ...)

Package

Convenzione sui nomi

Come scegliere un nome che nessun altro userà mai?

Le aziende usano il loro dominio internet al contrario come package "principale" dei loro progetti.

Dominio	Package
unica.it	it.unica
mysql.com	com.mysql
bitcoin.org	org.bitcoin

Le classi appartenenti all'interno di un package devono essere salvate in una **struttura di directory con lo stesso nome**.

Package

1. scegliere il nome del package (ad es. `it.unica.pr2`)
2. creare nel file system le directory necessarie (ad es. `it/unica/pr2/`)
   a. **`mkdir -p it/unica/pr2/`**
3. creare una classe di esempio Test.java dentro it/unica/pr2/
   a. **`gedit it/unica/pr2/Test.java`**
4. dichiarare il package all'interno della classe ➔
5. il nome esteso della classe è `it.unica.pr2.Test`
6. compilare ed eseguire la classe **dalla root del package**
   a. **`javac it/unica/pr2/Test.java`**
   b. **`java it.unica.pr2.Test`**

```
/*
 * classe
 * it.unica.pr2.Test
 */

package it.unica.pr2;

public class Test {

    /* main */
}
```

Import

Le classi all'interno dello stesso package non richiedono l'import.

Le classi in package differenti possono essere usate in tre modi

- usando il nome completo `it.unica.pr2.Test`
- importando la singola classe `import it.unica.pr2.Test;`
- importando tutte le classi di un package `import it.unica.pr2.*;`

Import

Il package `java.lang` è importato di default.

L'asterisco non importa le classi all'interno di sotto-package

```
import java.awt.*;          // non importa le classi dentro color
import java.awt.color.*;
```

Vedremo altri tipi di import quando farete *static* e le *classi annidate*.

equals(Object)

Il metodo equals(Object) della classe Object è un metodo fondamentale poiché definisce quando due oggetti devono ritenersi uguali.

Poiché il tipo aspettato è Object, **qualunque oggetto può essere passato come argomento di input**.

La relazione di equivalenza deve rispettare ovvie proprietà:

- riflessività: `x.equals(x)`
- simmetria: `x.equals(y) ⇒ y.equals(x)`
- transitività: `x.equals(y) && y.equals(z) ⇒ x.equals(z)`
- `x.equals(null) == false`

L'implementazione nella classe object confronta gli indirizzi di memoria degli oggetti. Per cui restituisce true solo se i due oggetti sono la stessa istanza

```
Persona p = new Persona();
assert p.equals(p);    assert !p.equals(new Persona());
```

equals(Object) - implementazione

```
public boolean equals(Object obj) {
    if (this == obj) return true;
    if (obj == null) return false;
    if (!(obj instanceof MiaClasse)) return false;
    MiaClasse other = (MiaClasse) obj;

    if (primitive != other.primitive) return false;

     if (objectAttribute == null) {
         if (other.objectAttribute != null)
             return false;
     } else if (!objectAttribute.equals(other.objectAttribute))
         return false;

     return true;
}
```