Metodologie di Programmazione

Lezione 20: Le classi interne

Lezione 20: Sommario

- Classi annidate
- Classi interne
- Classi annidate statiche
- Esercizi

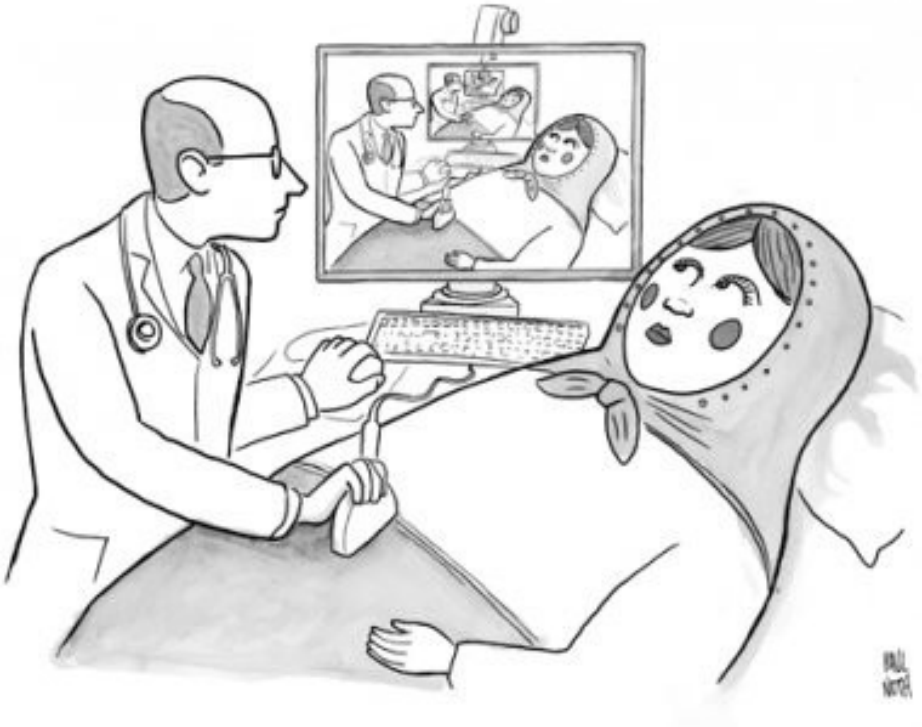
Classi top-level

- Le classi **usate finora** vengono dette top-level, cioè esse si trovano più in alto di tutte le altre e non sono contenute in altre classi

- Questo tipo di classi richiede un file .java dedicato con lo stesso nome della classe che esso contiene

Classi annidate (nested class)

- Java consente di scrivere classi all'interno di altre classi
- Le classi presenti all'interno sono chiamate classi annidate (nested class)
- Possono essere **static**
- Oppure **non-static**: in questo caso vengono dette **classi interne** (inner class)



Classi interne (inner class)

- Prima di poter creare un oggetto della classe interna è necessario istanziare la classe esterna (top-level) che la contiene
- Ciascuna classe interna, infatti, ha un riferimento implicito all'oggetto della classe che la contiene
- Dalla classe interna è possibile accedere a tutte le variabili e a tutti i metodi della classe esterna
- Come tutti i membri di una classe, le classi interne possono essere dichiarate public, protected o private

Classi interne (inner class)

- Se dalla classe interna viene usato soltanto **this** si fa riferimento ai campi e ai metodi di quella classe
- Per far riferimento alla classe esterna è necessario far precedere **this** dal nome della classe esterna e il punto

Classi interne (inner class)


```
public class Tastiera
{
    private String tipo;
    private Tasto[] tasti;

    public class Tasto
    {
        private char c;

        public Tasto(char c)
        {
            this.c = c;
        }

        public char premi()
        {
            return c;
        }

        public String toString()
        {
            return Tastiera.this.tipo + ": " + premi();
        }
    }

    public Tastiera(String tipo, char[] caratteri)
    {
        this.tipo = tipo;
        tasti = new Tasto[caratteri.length];

        for(int i=0; i<caratteri.length; i++)
            tasti[i] = new Tasto(caratteri[i]);
    }
}
```

La classe Tasto è una inner-class (quindi non statica)

Tasto ha accesso al campo (privato!) della classe esterna

Esercizio: Liste linkate di interi con classi interne

- Implementare le liste linkate di interi in modo da nascondere all'esterno l'implementazione del singolo elemento della lista

Classi annidate statiche (static nested class)

- Se invece la classe interna è statica allora essa non richiede l'esistenza di un oggetto appartenente alla classe esterna e non ha nemmeno un riferimento implicito ad essa
  - Come con i metodi statici, non può accedere allo stato dei singoli oggetti della classe esterna

- Da un punto di vista di comportamento, una classe annidata statica è equivalente ad una classe top-level inserita all'interno di un'altra classe top-level

- Sono accessibili tramite il nome della classe esterna che le contiene, secondo la forma **new** ClasseEsterna.ClasseInnestataStatica()

Classi annidate statiche (static nested class)


```
public class Tastiera
{
    private String tipo;
    private Tasto[] tasti;

    public static class Tasto
    {
        private char c;

        public Tasto(char c)
        {
            this.c = c;
        }

        public char premi()
        {
            return c;
        }

        public String toString()
        {
            return Tastiera.this.tipo + ": " + premi();
        }
    }

    public Tastiera(String tipo, char[] caratteri)
    {
        this.tipo = tipo;
        tasti = new Tasto[caratteri.length];

        for(int i=0; i<caratteri.length; i++)
            tasti[i] = new Tasto(caratteri[i]);
    }
}
```

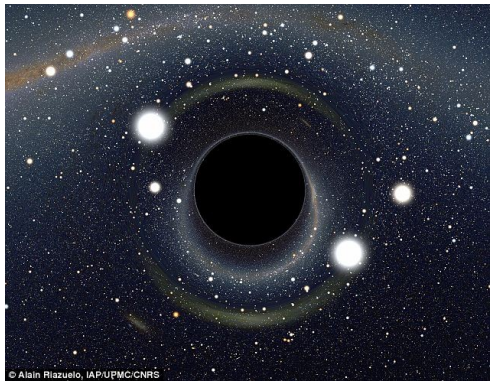
Errore in fase di compilazione: la classe static Tasto non ha accesso implicito ai membri della classe esterna

Ma la classe è visibile all'esterno

```
public class Computer
{
    public static void main(String[] args)
    {
        Tastiera.Tasto tasto = new Tastiera.Tasto('f');
        tasto.premi();
    }
}
```

Classi annidate: perché sono utili?

- Raggruppamento logico delle classi
  - Se una classe è **utile solo ad un'altra classe**, è logico inserirla al suo interno e tenere le due classi logicamente vicine
- Incrementa l'incapsulamento
  - Una classe B annidata in A può accedere ai membri di A (anche se privati), ma B può essere nascosta al mondo esterno
- Codice più leggibile e più facile da mantenere
  - La vicinanza spaziale è un fattore decisivo

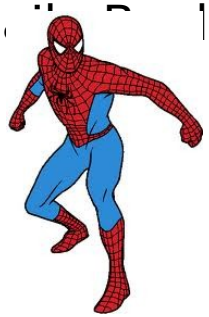


Esercizio: Disney vs. Marvel (1)

- Modellare uno scontro su campo tra personaggi Disney e personaggi Marvel
- Squadra Disney = Paperinik, Ciccionik, Superpippo
  - Paperinik, Ciccionik e Superpippo sono la **versione supereroe** dei corrispettivi personaggi (Paperino, Ciccio e Pippo)



- Squadra Marvel = Spiderman, La cosa, Magneto
  - Notare che Spiderman è la **versione supereroe** di Peter Parker, noto fotografo del D[illegible]le; La cosa e Mag[illegible]ce sono supereroi 24h



Esercizio: Disney vs. Marvel (2)

- I personaggi con una doppia vita devono esporre un'interfaccia DoppiaVita con i seguenti metodi:
  - assumiIdentitaSegreta(): consente di trasformarsi in supereroe
  - assumiIdentitaPubblica(): consente di assumere le sembianze 'umane'
- Ogni supereroe implementa l'interfaccia Supereroe con:
  - attacca(): sferra l'attacco specifico del supereroe
- Creare una partita su campo (ovvero una classe di test) in cui le due squadre si alternano in attacchi micidiali
  - Il nemico viene scelto casualmente dalla lista dei supereroi avversari

Esercizio: Disney vs. Marvel (3)

- Hint per versione **semplificata**: Senza utilizzare classi interne, ma solo interfacce ed ereditarietà
- Hint per versione **elaborata**: Per riflettere il fatto che nessuno è a conoscenza dei superpoteri in possesso dalle controparti umane, i supereroi, laddove possibile, dovrebbero estendere le classi umane corrispondenti ed essere implementati come loro classi annidate private (ad es. Paperinik estende Paperino, ma solo Paperino potrà restituire un'istanza di Paperinik tramite l'interfaccia DoppiaVita)