

Object Oriented per non credenti

Creative Commons Attribution-NonCommercial-ShareAlike 2.5

Bruno Bossola



About me

Creative Commons Attribution-NonCommercial-ShareAlike 2.5

- Sviluppatore C dal 1988
- Sviluppatore Java dal 1996
- Coach di un team XP nel 2000
- Socio fondatore del JUG Torino nel 2001
- Nominato Sun Java Champion nel 2005



Agenda

Creative Commons Attribution-NonCommercial-ShareAlike 2.5

- Ammettere che non sappiamo cosa cavolo sia Object Oriented



Agenda

Creative Commons Attribution-NonCommercial-ShareAlike 2.5

- Ammettere che non sappiamo cosa cavolo sia Object Oriented
- Ammettere che non sappiamo come cavolo farlo



Agenda

Creative Commons Attribution-NonCommercial-ShareAlike 2.5

- Ammettere che non sappiamo cosa cavolo sia Object Oriented
- Ammettere che non sappiamo come cavolo farlo
- Capire cosa è veramente Object Oriented!

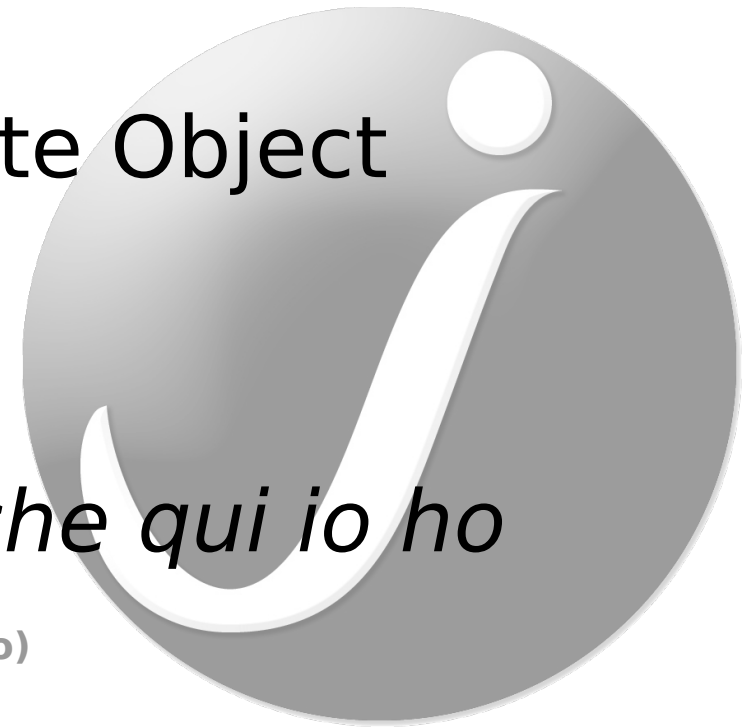


Agenda

Creative Commons Attribution-NonCommercial-ShareAlike 2.5

- Ammettere che non sappiamo cosa cavolo sia Object Oriented
- Ammettere che non sappiamo come cavolo farlo
- Capire cosa è veramente Object Oriented!

*...e poi sono fatti vostri che qui io ho
50 minuti!*



Il nostro problema

Creative Commons Attribution-NonCommercial-ShareAlike 2.5

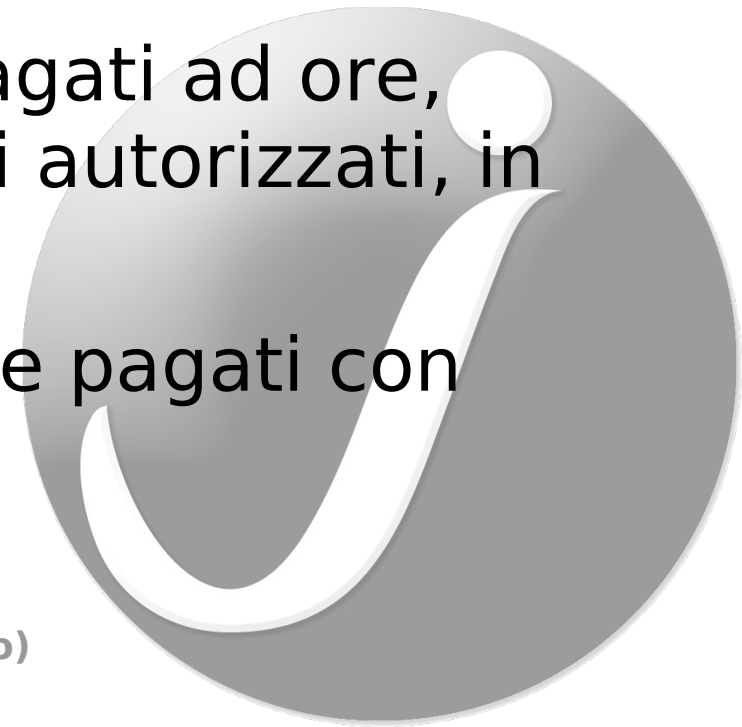
- Realizzare un'applicazione paghe
- Deve essere dotata di interfaccia web salvando i dati su un DB relazionale
- Ci limiteremo ad alcuni semplici requisiti funzionali



Requisiti

Creative Commons Attribution-NonCommercial-ShareAlike 2.5

- alcuni impiegati sono pagati ogni due settimane, alcuni una volta al mese
- i venditori sono pagati con un fisso più una provvigione, gli altri solo un fisso
- alcuni impiegati sono pagati ad ore, compresi gli straordinari autorizzati, in base alle timbrature
- si può scegliere di essere pagati con assegno o con bonifico



Ma in fondo...

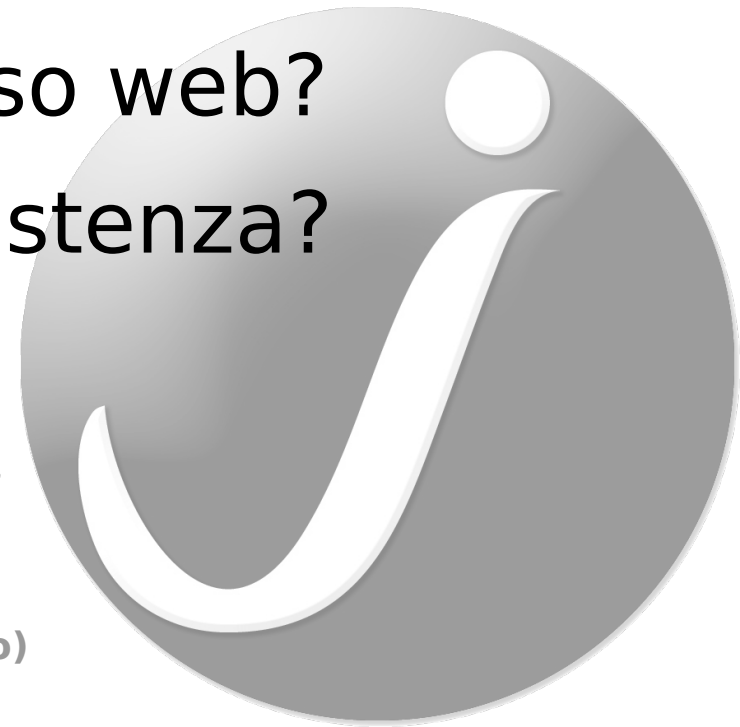
Creative Commons Attribution-NonCommercial-ShareAlike 2.5

...chi se ne frega!

Dobbiamo pensare all'architettura!

- come gestiamo l'accesso web?
- come gestiamo la persistenza?

Non è così che funziona?



Il web

Creative Commons Attribution-NonCommercial-ShareAlike 2.5

- Servlet?



Il web

Creative Commons Attribution-NonCommercial-ShareAlike 2.5

- ~~Servlet?~~



Il web

Creative Commons Attribution-NonCommercial-ShareAlike 2.5

- ~~Servlet?~~
- JSP + TagLib?



Il web

Creative Commons Attribution-NonCommercial-ShareAlike 2.5

- ~~Servlet?~~
- ~~JSP + TagLib?~~



Il web

Creative Commons Attribution-NonCommercial-ShareAlike 2.5

- ~~Servlet?~~
- ~~JSP + TagLib?~~



Il web

Creative Commons Attribution-NonCommercial-ShareAlike 2.5

- ~~Servlet?~~
- ~~JSP + TagLib?~~
- Velocity?



Il web

Creative Commons Attribution-NonCommercial-ShareAlike 2.5

- ~~Servlet?~~
- ~~JSP + TagLib?~~
- ~~Velocity?~~



Il web

Creative Commons Attribution-NonCommercial-ShareAlike 2.5

- ~~Servlet?~~
- ~~JSP + TagLib?~~
- ~~Velocity?~~
- Struts?



Il web

Creative Commons Attribution-NonCommercial-ShareAlike 2.5

- ~~Servlet?~~
- ~~JSP + TagLib?~~
- ~~Velocity?~~
- ~~Struts?~~



Il web

Creative Commons Attribution-NonCommercial-ShareAlike 2.5

- ~~Servlet?~~
- ~~JSP + TagLib?~~
- ~~Velocity?~~
- ~~Struts?~~
- WebWork?



Il web

Creative Commons Attribution-NonCommercial-ShareAlike 2.5

- ~~Servlet?~~
- ~~JSP + TagLib?~~
- ~~Velocity?~~
- ~~Struts?~~
- ~~WebWork?~~



Il web

Creative Commons Attribution-NonCommercial-ShareAlike 2.5

- ~~Servlet?~~
- ~~JSP + TagLib?~~
- ~~Velocity?~~
- ~~Struts?~~
- ~~WebWork?~~
- JSF + AJAX! (cool!)



La persistenza

Creative Commons Attribution-NonCommercial-ShareAlike 2.5

- JDBC?



La persistenza

Creative Commons Attribution-NonCommercial-ShareAlike 2.5

- ~~JDBC?~~



La persistenza

Creative Commons Attribution-NonCommercial-ShareAlike 2.5

- ~~JDBC?~~
- DAO?



La persistenza

Creative Commons Attribution-NonCommercial-ShareAlike 2.5

- ~~JDBC?~~
- ~~DAO?~~



La persistenza

Creative Commons Attribution-NonCommercial-ShareAlike 2.5

- ~~JDBC?~~
- ~~DAO?~~
- Entity beans?



La persistenza

Creative Commons Attribution-NonCommercial-ShareAlike 2.5

- ~~JDBC?~~
- ~~DAO?~~
- ~~Entity beans?~~



La persistenza

Creative Commons Attribution-NonCommercial-ShareAlike 2.5

- ~~JDBC?~~
- ~~DAO?~~
- ~~Entity beans?~~
- Hibernate?



La persistenza

Creative Commons Attribution-NonCommercial-ShareAlike 2.5

- ~~JDBC?~~
- ~~DAO?~~
- ~~Entity beans?~~
- ~~Hibernate?~~



La persistenza

Creative Commons Attribution-NonCommercial-ShareAlike 2.5

- ~~JDBC?~~
- ~~DAO?~~
- ~~Entity beans?~~
- ~~Hibernate?~~
- Ibatis?



La persistenza

Creative Commons Attribution-NonCommercial-ShareAlike 2.5

- ~~JDBC?~~
- ~~DAO?~~
- ~~Entity beans?~~
- ~~Hibernate?~~
- ~~Ibatis?~~



La persistenza

Creative Commons Attribution-NonCommercial-ShareAlike 2.5

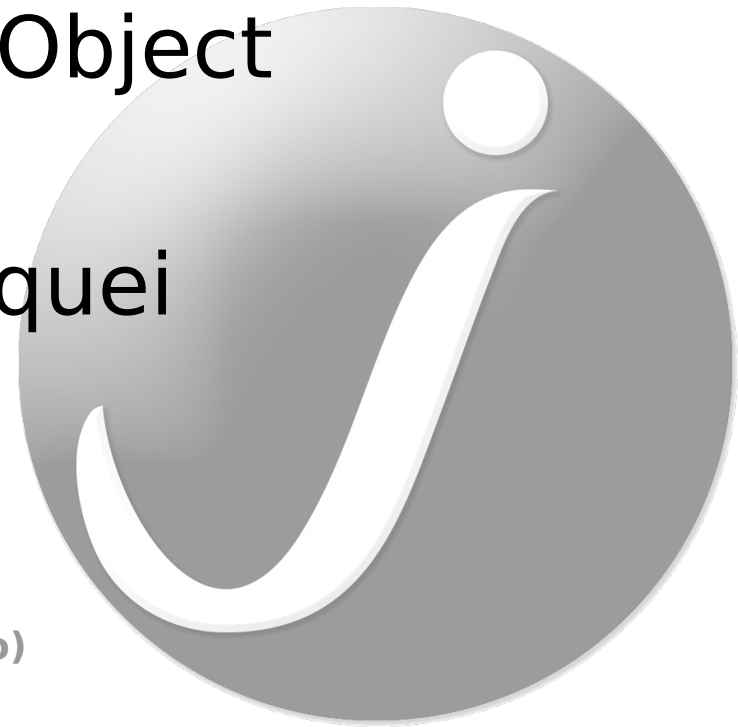
- ~~JDBC?~~
- ~~DAO?~~
- ~~Entity beans?~~
- ~~Hibernate?~~
- ~~Ibatis?~~
- JPA!!! (aka EJB3... cool!)



Uhm...

Creative Commons Attribution-NonCommercial-ShareAlike 2.5

- Ma dov'è che esattamente abbiamo usato Object Oriented?
- Ah, già, usiamo Java
- Quindi stiamo facendo Object Oriented, no?
- Poi usiamo anche tutti quei framework così fighi...



Questo non è OO!

Creative Commons Attribution-NonCommercial-ShareAlike 2.5

- ...e voi direte chissene frega :-)

Non è la prima volta che lo sento...



Perchè è nato OO?

Creative Commons Attribution-NonCommercial-ShareAlike 2.5

- a cosa serve esattamente?
- che vantaggi reali e concreti fornisce?



Perchè è nato OO?

Creative Commons Attribution-NonCommercial-ShareAlike 2.5

- consente di mantenere la tracciabilità dei requisiti fra i vari modelli
 - analisi
 - design
 - implementazione
- consente di mantenere bassa la complessità del sistema
- descrive un sistema in termini comprensibili



Cosa sono i requisiti?

Creative Commons Attribution-NonCommercial-ShareAlike 2.5

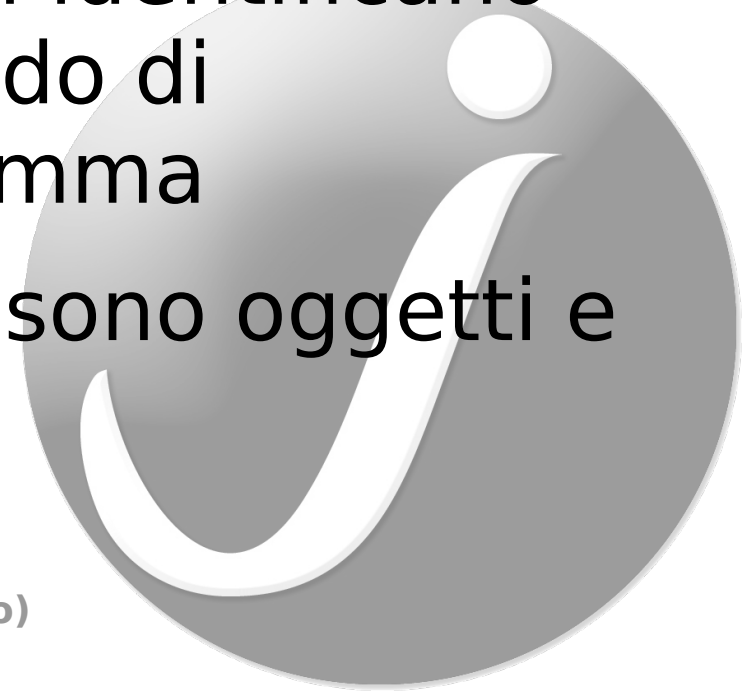
- i comportamenti desiderati dal sistema
- l'insieme dei requisiti porta a definire ciò che l'applicazione finale dovrà fare
- possono essere espressi in diverse forme



Cosa è l'analisi?

Creative Commons Attribution-NonCommercial-ShareAlike 2.5

- la riscrittura di un set di requisiti in termini adatti a descrivere un'applicazione software
- partendo dai requisiti si identificano quei componenti in grado di esprimerli in un programma
- nel caso di OOA questi sono oggetti e le loro collaborazioni



Object Oriented Analysis

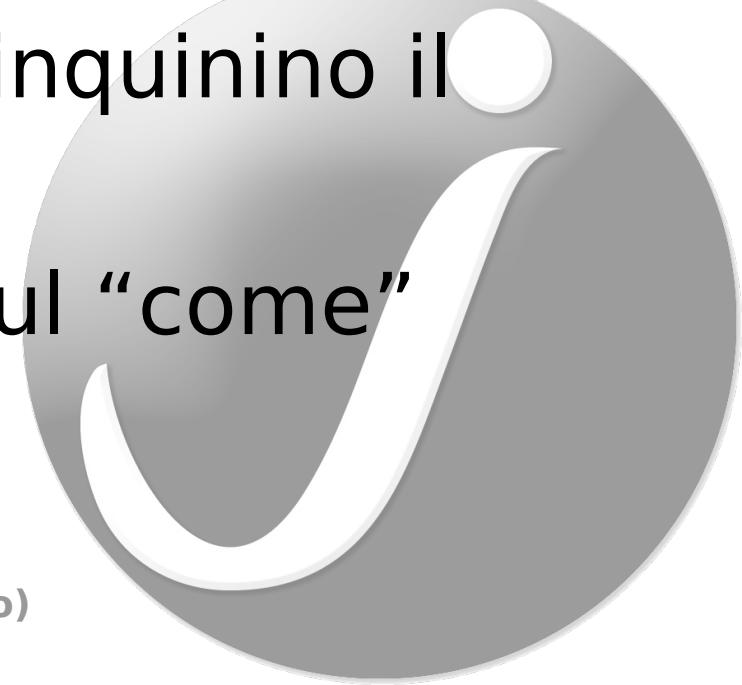
Creative Commons Attribution-NonCommercial-ShareAlike 2.5

- è la determinazione delle astrazioni che implementano i requisiti
- identifica ed espone i componenti del sistema in modo che possano essere successivamente implementati
- valuta il sistema sia in termini di sui comportamenti che di struttura
- l'analisi si concentra sul “cosa”

Object Oriented Design

Creative Commons Attribution-NonCommercial-ShareAlike 2.5

- è l'insieme delle decisioni che portano a definire come le astrazioni verranno implementate
- nasconde l'implementazione in modo che i suoi dettagli non inquinino il resto del sistema
- il design si concentra sul “come”



Ripartiamo!

Creative Commons Attribution-NonCommercial-ShareAlike 2.5

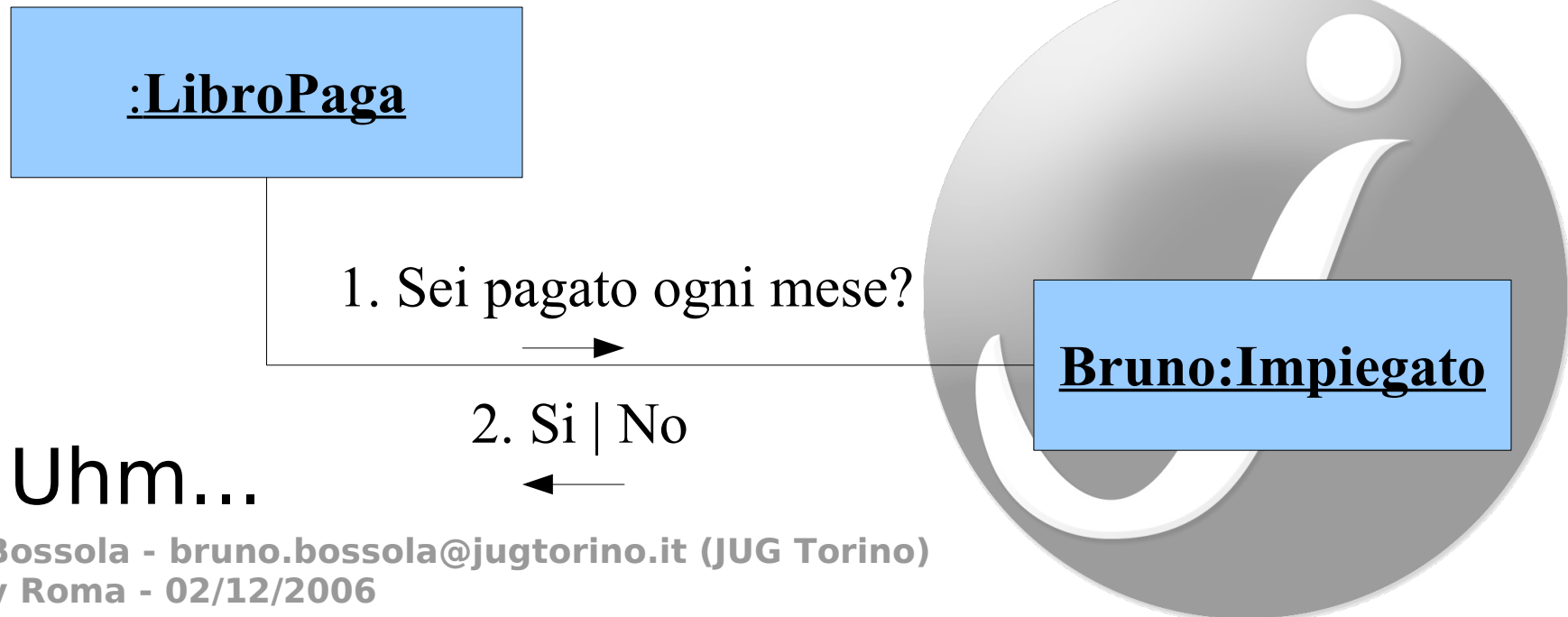
- E' meglio se proviamo a ripartire dai requisiti!
- Ma non sarà facile...



Analisi?

Creative Commons Attribution-NonCommercial-ShareAlike 2.5

- Requisito:
 - alcuni impiegati sono pagati ogni due settimane, alcuni ogni mese

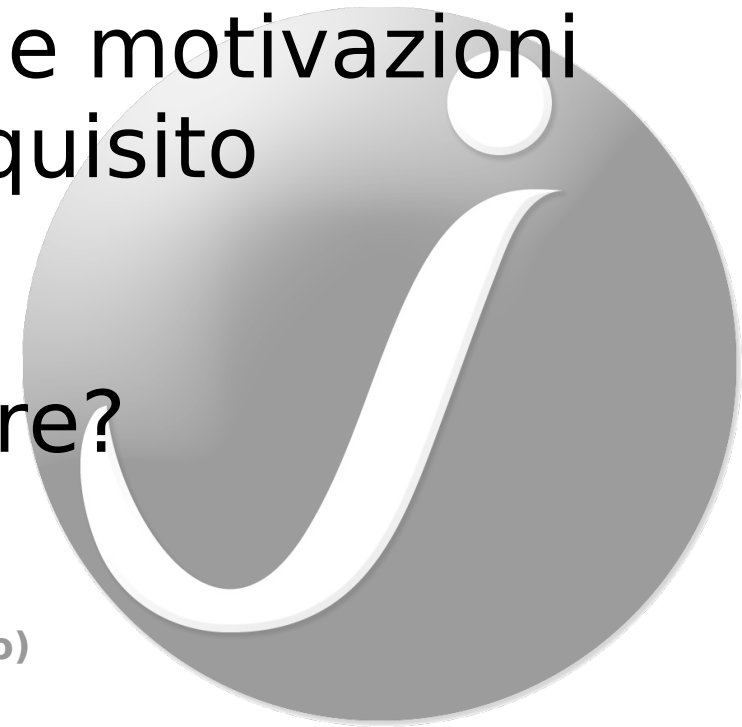


- Uhm...

Analisi!

Creative Commons Attribution-NonCommercial-ShareAlike 2.5

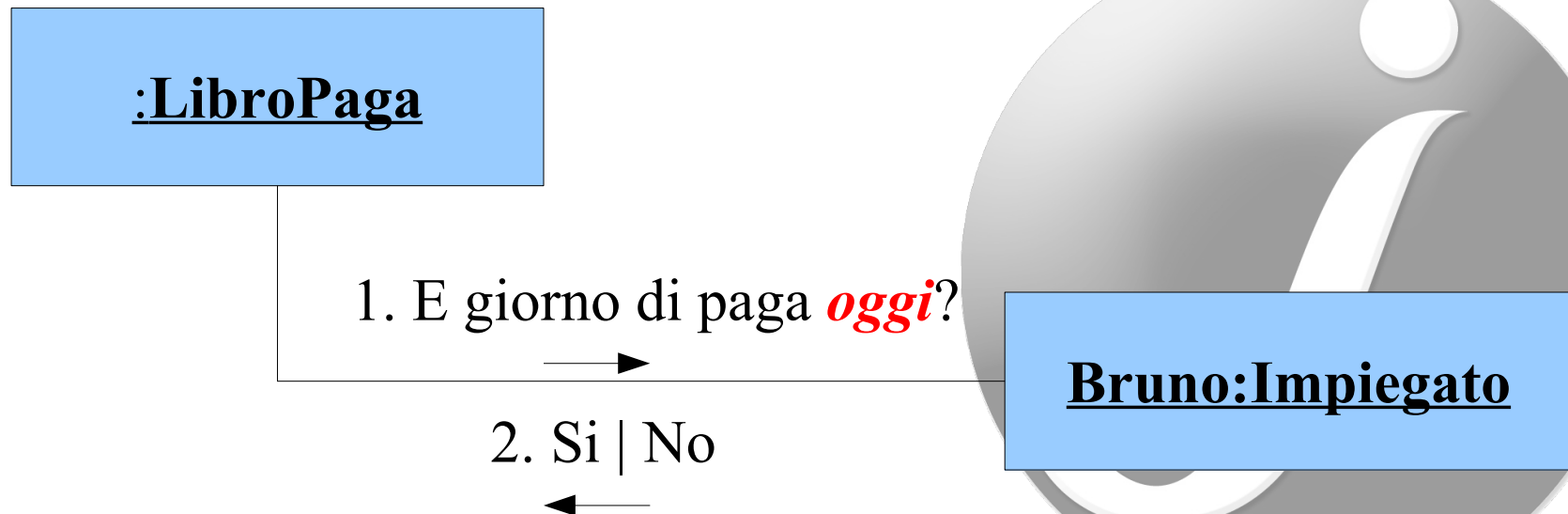
- Un po' troppo semplificato, non pensate?
- Dobbiamo identificare le motivazioni che sottintendono il requisito
- Che cosa è invariante?
- Che cosa potrà cambiare?



Analisi!

Creative Commons Attribution-NonCommercial-ShareAlike 2.5

- Invariante: ogni impiegato viene pagato secondo una scadenza
- Variante: la scadenza stessa



Design!

Creative Commons Attribution-NonCommercial-ShareAlike 2.5

- Disaccoppiamo le logiche
- Come viene associato un impiegato alla sua modalità di pagamento?



- Uhm... ma come funziona il codice qui?

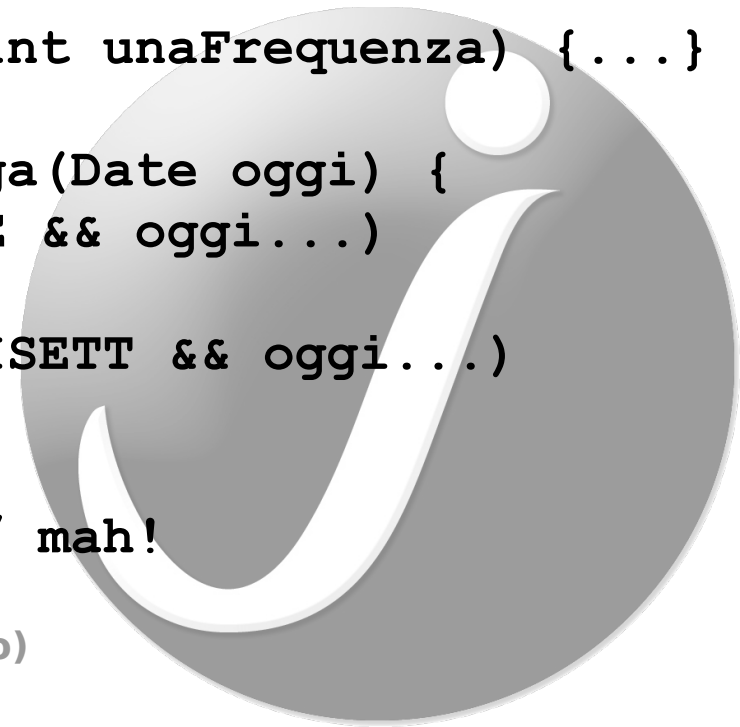
Design?

Creative Commons Attribution-NonCommercial-ShareAlike 2.5

```
public class FrequenzaPagamento {
    public static int MENSILE = 0;
    public static int BISETT = 1;

    public int frequenza;
    public FrequenzaPagamento (int unaFrequenza) {...}

    public boolean isGiornoDiPaga (Date oggi) {
        if (frequenza == MENSILE && oggi...)
            return true;
        else if (frequenza == BISETT && oggi...)
            return true;
        else
            return false;        // mah!
    }
}
```



Design?

Creative Commons Attribution-NonCommercial-ShareAlike 2.5

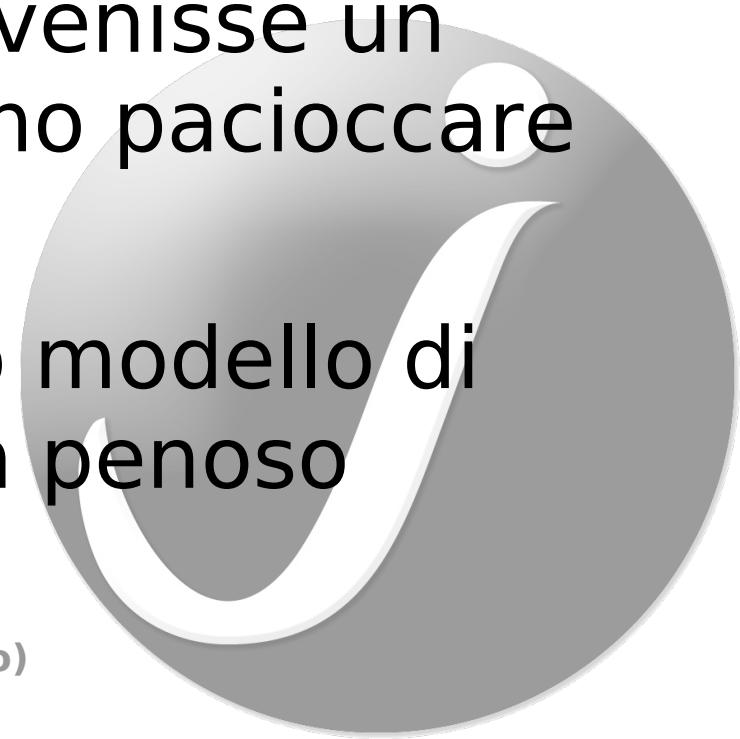
- Okay, ho scelto il modo peggiore ma si può fare anche più figo
 - frequenza di tipo short
 - frequenza è una istanza di una inner class privata di TipoPagamento
 - frequenza è un tipo Enum (forte! richiede JDK5! cool!)
- No, manca un pizzico di design!



Design!

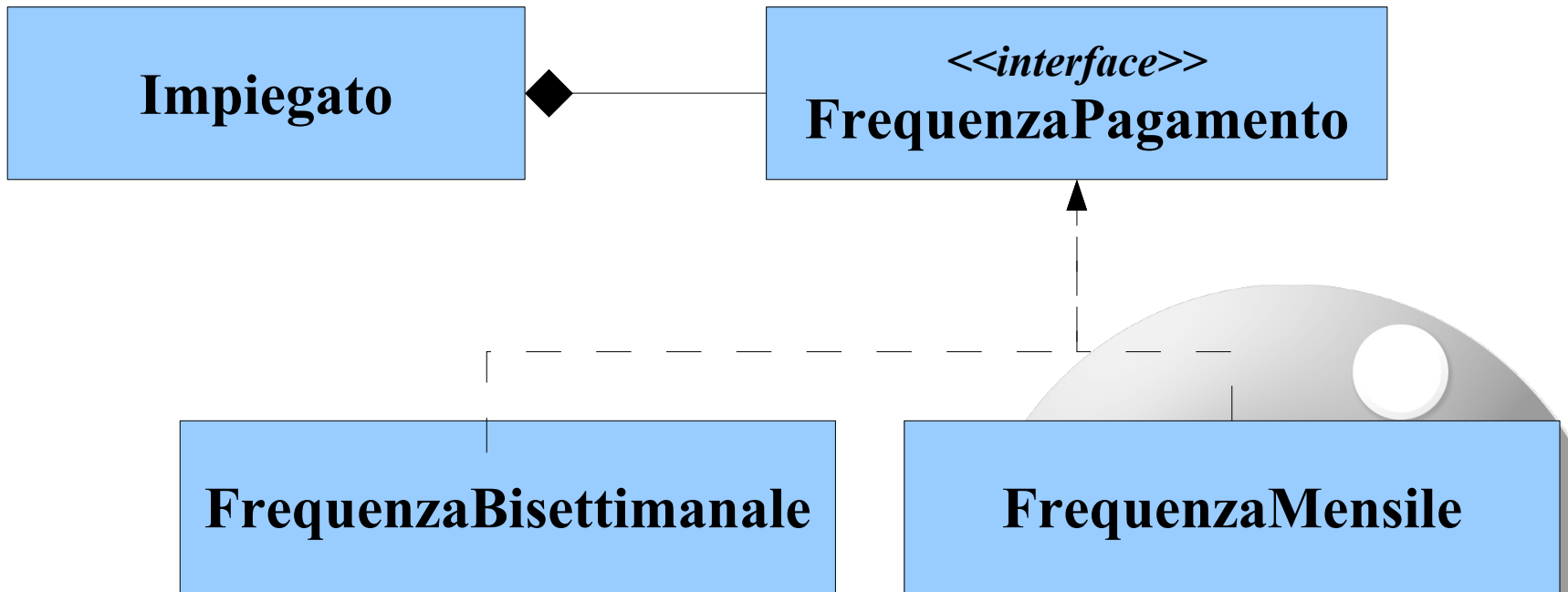
Creative Commons Attribution-NonCommercial-ShareAlike 2.5

- Già design... perché la schifezza che ho scritto rispetta il modello dell'analisi!
- Ma è inflessibile! Se avvenisse un cambiamento dovremmo pacioccare dentro quel codice!
- Si puo' avere un ottimo modello di analisi in coppia con un penoso modello di design!



Design!

Creative Commons Attribution-NonCommercial-ShareAlike 2.5



- Niente if. Polimorfismo, please
- Il design nasconde la complessità

Spot!

Creative Commons Attribution-NonCommercial-ShareAlike 2.5

Ci vediamo a **JavaPolis!**



Io e il JUG Torino presentiamo:

“Automatic testing
using Open Source tools”

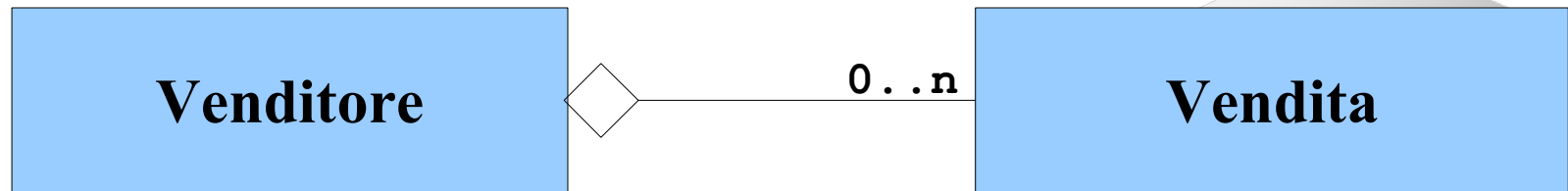


Analisi

Creative Commons Attribution-NonCommercial-ShareAlike 2.5

Requisito:

- i venditori sono pagati con un fisso più una provvigione



- Esprime tutte le motivazioni che sottintendono il requisito?

Analisi

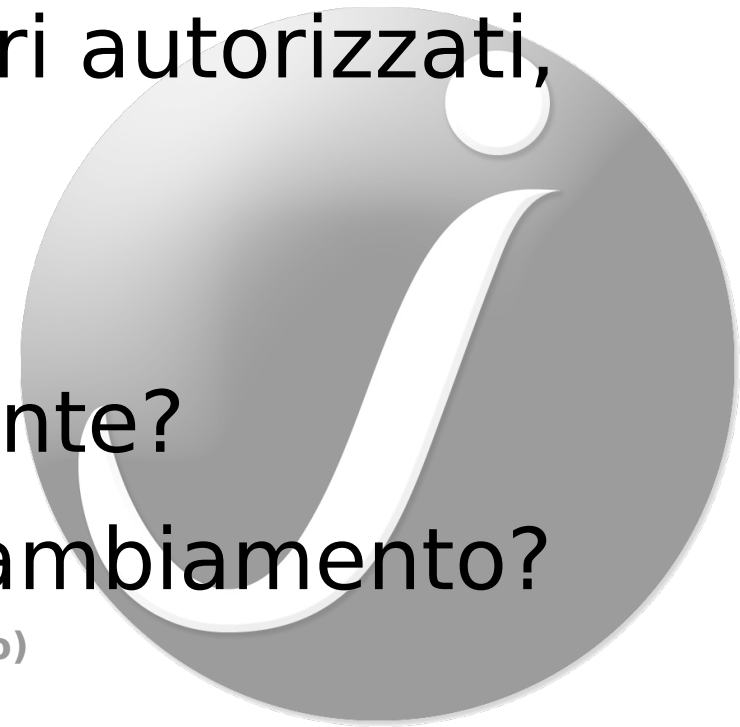
Creative Commons Attribution-NonCommercial-ShareAlike 2.5

Forse no se sbirciamo i due successivi

- alcuni impiegati ricevono un fisso
- alcuni impiegati sono pagati ad ore, compresi gli straordinari autorizzati, in base alle timbrature

Di nuovo, qual'è l'invariante?

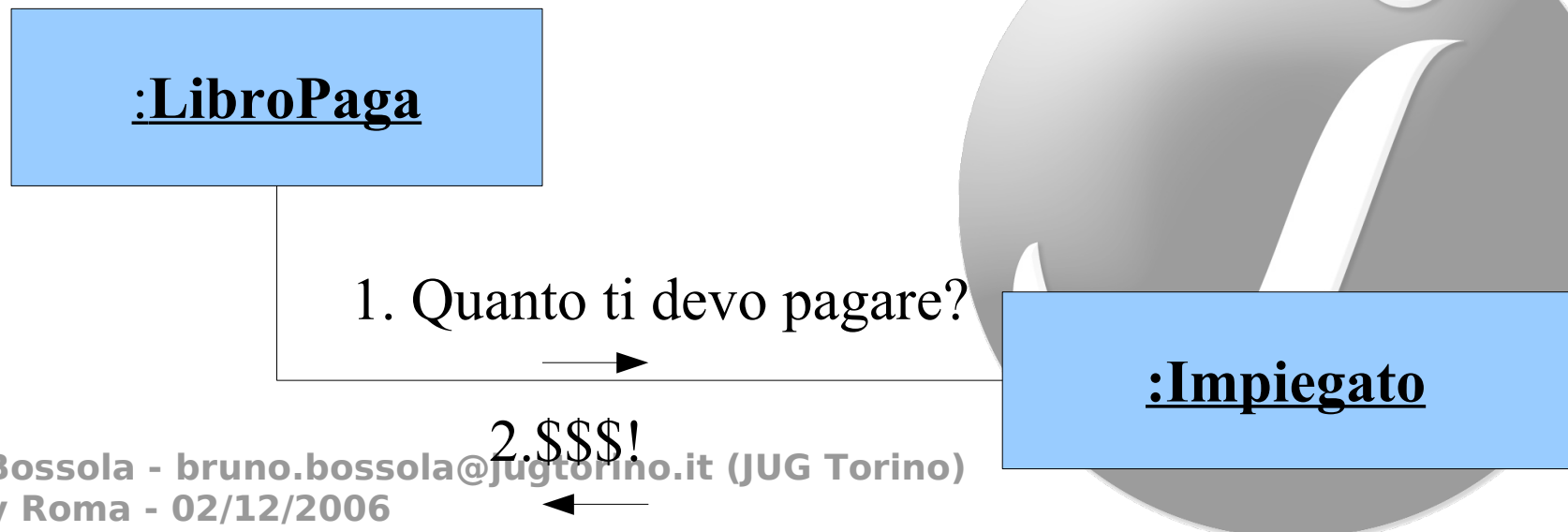
Che cosa è soggetto a cambiamento?



Analisi

Creative Commons Attribution-NonCommercial-ShareAlike 2.5

- Invariante: ogni impiegato viene pagato (di solito :-))
- Variante: la modalità del calcolo della retribuzione



Design!

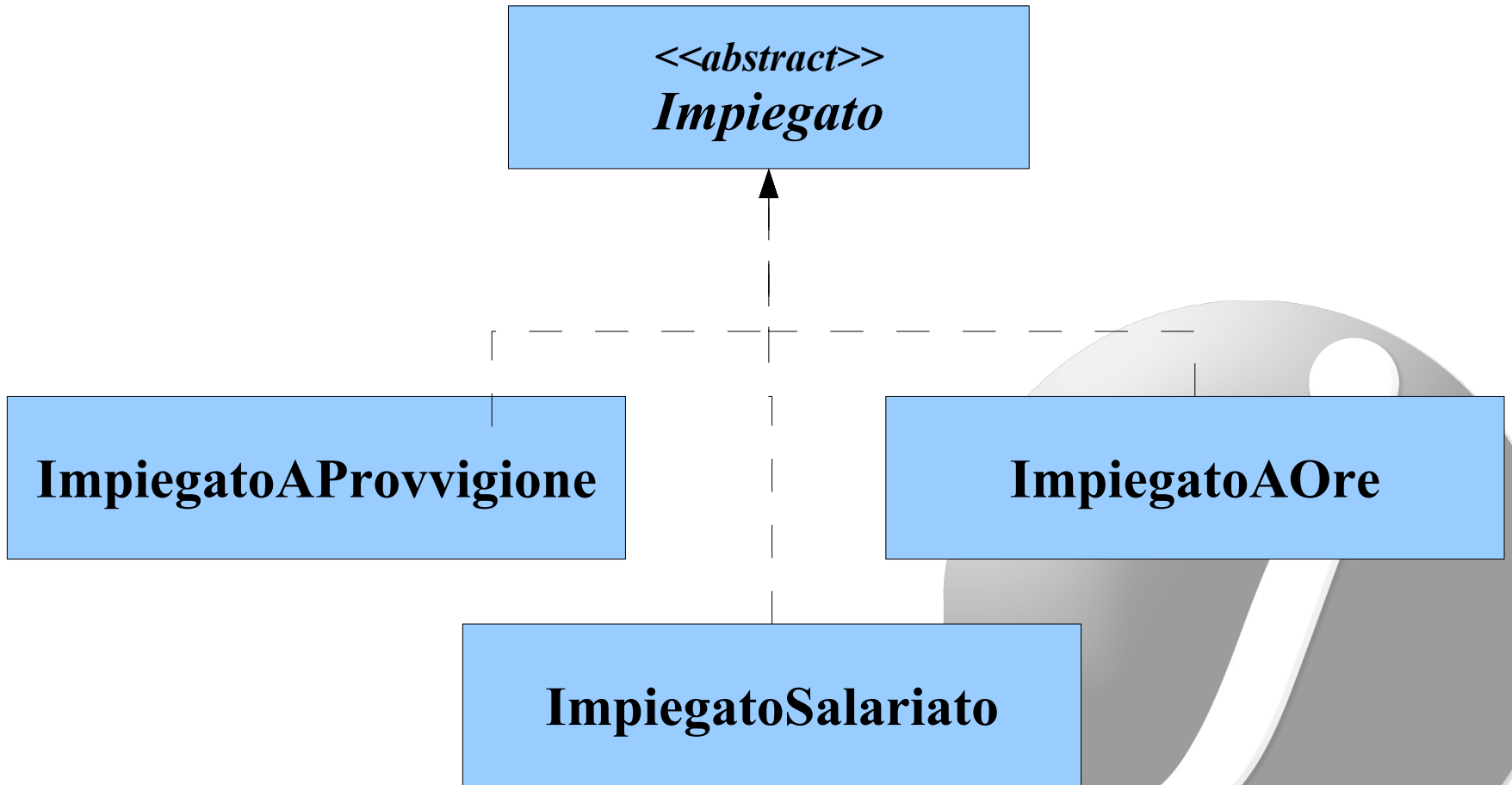
Creative Commons Attribution-NonCommercial-ShareAlike 2.5

- Proposte?



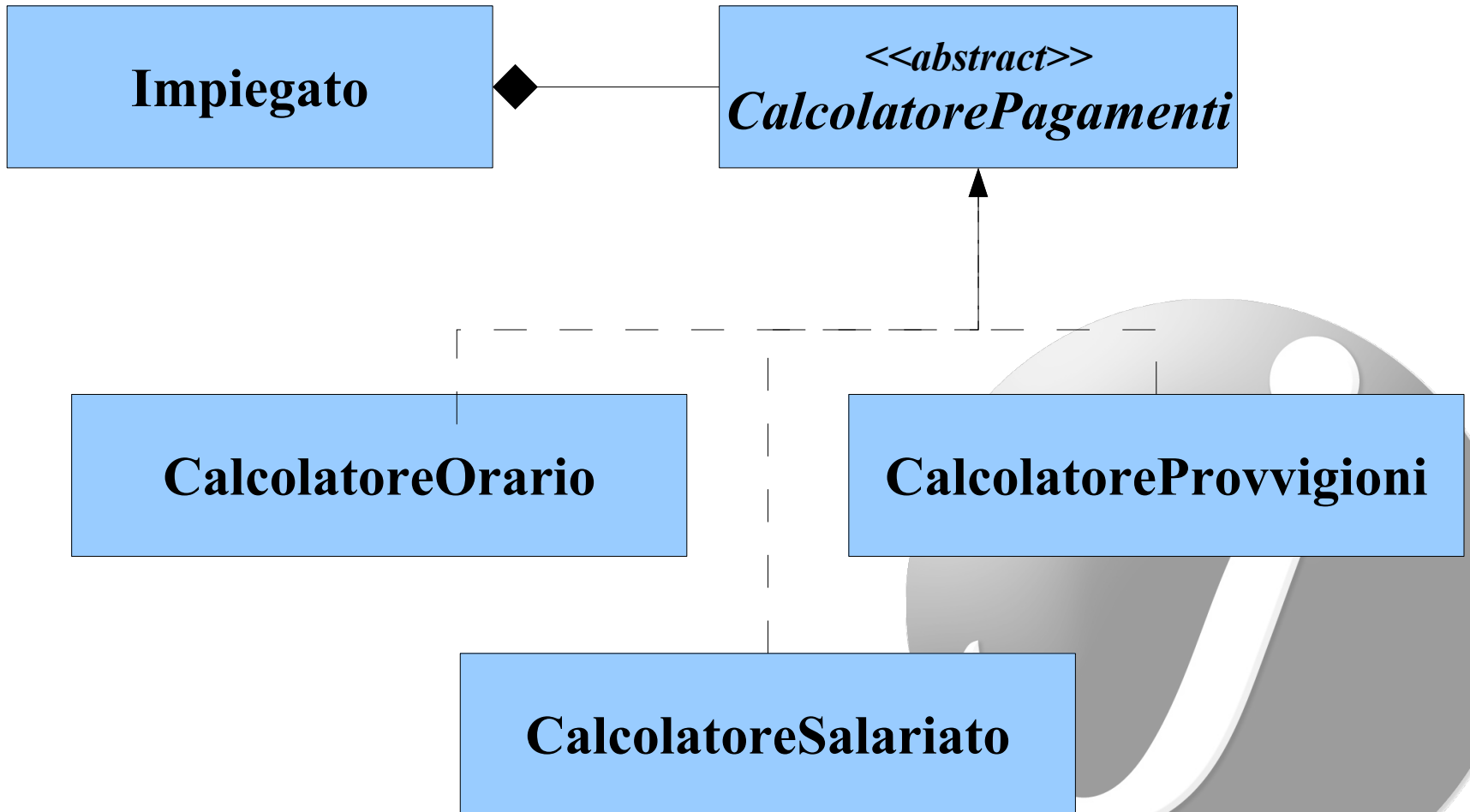
Design!

Creative Commons Attribution-NonCommercial-ShareAlike 2.5



Design!

Creative Commons Attribution-NonCommercial-ShareAlike 2.5



Design!

Creative Commons Attribution-NonCommercial-ShareAlike 2.5

Per ora buona la prima!

- il calcolatore per provvigioni deve vedere le vendite, che sono del dipendente
- il calcolatore orario deve vedere le timbrature, che sono del dipendente
- servirebbero comunque diversi tipi di dipendente



Design!

Creative Commons Attribution-NonCommercial-ShareAlike 2.5

- le astrazioni sono scelte in base alla natura del sistema
- una gerarchia come questo spesso è una cattivissima idea!



Design!

Creative Commons Attribution-NonCommercial-ShareAlike 2.5

- le astrazioni sono scelte in base alla natura del sistema
- una gerarchia come questo spesso è una cattivissima idea!
- Chi mi sa dire perché?



Design!

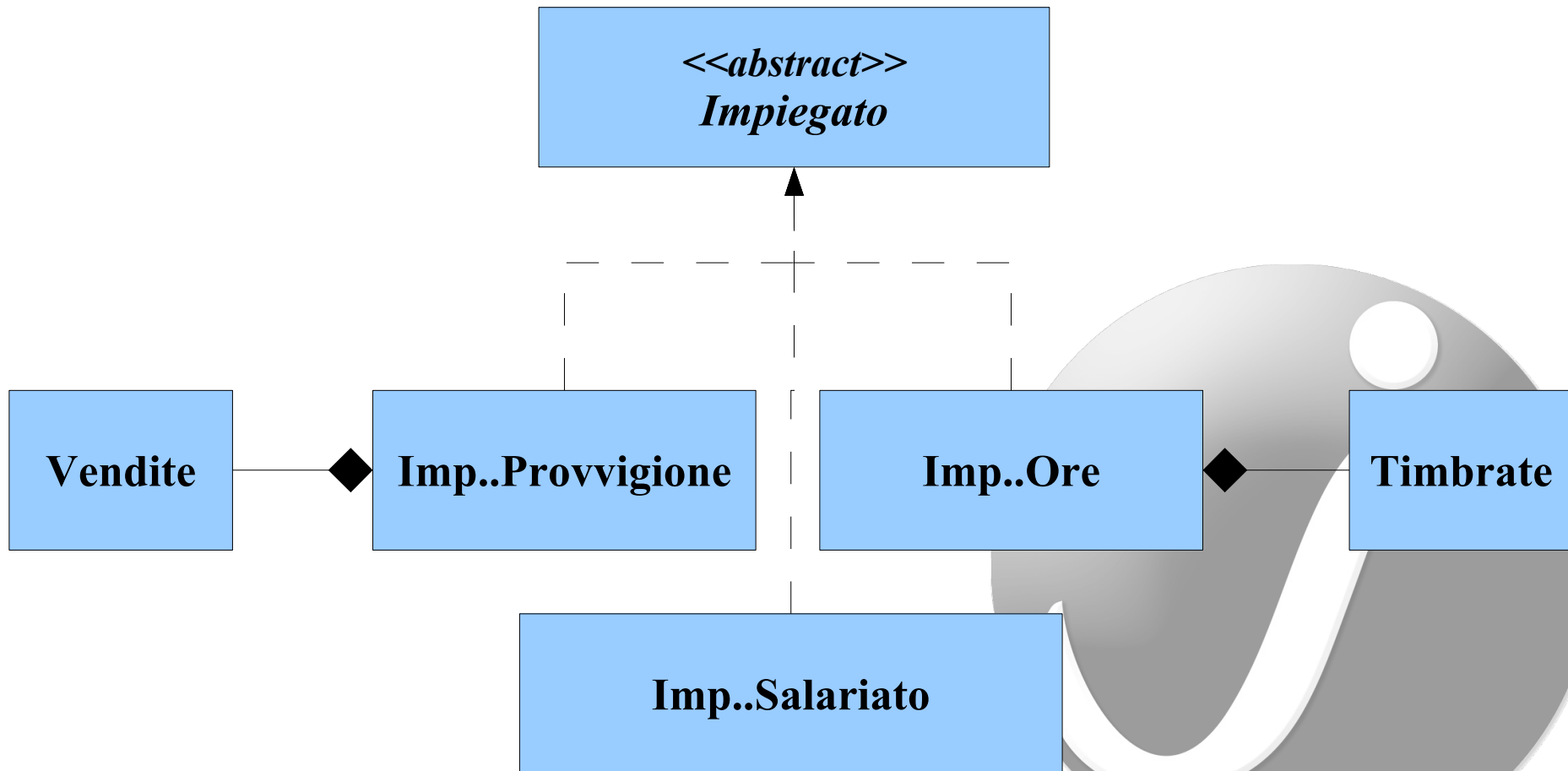
Creative Commons Attribution-NonCommercial-ShareAlike 2.5

- le astrazioni sono scelte in base alla natura del sistema
- una gerarchia come questo spesso è una cattivissima idea!
- in questo caso soddisfa il modello di analisi (e quindi i requisiti)
- occhio che il design **evolve** nel tempo, il calcolatore puo' emergere



Design!

Creative Commons Attribution-NonCommercial-ShareAlike 2.5

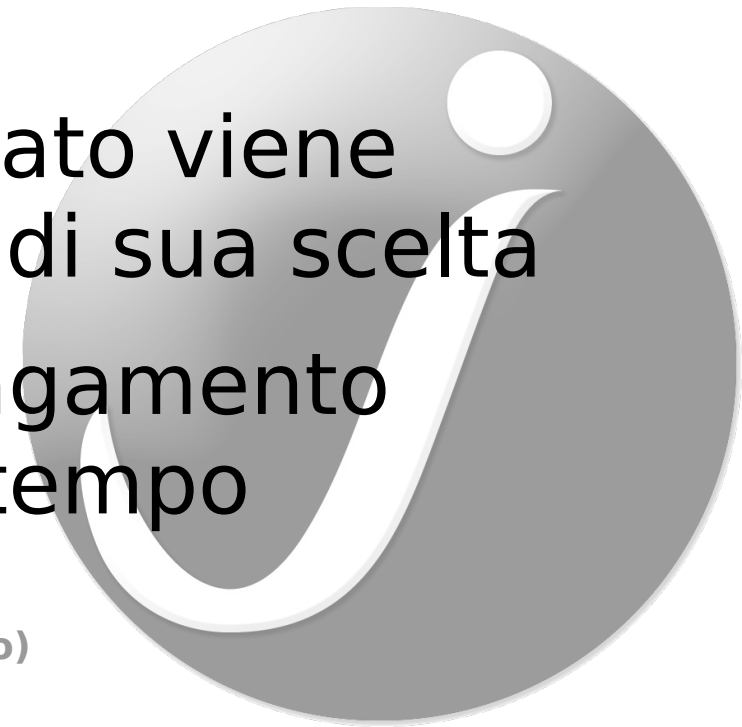


Analisi

Creative Commons Attribution-NonCommercial-ShareAlike 2.5

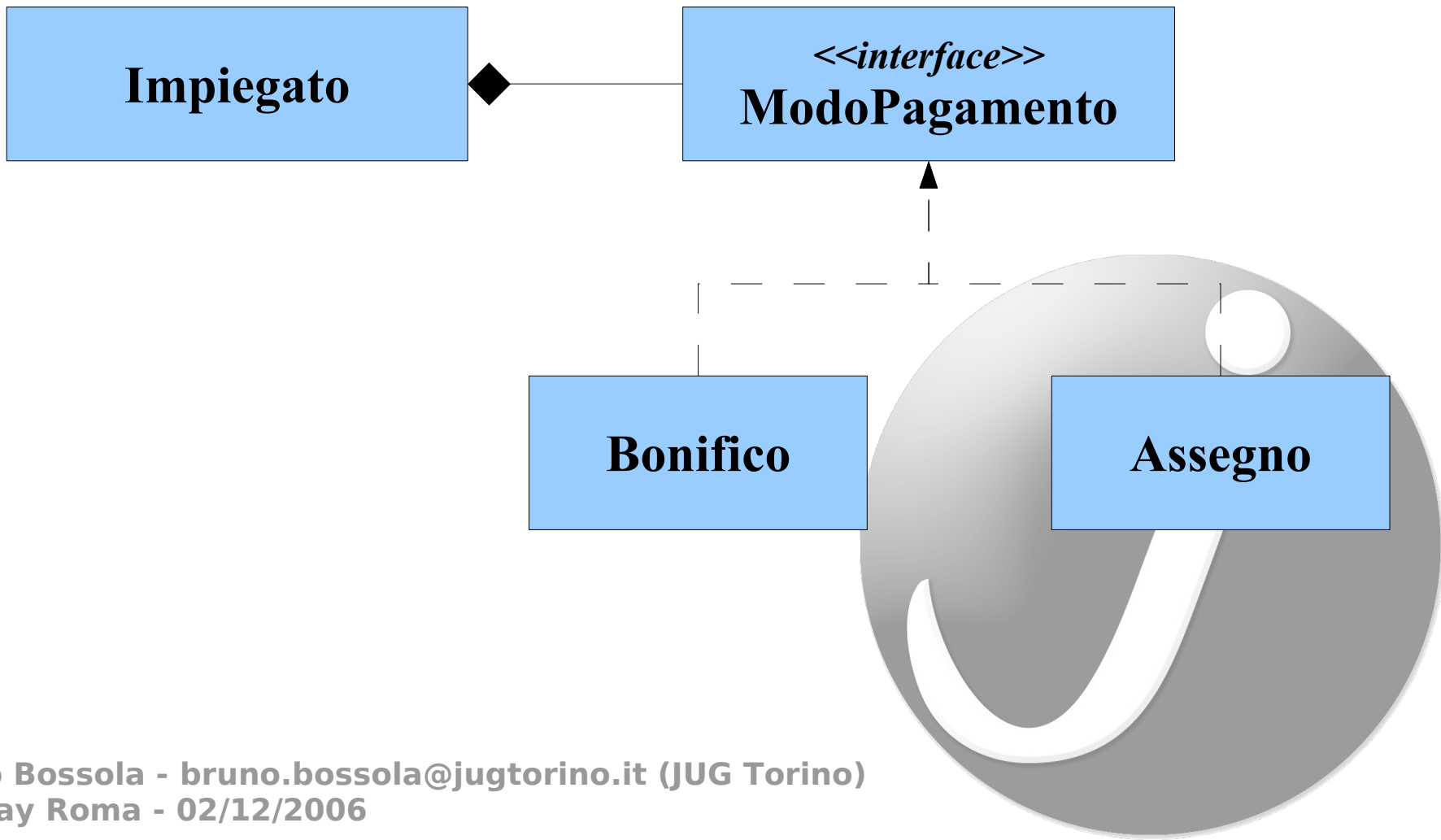
Requisito:

- si può scegliere di essere pagati con assegno o con bonifico
- invariante: ogni impiegato viene pagato con un metodo di sua scelta
- variante: i metodi di pagamento possono cambiare nel tempo



Design!

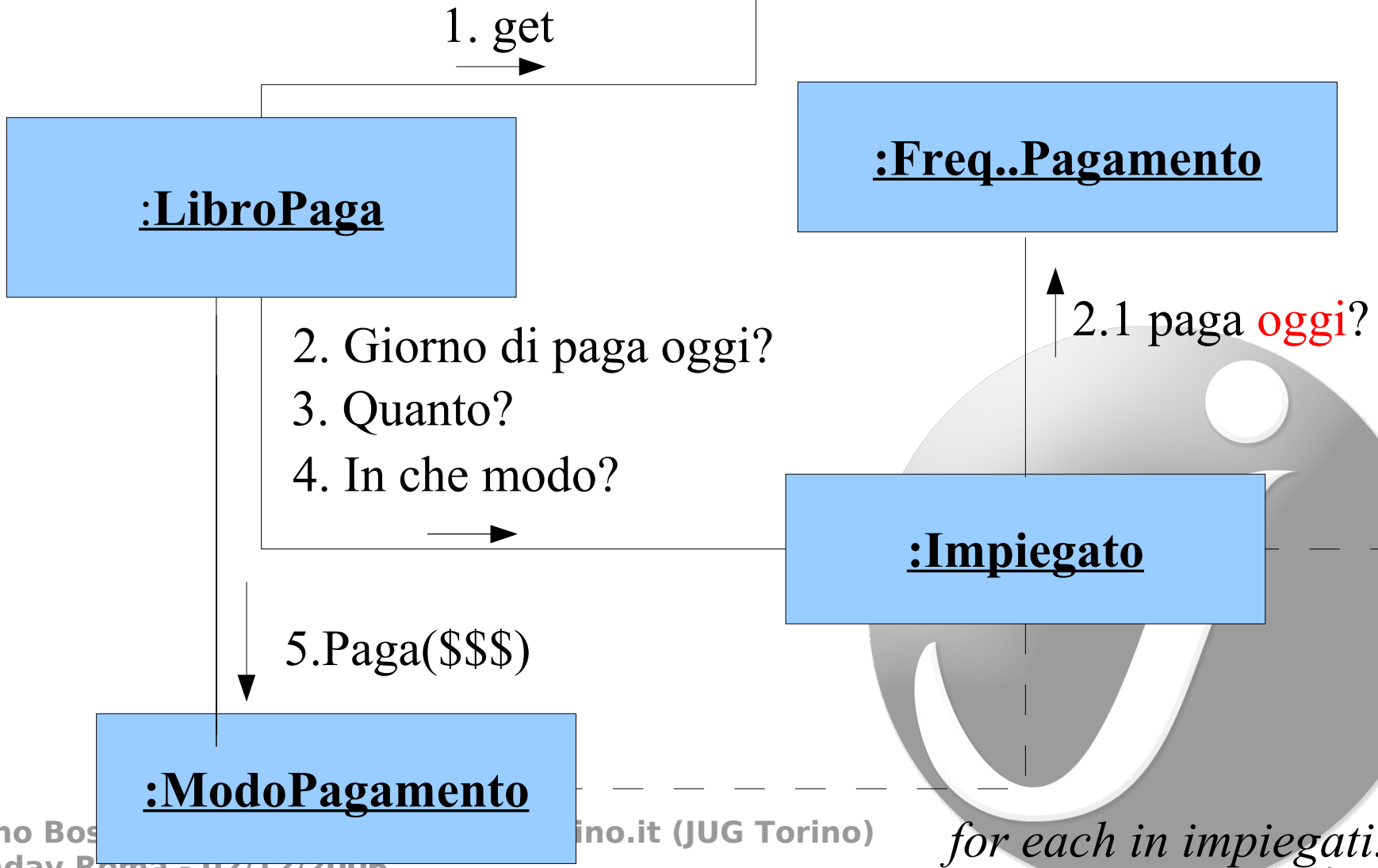
Creative Commons Attribution-NonCommercial-ShareAlike 2.5



Design!

impiegati:Collection

Creative Commons Attribution-NonCommercial-ShareAlike 2.5



for each in impiegati...

I requisiti non funzionali?

Creative Commons Attribution-NonCommercial-ShareAlike 2.5

- non devono inquinare il modello di analisi
- se possibile non devono inquinare il modello di design
- fanno parte della complessità dell'implementazione

Il DB e la GUI si “attaccano” per ultimi!



Perchè “x non credenti?”

Creative Commons Attribution-NonCommercial-ShareAlike 2.5



Bruno Bossola - bruno.bossola@jugtorino.it (JUG Torino)
Javaday Roma - 02/12/2006

Perchè “x non credenti?”

Creative Commons Attribution-NonCommercial-ShareAlike 2.5

- perché tanto non ci credete
- perché tanto non vi ho convinto
- domani scegliete il prossimo framework
- comincerete a progettare partendo dalla gui e dal DB

Così va il mondo :-)



Q&A

Creative Commons Attribution-NonCommercial-ShareAlike 2.5



Bruno Bossola - bruno.bossola@jugtorino.it (JUG Torino)
Javaday Roma - 02/12/2006