Laboratorio di Programmazione

A.A. 2002-2003

Specifiche di Progetto

Luca Padovani
lpadovan@cs.unibo.it

Sommario

Il progetto si compone di due parti. Nella prima parte si richiede l'implementazione di una gerarchia di classi per la realizzazione di semplici applicazioni grafiche interattive. Viene fornito un insieme minimale di interfacce, corrispondenti a semplici elementi grafici quali bottoni, etichette, che gli studenti devono implementare in una corrispondente gerarchia di classi.

Nella seconda parte del progetto si richiede la realizzazione di una calcolatrice grafica che utilizzi i componenti sviluppati nella prima parte del progetto.

1 Regole

Gli studenti sono liberi di adottare l'ambiente di sviluppo che preferiscono. Tuttavia, non è possibile fare uso di librerie di classi esterne a quelle fornite da Java (versioni 1.3.1 e successive), e l'intero progetto deve essere compilabile ed eseguibile anche utilzzando i comandi `javac` e `java` visti a lezione. Allo stesso tempo, essendo lo scopo del progetto quello di creare una gerarchia di componenti grafici, non è ammesso l'uso dei componenti grafici forniti nella libreria di Java, ad eccezione di `JFrame` e `JPanel` usati per creare la finestra principale dell'applicazione e l'area su cui i componenti sviluppati devono essere disegnati.

La valutazione del progetto riguarderà l'architettura interna delle classi, con particolare enfasi sugli aspetti di object-orientation (incapsulamento, uso corretto dell'ereditarietà e del meccanismo di late-binding) e di pulizia del codice, l'effettiva implementazione delle funzionalità richieste nelle due parti del progetto, ed infine, seppur in misura molto minore e solo se i requisiti minimali del progetto sono stati soddisfatti, eventuali estensioni alla libreria di componenti e/o all'applicazione che gli studenti decideranno di apportare.

2 Parte I: componenti dell'interfaccia

Si richiede l'implementazione di un insieme minimo di *componenti grafici*. Tale insieme è costituito da:

- *bottoni* che possono essere premuti;
- *etichette di testo*;
- *aree di testo editabile*.

Quelli elencati sono componenti *foglia*, nel senso di componenti che non contengono al loro interno altri componenti. Occorre però almeno un altro tipo di componente, detto *componente contenitore*, che non ha necessariamente un aspetto "visivo", ma serve ad organizzare geometricamente e gestire sotto-componenti.

Deve infine esistere una *finestra* che rappresenta la radice dell'albero dei componenti che realizzano l'interfaccia grafica. La Figura 1 rappresenta le gerarchie di interfacce e le classi astratte da implementare.

Si ricorda che la gerarchia delle interfacce *non* implica l'implementazione di una gerarchia *isomorfa* di classi. Se lo si ritene opportuno, è possibile raffinare ed estendere la gerarchia di classi sviluppate, sempre che questa implementi (nel senso di `implements` di Java) la gerarchia di interfacce della figura.

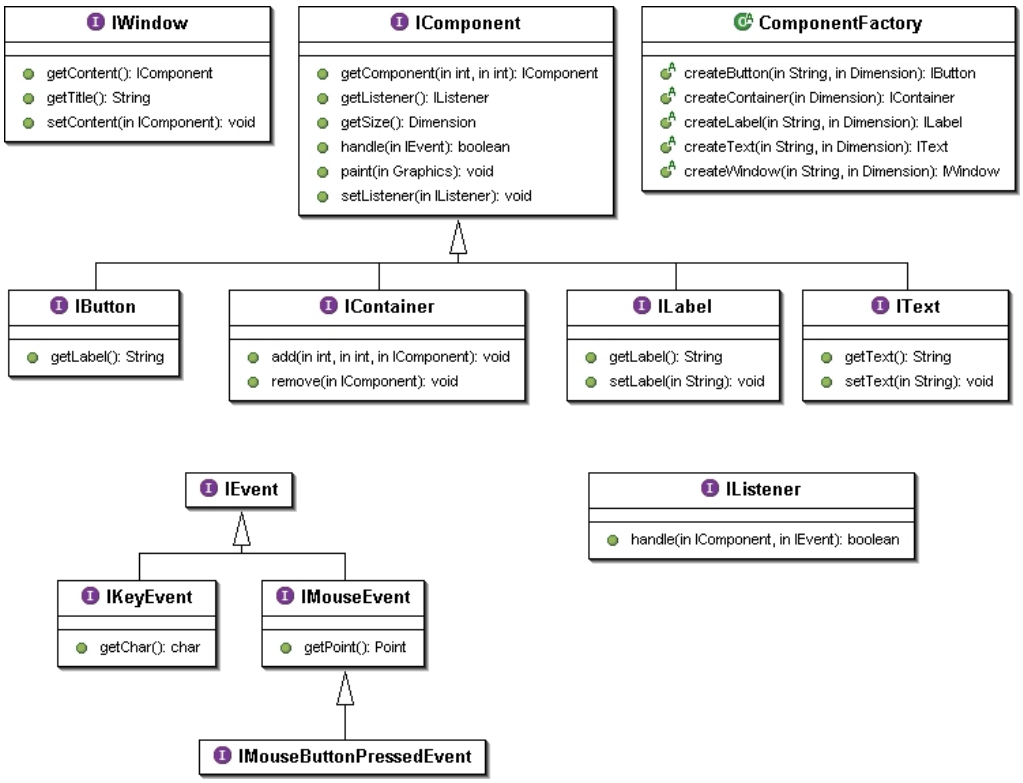

Figura 1: Interfacce e classi da implementare. Le frecce rappresentano la relazione di estensione delle interfacce: $A \rightarrow B$ significa che l'interfaccia A estende l'interfaccia B.

2.1 Struttura della GUI

Una GUI (Graphical User Interface) è rappresentata in memoria come un albero di oggetti. Le foglie dell'albero sono costituite da componenti foglia, mentre i nodi interni dell'albero sono componenti contenitori. La radice dell'albero *deve* essere un componente finestra, cioè un oggetto la cui classe implementa l'interfaccia `IWindow`.

Nota: non si confondano gli alberi di Figura 1 con l'albero di oggetti che rappresenta la GUI. Gli alberi della figura mostrano le *relazioni* tra le interfacce da implementare. L'albero di oggetti della GUI è un insieme di oggetti, ciascuno dei quali è l'istanza di una classe che implementa una delle interfacce di Figura 1 tra quelle con radice `IComponent`.

L'istanziazione dei componenti dell'interfaccia *deve* essere affidata ad una opportuna estensione della classe astratta `ComponentFactory` della figura. Supponendo che `MyComponentFactory` sia il nome di tale classe, deve essere possibile istanziarla invocando il costrutture di default (quello senza argomenti). Ovvero, la creazione dei componenti dell'interfaccia dovrà essere effettutata secondo lo schema seguente:

```
// Creo la factory dei componenti
ComponentFactory factory =
    new MyComponentFactory();

// Chiedo alla factory di istanziare
// i componenti desiderati
IButton button1 =
    factory.createButton(...);

IContainer container =
    factory.createContainer(...);

// ... altri componenti ...
```

2.2 Visualizzazione

La visualizzazione di un componente è affidata al metodo `paint`. Nel momento in cui tale metodo viene invocato su un componente, il componente deve disegnarsi utilizzanto i metodi messi a disposizione dall'oggetto `Graphics` passato come parametro di `paint` ed eventualmente, se il componente è un componente contenitore, invocare il metodo `paint` su tutti i sotto-componenti, dopo aver aggiornato opportunamente lo stato dell'oggetto `Graphics`.[1]

Il processo di visualizzazione deve essere implementato in modo *top-down*. Ovvero, un componente contenitore che "disegna qualcosa" sulla finestra di visualizzazione deve farlo *prima* di invocare il metodo `paint` sui propri sotto-componenti.

2.3 Gestione degli eventi

Gli eventi permettono di rendere l'interfaccia grafica interattiva. Per gli scopi del progetto distingueremo due tipi diversi di eventi:

- pressione di un tasto sulla tastiera;
- pressione di un bottone del mouse.

La "finestra", alla radice dell'albero, deve intercettare gli eventi AWT di Java, produrre (istanziare) oggetti corrispondenti ai tipi di eventi che siamo interessati a considerare, ed inoltrare la richiesta di gestione al componente opportuno. L'inoltro può essere fatto in due modi, in base al tipo di evento: l'evento tastiera deve essere inoltrato ai componenti di tipo `IText` presenti nell'albero GUI. L'evento corrispondente alla pressione di un bottone del mouse deve invece essere inoltrato al componente che si trova correntemente sotto il puntatore del mouse.

2.4 Descrizione sistematica delle interfacce

`IWindow`. Rappresenta la finestra principale dell'applicazione. Notare che `IWindow` non è un componente proprio, infatti la sua interfaccia non estende `IComponent`.

`getContent` ritorna il componente radice dell'albero GUI.

`setContent` setta il componente radice dell'albero GUI.

`getTitle` ritorna il titolo assegnato alla finestra al momento dell'istanziazione.

[1] La classe `Graphics` mette a disposizione un metodo `create` che consente di creare un nuovo oggetto `Graphics` aggiornato in cui l'origine è stata traslata orizzontalmente e verticalmente. La classe che implementa `IContainer` può sfruttare questo metodo per passare un oggetto `Graphics` aggiornato ai propri sotto-componenti.

`IComponent`. Interfaccia base di tutti i componenti:

`getSize` ritorna la dimensione rettangolare del componente. La dimensione viene stabilita una volta per tutte al momento dell'istanziazione del componente, e non può essere modificata in seguito.

`handle` chiede al componente di gestire un evento. Se il componente è effettivamente in grado di gestirlo, invocando l'oggetto listener associato, il metodo deve ritornare `true` e l'evento si dice *consumato*. Se il componente è un componente contenitore e non consuma lui stesso l'evento, allora deve passarlo ai suoi sotto-componenti finché uno di questi lo consuma. Se il componente non ha sotto-componenti o nessuno dei sotto-componenti consuma l'evento, il metodo deve ritornare `false`.

`paint` disegna il componente sullo schermo, insieme a tutti i suoi sotto-componenti se il componente è contenitore.

`getListener` ritorna l'oggetto listener associato al componente, o `null` se nessun listener è associato al componente.

`setListener` associa un listener al componente. Quando il metodo `handle` del componente viene invocato, ed il componente ha un listener associato, il componente invoca il listener con `this` come primo argomento e l'evento ricevuto come secondo.

`getComponent` ritorna il sotto-componente più profondo (o eventualmente il componente stesso) che comprende le coordinate passate come argomento. Ritorna `null` se le coordinate cadono al di fuori del componente su cui il metodo è stato invocato.

`IText`. Area di testo editabile, cioè che può variare nel tempo. L'area di testo deve essere visibilmente delimitata (da un rettangolo o da un'altra decorazione grafica).

`getText` ritorna il testo visualizzato dal componente.

`setText` setta il testo visualizzato dal componente.

`ILabel`. Visualizza una etichetta di testo. Il testo non deve necessariamente essere visibilmente delimitato. In ogni caso una etichetta deve essere facilmente distinguibile da un'area di testo. I metodi sono analoghi a quelli per `IText`, solo con nomi diversi.

`IButton`. Visualizza un "bottone", cioè un componente con una etichetta delimitata. Il bottone deve essere graficamente distinguibile dalle aree di testo e dalle etichette. L'etichetta è stabilita al momento dell'istanziazione del bottone, e non può essere modificata in seguito. L'etichetta deve essere *centrata* rispetto alla dimensione complessiva del bottone.

`IContainer`. Componente contenitore per sotto-componenti. Non ha necessariamente una decorazione grafica, ma deve disegnare tutti i suoi sotto-componenti quando richiesto.

`add` Aggiunge un sotto-componente alle coordinate specificate. Le coordinate fanno riferimento all'*angolo in alto a sinistra* del rettangolo che racchiude il sotto-componente aggiunto.[2]

`remove` Rimuove il sotto-componente indicato.

`IEvent`. Interfaccia base di tutti gli eventi.

`IMouseEvent`. Interfaccia degli eventi del mouse.

`getPoint` Ritorna le coordinate relative al puntatore del mouse al momento della pressione del bottone.

`IMouseButtonPressedEvent`. Interfaccia degli eventi relativi alla pressione di un bottone del mouse.

`IKeyEvent`. Interfaccia degli eventi relativi alla pressione di un tasto della tastiera.

`getChar` Ritorna il carattere corrispondente al tasto premuto. Da notare che ad alcuni tasti non corrisponde un carattere valido.

[2] Prestare particolare attenzione ai metodi di disegno della classe `Graphics`, in quando non tutti seguono la convenzione dell'"angolo in alto a sinistra".

3 Parte II: applicazione

In questa parte del progetto si richiede l'implementazione di una semplice applicazione, una calcolatrice, che utilizzi la libreria di classi definita ed implementata nella prima parte del progetto. Per quanto riguarda l'aspetto grafico di tale applicazione, gli studenti possono prendere spunto dalle applicazioni `xcalc` e `gcalc` normalmente presenti su ogni sistema Linux, ma hanno sostanzialmente completa libertà sulla disposizione dei componenti grafici.

L'insieme minimale di operazioni che la calcolatrice deve supportare include: le 4 operazioni aritmetiche di base (addizione, sottrazione, moltiplicazione, divisione), una operazione per azzerare la calcolatrice, tre funzioni trigonometriche (seno, coseno, tangente). Deve essere possibile inserire costanti numeriche intere con opzionalmente una parte decimale. Sarà dunque necessario predisporre un opportuno bottone ".", oltre a quelli per le cifre decimali e le operazioni richieste. Il bottone "=" deve consentire la valutazione dell'espressione inserita.

Oltre all'uso dell'interfaccia grafica, deve essere possibile inserire le cifre decimali anche premendo sulla tastiera i tasti corrispondenti (da 0 a 9). Infine, il tasto 'c' deve consentire di azzerare la calcolatrice, ed il tasto 'q' deve terminare l'applicazione.