Commenti alla soluzione dell'Esercitazione 6

In primo luogo, per risolvere il nostro problema dobbiamo lavorare con una matrice di elementi grafici: dunque il contenitore giusto è un GridPane. Gli elementi che dobbiamo aggiungere sono eterogenei: cerchi (Circle), quadrati (Rectangle) e triangoli (Polygon). Dobbiamo fare in modo che i click su questi elementi siano gestiti: dunque costruiremo un ascoltatore (uno solo, da specifica del problema) e gli faremo ascoltare tutti gli eventi. Quali eventi? Non gli ActionEvent (che sono ad appannaggio di widgets vari, come Button, ComboBox, TextField e vari Menu ma non degli Shape) ma piuttosto i MouseEvent, ed in particolare in MouseClicked.

Ora, il problema è che nel rispondere agli eventi siamo interessati a delle proprietà delle nostre figure, quali le coordinate ix e iy, e/o a dare a tali figure dei metodi opportuni (es. colora()). A questo punto dobbiamo decidere dove mettiamo queste proprietà e/o metodi. La prima cosa che ci viene in mente è di sottoclassare le figure che ci interessano aggiungendovi tali proprietà, ma se scegliamo questa strada dobbiamo farlo tre volte, una per ciascun tipo di figura, replicando il codice: brutta cosa. Questo è un classico caso nel quale ci farebbe comodo ereditarietà multipla: se la avessimo potremmo definire una classe "AiutanteDiFigura" nella quale mettere proprietà e/o metodi desiderati, e poi creare la classe MyCircle che eredita da Circle e da AiutanteDiFigura, e analogamente per le altre due figure. Ma siamo in Java, e non possiamo seguire questa strada.

Possiamo invece costruire una classe Figura che contenga queste informazioni, ed abbia come variabile di istanza uno Shape nel quale metteremo cerchio, triangolo o quadrato a seconda del caso. L'intuizione è buona, ma il problema che si pone ora è: quando ho un evento e ne recupero il Target ottengo il cerchio/quadrato/triangolo su cui si è cliccato, ma come faccio a sapere a quale Figura appartiene per recuperare le informazioni che vi sono codificate?

Una possibile soluzione è fare in modo che Figura (che contiene il mio cerchio/quadrato/ triangolo) si trovi lungo la catena di contenimento di JavaFX. Per fare questo scelgo di sottoclassare Figura come un tipo di Parent (ad esempio un Pane, o un HBox). A questo punto, per ogni cerchio/quadrato/triangolo, avrò che GridPane contiene, in ogni sua cella, una Figura (sottoclasse di HBox), e questa contiene (aggiunto con getChildren().add) il mio cerchio/quadrato/triangolo.
Ora posso mettere l'ascoltatore su Figura, e quando ottengo l'evento dovrò recuperarne la Source (e non il Target! Perché?). Nella mia Source (che è una Figura) trovo le informazioni/metodi che mi servono e lo Shape a cui applicarle. Il gioco è fatto.

La parte relativa ai KeyboardEvent è più diretta, per questa rimando alla lettura del codice della soluzione.

Due soluzioni alternative:

1) Un paio di vostri colleghi hanno ideato una soluzione che, nel caso specifico, risolve il problema in modo. Notando che ogni Node ha un campo "id" (e ovviamente Circle, rectangle e Polygon sono dei Node), hanno pensato di codificarvi le informazioni necessarie: in questo caso basta fare riga*10+colonna (oppure riga concatenato con

colonna, visto che id è una stringa) per avere un codice dal quale recuperare le informazioni. Bene, ma se avessimo avuto un caso nel quale dovevamo mantenere informazioni più articolate questa strada non sarebbe stata percorribile, almeno non in modo così diretto (ma avremmo sempre potuto crearci una mappa (associazione chiave – valore) nella quale usare l'ID (univoco!) come chiave per recuperare un oggetto che contenga tutte le informazioni necessarie).

2) Potremmo mettere la logica che ci serve nell'ascoltatore: data una figura, possiamo risalire alla sua posizione cercandola nel GridPane. Purtroppo nelle API di GridPane questa funzionalità non è presente, ma non è difficile implementarla (si veda la slide 28 nella lezione 13_static_layout del 20 aprile). Questo ci permette di ottenere le coordinate che ci servono. Nel nostro caso è sufficiente, se avessimo avuto bisogno di informazioni addizionali avremmo potuto costruirci una matrice di oggetti (ad esempio di una classe "InformazioniDiFigura" nella quale immagazzinare le informazioni necessarie, recuperandole poi dalla matrice stessa a partire dalle coordinate.