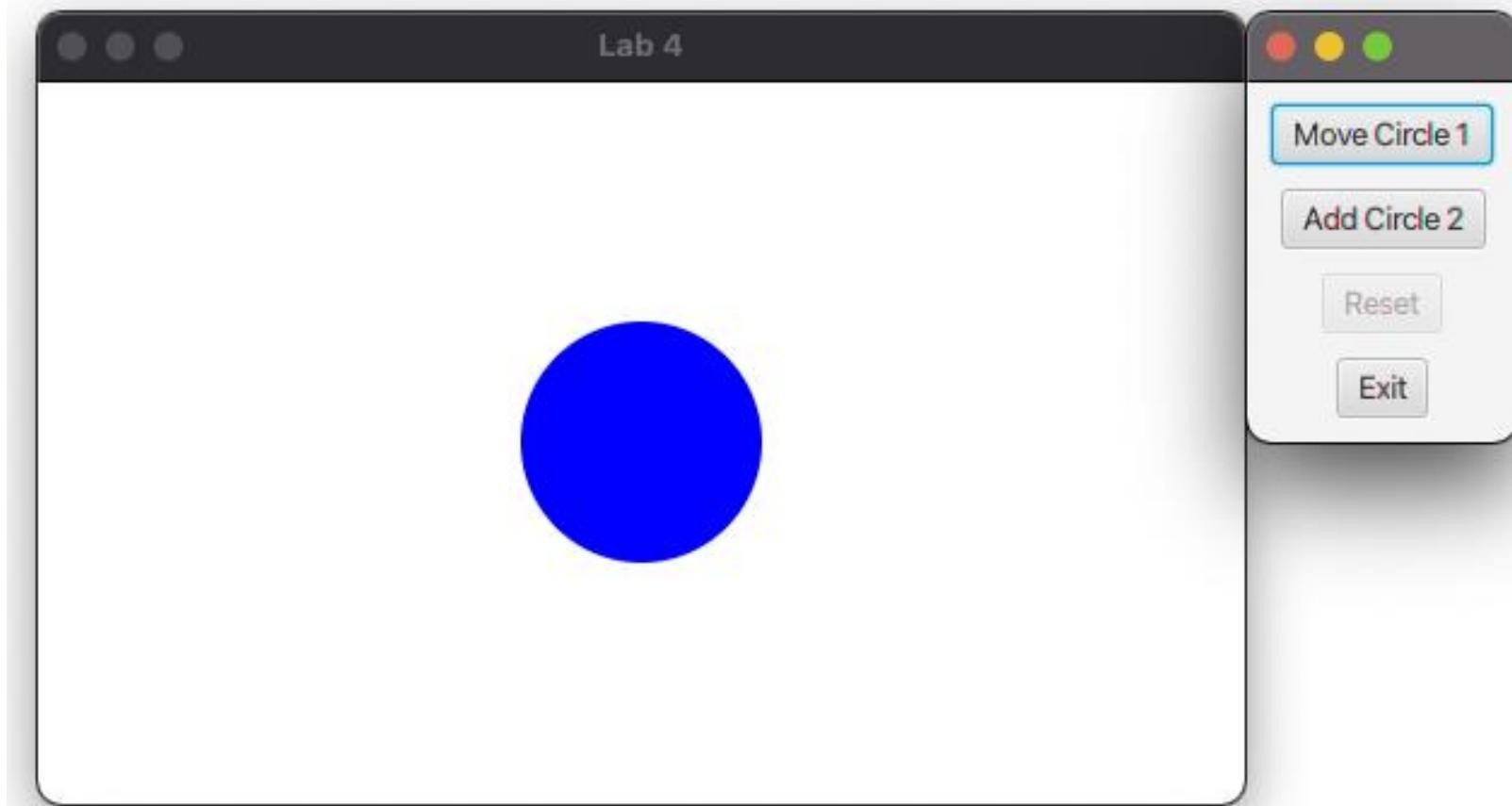


Laboratorio 4

Gian Pietro Picco

... in preparazione del Lab 5



Muovere elementi sullo schermo (1)

- ✖ All'avvio dell'applicazione, vengono visualizzate:
 - la finestra principale di 500 x 300 pixel, contiene il **cerchio 1** (blu) con raggio 50 pixel, posizionato nel centro
 - la finestra secondaria contenente i bottoni
- ✖ Il cerchio 1 viene spostato di 10 pixel:
 - a sinistra: quando il mouse entra nel cerchio
 - a destra: quando si clicca il bottone «Move Circle 1»
- ✖ Come creare un cerchio e spostarne il centro: Javadoc
- ✖ Suggerimenti:
 - Quali eventi devono essere gestiti e come?
 - Procedete per gradi:
 - ◆ prima costruite la scena con il cerchio
 - ◆ poi aggiungete una reazione semplice (es. stampa su console)
 - ◆ infine, trasformatela nella reazione desiderata (spostamento del cerchio)

Gestire due finestre...

- ✖ Ricordate: il **primaryStage** vi viene passato come parametro in **start()** mentre il secondo va creato esplicitamente
- ✖ Il cerchio va inserito in un contenitore, a sua volta passato alla scena: usate **Group**
- ✖ Per la seconda finestra, vi basta un layout molto semplice; potete usare **setSpacing** e **setPadding** per la spaziatura dei bottoni
- ✖ Vedere esempi su slide!

Gestire eventi da tastiera

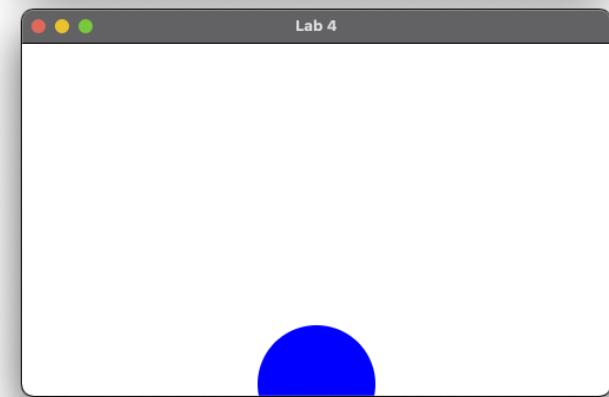
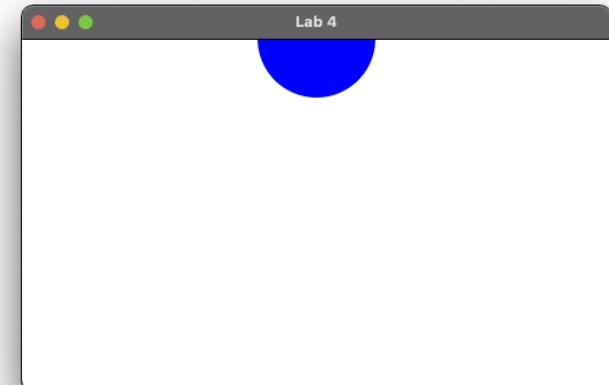
- ✖ Sono definiti dalla classe **KeyEvent**
- ✖ Solitamente serve reagire alla pressione di un tasto
 - **KeyPressed** oppure **KeyTyped**
- ✖ Sono disponibili *convenience method*:
 - es., **setOnKeyPressed**
- ✖ Per conoscere il codice del tasto a cui è associato l'evento e si usa il metodo **e.getCode()**
 - Il tipo del valore ritornato è **KeyCode**
 - è un'enumerazione e può essere usato come un **int**
 - Ad esempio all'interno di un'istruzione **switch** per eseguire azioni diverse in corrispondenza di tasti diversi

Muovere elementi sullo schermo (2)

- ✖ Ora aggiungiamo anche la tastiera...
- ✖ Premendo il tasto «I» o «K» il cerchio 1 si sposta di 10 pixel in alto o in basso, rispettivamente
- ✖ Suggerimenti:
 - In quale elemento dell'interfaccia è rilevata e gestita la pressione del tasto?
 - ◆ Dove devo chiamare il metodo `setOnKeyPressed`?
 - Prima stampate il tasto premuto in console e poi aggiungete il movimento del cerchio
 - Attenzione all'orientamento dell'asse delle y ...

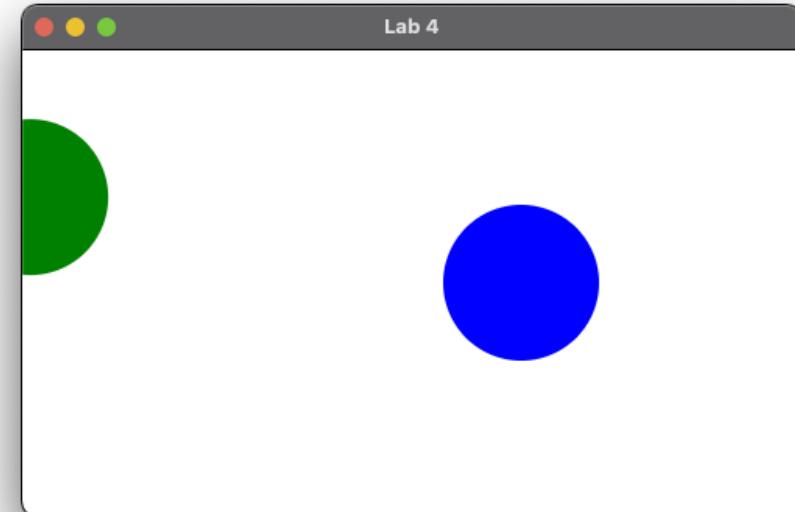
Muovere elementi sullo schermo (3)

- ✖ Aggiungiamo il «wrap around»:
quando il cerchio esce da un lato della finestra,
riappare su quello opposto
- ✖ Suggerimenti: attenzione...
 - ... alle dimensioni della finestra ...
 - ... all'orientamento degli assi ...
 - ... e al fatto che il centro e raggio
di un **Circle** sono **Double**



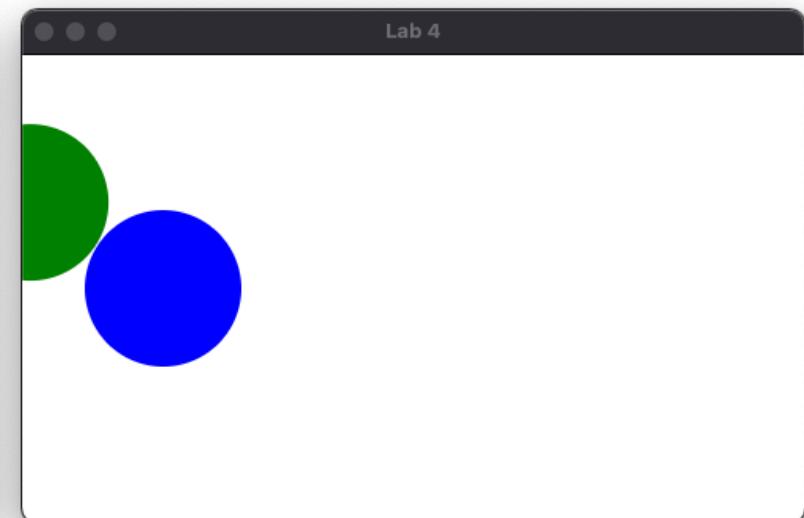
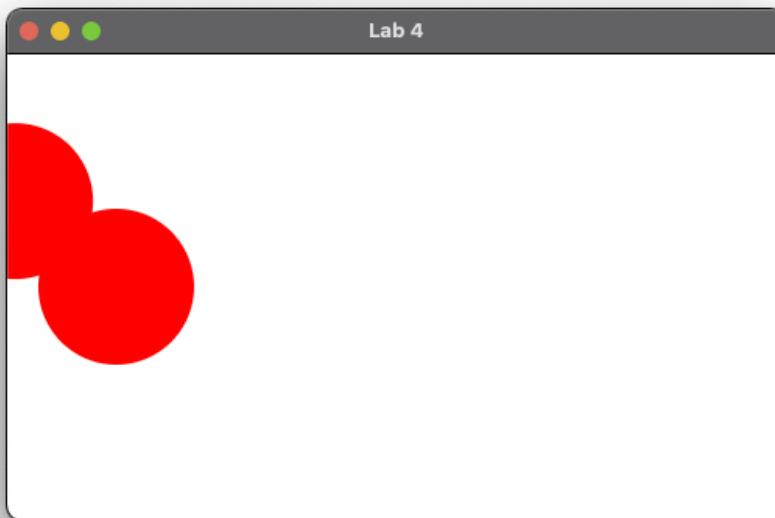
Aggiungere il secondo cerchio

- ✖ Quando si preme «Add Circle 2» viene aggiunto un secondo cerchio di colore verde
 - Su quale elemento dell'interfaccia aggiungerlo e come?
- ✖ La posizione del centro è **casuale** ma deve rimanere **dentro la finestra**
 - potete usare la classe **Random** (in **java.util**)
 - fornisce metodi per generare
 - ◆ un **int** (tra 0 e un max desiderato)
 - ◆ un **double** (tra 0 e 1)
- ✖ Il bottone viene disabilitato e viene invece abilitato «Reset»



Rilevare collisioni

- ✖ Quando il cerchio blu viene mosso in modo da intersecare quello verde ambedue diventano di colore rosso
- ✖ Quando la “collisione” viene rimossa, ambedue ritornano al colore originale



Rimuovere il secondo cerchio

- ✖ La pressione del tasto «Reset» riporta l'applicazione alla situazione originale
 - il cerchio 1 viene riposizionato al centro della finestra con il suo colore blu
 - il cerchio 2 viene rimosso dalla finestra
- ✖ Il bottone viene disabilitato e viene invece abilitato «Add Circle 2»
 - Come all'avvio dell'applicazione
- ✖ Suggerimenti:
 - Come rimuovere il cerchio 2? Ricordate che il metodo `getChildren()` ritorna una lista...