Esercitazione 5

Programmazione 2

Nota: per esclusivo uso interno al corso, riferimenti bibliografici forniti a lezione

Esercizio

Riprendere l'esercizio carte da gioco e implementare il metodo clone (deep copy) nelle classi:

- Carta
- MazzoDiCarte

Esercizio – clone in Carta

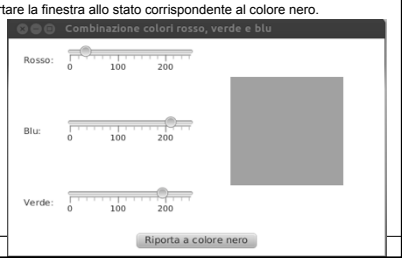
```
public class Carta implements Comparable, Cloneable {

    protected String seme;
    protected int valore;

    /*...skip code */

    @Override
    public Object clone()   {
        Carta copia = null;
        try {
            copia = (Carta) super.clone();
        } catch (CloneNotSupportedException ex) {
            System.err.println("ERRORE: L'oggetto "
                    + this + "non puo' essere clonato.");
        }
        copia.seme = new String(this.seme);
        return copia;
    }
}
```

Esercizio – clone in MazzoDiCarte

```
public class MazzoDiCarte implements Cloneable{

  private LinkedList carte;

  /*  skip code ... */

  @Override
    public Object clone()   {
        //crea una shallow copy dell'oggetto this
        MazzoDiCarte copia = null;
        try {
            copia = (MazzoDiCarte) super.clone();
        } catch (CloneNotSupportedException ex) {
            System.err.println("ERRORE: L'oggetto " +
                    this + "non puo' essere clonato.");
        }
        //crea una shallow copy della lista di carte
        LinkedList copiaLista = (LinkedList) carte.clone();
        //sostituisce le carte nella copia della lista con
        //copia delle carte
        for (Iterator it = carte.iterator(); it.hasNext();) {
          Carta carta = (Carta) it.next();
          copiaLista.set(carte.indexOf(carta),carta.clone());
        }
        copia.carte = copiaLista;
        return copia;
    }
}
```

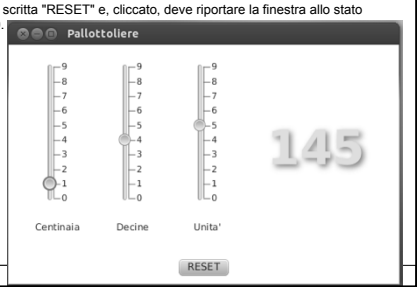
Esercizio

Creare un'applicazione che mostri una finestra (che all'inizio deve avere dimensioni 500*300) contenente tre controlli scorrevoli, un'area colorata, un pulsante e descrizioni opportune.
I tre selettori devono essere associati ai tre valori rosso, verde e blu del modello di colori RGB.
L'area colorata deve visualizzare in ogni momento il colore determinato dalla combinazione dei tre valori.
Un clik sul pulsante deve riportare la finestra allo stato corrispondente al colore nero.

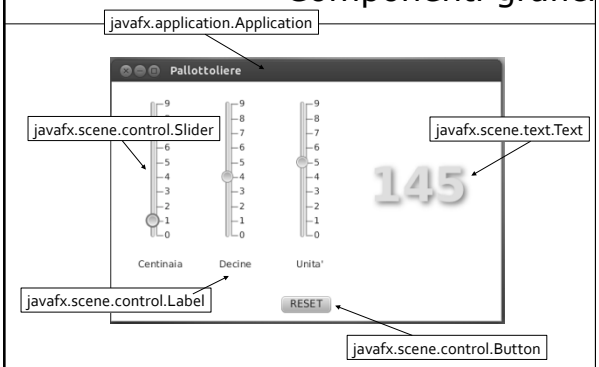


Esercizio / alternativa

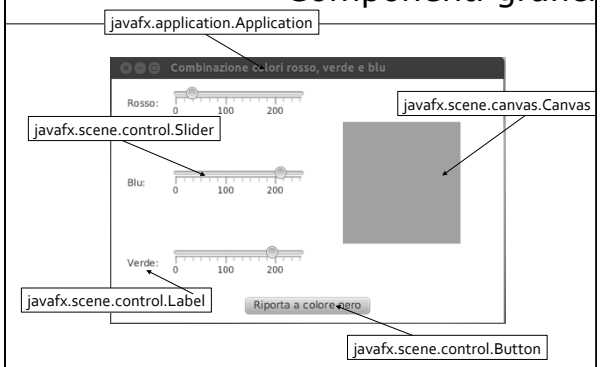
Creare un'applicazione che mostri una finestra (che all'inizio deve avere dimensioni 500*300) contenente tre controlli scorrevoli, un testo, un pulsante e descrizioni opportune.
I tre selettori devono essere associati a unita', decine e centinaia.
Il testo deve visualizzare in ogni momento il numero determinato dalla somma di unita', decine e centinaia.
Il pulsante deve riportare la scritta "RESET" e, cliccato, deve riportare la finestra allo stato corrispondente al numero 0.



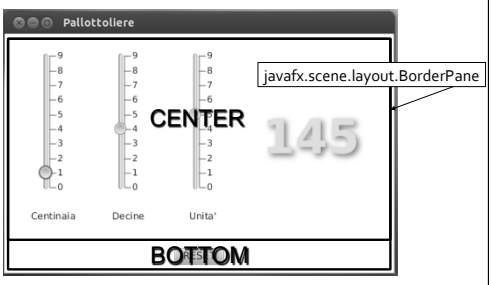
Componenti grafici



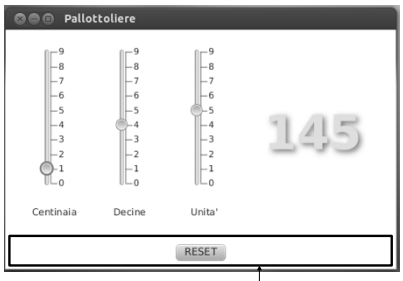
Componenti grafici



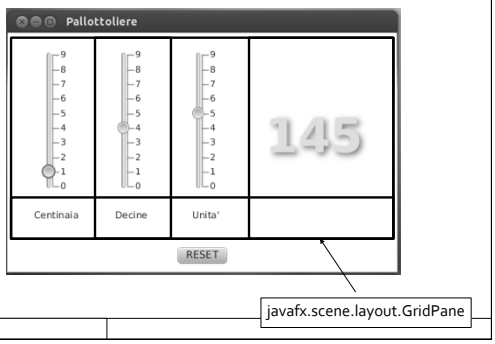
Impaginazione



Impaginazione

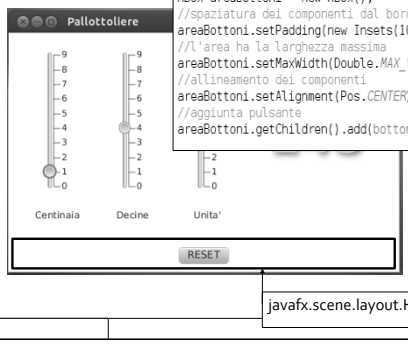


Impaginazione

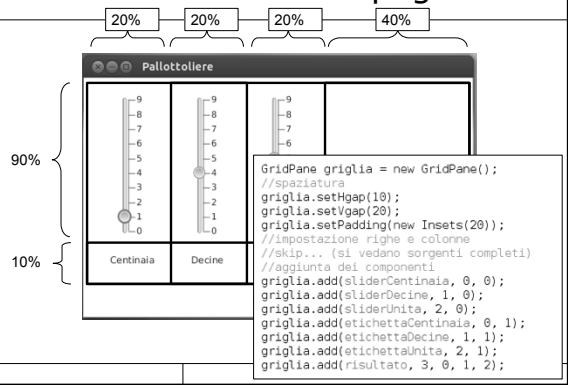


Impaginazione

```
HBox areaBottoni = new HBox();
//spaziatura dei componenti dal bordo
areaBottoni.setPadding(new Insets(10));
//l'area ha la larghezza massima
areaBottoni.setMaxWidth(Double.MAX_VALUE);
//allineamento dei componenti
areaBottoni.setAlignment(Pos.CENTER);
//aggiunta pulsante
areaBottoni.getChildren().add(bottoneReset);
```

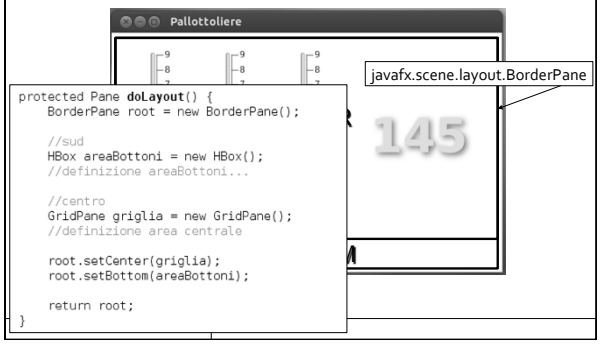


Impaginazione



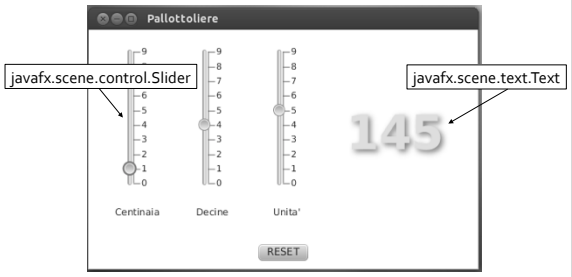

```
GridPane griglia = new GridPane();
//spaziatura
griglia.setHgap(10);
griglia.setVgap(20);
griglia.setPadding(new Insets(20));
//impostazione righe e colonne
//skip... (si vedano sorgenti completi)
//aggiunta dei componenti
griglia.add(sliderCentinaia, 0, 0);
griglia.add(sliderDecine, 1, 0);
griglia.add(sliderUnita, 2, 0);
griglia.add(etichettaCentinaia, 0, 1);
griglia.add(etichettaDecine, 1, 1);
griglia.add(etichettaUnita, 2, 1);
griglia.add(risultato, 3, 0, 1, 2);
```

Impaginazione

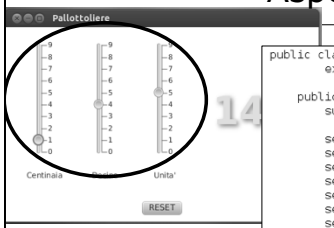



```
protected Pane doLayout() {
    BorderPane root = new BorderPane();

    //sud
    HBox areaBottoni = new HBox();
    //definizione areaBottoni...

    //centro
    GridPane griglia = new GridPane();
    //definizione area centrale

    root.setCenter(griglia);
    root.setBottom(areaBottoni);

    return root;
}
```

Componenti grafici



Aspetto componenti




```
public class SelettoreValore
        extends javafx.scene.control.Slider {

    public SelettoreValore() {
        super(0, 9, 0);

        setShowTickMarks(true);
        setShowTickLabels(true);
        setSnapToTicks(true);
        setMajorTickUnit(1);
        setMinorTickCount(0);
        setBlockIncrement(1);
        setOrientation(Orientation.VERTICAL);
    }
}
```

```
protected javafx.scene.control.Slider
        sliderUnita, sliderCentinaia, sliderDecine;

void inizializzaSlider() {
    sliderUnita = new SelettoreValore();
    sliderDecine = new SelettoreValore();
    sliderCentinaia = new SelettoreValore();
}
```

Aspetto componenti

```
protected javafx.scene.text.Text risultato;

void inizializzaTesto() {
    risultato = new Text("0");
    //impostazione colore, font e dimensione
    risultato.setFill(Color.GREENYELLOW);
    risultato.setFont(Font.font("Arial", FontWeight.BOLD, 60));
    //effetto ombra
    DropShadow ombra = new DropShadow();
    ombra.setOffsetX(3.0);
    ombra.setOffsetY(3.0);
    ombra.setColor(Color.GRAY);
    risultato.setEffect(ombra);
}
```

Intercettazione dell'evento di modifica del valore dei selettori

```
protected void creaAscoltatoreSelettori() {
    //istanzia un ascoltatore di tipo ChangeListener
    ChangeListener eh = new ChangeListener() {

        @Override
        public void changed(ObservableValue observable,
                Object oldValue, Object newValue) {
            aggiornaValoreNumerico();
        }

    };
    //registra l'ascoltatore sulla proprieta' valore
    //dei tre selettori
    sliderCentinaia.valueProperty().addListener(eh);
    sliderDecine.valueProperty().addListener(eh);
    sliderUnita.valueProperty().addListener(eh);
}
```

Definizione di una classe anonima che implementa ChangeListener

```
protected void aggiornaValoreNumerico() {
    //legge i valori correnti dei tre selettori
    double cent = sliderCentinaia.getValue();
    double dec = sliderDecine.getValue();
    double unit = sliderUnita.getValue();
    //calcola il risultato
    int numero = (int) (unit + (10 * dec) + (100 * cent));
    //lo visualizza sul componente testo
    risultato.setText("" + numero);
}
```

Riferimenti

Eseguire copie di oggetti:

Eckel, Thinking in Java → appendice A →
"Making local copies"

Layout dei componenti in JavaFX:

Sito oracle.com → JavaFX tutorials →
"Working with Layouts in JavaFX"

Componenti grafici in JavaFX:

Sito oracle.com → JavaFX tutorials →
"Using JavaFX UI Controls"