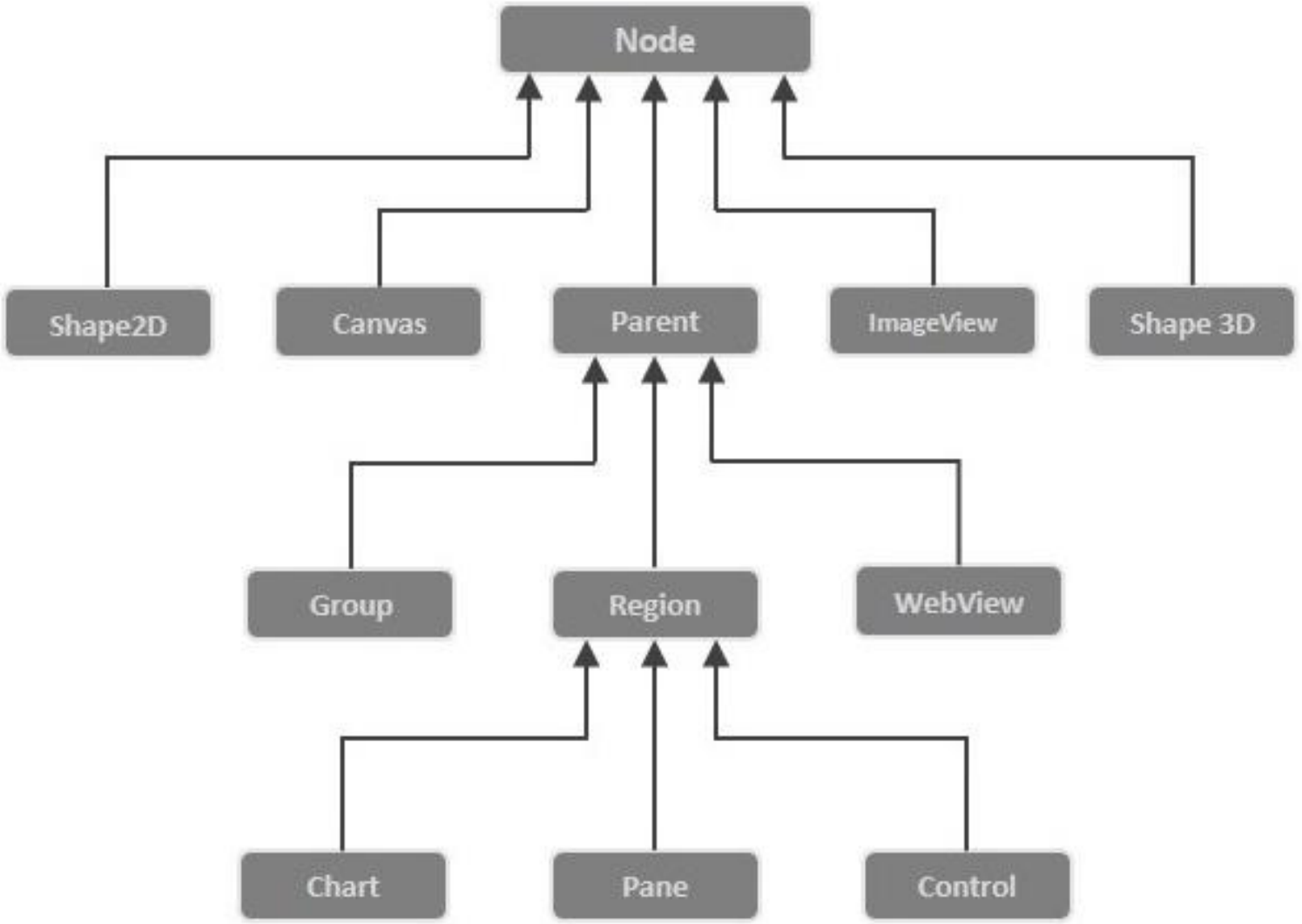
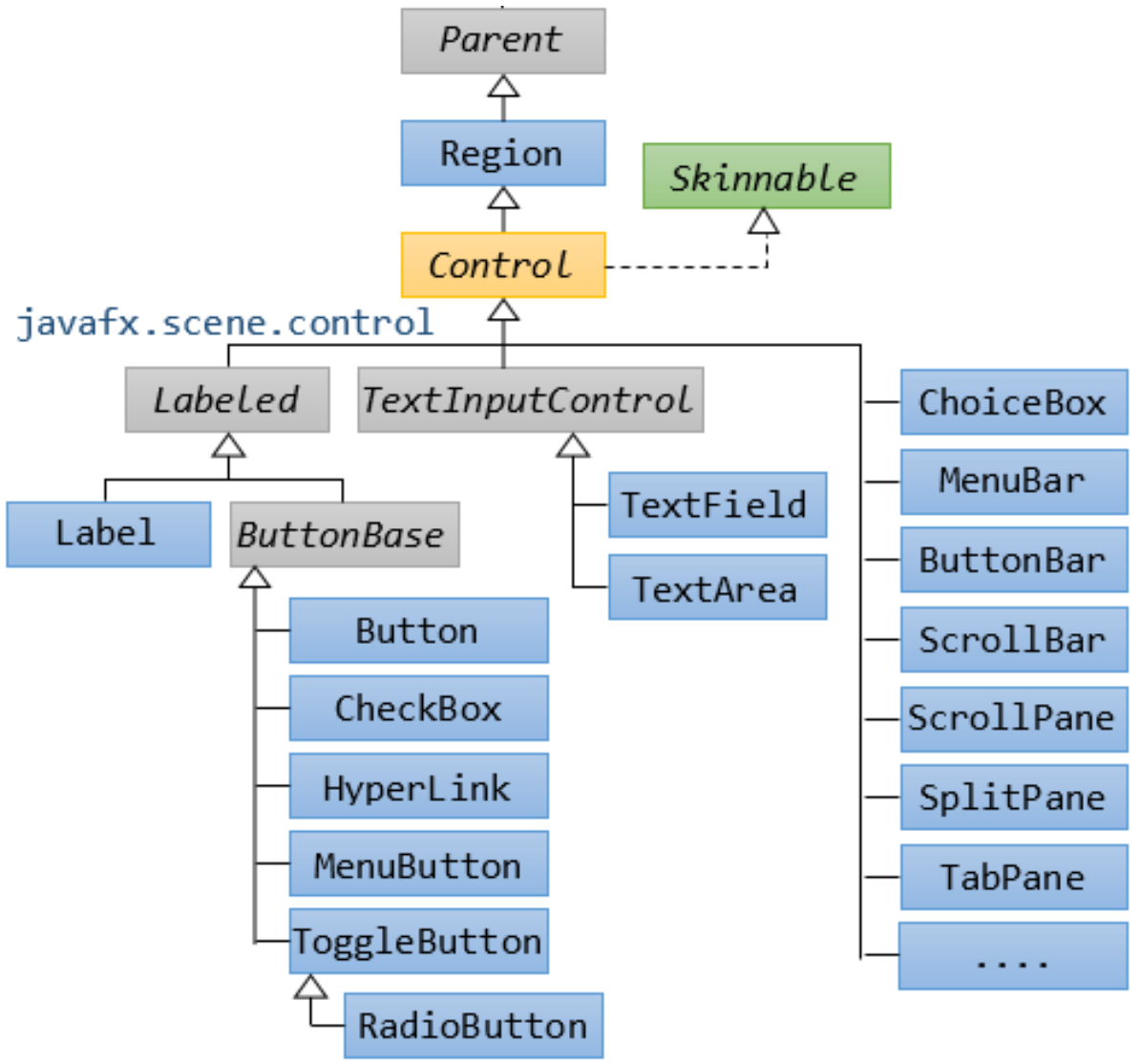
Компоненты JavaFX

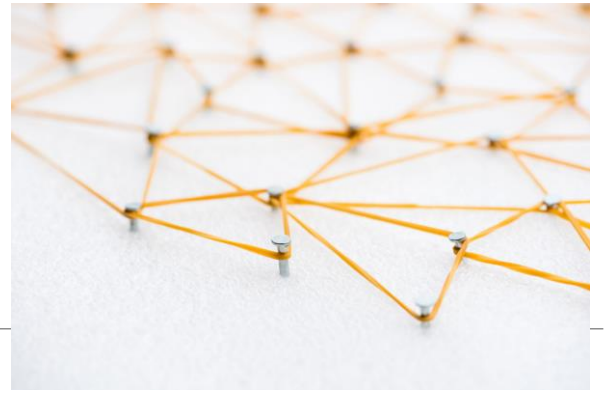
Узлы сцены


Node
Shape2D
Canvas
Parent
ImageView
Shape 3D
Group
Region
WebView
Chart
Pane
Control

Controls


Parent
Region
Skinnable
Control
javafx.scene.control
Labeled
TextInputControl
ChoiceBox
MenuBar
ButtonBar
ScrollBar
ScrollPane
SplitPane
TabPane
....
Label
ButtonBase
TextField
TextArea
Button
CheckBox
HyperLink
MenuButton
ToggleButton
RadioButton

Узлы сцены



Однотипные узлы сцены можно хранить в коллекциях

Как получить доступ к конкретному узлу сцены из коллекции?

Любому узлу можно назначить идентификатор и пользовательские данные

void setId (String value)

void setUserData(Object value)

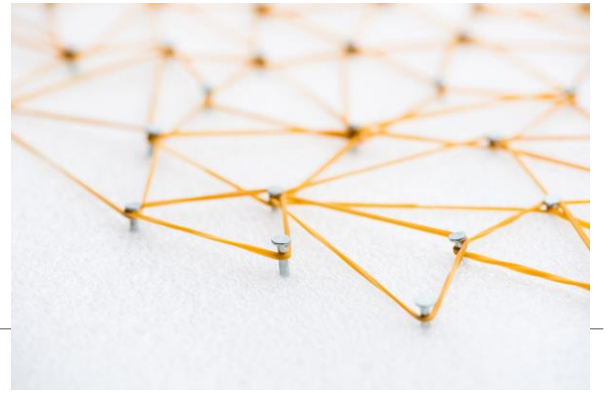
И опросить их через методы

String getId()

StringProperty idProperty()

Object getUserData()

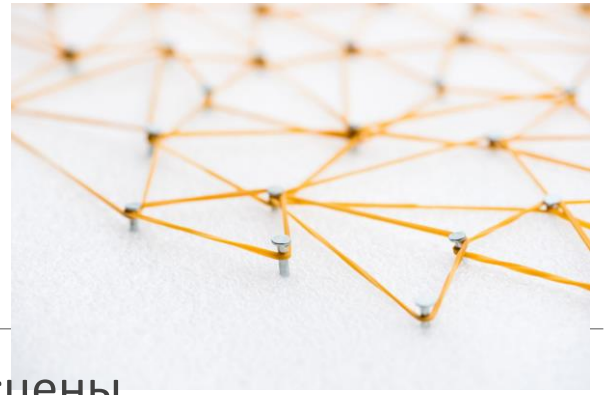
Узлы сцены



Можем проанализировать нажатие любой кнопки на сцене

```
scene.addEventHandler(ActionEvent.ACTION, event -> {
  if (event.getTarget() instanceof Button) {
            Button b = (Button) event.getTarget();
            System.out.println( b.getId());
            System.out.println( b.getIText());
            System.out.println( b.getId());
       }
 });
```

Узлы сцены



Имея ссылку на сцену, можно получить доступ ко всем узлам сцены

```
//Scene s= b.getScene();
Parent r = s.getRoot();
for (Node n: r.getChildrenUnmodifiable() ){
        System.out.println("  ->"+ n);
}
```

Выполняем и разбираем пример App1.java

Основные компоненты

Label

Button

ToggleButton

ToggleGroup

RadioButton

CheckBox

Hyperlink

Slider

ProgressIndicator

ProgressBar

ScrollBar

ScrollPane

ToolBar

Separator

Выполняем и разбираем пример Buttons.java

JavaFX свойства

ДОСТУП К ПОЛЮ

```
scene.setRoot(pane);
System.out.println(scene.getRoot());
```

ДОСТУП К СВОЙСТВУ

```
import javafx.beans.property.*;

scene.rootProperty().setValue(pane);
System.out.println(scene. rootProperty(). getValue());
```

Обработчики свойств

ИНТЕРФЕЙСЫ

***InvalidationListener* – недостоверное значение свойства**

***Observable* – назначение/удаление обработчика недостоверности свойства**

```
InvalidationListener inv = new InvalidationListener(){
 @Override
 public void invalidated (Observable o){
          System.out.println(“window invalidated”);
}};
window.widhtProperty().addListener(inv);

//для удаления обработчика
button.setOnAction(event->{
window.widthProperty().removeListener(inv);
});
```

```
window.widhtProperty().addListener(observable -> {
  System.out.println(“window invalidated”);
  //System.out.println(window.getWidth());
  });
  // «ленивые» вычисления
```

Обработчики свойств

ИНТЕРФЕЙСЫ

***ChangeListener* – изменение значения свойства**

***ObservableValue*– назначение/удаление обработчика изменения свойства**

ChangeListener Описывает событие изменения свойства

Метод

void changed (ObservableValue <T> ov, T oldValue, T newValue)

Обработчики свойств

ИНТЕРФЕЙСЫ

***ChangeListener* – изменение значения свойства**

***ObservableValue*– назначение/удаление обработчика изменения свойства**

```
ChangeListener<? super Number>      changeLlistener = new ChangeListener<Number>() {
        @Override
        public void changed(ObservableValue<? extends Number> observable,
                        Number oldValue, Number newValue) {
                System.out.println("ChangeListener " + observable.getValue());
                System.out.println("oldValue = " + oldValue);  System.out.println("newValue = " + newValue);
        } };
    slider.valueProperty().addListener(changeLlistener);
// используя лямбда-выражение
slider.valueProperty().addListener((obj, oldValue, newValue) -> {
        rect.setTranslateX(newValue.doubleValue());
    });
```

Связывание обработчиков

Двунаправленное связывание

```
rect.translateXProperty().bindBidirectional(slider.valueProperty());

textField.textProperty().bindBidirectional(slider.valueProperty(),
                        new NumberStringConverter());

textField2.textProperty().bindBidirectional(slider.valueProperty(),
                        NumberFormat.getInstance());
```

Связывание обработчиков

Однонаправленное связывание

```
slider2.valueProperty().bind(slider.valueProperty());
```

Связывание через выражение

```
slider2.valueProperty().bind(slider.valueProperty().add(50));
//add()  subtract()  multiply()  divide()   negate()
```

Выполняем и разбираем пример AppProp.java