CSCI 2120:
Software Design & Development II

UNIT4: UI management

GUI framework

JavaFX Core: Node

Overview

1. Introduction
2. JavaFX Node Basics
3. JavaFX Node Properties
4. JavaFX Node Coordinate System
5. JavaFX Node Bounding Box
6. layoutX and layoutY
7. Preferred Width and Height
8. Minimum Width and Height
9. Maximum Width and Height
10. User Data
11. Items or Child Nodes

Introduction

- The *JavaFX* `Node` class, `javafx.scene.Node`, is the base class (superclass) for all components added to the **JavaFX Scene Graph**.
- The JavaFX `Node` class is abstract, so you will only add subclasses of the `Node` class to the scene graph.
- All JavaFX `Node` instances in the scene graph share a set of common properties which are defined by the JavaFX `Node` class.

These common properties will be covered in this JavaFX `Node` lecture.

JavaFX Node Basics

- Each JavaFX `Node` (subclass) instance can only be added to the JavaFX scene graph once. In other words, each `Node` instance can only appear in one place in the scene graph. If you try to add the same `Node` instance, or `Node` subclass instance, to the scene graph more than once, JavaFX will throw an exception!

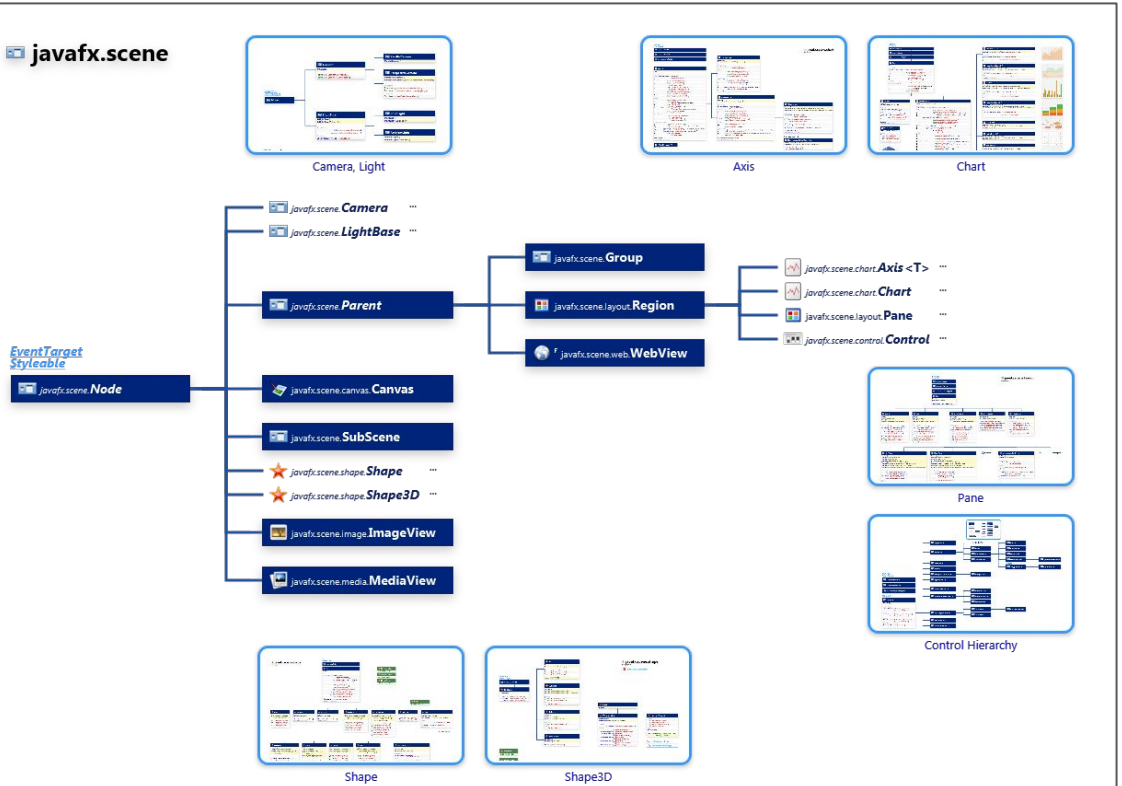
- A JavaFX `Node` can sometimes have sub-items - which are also called `children`. Whether a given `Node` instance can have children or not depends on the concrete `Node` subclass. A special subclass of JavaFX `Node` named `Parent` is used to model `Node` instance which can have `children`. Thus, `Node` instances that can have `children` are typically `children` of the `Parent` class - and not the `Node` class directly.

- The **JavaFX Stage** and **JavaFX Scene** classes are not subclasses of the JavaFX `Node` class. While these two classes are used to display the JavaFX scene graph, only `Node` instances added to a JavaFX `Scene` instance are considered part of the JavaFX Scene graph.

- Once a `Node` instance is attached to the scene graph, only the JavaFX Application Thread, the thread managing the JavaFX scene graph, is allowed to modify the `Node` instance.

Node class hierarchy

- Node
  - Shape
    - Text
    - Line
    - Rectangle
    - Circle
    - Polygon
  - Parent: node that contains other nodes
    - Group
    - Region: parent that covers a specified area
      - Control: some useful predefined GUI widgets
        - Accordion
        - ChoiceBox
        - Separator
        - Labeled
          - ButtonBase
            - Button
            - CheckBox
            - Hyperlink
        - TextInputControl
          - TextArea
          - TextField
      - Pane
        - HBox
        - VBox
        - StackPane
        - GridPane
        - FlowPane
        - BorderPane
      - Chart
      - Axis
  - Canvas
  - ImageView
  - SwingNode



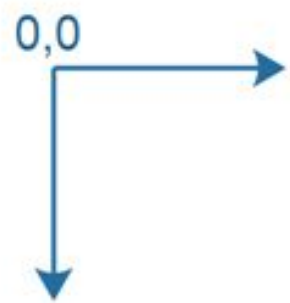
JavaFX Node Properties

The JavaFX Node class, and thus all subclasses of Node, has the following common properties:

- A cartesian coordinate system
- A bounding box delimited by:
  - Layout bounds
  - Bounds in local
  - Bounds in parent
- layoutX
- layoutY
- Preferred height
- Preferred width
- Minimum height
- Minimum width
- Maximum height
- Maximum width
- User data
- Items (Child nodes)

JavaFX Node Coordinate System

Each JavaFX Node has its own cartesian coordinate system. The only difference from a regular cartesian coordinate system is that the Y axis is reversed. That means, that the origin of the coordinate system is in the upper left corner of the coordinate system. As Y values increase, the point moves down from the top of the coordinate system. This reversal of the Y axis is normal in 2D graphics coordinate systems.

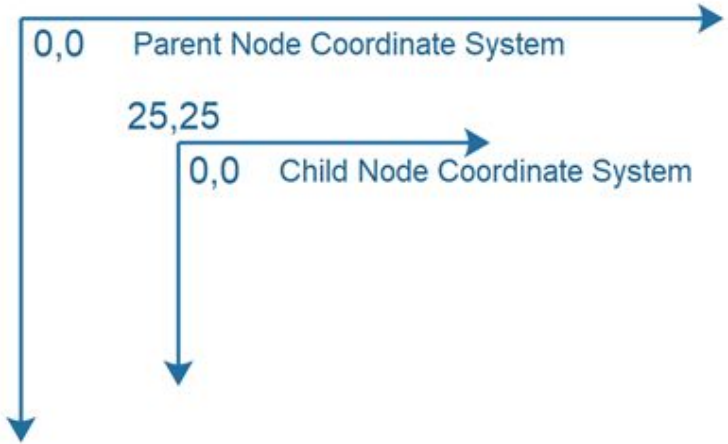


It is possible for a JavaFX Node to have negative X and Y coordinates.

JavaFX Node Coordinate System

Each Node has its own coordinate system. This coordinate system is used to position child Node instances within the parent Node, or when drawing on a JavaFX Canvas. That means, that a Node that is a child of another Node both has its own coordinate system, and a location (X,Y) within its parent Node's coordinate system.

Below is an example of a parent Node coordinate system in which a child Node is located at (25,25) in the parent Node coordinate system. The child Node also has its own coordinate system which has its (0,0) where the child Node is located in the parent coordinate system - meaning at (25,25) in the parent Node coordinate system.



JavaFX Node Bounding Box

A JavaFX Node has a *bounding box*. The bounding box of a JavaFX Node is a logical box around the shape of the Node. The full Node is located inside the bounding box - graphically that is. In other words, all corners and edges of the Node is contained within the bounding box, and there is no extra space around the Node, unless added via an effect, padding or something else applied to the Node.

Actually, a Node has 3 bounding boxes:

Name	Description
`layoutBounds`	The bounds of the Node in its own coordinate space - without any effects, clip or transformations applied.
`boundsInLocal`	The bounds of the Node in its own coordinate space - with effects and clip applied, but without transformations.
`boundsInParent`	The bounds of the Node in its parent coordinate space - with effects, clip and transformations applied.

Each of these bounding box dimensions can be read from their corresponding properties with the same name, meaning properties named `layoutBounds, boundsInLocal` and `boundsInParent`.

The `boundsInParent` bounding box is used by the parent Node (Parent) to layout its children. The parent Node needs to know the total space including all effects, clip and transformations of a Node in order to be able to allocate space for it.

layoutX and layoutY

The `layoutX` and `layoutY` properties of a JavaFX Node object contain the X and Y of the Node inside of its parent. In other words, layoutX and layoutY are the offsets from the 0,0 (origo) of the parent Node.

Not all layout classes will respect the layoutX and layoutY of a Node. It depends on what the exact layout algorithm of the parent Node is.

```
node.setLayoutX(100);
node.setLayoutY(200);
```

Preferred Width and Height

The preferred width and height properties of a JavaFX Node object specify the preferred width and height of the given Node object (subclass of Node normally). Not all parent Node classes will respect the `preferredWidth` and `preferredHeight` of a child Node. It depends on the concrete parent Node implementation.

```
node.setPrefWidth(100);
node.setPrefHeight(100);
```

Minimum Width and Height

The minimum width and height properties of a JavaFX Node object specify the minimum width and height a Node wants (or needs) to display itself. Not all parent Node classes may respect these properties. It depends on the concrete parent Node implementation.

```
node.setMinWidth(50);
node.setMinHeight(50);
```

Maximum Width and Height

The maximum width and height properties of a JavaFX Node object specify the maximum width and height a Node wants (or needs) to display itself. Not all parent Node classes may respect these properties. It depends on the concrete parent Node implementation.

```
node.setMaxWidth(50);
node.setMaxHeight(50);
```

User Data

You can set user data on a JavaFX node using the `setUserData()` method. This method takes any Java object of your own choosing. This way you can attach e.g. business objects to JavaFX Node instances. Here is an example of attaching some user data to a Node instance:

```
node.setUserData(new MyObject("Hey - some data"));
```

Items or Child Nodes

Many JavaFX `Node` subclasses can contain items or child nodes. How exactly you add and access these child Nodes depends on the concrete `Node` subclass. Some classes have a `getItems()` method returning a List of the items. Other classes have a `getChildren()` method doing the same. You will have to check the concrete `Node` subclass to find out if it can have items or child nodes, and how you add and access them.

END