WIESLAW ZIELONKA
ZIELONKA@IRIF.FR
WWW.IRIF.FR/~ZIELONKA

INTERFACES GRAPHIQUES

JavaFX

JavaFX

- upgrade de Swing
- inclus dans Java JDK (Java 8)
- grand choix de composants
- composant web
- les moteurs graphiques et media (son, video, images)
- FXML pour décrire la vue en xml

SCENE GRAPH

- un arbre hiérarchique des éléments qui forment l'interface graphique (des noeuds)
- peut être rendu sur l'écran

Un noeud:

- un élément de l'interface graphique
- possède ID et le style
- nous pouvons appliquer sur un noeud des transformations, des effets visuels, définir la transparence(ou opacité), définir event handlers (event handlers — les classes qui interceptent et réagissent sur les événements)

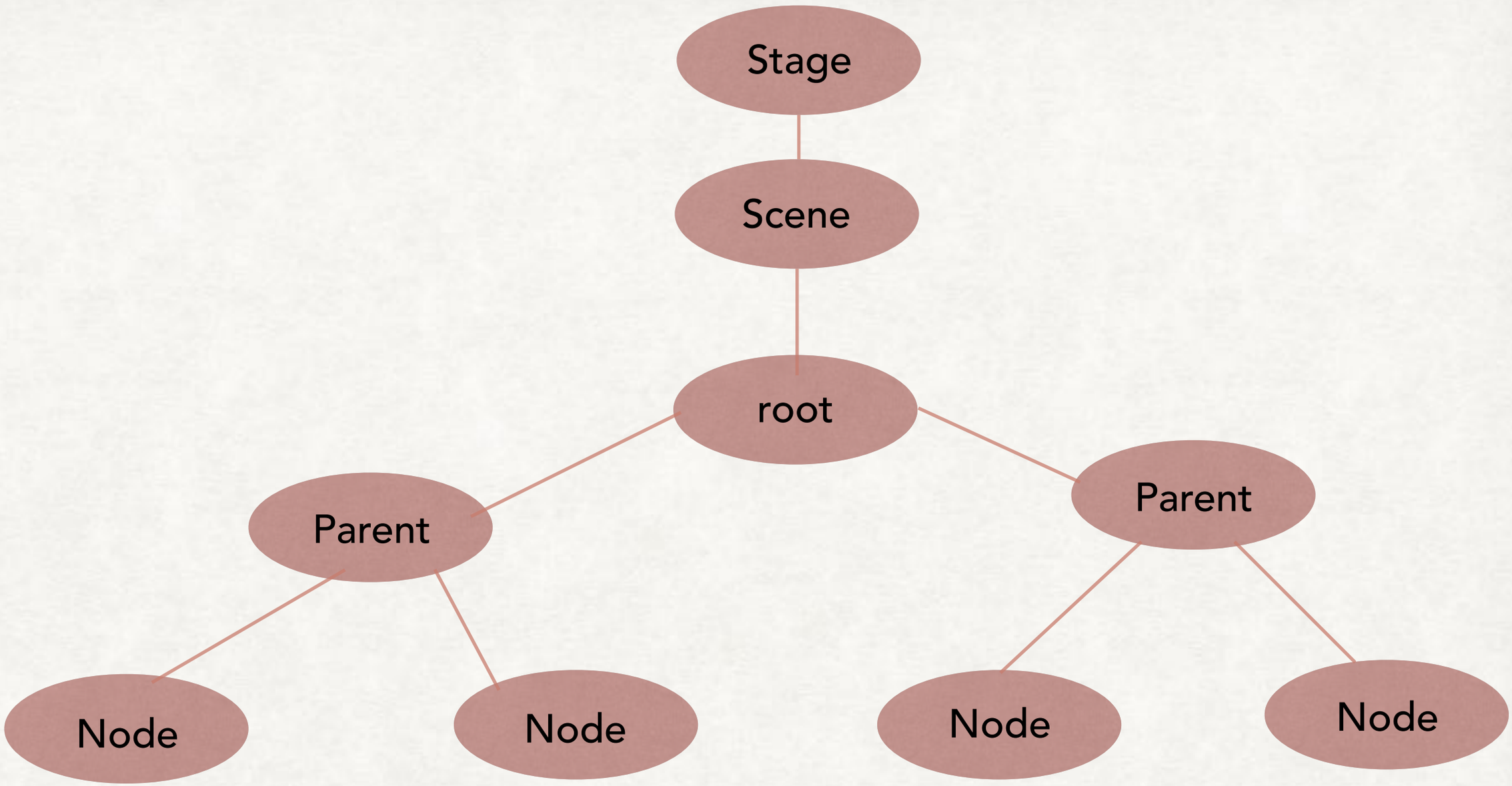
STAGE ET SCENE

Hélas Stage et Scene semblent avoir la même traduction en français.

Stage — la fenêtre principale, javaFx nous donne la référence vers un seul objet Stage (le paramètre de la méthode start())

Scene — nous pouvons construire dans notre programme autant d'objets de la classe Scene qu'on veut. Comme dans un théâtre Stage reste toujours le même mais un objet Scene qui représente ce qui est affiché sur Stage peut être remplacé à tout moment par un autre objet Scene.

Scene est un conteneur de base qui peut comporter un ou plusieurs noeuds, la propriété root de Scene doit référencer la racine de l'arbre de noeuds qui sont affichés sur la scène.



Nodes : Button, Label, …, Ellipse, Rectangle,…, Sphere, Box, Cylinder

Parents, différents Panes : StackPane, GridPane, BorderPane …

package javafx.scene

- classe Node
  - la classe de base pour tous les éléments graphiques
  - plusieurs propriétés et méthodes
  - permet d'installer event handlers pour les événements du clavier et de la souris
- classe Scene
  - tous les noeuds de l'application
- classe Group
  - permet de regrouper les Nodes
  - permet d'appliquer une transformation sur tous les Nodes du groupe
  - permet de définir les propriétés qui s'appliquent à tous les membres du groupe

la structure du programme javaFX

```
public class EmptyScene extends Application {

    @Override
    public void start(Stage primaryStage) {
      /* construire interface graphique */
     }

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        launch(args); /* lance l'application */
    }

}
```

Fenêtre avec titre et changement de couleur background de Scene

```
@Override

public void start(Stage primaryStage) {

    Group root = new Group();

    Scene scene = new Scene(root, 300, 200);

    /* couleur backGround propriété fill */
    scene.setFill(Color.DEEPSKYBLUE);

    primaryStage.setTitle("Scene vide avec titre");

    primaryStage.setScene(scene);

    primaryStage.show();

}
```

Events and handlers — introduction rapide

Les actions de l'utilisateur provoquent envoie des événements vers les noeuds : entrée clavier, click de la souris, geste sur un écran tactile.

Les événements sons les classes dérivées de

javafx.event.Event

Pour capturer les événements et y répondre on enregistre EventHandlers sur les composants de l'interface graphique.

Events and handlers

Exemple:

```
Button plus new Button("+");

plus.setOnAction(new EventHandler<ActionEvent>(){

            public void handle(ActionEvent e){

                //implementer les actions a executer

                //si on clique sur le bouton

        }

});
```

Events and handlers

Exemple:

```
Button plus new Button("+");

plus.setOnAction(new EventHandler<ActionEvent>(){

            public void handle(ActionEvent e){

                //implementer les actions a executer

                //si on clique sur le bouton

        }

});
```

Events and handlers

Inconvénient de classes anonymes avec une seule méthode : beaucoup de code à écrire pour juste faire passer une méthode dans une classe. La solution en java 8 : lambda expressions.

Exemple:

```
Button plus new Button("+");

plus.setOnAction(
        (ActionEvent e) -> {
            //implementer les actions a executer
            //si on clique sur le bouton
        }
);
```

Events and handlers

Encore plus simple, sans écrire le type de l'argument:

Exemple:

```
Button plus new Button("+");

plus.setOnAction(

        e -> {

            //implementer les actions a executer

            //si on clique sur le bouton

        }

);
```

Events and handlers

La méthode setOnAction() permet d'enregistrer un seul handler. Si on enregistre un nouveau handler alors l'enregistrement de handler précédent est annulé.