Java FX

Switching Scenes:

Scene game = new Scene(FXMLLoader.*load*(getClass().getResource("Game.fxml")));  
Stage initializer = (Stage) ((javafx.scene.Node) event.getSource()).getScene().getWindow();  
initializer.setScene(game);  
initializer.show();

Initializing right after scene execution:

public class MapController implements Initializable

@Override  
public void initialize(URL fxmlFileLocation, ResourceBundle resources)

Making a node draggable:

MouseControlUtil.makeDraggable(player1);

Setting up Drag and Drop:

@Override public void start(Stage stage) {  
 stage.setTitle("Hello Drag And Drop");  
  
 Group root = new Group();  
 Scene scene = new Scene(root, 400, 200);  
 scene.setFill(Color.*LIGHTGREEN*);  
  
 final Text source = new Text(50, 100, "DRAG ME");  
 MouseControlUtil.*makeDraggable*(source);  
 source.setScaleX(2.0);  
 source.setScaleY(2.0);  
  
 final Text target = new Text(250, 100, "DROP HERE");  
 target.setScaleX(2.0);  
 target.setScaleY(2.0);  
  
 source.setOnDragDetected(new EventHandler <MouseEvent>() {  
 public void handle(MouseEvent event) {  
 /\* drag was detected, start drag-and-drop gesture\*/  
 System.*out*.println("onDragDetected");  
  
 /\* allow any transfer mode \*/  
 Dragboard db = source.startDragAndDrop(TransferMode.*MOVE*);  
  
 /\* put a string on dragboard \*/  
 ClipboardContent content = new ClipboardContent();  
 content.putString(source.getText());  
 db.setContent(content);  
  
 event.consume();  
 }  
 });  
  
 target.setOnDragOver(new EventHandler <DragEvent>() {  
 public void handle(DragEvent event) {  
 /\* data is dragged over the target \*/  
 System.*out*.println("onDragOver");  
  
 /\* accept it only if it is not dragged from the same node  
 \* and if it has a string data \*/  
 if (event.getGestureSource() != target &&  
 event.getDragboard().hasString()) {  
 /\* allow for both copying and moving, whatever user chooses \*/  
 event.acceptTransferModes(TransferMode.*COPY\_OR\_MOVE*);  
 }  
  
 event.consume();  
 }  
 });  
  
 target.setOnDragEntered(new EventHandler <DragEvent>() {  
 public void handle(DragEvent event) {  
 /\* the drag-and-drop gesture entered the target \*/  
 System.*out*.println("onDragEntered");  
 /\* show to the user that it is an actual gesture target \*/  
 if (event.getGestureSource() != target &&  
 event.getDragboard().hasString()) {  
 target.setFill(Color.*GREEN*);  
 }  
  
 event.consume();  
 }  
 });  
  
 target.setOnDragExited(new EventHandler <DragEvent>() {  
 public void handle(DragEvent event) {  
 /\* mouse moved away, remove the graphical cues \*/  
 target.setFill(Color.*BLACK*);  
  
 event.consume();  
 }  
 });  
  
 target.setOnDragDropped(new EventHandler <DragEvent>() {  
 public void handle(DragEvent event) {  
 /\* data dropped \*/  
 System.*out*.println("onDragDropped");  
 /\* if there is a string data on dragboard, read it and use it \*/  
 Dragboard db = event.getDragboard();  
 boolean success = false;  
 if (db.hasString()) {  
 target.setText(db.getString());  
 success = true;  
 }  
 /\* let the source know whether the string was successfully  
 \* transferred and used \*/  
 event.setDropCompleted(success);  
  
 event.consume();  
 }  
 });  
  
 source.setOnDragDone(new EventHandler <DragEvent>() {  
 public void handle(DragEvent event) {  
 /\* the drag-and-drop gesture ended \*/  
 System.*out*.println("onDragDone");  
 /\* if the data was successfully moved, clear it \*/  
 if (event.getTransferMode() == TransferMode.*MOVE*) {  
 source.setText("");  
 }  
  
 event.consume();  
 }  
 });  
  
 root.getChildren().add(source);  
 root.getChildren().add(target);  
 stage.setScene(scene);  
 stage.show();  
}

Scenebuilder configuration with CSS files :

Style : Write your code here

Style Class : define class

Stylesheets : import custom CSS file

Pane: Regions to which you can add children using the getChildren() api. Pane is very similar to a group; it has a simple api for adding children and does not explicitly layout the location of the children.