Mouse Events

A MouseEvent is fired whenever a mouse button is pressed, released, clicked, moved, or dragged on a node or a scene.

- The MouseEvent object captures the event, such as the number of clicks associated with it, the location (the *x*- and *y*-coordinates) of the mouse, or which mouse button was pressed .
- Four constants—PRIMARY, SECONDARY, MIDDLE, and NONE—are defined in MouseButton to indicate the left, right, middle, and none mouse buttons.
- You can use the getButton() method to detect which button is pressed. For example, getButton() == MouseButton.SECONDARY indicates that the right button was pressed.

```
package javafxapplication55;

import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.layout.Pane;
import javafx.scene.text.Text;

public class JavaFXApplication55 extends Application {

  @Override
  public void start(Stage primaryStage) {
    // Create a pane and set its properties
    Pane pane = new Pane();
    Text text = new Text(20, 20, "Programming is fun");
    pane.getChildren().addAll(text);
    text.setOnMouseDragged(e -> {
        text.setX(e.getX());
        text.setY(e.getY());
    });

    // Create a scene and place it in the stage
    Scene scene = new Scene(pane, 300, 100);
    primaryStage.setTitle("MouseEventDemo"); // Set the stage title
    primaryStage.setScene(scene); // Place the scene in the stage
    primaryStage.show(); // Display the stage
    }
}
```

Key Events

A KeyEvent is fired whenever a key is pressed, released, or typed on a node or a scene.

- *Key events* enable the use of the keys to control and perform actions or get input from the keyboard.

- The KeyEvent object describes the nature of the event (A key has been pressed, released, or typed) and the value of the key, as shown in Figure 15.12.
- Every key event has an associated code that is returned by the getCode() method in KeyEvent.
- The *key codes* are constants defined in KeyCode.

TABLE 15.2 KeyCode Constants

Constant	*Description*	*Constant*	*Description*
HOME	The Home key	CONTROL	The Control key
END	The End key	SHIFT	The Shift key
PAGE_UP	The Page Up key	BACK_SPACE	The Backspace key
PAGE_DOWN	The Page Down key	CAPS	The Caps Lock key
UP	The up-arrow key	NUM_LOCK	The Num Lock key
DOWN	The down-arrow key	ENTER	The Enter key
LEFT	The left-arrow key	UNDEFINED	The keyCode unknown
RIGHT	The right-arrow key	F1 to F12	The function keys from F1 to F12
ESCAPE	The Esc key	0 to 9	The number keys from 0 to 9
TAB	The Tab key	A to Z	The letter keys from A to Z

- KeyCode is an enum type. For use of enum types, see Appendix I. For the key-pressed and key-released events, getCode() returns the value as defined in the table, getText() returns a string that describes the key code, and getCharacter() returns an empty string. For the key-typed event, getCode() returns UNDEFINED and getCharacter() returns the Unicode character or a sequence of characters associated with the key-typed event.

```
package javafxapplication56;

import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.layout.Pane;
import javafx.scene.text.Text;

public class JavaFXApplication56 extends Application {

  @Override
  public void start(Stage primaryStage) {
  // Create a pane and set its properties
  Pane pane = new Pane();
  Text text = new Text(20, 20, "A");

  pane.getChildren().add(text);
  text.setOnKeyPressed(e -> {
  switch (e.getCode()) {
  case DOWN: text.setY(text.getY() + 10); break;
  case UP: text.setY(text.getY() - 10); break;
  case LEFT: text.setX(text.getX() - 10); break;
  case RIGHT: text.setX(text.getX() + 10); break;
```

```
    default:
    if (Character.isLetterOrDigit(e.getText().charAt(0)))
       text.setText(e.getText());
    }
    });

    // Create a scene and place it in the stage
    Scene scene = new Scene(pane, 200,200);
    primaryStage.setTitle("KeyEventDemo"); // Set the stage title
    primaryStage.setScene(scene); // Place the scene in the stage
    primaryStage.show(); // Display the stage
    text.requestFocus(); // text is focused to receive key input
  }
}
```

Another Example : ControlCircle

```
package javafxapplication57;

import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.scene.layout.HBox;
import javafx.geometry.Pos;
import javafx.scene.input.KeyCode;
import javafx.scene.input.MouseButton;
import javafx.scene.layout.*;
import javafx.scene.paint.Color;
import javafx.scene.shape.Circle;

public class JavaFXApplication57 extends Application {

    private CirclePane circlePane = new CirclePane();

           @Override // Override the start method in the Application class
           public void start(Stage primaryStage) {
           // Hold two buttons in an HBox
           HBox hBox = new HBox();
           hBox.setSpacing(10);
           hBox.setAlignment(Pos.CENTER);
           Button btEnlarge = new Button("Enlarge");
           Button btShrink = new Button("Shrink");
           hBox.getChildren().add(btEnlarge);
           hBox.getChildren().add(btShrink);

           // Create and register the handler
           btEnlarge.setOnAction(e -> circlePane.enlarge());
           btShrink.setOnAction(e -> circlePane.shrink());

           circlePane.setOnMouseClicked(e -> {
           if (e.getButton() == MouseButton.PRIMARY) {
           circlePane.enlarge();
           }
           else if (e.getButton() == MouseButton.SECONDARY) {
           circlePane.shrink();
           }
           });

           circlePane.setOnKeyPressed(e -> {
           if (e.getCode() == KeyCode.U) {
           circlePane.enlarge();
           }
           else if (e.getCode() == KeyCode.D) {
           circlePane.shrink();
           }
           });

           BorderPane borderPane = new BorderPane();
           borderPane.setCenter(circlePane);
           borderPane.setBottom(hBox);
           BorderPane.setAlignment(hBox, Pos.CENTER);

           // Create a scene and place it in the stage
           Scene scene = new Scene(borderPane, 200, 150);
           primaryStage.setTitle("ControlCircle"); // Set the stage title
           primaryStage.setScene(scene); // Place the scene in the stage
           primaryStage.show(); // Display the stage

            circlePane.requestFocus(); // Request focus on circlePane
       }

class EnlargeHandler implements EventHandler<ActionEvent>{
   @Override
   public void handle(ActionEvent e){
       circlePane.enlarge();
   }
}

class shrinkHandler implements EventHandler<ActionEvent>{
   @Override
   public void handle(ActionEvent e){
      circlePane.shrink();
   }

}

class CirclePane extends StackPane{
   private Circle circle = new Circle(50);

   public CirclePane(){
       circle.setStroke(Color.BLACK);
       circle.setFill(Color.BLUE);
       getChildren().add(circle);
}
```

```
public void enlarge(){
   circle.setRadius(circle.getRadius()+5);
}

public void shrink(){
   circle.setRadius(circle.getRadius()> 5 ? circle.getRadius()- 5 :circle.getRadius());
}
}
}
```