



ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ  
ΑΝΑΤΟΛΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ ΚΑΙ ΘΡΑΚΗΣ  
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΤΕ

## Προγραμματισμός Διεπαφής Χρήστη

### Ενότητα 4: Διαχείριση γεγονότων (Event handling)

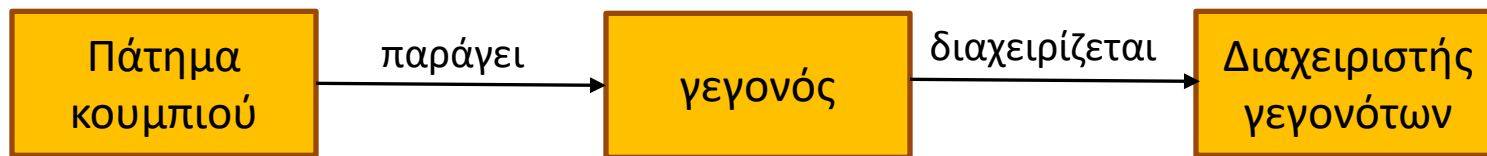
Κωνσταντίνος Τσίκνας  
ktsik@teiemt.gr

# Περιεχόμενα ενότητας

- Γεγονότα και πηγές προέλευσής τους.
- Εγγραφή διαχειριστών και διαχείριση γεγονότων
- Παραδείγματα.

# Εισαγωγή στη διαχείριση γεγονότων (2)

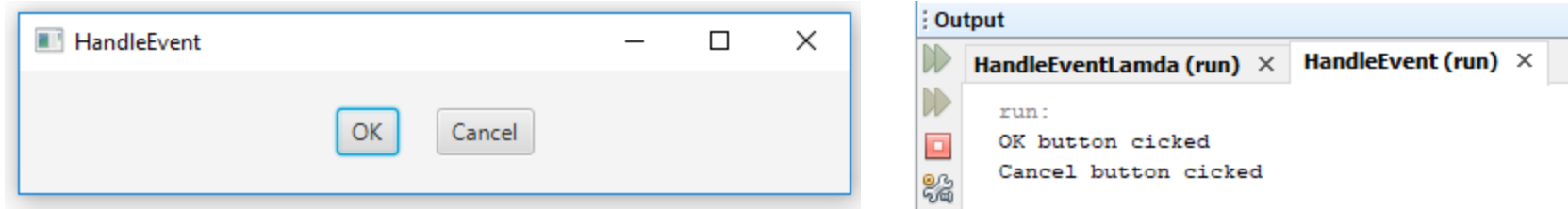
- Όταν εκτελούμε ένα πρόγραμμα Java GUI, αυτό αλληλεπιδρά με το χρήστη και «παράγονται» γεγονότα που καθοδηγούν την εκτέλεσή του.
- **Ενα γεγονός** μπορεί να ορισθεί ως ένα σήμα στο πρόγραμμα ότι κάτι έχει συμβεί.



- Τα γεγονότα πυροδοτούνται από εξωτερικές ενέργειες του χρήστη, όπως είναι οι κινήσεις του ποντικού, κλικ, το πάτημα ενός κουμπιού, ή τα χτυπήματα των πλήκτρων του πληκτρολογίου.
- Το πρόγραμμα μπορεί να επιλέξει να αποκριθεί ή να αγνοήσει το γεγονός.
- Τα γεγονότα στη JavaFx τα διαχειρίζεται μια ειδική κλάση που ονομάζεται **Διαχειριστής Γεγονότος (Event Handler)**

# Εισαγωγή στη διαχείριση γεγονότων (1)

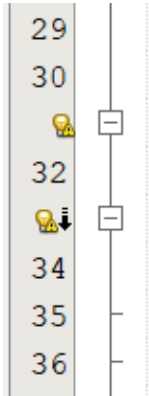
- Εστω ένα πρόγραμμα που εμφανίζει δύο κουμπιά σε ένα παράθυρο.



- Το κάθε κουμπί είναι ένα αντικείμενο προέλευσης γεγονότος (source event), όπου παράγει μια ενέργεια όταν αυτό πατηθεί.
- Για τη διαχείριση της ενέργειας (ή του γεγονότος αυτού) δημιουργούμε ένα νέο αντικείμενο, το οποίο μας επιτρέπει να διαχειριστούμε το γεγονός που παράχθηκε όπως επιθυμούμε.
- Το αντικείμενο αυτό ονομάζεται **διαχειριστής γεγονότος (Event Handler)**
- Στο παράδειγμά μας επιθυμούμε ο διαχειριστής του γεγονότος που συνδέεται με το πάτημα του κουμπιού να εμφανίζει ένα μήνυμα. Σε αυτήν την περίπτωση ο διαχειριστής γεγονότος θα πρέπει να το εμφανίζει.

# Κώδικας παράδειγματος

- Ο κώδικας που διαχειρίζεται τη δημιουργία ενός κουμπιού δίνεται στο παράδειγμα που ακολουθεί



```
Button btnOK = new Button("OK");

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

    public void handle(ActionEvent event){
        System.out.println("OK button cicked");
    }

});
```

## Επεξήγηση:

- Η `setOnAction()` είναι η μέθοδος που καλείται όταν το αντικείμενο προέλευσης (κουμπί) πατηθεί.
- Εντός της `setOnAction()` δημιουργούμε ένα νέο αντικείμενο της διεπαφής.

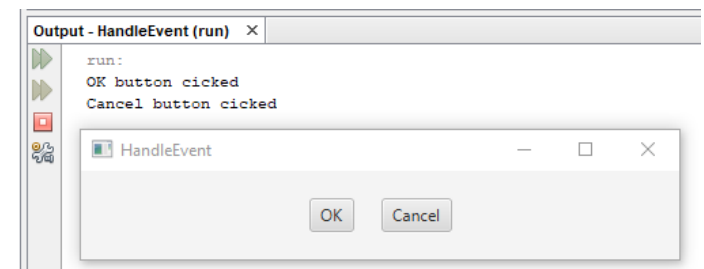
**`EventHandler<T Extends Event>`** για το χειρισμό του γεγονότος με τον τρόπο που εμείς επιθυμούμε.

Στο συγκεκριμένο παράδειγμα καλείται η **`EventHandler<ActionEvent>`**

- Εντός της `EventHandler<ActionEvent>` γράφουμε την `public void handle(ActionEvent event)` για τη διαχείριση του γεγονότος.

# Ο κώδικας του παραδείγματος

```
22 public class HandleEvent extends Application{
23     @Override
24     public void start(Stage myStage){
25         FlowPane pane = new FlowPane();
26         pane.setPadding(new Insets(20, 20, 20, 20));
27         pane.setHgap(20);
28         pane.setAlignment(Pos.CENTER);
29
30         Button btnOK = new Button("OK");
31         btnOK.setOnAction(new EventHandler<ActionEvent>(){
32             public void handle(ActionEvent event){
33                 System.out.println("OK button clicked");
34             }
35         });
36
37         Button btnCancel = new Button("Cancel");
38         btnCancel.setOnAction(new EventHandler<ActionEvent>(){
39             public void handle(ActionEvent event){
40                 System.out.println("Cancel button clicked");
41             }
42         });
43
44         pane.getChildren().addAll(btnOK, btnCancel);
45         Scene scene = new Scene(pane);
46         myStage.setTitle("HandleEvent");
47         myStage.setScene(scene);
48         myStage.show();
49     }
```



# Γεγονότα και διαχείρισή τους

- Ενα γεγονός είναι ένα αντικείμενο της κλάσης γεγονότων.
- Η κλάση-ρίζα των κλάσεων γεγονότων του εργαλείου JavaFx είναι η `javafx.event.Event`
- Οι ιεραρχικές σχέσεις ορισμένων από τις κλάσεις γεγονότων, απεικονίζονται στο παρακάτω σχήμα. Ανάλογα με το αντικείμενο προέλευσης (`button`, `mouse`, `key`), υπάρχει το κατάλληλο είδος γεγονότος που μπορεί να ενεργοποιηθεί, καθώς και η αντίστοιχη μέθοδος για την ανίχνευση του γεγονότος αυτού.
- Ορισμένες ενέργειες μαζί με τις μεθόδους με τις οποίες αυτές ανιχνεύονται δίνονται στον παρακάτω πίνακα.

Ενέργεια χρήστη - Αντικείμενο προέλευσης	Είδος γεγονότος	Μέθοδος ανίχνευσης γεγονότος
Πάτημα κουμπιού ( <code>Button</code> )	<code>ActionEvent</code>	<code>setOnAction (EventHandler&lt;ActionEvent&gt;)</code>
Πάτημα πλήκτρου Enter ( <code>TextField</code> )	<code>ActionEvent</code>	<code>setOnAction (EventHandler&lt;ActionEvent&gt;)</code>
Κλικ ποντικιού ( <code>node</code> , π.χ. <code>Rectangle</code> )	<code>ActionEvent</code>	<code>setOnAction (EventHandler&lt;ActionEvent&gt;)</code>
Πάτημα ποντικιού ( <code>node</code> , π.χ. <code>Rectangle</code> )	<code>MouseEvent</code>	<code>setOnMousePressed (EventHandler&lt;MouseEvent&gt;)</code>
Εισαγωγή ποντικιού ( <code>node</code> , π.χ. <code>Rectangle</code> )	<code>MouseEvent</code>	<code>setOnMouseEntered (EventHandler&lt;MouseEvent&gt;)</code>
Εξοδος ποντικιού ( <code>node</code> , π.χ. <code>Rectangle</code> )	<code>MouseEvent</code>	<code>setOnMouseReleased (EventHandler&lt;MouseEvent&gt;)</code>

# Εκφράσεις λάμδα

- Οι εκφράσεις λάμδα έχουν εισαχθεί στη Java 8 για την απλοποίηση του κώδικα που γράφουμε. Γενική σύνταξη:

```
e -> {  
    //στην περιοχή αυτή γράφουμε τον κώδικα επεξεργασίας του γεγονότος  
}  
ή  
e -> ΟνομαΜεθόδου()
```

## Παράδειγμα:

```
btnOK.setOnAction( e -> {  
    System.out.println("OK button cicked");  
}  
);
```

Δηλαδή αποφεύγουμε να γράψουμε την **new EventHandler<T Extends Event>** καθώς και την κλήση της **public void handle(ActionEvent event){**

```
.....  
}
```

Δηλαδή εντός της `setOnAction( )` γράφουμε το σύμβολο `e ->` ακολουθούμενο από τις ενέργειες που επιθυμούμε να πραγματοποιηθούν μόλις ανίχνευθεί το γεγονός για το συγκεκριμένο κουμπί προέλευσης.

Εναλλακτικά, μετά το σύμβολο μπορεί να ακολουθήσει το όνομα μιας μεθόδου που θα περιέχει συγκεντρωτικά τις απαιτούμενες ενέργειες.



# Παραδείγματα

- Η παρακάτω έκφραση:

```
29      Button btnOK = new Button("OK");
30
31      btnOK.setOnAction(new EventHandler<ActionEvent>() {
32
33          public void handle(ActionEvent event) {
34              System.out.println("OK button cicked");
35          }
36      });
```

- Γράφεται πιο απλά με έκφραση λάμδα ως:

```
30      Button btnOK = new Button("OK");
31      btnOK.setOnAction(e->{
32          System.out.println("OK button cicked");
33      });
```

- ή ισοδύναμα ως:

```
30      Button btnOK = new Button("OK");
31      btnOK.setOnAction(e->displayMessage());
32
33      }
34      void displayMessage() {
35          System.out.println("OK button cicked");
36      }
```

# Ο κώδικας του παραδείγματος με εκφράσεις λάμδα

```
public void start(Stage myStage) {  
    25     FlowPane pane = new FlowPane();  
    26     pane.setPadding(new Insets(20, 20, 20, 20));  
    27     pane.setHgap(20);  
    28     pane.setAlignment(Pos.CENTER);  
    29  
    30     Button btnOK = new Button("OK");  
    31     btnOK.setOnAction( e -> {  
    32         System.out.println("OK button cicked");  
    33     }  
    34 );  
    35  
    36     Button btnCancel = new Button("Cancel");  
    37     btnCancel.setOnAction( e -> {  
    38         System.out.println("Cancel button cicked");  
    39     }  
    40 );  
    41  
    42     pane.getChildren().addAll(btnOK, btnCancel);  
    43  
    44     Scene scene = new Scene(pane);  
    45     myStage.setTitle("HandleEvent");  
    46     myStage.setScene(scene);  
    47     myStage.show();  
    48 }
```

