# 8. előadás 1. rész – Eseménykezelés

## Múlt alkalommal az előadás elmaradt

Mert ZH-t írtak sokan. Csak ketten voltunk előadáson.

## Az eseménykezelés alapjai

Az eseménykezeléssel működő programok alapötlete az, hogy a program futása közben események hatására mennek végbe bizonyos folyamatok

Események váltódnak ki, melyekre az eseménykezelők reagálnak

Egy esemény a program bármely részében kiváltódhat, az erre reagáló eseménykezelő pedig tipikusan a program egy teljesen más részében fut le

Objektumorientált programozási nyelvben objektumok és függvényeik segítségével valósul meg

## Eseményobjektum, eseményosztály

Minden kiváltott eseményhez tartozik egy eseményobjektum

Hordozza az esemény tulajdonságait – az objektum saját változói segítségével

Rendelkezik egy referenciával arról is, hogy hol váltódott ki az esemény – ez az ú.n. forrás (source)

A hasonló típusú eseményekhez természetesen létrehozható egy megfelelő eseményosztály

Java-ban az események ősosztálya a java**.**util**.**EventObject osztály, tartalmazza a forrás kezelését

## Eseménykezelő objektum, - osztály

Alkalmas egy adott típusú esemény kezelésére, azaz reagálni arra, ha az esemény kiváltódik

Rendelkezik egy vagy több eseménykezelő függvénnyel, mind 1-1 esemény kezelésére szolgál

A függvény(ek) megkapja/megkapják a kiváltott eseményobjektumot

Az eseményobjektum forrása és tulajdonságai alapján – és persze a saját feladatának megfelelően – végzi el az esemény kezelését

A hasonló típusú eseménykezelő objektumok számára természetesen létrehozható egy megfelelő eseménykezelő osztály

Java-ban az eseménykezelők a java**.**util**.**EventListener interfészből származnak, tipikusan maguk is csak interfészek, hiszen az eseménykezelő függvények konkrét működése általában nem ismert

## Eseménykezelő függvény

Egy eseménykezelő függvény egy esemény kezelésére szolgál, tartalmazza azt a tevékenységet, amelyet az adott eseményre reagálásképp végre kell hajtani

Paraméterként megkapja a kiváltott eseményt, amelyre éppen reagálni kell, azaz amelyet kezelni kell

Működése közben kiválthat más eseményeket is

## Esemény kiváltása és kezelése

Egy esemény kiváltásakor ismerni kell az eseményt kezelő objektumokat. Az esemény kiváltásának első lépése az eseményobjektum megkonstruálása (forrásként rendszerint beállítható a **this** referencia), második lépése az eseménykezelő objektumok eseménykezelő függvényeinek meghívása az eseményobjektum átadásával

Az esemény kezelése az eseménykezelő függvényben történik, tehát ezen függvények törzsét, működését kell megadni

## Események és eseménykezelők összekapcsolása

Ahhoz, hogy egy esemény kiváltásának pillanatában ismerjük az eseménykezelőket, össze kell kapcsolni azokat egymással

Kifejezések:

* regisztráljuk az eseménykezelőt az eseménynél;
* az eseménykezelő feliratkozik az eseményre;
* az eseménykezelőt hozzárendeljük az eseményhez

Természetesen később szétkapcsolni is lehet ezeket

## Az eseménykezelés absztrakt szerkezete

Természetesen szóba jöhet többfajta esemény és eseménykezelő osztály is



$registerListener$ mellett van még $addEventListener$, $removeEventListener$.

Az eseménykezelő ismeri a kiváltót, nem fordítva.

A bal alsó ciklust nekünk kell megírni. Nem automatikus.

## Egy esettanulmány

Töltsünk le egy HTML oldalt, majd írjuk ki a kimenetre annak szöveges tartalmát

A HTML forrás feldolgozása és tartalmának kiírása a program két külön komponense

A forrás feldolgozása közben váltunk ki különböző eseményeket, melyeket a kiíró komponens kezel

A program osztálydiagramja: <https://lengyel.web.elte.hu/store/prt1/ea/HtmlBrowserClassDiagram.png>

A program egy NetBeans projektben: <https://lengyel.web.elte.hu/store/prt1/ea/ConsoleHtmlBrowser.zip>

# 8. előadás 2. rész – Swing

Nagyon bonyolult keretrendszer, nem is fogjuk végigvenni. Célunk csak annyi, hogy nagyjából megismerjük, a fő részeit. Konkrét dolgok a gyakorlaton lesznek.

## Felépítése – Áttekintés

Swing = Ablakozó keretrendszer.

Minden vezérlőelem egy-egy objektum. Maga az ablak is. Az ablak tartalmát tartalmazó Container is.

Másik ablakozó keretrendszerre épül, ami az $awt$. Korábban készült, primitívebb.

Csomagja $javax.swing$.

Komponens – vezérlőelem – grafikus megjelenítéssel rendelkező elemek.

Ősosztálynak a $JComponent$ osztályt tekintjük.

Az $awt$ primitívebb osztálytípusait nem szoktuk használni.

A $Component$ megjelenítő osztály.

A $Container$ osztály kompozícióként tartalmaz többszörös multiplicitással ($0..\*$) $Component$ típusú objektumokat.

## A legfontosabb $JComponent$ek

### $$AbstractButton$$

Tulajdonságait getter és setter függvényekkel lehet beállítani.

Van csak olvasható és csak írható tulajdonsága.

Van olyan függvénye, ami nem tulajdonság kezelő. Például $Listener$ek hozzáadása/levétele,

Ebből származik a sokféle gomb:

* $JButton$ – sima, szokásos gomb
* $Togglebutton$ – két állással rendelkezik
* $RadioButton$ – kijelöléssel rendelkezik, csak egy lehet kijelölve egy csoportban
* $CheckBox$ – pipa
* $MenuItem$ – A menük gombjai is innen származnak.
* $JMenu$ – Ez sok $MenuItem$et tartalmaz. De ez is gomb, rá lehet kattintani, az almenü kinyitható.

Minden függvény jól van dokumentálva.

### $$JTextComponent$$

$JTextField$ – egysoros

$JTextArea$ – többsoros

$$JEditorPane$$

$JFormattedTextField$ – Formátumnak megfelelő (pl. csak email cím, telefonszám)

$JPasswordfield$ – Rejtett karakterek.

### További

* $JComboBox$ – Legördülő lista
* $JList$ – Sima lista
* $JProgressBar$ – Folyamatjelző, %-osan ábrázolja hol tart
* $JSlider$ – Csúszka
* $JSpinner$ – Le-fel nyíl
* $JLabel$ – Egyszerű címke.
* $JTooltip$ – Egeret rávisszük pl. gombra, előjön egy tooltip
* $JSeparator$ – Menüelemeket elválasztó vízszintes csík
* $JTable$ – Táblázat, 2D mátrix.
* $JTableHeader$ - osztott fejlécek, táblázatokba
* $JTree$ – Fa, pl. könyvtárlistához

### Ritkábban használt

* $JSplitPane$ - Kettéosztó
* $JTabbedPane$ – Fülek
* $JScrollPane$ – Beágyazott vezérlőelem köré görgetősávokat rajzol, ha szükséges.
* $JScrollBar$ – Görgetősáv
* $JToolBar$ – Eszköztár
* $JMenuBar$ – Ablak menüsávja felül, ez tartalmaz Menüket, amiben vannak a menüpontok.
* $JPopupMenu$ – Context menü
* $JColorChooser$ – színpaletta
* $JFilechooser$ – Fájlok között tallózás
* $JOptionPane$ – Dialógus ablakot lehet vele készíteni. YES/NO/CANCEL, OK, stb. (?Biztos?)
* $JPanel$ – Semmi extra tulajdonsággal nem rendelkezik, de közrefog vezérlőelemeket. Természetesen lehet kerete.

Ez így önmagában kevés. Például táblázat esetében:

## $$TableModel$$

Minden $JTable$ben van. Táblázatbeli adatokat tartalmazza.

Táblázat egyes celláinak értékeit le lehet ezen keresztül kérdezni, stb.

$DeafultTableModel$ – sima 2D mátrix

## A $ListModel$ család

A $JList$ csak listaelemeket jelenít meg GUI-ban. A konkrét elemeket a $ListModel$ tartalmazza.

$AbstractListModel$ az ős.

$DefaultListModel$, amit használunk.

$ComboBoxModel$ interfész is van a legördülő listáknak.

## Eseménykezelés

A viselkedés így szabályozható.

TFH $JButton$ osztályt definiálunk. Rákattintásra átszíneződik, de ez nekünk nem fontos. Minden program így működik.

De ami fontos, az hogy hogyan reagálunk rá. Ezt a programozó valósítja meg.

Program alapértelmezésből nem csinál semmit – vár.

Akkor történik valami, ha a felhasználó csinál valamit.

Mikor a legelső grafikus objektumot felhelyezzük a képernyőre, akkor megjelenik egy háttérben futó folyamat, ami vár a felhasználóra, hogy csináljon valamit. Ha történik valami, eseményt vált ki. Ez az "Event Dispatching".

Ha van eseménykezelő a generált eseményre, akkor csinál valamit a program.

Minden komponens minden eseményére megmondhatjuk, milyen eseménykezelést szeretnénk elvégezni.

### Az awt-s $EventObject$ család

awt.event.\*

* $ActionEvent$ – nyomógomb lenyomásakor egy ilyen akció fog kiváltódni.
* $KeyEvent$ – Billentyűt lenyomunk, minden billentyűre van VK\_\* konstans. VK\_SPACE, VK\_A, VK\_B, VK\_F10, VK\_F11, VK\_F12, stb.
* $MouseEvent$ – Kattintás, bárhol, nem csak gombokon. Melyik egérgomb: $getButton\left(​\right);$, hol: $getX\left(​\right);$
* $MouseWheelEvent$ – Görgő. Merre, hányat.
* Egér mozgatást is lehet figyelni. Pixelenként.
* $WindowEvent$ – Ablakot átméretezzük, bezárjuk, minimalizáljuk, stb.
* stb.

### Swing-es kiegészítések az $EventObject$-hez

* $DragMouseEvent$ – Fogd és vidd
* $ListSelectedEvent$ – Lista milyen indexű eleme van kiválasztva.

## Eseménykezelők

A legtöbb eseménykezelő interfész. Az $EventListener$ ősinterfészből származik mind.

* $WindowAdapter$ – $WindowListener$t valósít meg. A 7 absztrakt funkciót üres függvénytörzzsel megvalósítja, ezért csak azokat kell felüldefiniálnunk, amire tényleg szükségünk van.
* $MouseWheelListener$ – Görgőre figyel. $mouseWheelMoved$
* $MouseListener$ – Egérre figyel. $mouseClicked$, $mouseEntered$, stb.
* $MouseMotionListener$ – Egérmozgásra figyel. $mouseMoved$, $mouseDragged$
* $KeyListener$ és adaptere $KeyAdapter$. A három eseménye: $keyPressed$ (lenyom) $keyReleased$ (felenged) $keyTyped$ (lenyom és felenged)
* $ActionListener$ – Rengetegszer fogjuk használni. Eseménye az $actionPerformed$.

## Elrendezések (Layout)

$Container$ek egyszerre több vezérlőt is tudnak tartalmazni / megjeleníteni. Érdemes elrendezéseket használni.

Ősinterfész a $LayoutManager$

* $GridLayout$ – sorokba és oszlopokba rendezi az elemeket.
* $FlowLayout$ – Középre igazítva sorba az elemeket, ha elfér. Ha kevés hely van, akkor új sort kezd.
* $GridBagLayout$ – Felosztja rácspontokra a $Container$t. Egyes cellákat össze lehet vonni (insets)
* $BorderLayout$ – 5 részre osztja a $Container$t. Középre és a négy szélére. Égtájak szerint vannak elnevezve.
* $BoxLayout$ – $FlowLayout$ra hasonlít, de mindig egy sor/oszlopba.

Ha a program futása közben a $Container$ mérete megváltozhat, érdemes $LayoutManager$t használni. Így a program követi ezeket a méretváltozásokat.

Egy $Container$hez csak egy $LayoutManager$ tartozhat. Egymásba ágyazás: Lehet, $Container$t $Container$be. Ekkor lehet az egyik $FlowLayout$ a másik $GridLayout$.

## $$JFrame$$

Kimaradt.

Ezt használjuk leggyakrabban ablak megjelenítésére.

A $JDialog$ párbeszédablak.

Ide jönne egy jó ábra.