



# Programozási technológia

Swing GUI készítése  
NetBeans IDE segítségével

Dr. Szendrei Rudolf  
ELTE Informatikai Kar  
2020.

# Bevezető

- ▶ Eddig a grafikus felhasználói felületet kódból hoztuk létre kézzel.
- ▶ A mi feladatunk volt az eseménykezelők implementálása és azoknak a megfelelő grafikus vezérlőelemekhez rendelése.
- ▶ Szeretnénk, hogy
  - ▶ grafikus tervező felületen készíthessük el a felhasználói interfészeket, és
  - ▶ az eseménykezelőket még a tervezőben hozzárendelhessük a vezérlőelemekhez (nekünk csak azok megvalósításával kelljen foglalkoznunk).

# Bevezető

- A NetBeans IDE fejlesztő környezet mindkét elvárást teljesíti (ha Swing GUI-t akarunk készíteni).
- Nincs szükségünk a LayoutManager működésének megértésére.
- Egyszerű húzd és ejtsd (Drag&Drop) módon helyezhetjük el a kívánt vezérlő elemeket a form-on.
- A tervező segítségével a GUI készítés gyorsabbá tehető.

# Projekt létrehozása

- Készítsünk egy egyszerű form-ot, amin megadhatjuk egy személy névjegyét.
  - Hozzunk létre egy új projektet a NetBeans-ben (File > New Project)
  - Válasszuk a Java-t a kategóriák közül, majd a Java Application-t a projektekből
  - Legyen a projekt neve `ContactEditor`
  - Use Dedicated Folder for Storing Libraries: **Nem**
  - Create Main Class: **Igen**
  - Kattintsunk a `Finish` gombra

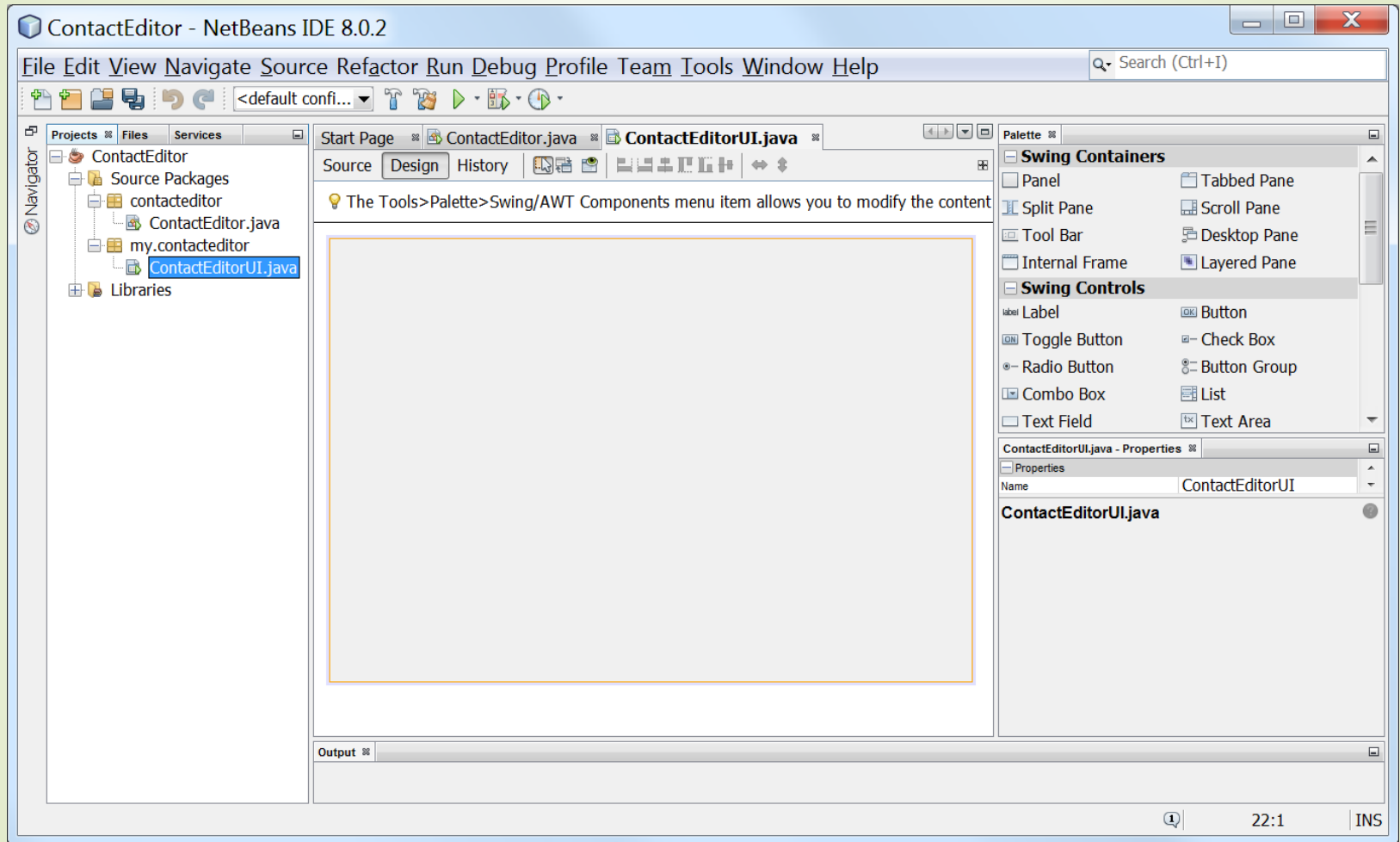
# JFrame létrehozása

- A programunk felhasználói felületének vezérlőelemeit a `JFrame` tartalmazza, amit a következő módon hozunk most létre:
  - A projekt ablakban kattintsunk jobb gombbal a projektünk nevére (`ContactEditor`), majd válasszuk a `New > JFrame Form... -ot`
  - Osztálynév: `ContactEditorUI`
  - Csomag neve: `my.contacteditor`
  - Kattintsunk a Finish gombra

# GUI Builder

- ▶ A `JFrame` létrehozása után a NetBeans átvált a GUI tervező nézetbe, és megjelennek a főbb ablakai:
  - ▶ Tervező ablak: Itt helyezhetjük el a vezérlőelemeket a formon,
    - ▶ Source gomb: megnézhetjük az osztály forráskódját
    - ▶ Design gomb: átválthatunk a grafikus megjelenítésre
    - ▶ History gomb: eddigi változtatások listája
  - ▶ Navigátor ablak:
    - ▶ Tartalmazza az összes komponenst (a láthatóakat és nem láthatóakat egyaránt) egy fa szerkezetben.
  - ▶ Palette ablak:
    - ▶ Tartalmazza a használható vezérlő komponenseket, layout menedzsereket kategóriákba rendezve
    - ▶ A tartalmát átrendezhetjük
  - ▶ Properties ablak:
    - ▶ megmutatja a kiválasztott vezérlőelem tulajdonságait

# GUI Builder





# GUI kialakítása – Beviteli mezők

- Tegyük a formra egymás alá két `JPanel`-t
- A `JPanel` Properties ablakában a `Border`-t választva a megjelenő ablakban válasszuk a `Titled Border`-t
- Adjuk meg a text mezőben a keret címkéjét (Név, E-Mail)
- Adjuk hozzá a Név kerethez a következőket:
  - 'Vezetéknév' feliratú címke (`JLabel`)
  - Szövegbeviteli mező a vezetéknévnek (`JTextField`)
  - 'Utónév' feliratú címke (`JLabel`)
  - Szövegbeviteli mező az utónévnek (`JTextField`)
- Adjuk hozzá az E-Mail kerethez a következőket:
  - 'E-Mail cím' feliratú címke (`JLabel`)
  - Szövegbeviteli mező az E-Mail címnek (`JTextField`)
  - E-Mail címek listája (`JList`)



# GUI kialakítása – Beviteli mezők

Név:

Vezetéknév:	<input type="text" value="jTextField1"/>	Utónév:	<input type="text" value="jTextField2"/>
Megszólítás:	<input type="text" value="jTextField3"/>	Becenév:	<input type="text" value="jTextField4"/>
Megjelenítés:	<input type="text" value="Item 1"/>		

E-Mail:

E-Mail cím:

- Item 1
- Item 2
- Item 3
- Item 4
- Item 5

# GUI kialakítása – Kezelőgombok

- Egészítsük ki a form-ot a megfelelő funkciókhoz tartozó gombokkal:
  - Hozzáad (JButton)
  - Szerkeszt (JButton)
  - Eltávolít (JButton)
  - Alaphelyzet (JButton)

# GUI kialakítása – Kezelőgombok

Név:

Vezetéknév:	<input type="text" value="jTextField1"/>	Utónév:	<input type="text" value="jTextField2"/>
Megszólítás:	<input type="text" value="jTextField3"/>	Becenév:	<input type="text" value="jTextField4"/>
Megjelenítés:	<input type="text" value="Item 1"/>		

E-Mail:

E-Mail cím:	<input type="text" value="jTextField5"/>	<input type="button" value="Hozzáad"/>
<input type="text" value="Item 1"/> <input type="text" value="Item 2"/> <input type="text" value="Item 3"/> <input type="text" value="Item 4"/> <input type="text" value="Item 5"/>	<input type="button" value="Szerkeszt"/> <input type="button" value="Eltávolít"/> <input type="button" value="Alaphelyzet"/>	

# GUI kialakítása – Rádiógombok

- A form alján adhassuk meg, hogy milyen típusú levelet kaphat a személy az adott E-Mail címére:
  - 'E-Mail formátuma:' (JLabel)
  - 'HTML' (JRadioButton)
  - 'Egyszerű szöveges' (JRadioButton)
  - 'Egyéni' (JRadioButton)

# GUI kialakítása – Rádiógombok

Név:

Vezetéknév:  Utónév:

Megszólítás:  Becenév:

Megjelenítés:

E-Mail:

E-Mail cím:

E-Mail formátuma:

HTML  Egyszerű szöveges  Egyéni

# GUI kialakítása – Rádiógombok

- A rádiógombok csak akkor alkotnak egy csoportot, ha azokat egy gomb csoporthoz társítjuk (`ButtonGroup`).
- A Palette ablakban válasszuk ki a `ButtonGroup` komponenst és kattintsunk a form egy tetszőleges helyére.
- A rádió gombok egy csoportba fogásához jelöljük ki mindhármát egyszerre, majd a Properties ablakban állítsuk be a `buttonGroup` tulajdonság legördülő menüjében az imént létrejött `buttonGroup1` komponenst.

# GUI Builderrel generált kód

- A GUI Builder által generált kód a tervezőben a Source gomb megnyomásával érhető el.
- A formon elhelyezett valamennyi komponensnek a változóját megtaláljuk a `ContactEditorUI.java` fájl végén, a hozzájuk kapcsolódó elrendezéseket pedig a NetBeans az `initComponents` függvényben helyezi el.
- A létrehozott osztály konstruktora kezdetben csak egyetlen hívást tartalmaz az `initComponents` függvényre.
- Az osztályban automatikusan generálódik egy `main` függvény is, aminek a segítségével közvetlenül is futtathatjuk a létrehozott formból álló alkalmazásunkat. (Ehhez azonban a projekt beállítások között be kell állítanunk a `main class`-nak a `ContactEditorUI` osztályt, vagy Shift+F6-al kell futtatnunk a projektet.)



# GUI Események kezelése

- ▶ Ahhoz, hogy a létrehozott gombokhoz tevékenységeket adhassunk meg, hozzá kell rendelnünk egy eseménykezelőt az egyes eseményekhez. Ehhez `ActionListener`-t fogunk rendelni `ActionEvent`-ekhez.
- ▶ Jobb egérgombbal kattintsunk a 'Hozzáad' gombra a formon.
- ▶ Válasszuk az `Events > Action > actionPerformed`-ot.
- ▶ A tervező automatikusan hozzáadja a gombhoz az `ActionListener`-t és legenerálja az eseménykezelő függvényt a listener `actionPerformed` metódusához.

```
private void btnAddActionPerformed(ActionEvent e) {  
    // TODO add your handling code here  
}
```

# GUI Események kezelése

- A kódban az eseménykezelő nem törölhető, ezért ha erre van szükségünk, akkor a tervezőben válasszuk ki a gombot, majd a tulajdonságok között az Events lapon állítsuk vissza az `actionPerformed`-ot `<none>`-ra.
- Az eseménykezelők nevei mindig a vezérlőelem változónevéből és az esemény nevéből generálódnak, ezért is érdemes a vezérlőelemek neveit már a létrehozáskor megváltoztatni.
- A vezérlőelem neve, amit a paraméter ablakban adhatunk meg, nem tévesztendő össze a vezérlőelemhez tartozó változó nevével!
- A vezérlőelemhez tartozó változó nevét a vezérlőelemre jobb egérgombbal való kattintáskor megjelenő menü `Change Variable Name...` pontjával változtathatjuk meg.

# GUI – Hozzáad gomb

```
private void btnAddActionPerformed(ActionEvent evt) {
    DefaultListModel model;
    try {
        model = (DefaultListModel)listEmails.getModel();
    } catch (ClassCastException e) {
        model = new DefaultListModel();
        listEmails.setModel(model);
    }
    model.addElement(txtFieldEmailAddress.getText());
}
```

# GUI – Szerkeszt gomb

```
private void btnEditActionPerformed(ActionEvent evt) {
    DefaultListModel model;
    try {
        model = (DefaultListModel)listEmails.getModel();
        int i = listEmails.getSelectedIndex();
        if (i == -1) return;
        model.setElementAt(txtFieldEmailAddress.getText(), i);
    } catch (ClassCastException e) { }
}
```

# GUI – Eltávolít gomb

```
private void btnRemoveActionPerformed(ActionEvent evt) {  
    DefaultListModel model;  
    try {  
        model = (DefaultListModel)listEmails.getModel();  
        int i = listEmails.getSelectedIndex();  
        if (i != -1) model.removeElementAt(i);  
    } catch (ClassCastException e) { }  
}
```



# GUI – Alaphelyzet gomb

```
private void btnDefaultActionPerformed(ActionEvent evt) {  
    try {  
        listEmails.setModel(new DefaultListModel());  
    } catch (ClassCastException e) { }  
}
```

# GUI – Listaelem kiválasztása

- A szerkesztés funkcióhoz általában társul az is, hogy a listában kiválasztott elem tartalma megjelenik a szerkesztő mező(k)ben.
- Ehhez egy eseménykezelőt kell rendelnünk a listához.
- Kattintsunk a tervezőben a listára a jobb egér gombbal, majd a menüben válasszuk ki az `Events > ListSelection > valueChanged` eseményt.



# GUI – Listaelem kiválasztása

Írjuk be az alábbi kódot a megvalósításhoz:

```
private void listEmailsValueChanged(ListSelectionEvent evt) {  
    DefaultListModel m;  
    try {  
        m = (DefaultListModel) listEmails.getModel();  
        int i = listEmails.getSelectedIndex();  
        if (i != -1) {  
            txtFieldEmailAddress.setText(m.getElementAt(i).toString());  
        }  
    } catch (ClassCastException ex) { }  
}
```