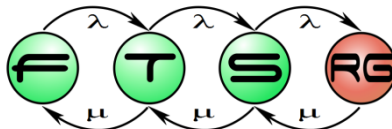


Grafikus felületek I. - SWT



SWT történelem

- Java grafikus toolkitek: AWT, Swing
- Problémái:
 - “Nem úgy néz ki, mint a Word” probléma
 - Nem veszi át az ablakkezelő look-and-feel beállításait
 - Gyakran a nyelvi beállításokat sem
 - “Nem szép”
 - Swing:
 - memóriafogyasztás
 - teljesítményproblémák
 - mára (Java 6.0) ez elfogadható
 - AWT: alacsony szintű

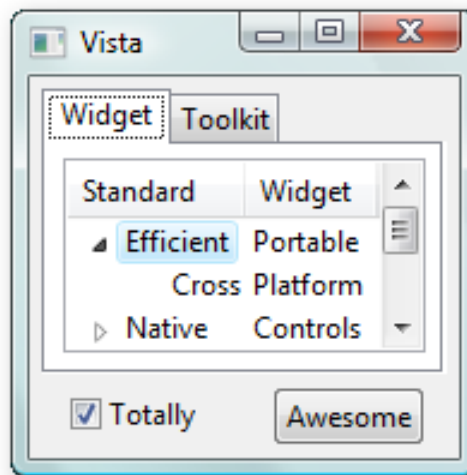
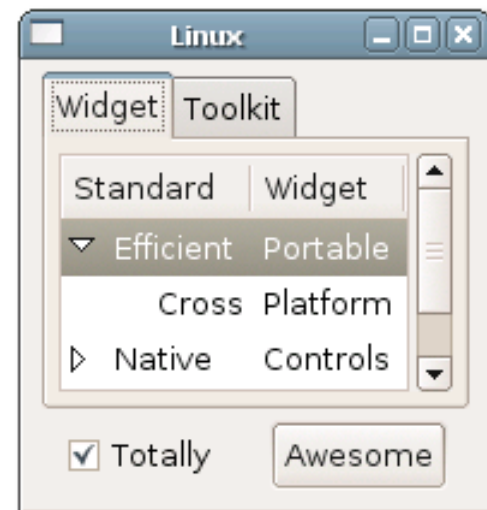
SWT történelem

- SWT – Standard Widget Toolkit
 - IBM fejlesztés
 - Swing helyett
 - Eclipse projekt indulásakor
 - Előzmény
 - Natív GUI komponensek elérése Smalltalk nyelven
 - Cél
 - Natív elemekből felépített GUI keretrendszer

SWT tulajdonságok

■ Natív

- Platform API-t használja
- Gyors
- Look-and-feel a platformé
 - Minden szolgáltatás elérhető
 - OLE, drag-n-drop, ...
 - Portolni kell
 - Eltérő megjelenés



SWT tulajdonságok

- Átgondolt struktúra
 - Egyszerű komponensek
 - Hierarchikus szerkezetek
 - Átlátható esemény-kezelés
 - Átlátható API
- Bővíthető
 - Saját widget-ek (nem natív)
 - Optimális funkció-teljesítmény arány érhető el

SWT alapelemek

■ Display

- Kapcsolat operációs rendszer és az SWT alkalmazás között
 - Egy SWT program – egy Display
- Feladat: Esemény-szétosztás
 - Egér
 - Billentyűzet
 - Monitor
 - ...
- Felső szintű Shell és Monitor objektumok

■ Shell

- Egy “ablak” megfelelője

SWT alapelemek

- Composite
 - Konténer elem, más elemeket (composite, control) tartalmazhat
- Control
 - Egy operációs rendszer szintű vezérlőt reprezentál (pl. Button, Label...)
 - A Shell és a Composite is ez
- Az összes osztály a Widget leszármazottja

SWT alapelemek

■ Eseményhurok

- Explicit - az alkalmazásban le kell kódolni
- Egy ciklusban
 - bejövő események olvasása
 - események feldolgozása
- Ciklus vége: az alkalmazás véget ért (főprogram bezárva)
- Nagyon hasonlít a Windows API programozáshoz

```
while(!shell.isDisposed()){  
    if(!display.readAndDispatch())  
        display.sleep ();  
}
```


SWT események

- Mi is az az esemény?
 - Minden esemény, ami számít
 - Felhasználó vagy az operációs rendszer generálja
 - Listenerek kezelik az eseményeket
 - Típus nélküli Listener (minden Widgeten)
 - Típusos Listener (a megfelelő controlokon)
- Szabályok
 - Ha több figyelőt adunk hozzá
 - A hozzáadás sorrendjében hívódnak meg
 - Egy figyelőt többször adunk hozzá
 - Többször hívódik meg
 - Többször kell eltávolítani

SWT események

■ Típus nélküli listener

○ Listener hozzáadása

- `addListener(int, Listener)`
- Első paraméter:
 - Típuszűrő
 - Integer mask
 - » Pl.: `SWT.Selection | SWT.Show`
- Csak a megfelelő események továbbítódnak

○ Listener

- `handleEvent(Event)`

○ Programozott eseményküldés

- `notifyListeners(int, Event)`

SWT események

■ Típusos listener

○ Listener hozzáadása

- addXYZListener(XYZListener)
 - Csak megfelelő típusú kezelő adható meg

○ Listener

- specifikus metódusok
- specifikus argumentumok
- Inkább objektum orientált megközelítés

○ Adapter osztályok

- Ősosztályok Listenerekhez
- Ne kelljen a nem figyelt eseményekhez üres metódus

Billentyűzet kezelése

- “Keyboard focus” - billentyűleütések célpontja
 - `setFocus(Control)`
 - `isFocusControl(Control)`
 - `forceFocus(Control)`
- Események
 - `KeyDown`
 - `KeyUp`
- `FocusEvent` - `FocusListener`

Billentyűzet kezelése

- Speciális karakterek azonosítás SWT konstanssal:
 - BS, CL, ESC, DEL, LF, TAB
- stateMask - különleges gombok állapota
 - SHIFT, ALT, CTRL, COMMAND
- “Traversal key”
 - Kontrollok bejárési sorrendje
 - A kontroll dönt a feldolgozásról

Billentyűzet kezelése

- AcceleratorKey - gyorsbillentyű
 - Kontrollhoz kötjük
 - `setAccelerator(int)`
 - A paraméter: Karakter + módosító maszk
- Window Manager gyorsbillentyűk
 - Nem jutnak el a programhoz
 - Az ablakkezelő rendszer “elnyeli”
 - Pl. Alt+F4 Windows-on

Egér kezelése

■ Mutatóeszköz

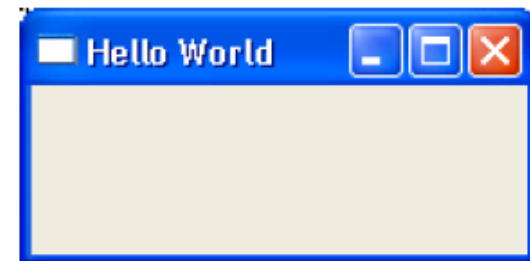
- 1-2-3 gomb
- Kurzor
 - Pozíció jelzése
 - Változó alakú (alatta levő elemtől (is) függően)

■ Események

- Enter, Exit, Down, Up, Move, Hover, DoubleClick
- Elemek
 - Koordináták - control-relatív
 - Button - az akcióban résztvevő gomb
 - Gomb maszk - A stateMask része
 - Button1, Button2, Button3
 - Button_MASK
 - State maszk: billentyűzet információ is!

SWT Hello, world

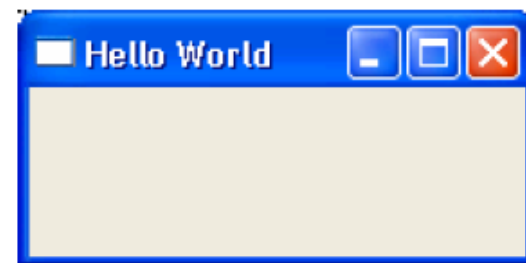
```
1 import org.eclipse.swt.*;
2 import org.eclipse.swt.graphics.*;
3 import org.eclipse.swt.widgets.*;
4 public class HelloWorld{
5     public static void main(String[] args) {
6         Display display = new Display();
7         Shell shell = new Shell(display);
8         shell.setText("Hello, World!");
9         shell.setSize(200, 100);
10        shell.open ();
11        while (!shell.isDisposed()) {
12            if (!display.readAndDispatch())
13                display.sleep ();
14        }
15        display.dispose ();
16    }
```



SWT Hello, world

```
1 import org.eclipse.swt.*;
2 import org.eclipse.swt.graphics.*;
3 import org.eclipse.swt.widgets.*;
4 public class HelloWorld{
5     public static void main(S
6         Display display = new Di
7         Shell shell = new Shell(display);
8         shell.setText("Hello, World!");
9         shell.setSize(200, 100);
10        shell.open ();
11        while (!shell.isDisposed()) {
12            if (!display.readAndDispatch())
13                display.sleep ();
14        }
15        display.dispose ();
16 }
```

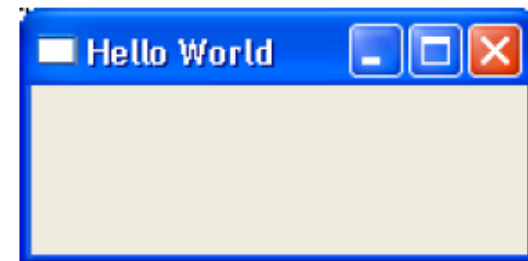
SWT könyvtárak



SWT Hello, world

```
1 import org.eclipse.swt.*;
2 import org.eclipse.swt.grap
3 import org.eclipse.swt.widg
4 public class HelloWorld{
5     public static void main(String[] args) {
6         Display display = new Display();
7         Shell shell = new Shell(display);
8         shell.setText("Hello, World!");
9         shell.setSize(200, 100);
10        shell.open ();
11        while (!shell.isDisposed()) {
12            if (!display.readAndDispatch())
13                display.sleep ();
14        }
15        display.dispose ();
16    }
```

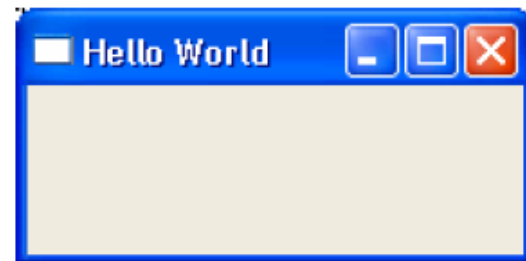
Display
referencia



SWT Hello, world

```
1 import org.eclipse.swt.*;
2 import org.eclipse.swt.graphics.*;
3 import org.eclipse.swt.widgets.*;
4 public class HelloWorld{
5     public static void main(String[] args) {
6         Display display = new Display();
7         Shell shell = new Shell(display);
8         shell.setText("Hello, World!");
9         shell.setSize(200, 100);
10        shell.open ();
11        while (!shell.isDisposed()) {
12            if (!display.readAndDispatch())
13                display.sleep ();
14        }
15        display.dispose ();
16    }
```

Új ablak a
képernyőn

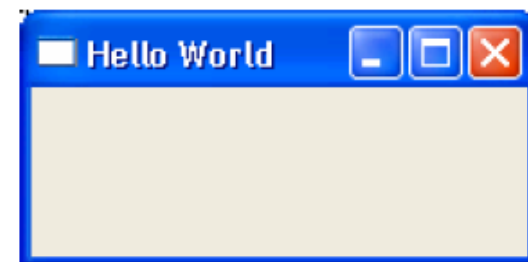


SWT Hello, world

```
1 import org.eclipse.swt.*;
2 import org.eclipse.swt.graphics.*;
3 import org.eclipse.swt.widgets.*;
4 public class HelloWorld{
5     public static void main(String[] args) {
6         Display display = new Display();
7         Shell shell = new Shell(display);
8         shell.setText("Hello, World!");
9         shell.setSize(200, 100);
10        shell.open ();
11        while (!shell.isDisposed()) {
12            if (!display.readAndDispatch())
13                display.sleep ();
14        }
15        display.dispose ();
16    }
```

Ablak cím

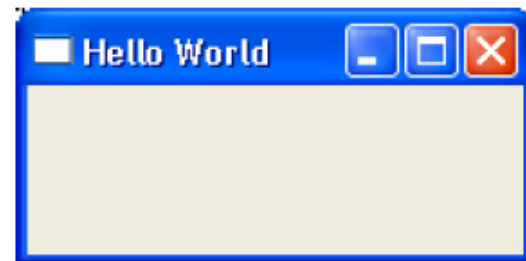
beállítása



SWT Hello, world

```
1 import org.eclipse.swt.*;
2 import org.eclipse.swt.graphics.*;
3 import org.eclipse.swt.widgets.*;
4 public class HelloWorld{
5     public static void main(String[] args) {
6         Display display = new Display();
7         Shell shell = new Shell(display);
8         shell.setText("Hello, World!");
9         shell.setSize(200, 100);
10        shell.open ();
11        while (!shell.isDisposed()) {
12            if (!display.readAndDispatch())
13                display.sleep ();
14        }
15        display.dispose ();
16    }
```

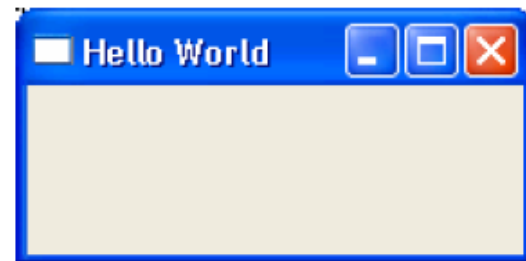
Ablak méret
beállítása



SWT Hello, world

```
1 import org.eclipse.swt.*;
2 import org.eclipse.swt.graphics.*;
3 import org.eclipse.swt.widgets.*;
4 public class HelloWorld{
5     public static void main(String[] args) {
6         Display display = new Display();
7         Shell shell = new Shell(display);
8         shell.setText("Hello World");
9         shell.setSize(200, 100);
10        shell.open ();
11        while (!shell.isDisposed()) {
12            if (!display.readAndDispatch())
13                display.sleep ();
14        }
15        display.dispose ();
16    }
```

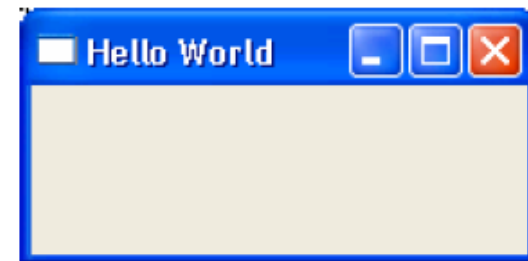
Ablak
megnyitása



SWT Hello, world

```
1 import org.eclipse.swt.*;
2 import org.eclipse.swt.graphics.*;
3 import org.eclipse.swt.widgets.*;
4 public class HelloWorld{
5     public static void main(String[] args) {
6         Display display = new Display();
7         Shell shell = new Shell(display);
8         shell.setText("Hello, World!");
9         shell.setSize(200, 100);
10        shell.open ();
11        while (!shell.isDisposed()) {
12            if (!display.readAndDispatch())
13                display.sleep ();
14        }
15        display.dispose ();
16    }
```

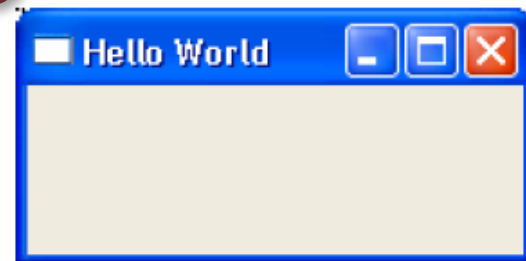
Eseményhurok



SWT Hello, world

```
1 import org.eclipse.swt.*;
2 import org.eclipse.swt.graphics.*;
3 import org.eclipse.swt.widgets.*;
4 public class HelloWorld{
5     public static void main(String[] args) {
6         Display display = new Display();
7         Shell shell = new Shell(display);
8         shell.setText("Hello, World!");
9         shell.setSize(200, 100);
10        shell.open ();
11        while (!shell.isDisposed()) {
12            if (!display.readAndDispatch())
13                display.sleep ();
14        }
15        display.dispose ();
16    }
```

Erőforrások
felszabadítása



■ Platform Interface

- Java Native Interface
 - Platform API elérése
- Egy-az-egyhez leképezés
 - Platform függvénynevei és paraméterei
 - Azonos elnevezési konvenciók
- Minden lényegi rész Java kódban
- Platform API ismeretében portolható az SWT

SWT runtime

■ Futtatáshoz szükséges

○ SWT jar fájlok

- swt.jar - az SWT alap csomagja
- Egyéb, platformfüggő jar fájlok
- Fordításhoz is kellenek

○ SWT osztott könyvtárak

- SWT Platform Interface (PI)
- Platform specifikus név, pl. swt-XX.dll (Windows), libswt-XX.so (Linux)
- Egyéb platform-függő libek halmaza

Az SWT belülről

Az SWT osztály

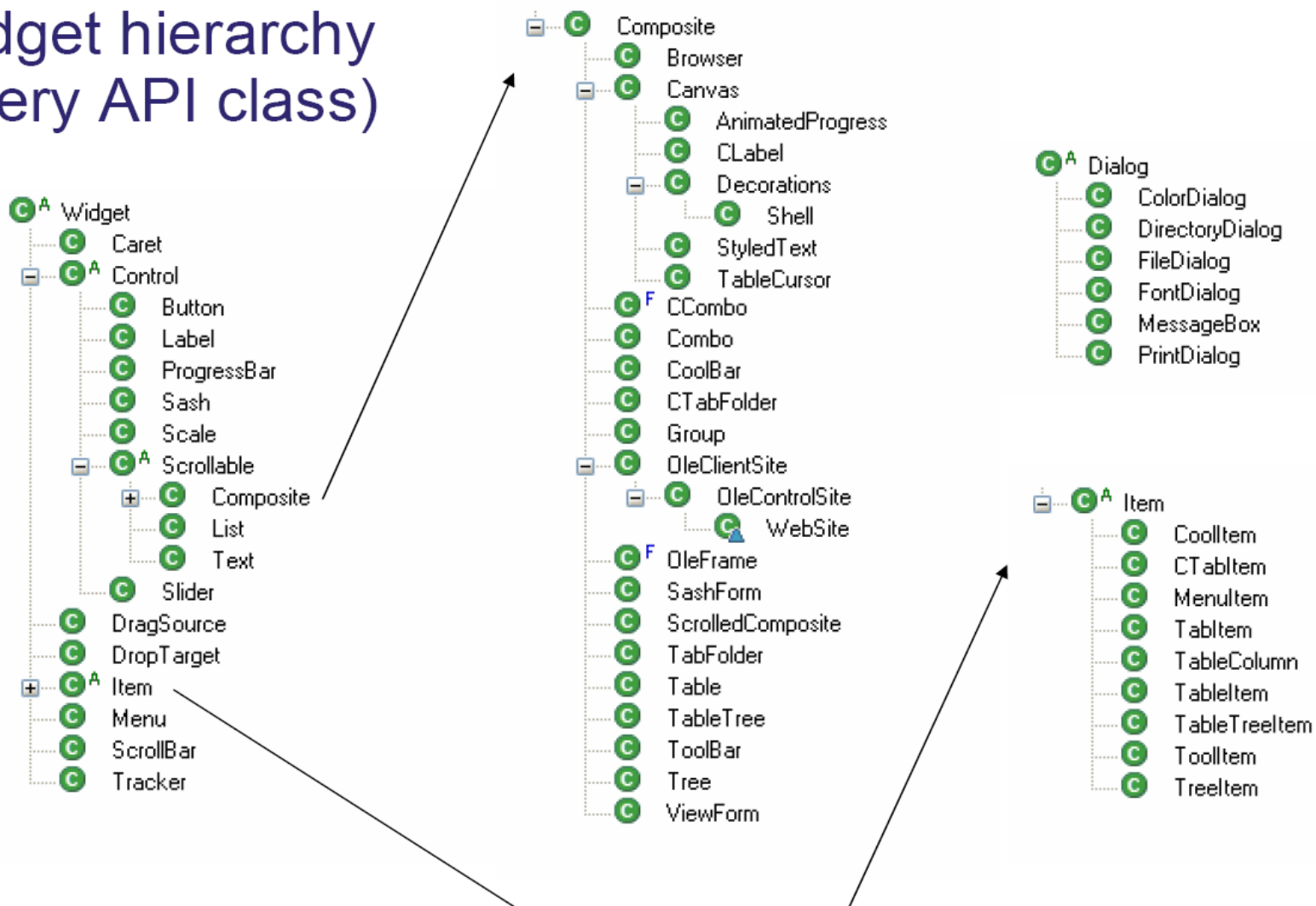
- SWT konstansok
 - SWT.PUSH, SWT.RADIO
 - SWT.Selection
- Fontos metódusok
 - getPlatform()
 - getVersion()
 - error()
- Megjegyzések
 - Platform: string (pl. “win32”, “gtk”)
 - Verzió - int

SWT hibák

- Háromféle kivétel
- SWTError - kritikus, végzetes hiba
 - Code: SWT hibakód
 - Pl.: `SWT.ERROR_CANNOT_SET_ENABLED`
 - engedélyezés nem beállítható
 - Throwable: a hibát okozó kivétel
 - getMessage: szöveges kivételleírás
- SWTException
 - Kevésbé kritikus hiba
 - Pl. I/O error (fájl nem található)
 - Mezők: mint SWTError-nál
- IllegalArgumentException
 - Hibás bemenő paraméter esetén

Widget hierarchy

Widget hierarchy (every API class)



Widget konstruktorok

- Widgeteknek mindig van szülőjük
- Tipikus konstruktor: `Widget(parent, style)`
- Stílus: SWT konstansok kombinációi (bitszintű vagy)
- Példák:
 - `new Label(shell, SWT.NONE);`
 - `Button push = new Button(shell, SWT.PUSH);`
 - `Button radio = new Button(shell, SWT.RADIO);`
- Kivétel: shell szülője `Shell` vagy `Display`

SWT widgetek

Widget

- Minden UI osztály közös absztrakt őse
- Konstruktorral példányosítjuk (nem factory)
 - Létrehozásakor OS erőforrások lefoglalódnak
- Törlés programozottan (dispose) vagy felhasználói művelet során
 - Elengedi az OS erőforrásokat
 - Eltér a Java garbage collection filozófiától!
- Alkalmazás-specifikus adatok tárolása:
 - `getData (Object)` – `getData ()`
 - egyetlen objektum
 - `setData (String, Object)` – `getData (String)`
 - kulcs-érték párok
 - sok érték esetén lassú
 - Alkalmazható MVC minta modellreferencia tárolására

Elemek törlése

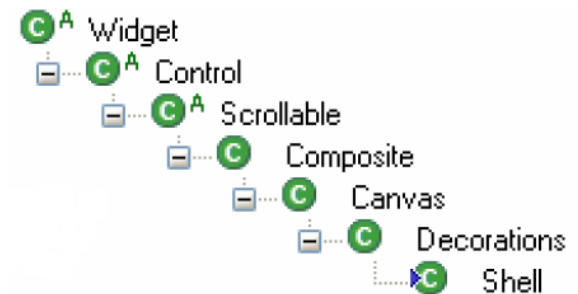
- Az erőforrásalapú elemeket explicit meg kell semmisíteni
- Ilyenek: Widget, Color, Cursor, Font, GC, Image, Region, Device, ...
- Mantra: Te csináltad, te semmisíted meg!
 - Programozó semmisíti meg:
 - `Font font = new Font(...);`
 - Programozó nem semmisítheti meg!
 - `Font font = control.getFont();`
- Szabály: Egy elem megsemmisítése a gyerekeit is megsemmisíti
 - `shell.dispose()` - az ablak minden eleme
 - `menu.dispose()` - a menü minden eleme
 - `tree.dispose()` - a fa minden eleme
 - Fontos: a `setMenu()` segítségével beállított menü is törlődik

Control osztály

- Ősosztály minden “nehézsúlyú” UI elemnek
- Stílusok
 - BORDER, LEFT_TO_RIGHT, RIGHT_TO_LEFT
- Események
 - FocusIn, FocusOut, KeyDown, KeyUp, Traverse, MouseDown, MouseUp, MouseDoubleClick, MouseEnter, MouseExit, MouseMove, Move, Resize, Paint, Help

Shell osztály

- `new Shell(display, SWT.SHELL_TRIM) ;`
- `new Shell(shell, SWT.DIALOG_TRIM) ;`
- **Stílusok**
 - `BORDER, CLOSE, MIN, MAX, NO_TRIM, RESIZE, TITLE, APPLICATION_MODAL, MODELESS, PRIMARY_MODAL, SYSTEM_MODAL`
- **Események**
 - `Activate, Close, Iconify,`
 - `Deiconify, Deactivate`



Shell – Az ablak

■ Metódusok

- `open()`
- `close()`
- `setActive()`

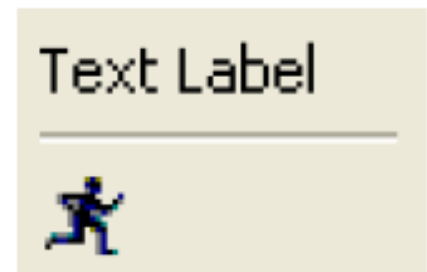
■ Megjegyzés

- A legfelső ablak szülője a display
- Dialógusok szülője a legfelső szintű ablak

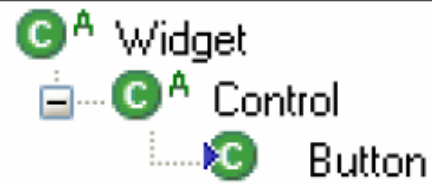
Label - Címkék



- `new Label(parent, SWT.NONE)`,
- Statikus szöveg vagy kép megjelenítése
- Stílusok
 - WRAP, LEFT, CENTER, RIGHT, SEPARATOR, HORIZONTAL, VERTICAL, SHADOW_IN, SHADOW_OUT
- Fontos metódusok
 - `setText(String)`
 - `setImage(Image)`
 - `setAlignment(Left | Center | Right)`



Button - Gombok



- `new Button(parent, SWT.PUSH);`
- Többféle gomb megjelenítése
- Stílusok
 - ARROW, CHECK, PUSH, RADIO, TOGGLE, FLAT, UP, DOWN, LEFT, CENTER, RIGHT
- Események
 - Kiválasztás

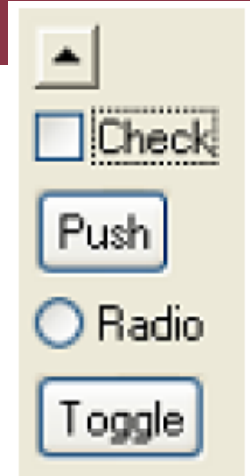
Button - Gombok

■ Metódusok

- `setText (String)`
- `setImage (Image)`
- `setAlignment (int)`
- `setSelection (boolean)`
 - check, radio és toggle stílusok esetén

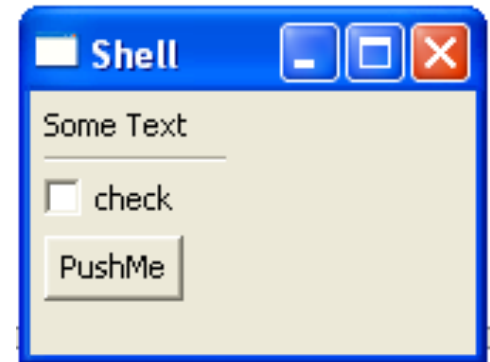
■ Megjegyzés

- Rádió gombok csoportosíthatóak
- Az “arrow” gombok iránnyal rendelkezhetnek



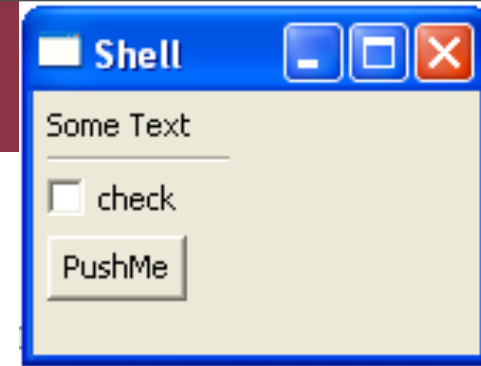
Példa

```
public static void main(String[] args) {  
    Display display = Display.getDefault();  
    MyDlg thisClass = new MyDlg();  
    thisClass.createSShell();  
    thisClass.sShell.open();  
  
    while (!thisClass.sShell.isDisposed()) {  
        if (!display.readAndDispatch())  
            display.sleep();  
    }  
    display.dispose();  
}
```

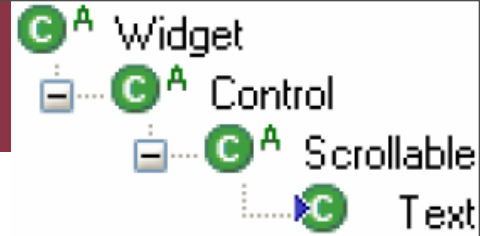


Példa

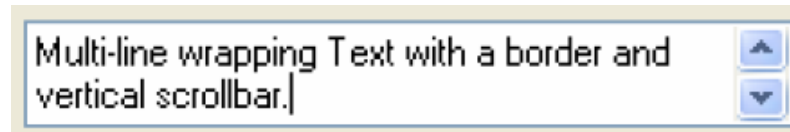
```
private void createSShell() {  
    sShell = new Shell();  
    sShell.setText("Shell");  
    sShell.setLayout(new GridLayout());  
    sShell.setSize(new Point(90,127));  
    label1 = new Label(sShell, SWT.NONE);  
    label1.setText("Some Text");  
    label2 = new Label(sShell, SWT.SEPARATOR |  
SWT.HORIZONTAL);  
    label2.setText("Label");  
    checkBox = new Button(sShell, SWT.CHECK);  
    checkBox.setText("check");  
    button = new Button(sShell, SWT.NONE);  
    button.setText("PushMe");  
}
```



Text - Szövegbevitel



- Szövegbeviteli komponens (család)
 - Egy- vagy többsoros
- Stílusok
 - SINGLE, MULTI, READ_ONLY, WRAP, LEFT, CENTER, RIGHT, PASSWORD
- Események
 - Modify, Verify, DefaultSelection (Enter)



Text - Szövegbevitel

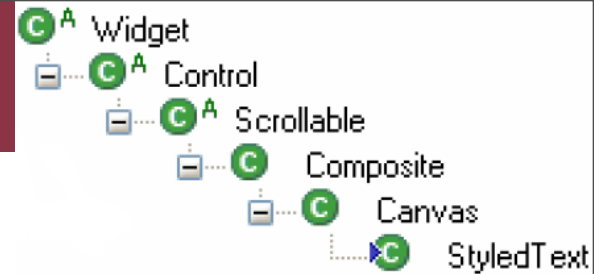
■ Metódusok

- `setText (String)`
- `setSelection (int, int)`
- `cut () , copy () , paste ()`
- `insert (String) , append (String)`
- `getLineCount () , getLineHeight ()`

■ Megjegyzés

- Kurzor (caret) jelzi az aktuális beviteli pontot
- Indexek 0 bázisúak

StyledText

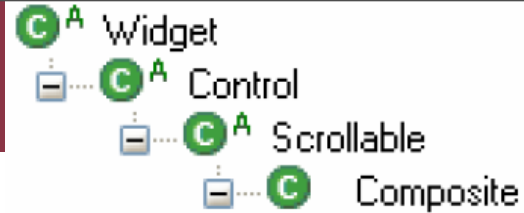


- Formázott szöveg - **nem natív!**
 - `new StyledText(parent, style)`
- Stílusok
 - SINGLE, MULTI, READ_ONLY, WRAP, FULL_SELECTION
- Események
 - DefaultSelection, Modify, Verify, ExtendedModify
- Akciók
 - `invokeAction(int act)`
 - `setKeyBinding(int key, int act)`

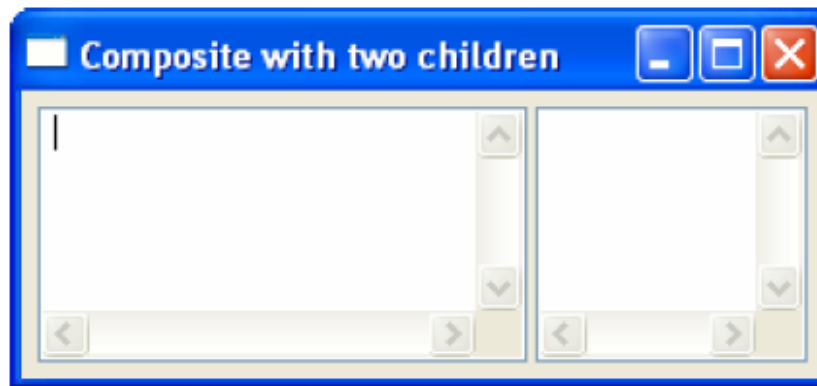
StyledText

- **Statikus tartalom manipuláció**
 - `setText(String)`
 - `setLineBackground(int, int, Color)`
 - `setStyleRanges(StyleRange[])`
- **Dinamikus tartalom manipuláció**
 - `setContent(StyledTextContent)`
 - Saját tároló implementáció
 - `addLineBackgroundListener(...)`
 - `addLineStyleListener(...)`
- **Megjegyzés**
 - Példák: `TextEditor`, `JavaEditor` (eclipse.org)

Composite



- Konténer más elemek tárolására
 - `new Container(parent, SWT.NONE)`
- Stílusok
 - `NO_BACKGROUND`, `NO_FOCUS`,
`NO_MERGE_PAINTS`, `NO_RADIO_GROUP`,
`NO_REDRAW_RESIZE`



Composite

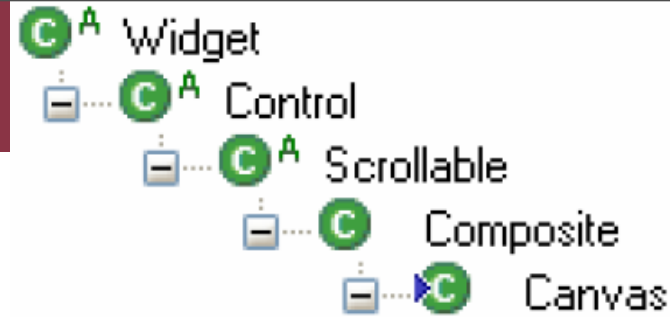
■ Metódusok

- `getChildren()`
- `setLayout(Layout)`
- `layout(boolean)`
- `setTabList(Control[])`

■ Megjegyzés

- Lehetnek gyermekei
- Lehet layout-ja
- Örököltethető, hogy saját elemeket hozhassunk létre

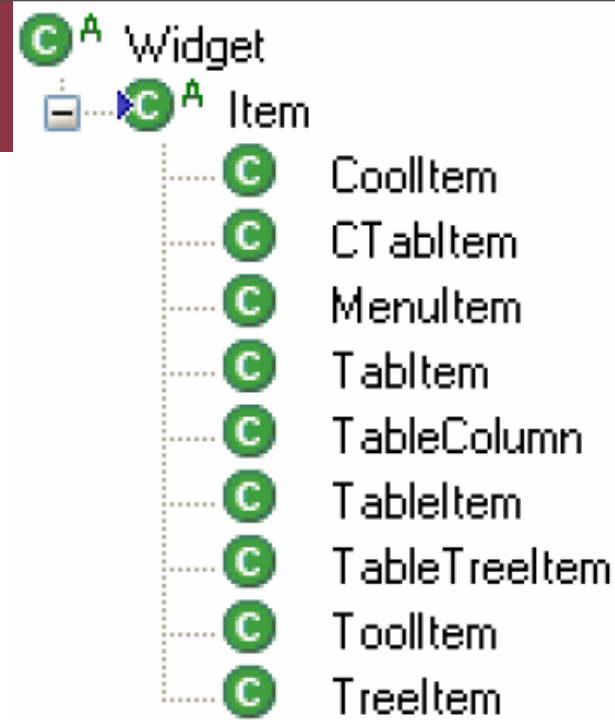
Canvas



- Festővászon
- `new Canvas(parent, style)`
- Szabad grafika készítésének alapja
- Saját kurzor (Caret) lehet benne
- Rajzolás
 - Saját alrendszer az SWT-ben
 - Később!

Item

- Általános elem osztály
- Mindig felsorolás jellegű elemen belül
 - Tree -> TreeItem
 - Menu -> MenuItem
- Általános metódusok
 - `setText (String)`
 - `setImage (Image)`



Menu

- Menü megjelenítése
- `new Menu(parent, SWT.BAR);`
 - `BAR` – felső szintű menü
 - `DROP_DOWN` – almenü
 - `POP_UP` – helyi menü
- **Fontos:** `shell.setMenuBar(Menu)`
- Menü elemek: `MenuItem`
 - `SWT.CASCADE` – almenüje van
 - `SWT.PUSH` – "sima" elem
- Egy item-nek is lehet menüje (almenü)

Toolbar

- `new ToolBar(parent, SWT.HORIZONTAL)`
- Eszközkészlet megjelenítése
- Stílusok
 - `HORIZONTAL, VERTICAL`
 - `WRAP, FLAT, RIGHT`
 - `SHADOW_OUT`
- Események: nincs
- Metódusok: `getItems()`, `getItem(int)`

ToolItem

- `new ToolItem(toolbar, stílus)`
- **Stílusok**
 - `PUSH, RADIO, DROP_DOWN, SEPARATOR, CHECK`
- Az API hasonlít a `Button`-éhoz
- `SEPARATOR` esetén a szélesség (`setWidth(int)`) beállítható

CoolBar, CoolItem

- `new CoolBar(parent.SWT.FLAT)`
- **Eszköz-kontrollok elhelyezése**
 - Átrendezhető, testre szabható
- `new CoolItem(coolBar, SWT.NONE)`
 - Tartó a kontrol számára
 - Méretezhető (`preferredSize`, `minimumSize`)
- **Fontos:** `CoolItem.setControl()`
- `CoolBar.setLocked()`

Tree

- `new Tree(parent.SWT.SINGLE)`
- **Stílusok**
 - `SINGLE` – egyszeres kijelölés
 - `MULTI` – többes kijelölés
 - `CHECK` – checkbox-os tree
- **Események**
 - `Selection` – egy elemet kiválasztottak
 - `defaultSelection` – egy elemen Enter-t ütöttek
 - `Collapse` – bezártak egy részfat
 - `Expand` – kinyitottak egy részfat
- **Mindig van scrollbar!**

Treeltem

- `new TreeItem(parent, SWT.NONE)`
- Szülő
 - Tree – gyökérelem
 - Treeltem - gyerek
- Lehet
 - Szöveg, ikon, checkbox
- Az expand eseménnyel lehet lazy-load

Tree és Item

■ Érdekes metódusok

- `showSelection()` – megmutatja a kijelölt elemet
- `showItem(TreeItem)` – megmutatja az itemet
- `setSelection(TreeItem[])` – `getSelection()`
- `selectAll()` – `deselectAll()`
- `setTopItem()` – a legfelső láthatóvá teszi az elemet, ha lehet
- `getTopItem()` – visszaadja a legfelső elemet

Table

- Egyszerű táblázat
- `new Table(parent, SWT.SINGLE)`
- Stílusok
 - SINGLE – egy kijelölés
 - MULTI – több kijelölés
 - CHECK – check boxok
 - FULL_SELECTION – teljes sort lehet kijelölni
 - HIDE_SELECTION – ha nincs fókusz nem mutatja a kijelölést

Table

■ Metódusok

- `setHeaderVisible(boolean)`
- `setLinesVisible(boolean)`
- `getHeaderHeight()`, `getItemHeight()`
- `indexOf(TableColumn)`, `indexOf(TableItem)`
- `getColumn(int)`, `getItem(int)`
- `setTopIndex(int)`
- `isSelected(int)`, `setSelection(int[])`,
`setSelection(TableItem[])`

TableColumn és Item

- `new TableColumn(table, SWT.LEFT)`
 - Lehet szövege, képe, igazítás
 - Állítható szélesség
 - Move, Resize, Selection események
- `new TableItem(table, SWT.NONE)`
 - Szöveg, kép, checkbox lehet

Table példa

```
Table table = new Table(shell, SWT.BORDER) ;
table.setHeaderVisible(true) ;
for(int i = 0; i < 4; i++) {
    TableColumn column = new
        TableColumn(table, SWT.NONE) ;
    column.setText("Column"+ i) ;}

for(int i = 0; i < 5; i++) {
    TableItem item = new TableItem(table,
        SWT.NULL) ;
    for(int j = 0; j < 4; j++) {
        item.setText(j, "Item "+ i + "-" + j) ;}}
    for(int i = 0; i < 4; i++){
        table.getColumn(i).pack() ;}
table.pack() ;
```

Table példa

```
Table table = new Table(shell, SWT.BORDER) ;  
table.setHeaderVisible(true) ;  
for(int i = 0; i < 4; i++) {  
    TableColumn column = new  
        TableColumn(table, SWT.NONE,  
            column.setText("Column"+ i) ; }
```

Új tábla

```
for(int i = 0; i < 5; i++) {  
    TableItem item = new TableItem(table,  
        SWT.NULL) ;  
    for(int j = 0; j < 4; j++) {  
        item.setText(j, "Item "+ i + "-" + j) ; } }  
    for(int i = 0; i < 4; i++) {  
        table.getColumn(i).pack() ; }  
table.pack() ;
```

Table példa

```
Table table = new Table(shell, SWT.BORDER) ;
table.setHeaderVisible(true) ;
for(int i = 0; i < 4; i++) {
    TableColumn column = new
        TableColumn(table, SWT.NONE, 4,
            "Column " + i);
    column.setText("Column" + i);
}

for(int i = 0; i < 5; i++) {
    TableItem item = new TableItem(table,
        SWT.NULL);
    for(int j = 0; j < 4; j++) {
        item.setText(j, "Item " + i + "-" + j);
    }
    for(int i = 0; i < 4; i++) {
        table.getColumn(i).pack();
    }
}
table.pack();
```

Négy oszlop
létrehozása

Table példa

```
Table table = new Table(shell, SWT.BORDER) ;
table.setHeaderVisible(true) ;
for(int i = 0; i < 4; i++) {
    TableColumn column = new
        TableColumn(table, SWT.NONE) ;
    column.setText("Column"+ i) ;}

for(int i = 0; i < 5; i++) {
    TableItem item = new TableItem(table,
        SWT.NULL) ;
    for(int j = 0; j < 4; j++) {
        item.setText(j, "Item "+ i + "-" + j) ;}}
    for(int i = 0; i < 4; i++) {
        table.getColumn(i).pack() ;}
table.pack() ;
```

Értékek beállítása
soronként és
oszloponként

Table példa

```
Table table = new Table(shell, SWT.BORDER) ;
table.setHeaderVisible(true) ;
for(int i = 0; i < 4; i++) {
    TableColumn column = new
        TableColumn(table, SWT.NONE) ;
    column.setText("Column"+ i) ;}

for(int i = 0; i < 5; i++) {
    TableItem item = new TableItem(table,
        SWT.NULL) ;
    for(int j = 0; j < 4; j++) {
        item.setText("Item "+ i + "-" + j) ;}}
    for(int i = 0; i < 5; i++) {
        table.getColumn(i).pack() ;}
table.pack() ;
```

Csomagolás

Browser – Beépített böngésző

- `new Browser(parent, SWT.NONE)`
- Alapvetően operációs rendszer böngészőjét használja
 - DE: `SWT.MOZILLA` stílussal Mozilla böngészőt használ
 - Ha telepítve van a Mozilla XULRunner
 - Hasznos lehet, mert így minden platformon egyforma böngésző
- Események: Show, Hide, Open, Close, (URL)Changed, ...
- Metódusok
 - `setURL(String)`
 - `setText(String)`
 - `back()`, `forward()`, `stop()`

TabFolder

- `new TabFolder(parent, SWT.TOP)`
- „Lapozós dialógusablak”
- Stílusok
 - TOP – fülek felül
 - BOTTOM – fülek alul
- Esemény: `selection` – egy elem kiválasztva

TabItem

- `new TabItem(folder, SWT.NONE)`
- Egy lapot jelképez
- Metódusok
 - `setText()` - fül felirat
 - `setImage()` – fül kép
 - `setToolTipText()` – tool tipp
 - `setControl()` – a kontroll, mely az item területét elfoglalja

További widgetek

- Combo
- Group
- List
- ProgressBar
- ScrollBar
- Scale, Slider, Spinner,
- TableTree – tábla, de az első oszlop fa struktúra
- ...

Összetett űrlapok SWT-ben

Layout

Grafikus felülettervezők

Layout – AWT (gyökerek)

- Layout
 - Elemek elrendezése
 - Automatikus módszer
- A programozók elégedetlenek
 - Eddig: grafikus UI builder
 - Erőforrás fájlok
 - Most: kódolás

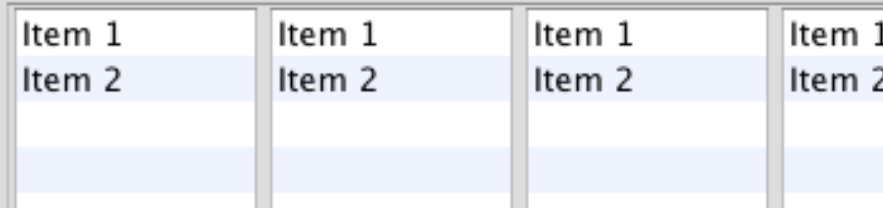
Layout - SWT

- Megtartották a layout koncepciót
 - FillLayout
 - RowLayout
 - GridLayout
 - FormLayout
 - StackLayout
- Lehet nélkülük is dolgozni

Layout

- Tartalom és elrendezés elválasztása
 - Elemek elrendezése
 - Relatív elrendezés
 - Követi a konténer méretének változását
- Közös űs: Layout
 - Nincs publikus API - nem hívjuk meg

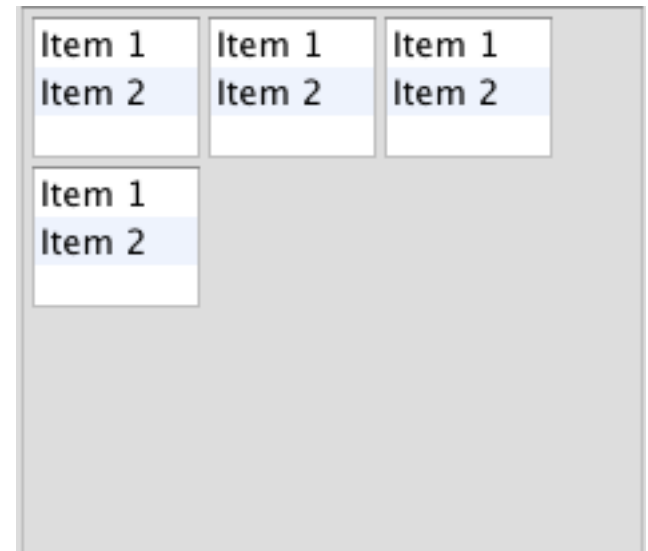
FillLayout

- Vízszintes/függőleges irányban egymás melletti elemek kitöltik a composite-ot
 - Az elemek saját méretét figyelmen kívül hagyja!
 - Primitív elrendezés
 - Használható egymásba ágyazott composite elemek esetén
- 

[illegible]

RowLayout

- Hasonló a FillLayout-hoz
 - Sorokba vagy oszlopokba rendezi az elemeket
 - Elemek méretét figyelembe veszi
- DE: minden elemnek egyedi mérete lehet
- RowData (LayoutData) : height, width
 - Az elemek méretét adja meg
- Az elemeket egyenletesen osztja el a meglevő helyen



GridLayout

- Grid (táblázat) jellegű elrendezés

- Oszlopok száma megadható

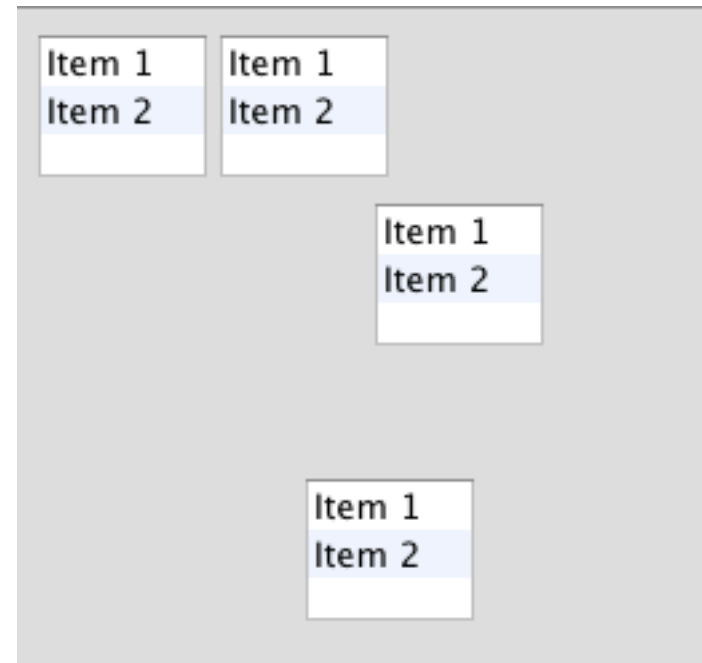
- Adattagok

- `horizontalSpacing` – elemek közötti szünet (pixel)
- `makeColumnsEqualWidth` – egyenlőek az oszlopok?
- `marginHeight` - margó magassága felül és alul
- `marginWidth`- margó szélessége jobb- és baloldalt
- `numColumns` - oszlopok száma
- `verticalSpacing` – elemek közötti szünet



FormLayout

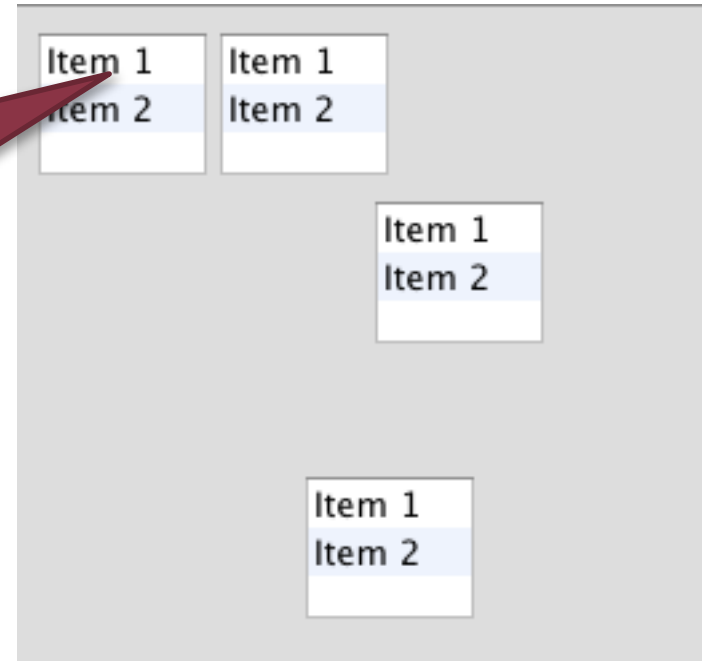
- A legkomplexebb layout
- A Data mellett egy Attachment osztályt is használ
 - Egy Data 4 attachmentet tartalmazhat (négy oldalhoz)
 - Az attachment a méretet adja meg
 - Alapképlet: $y=ax+b$ (y magasság, x szélesség)
 - a : a vonatkozó widgethez képesti arány, b offset



FormLayout

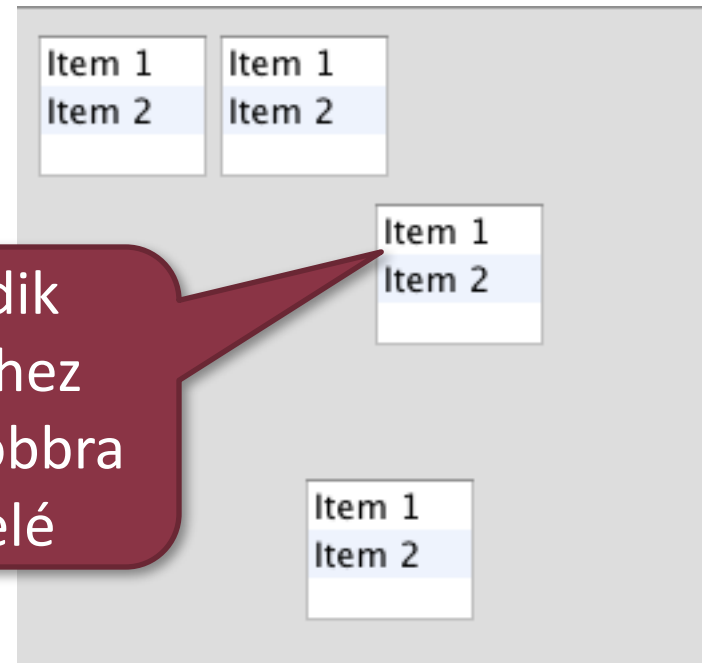
- A legkomplexebb
- A Data mellett egy `FormLayout` osztályt is használhatunk
 - Egy Data 4 attachmenetet tartalmazhat (négy oldalhoz)
 - Az attachmenet a méretet adja meg
 - Alapképlet: $y=ax+b$ (y magasság, x szélesség)
 - a : a vonatkozó widgethez képesti arány, b offset

Abszolút
pozíció a bal
felső sarokhoz
képest



FormLayout

- A legkomplexebb layout
- A Data mellett egy Attachment osztályt is használ
 - Egy Data 4 attachmentet tartalmazhat (négy oldal)
 - Az attachment a méretet adja meg
 - Alapképlet: $y=ax+b$ (y magasság, x szélesség)
 - a : a vonatkozó widgethez képesti arány, b offset



FormLayout

■ FormAttachment

- Alignment – a csatolt control-hoz képesti elrendezés (top, center, bottom, left, center, right)
- Control – a csatolt Control
- Denominator – a nevezője (def. 100)
- Numerator – a számlálója
- Offset – b

FormLayout

■ FormData

- `Left`, `right`, `top`, `bottom`: a négy `FormAttachment`
- `Height` : a control vízszintes mérete (kért)
- `Width` : a control függőleges mérete (kért)

StackLayout

- Minden elem azonos méretű, azonos helyen van
- Csak a legfelső látszik
 - `StackLayout.topControl`
 - Ha átállítjuk, a konténerre meg kell hívni a `layout()` függvényt, hogy aktualizálódjon a GUI
- Megadható a margó
 - `marginHeight`
 - `marginWidth`

Layout

- Sokféle van
- Sajátot is készíthetünk a Layout osztály örököltetésével
- Nem kell használni
 - `Widget.setBounds(x,y,w,h)`
 - Abszolút méret megadása

Felhasználói felület grafikus tervezése

- Alapvető elvárások
 - Támogassa közvetlen szerkesztést
 - Használhassuk a Layoutokat
 - Forrás és grafikus szerkesztés párhuzamosan
- Egyre jobb támogatás
 - Többféle eszköz
 - DE: zömmel kereskedelmiek
 - DE: kisebb-nagyobb problémák

Eclipse Visual Editor Projekt

- Nyílt forrású grafikus felület szerkesztő
 - AWT, Swing,
 - SWT és
 - Eclipse Forms technológiákhoz
 - JFace API nem támogatott
- Új szerkesztő java forrásfájlokhoz
 - Kód és grafikus felület egyszerre látszik
 - A szerkesztő mellett egy külön ablakban megnyitja a készülő formot
- Két év szünet után kezdték újra fejleszteni
 - Alapfeladatokra megfelelő lehet

Visual Editor Projekt

The screenshot displays the Eclipse IDE with a Java Swing application named "Shell". The application window contains two radio buttons labeled "Radio1" and "Radio2", a text box with the text "Text box with long text content. This is really nice.", and a button labeled "Test Button".

The code in the editor is as follows:

```
24 sShell = new Shell();
25 sShell.setText("Shell");
26 sShell.setLayout(null);
27 sShell.setSize(new Point(300, 200));
28 button = new Button(sShell, SWT.NONE);
29 button.setText("Test Button");
30 button.setBounds(new Rectangle(66, 102, 121, 56));
31 radioButton = new Button(sShell, SWT.RADIO);
32 radioButton.setBounds(new Rectangle(70, 15, 84, 22));
33 radioButton.setText("Radio1");
34 radioButton1 = new Button(sShell, SWT.RADIO);
35 radioButton1.setBounds(new Rectangle(70, 46, 71, 22));
36 radioButton1.setText("Radio2");
37 text = new Text(sShell, SWT.BORDER | SWT.MULTI | SWT.WRAP);
38 text.setBounds(new Rectangle(163, 28, 124, 59));
39 text.setEditable(false);
```

The Properties view at the bottom shows the following properties for the selected component:

Property	Value
background	Color {239, 235, 231}
border	<not set>
bounds	0,0,300,200
enabled	true
>field name	sShell
font	Sans,10

Visual Editor Projekt

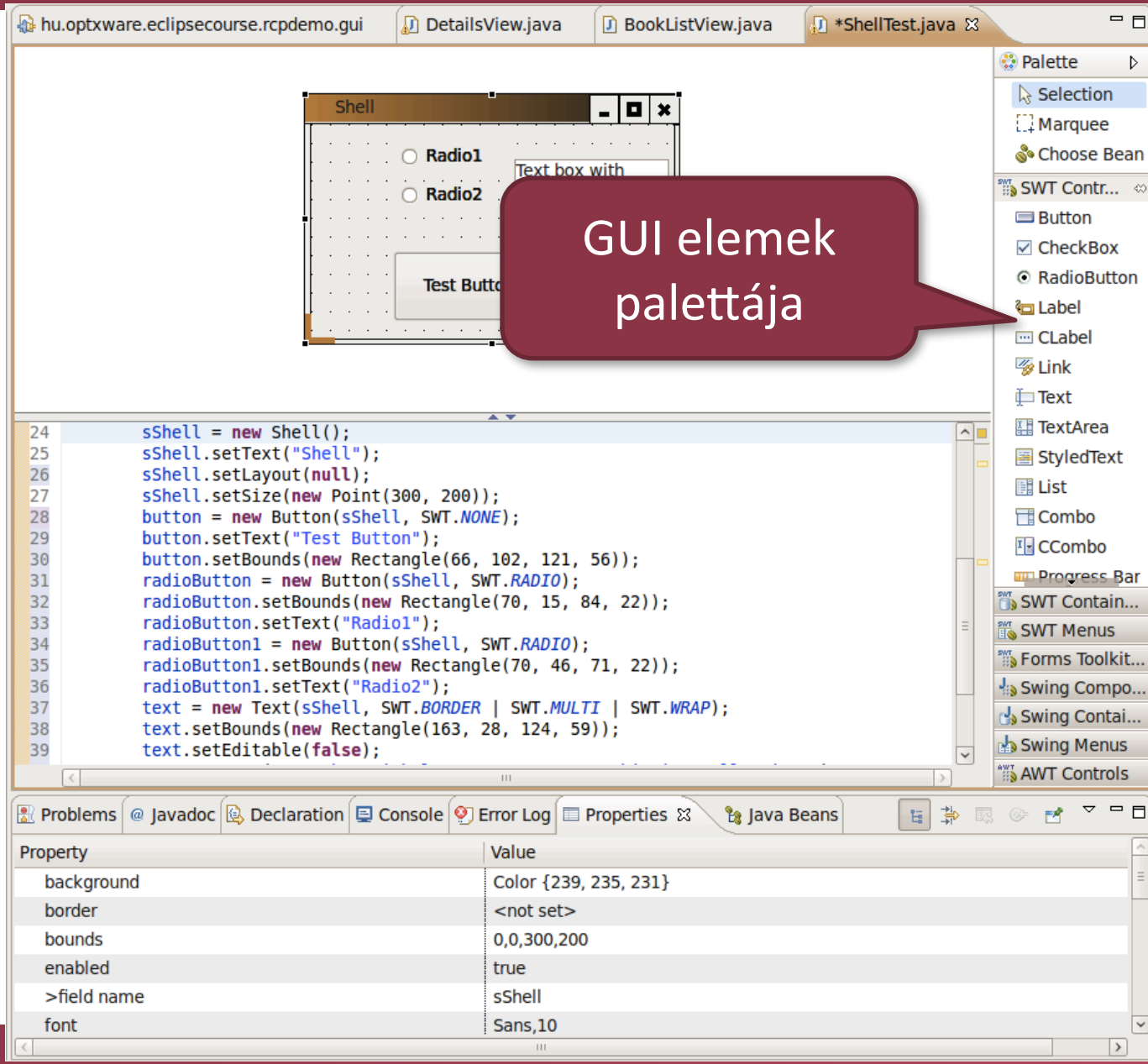
Form szerkesztő
terület

The screenshot displays the Eclipse IDE's Visual Editor for a Java SWT application. The top part shows a preview of the 'Shell' window, which contains two radio buttons labeled 'Radio1' and 'Radio2', a text box with the text 'Text box with long text content. This is really nice.', and a 'Test Button'. The bottom part shows the corresponding Java code in the editor, which creates the shell, sets its text, size, and layout, and adds the button, radio buttons, and text box. The right sidebar contains the SWT Controls palette, which lists various controls like Button, CheckBox, RadioButton, Label, CLabel, Link, Text, TextArea, StyledText, List, Combo, CCombo, Progress Bar, and others. The bottom status bar shows the Properties view for the 'sShell' object, listing properties like background, border, bounds, enabled, >field name, and font.

```
24 sShell = new Shell();
25 sShell.setText("Shell");
26 sShell.setLayout(null);
27 sShell.setSize(new Point(300, 200));
28 button = new Button(sShell, SWT.NONE);
29 button.setText("Test Button");
30 button.setBounds(new Rectangle(66, 102, 121, 56));
31 radioButton = new Button(sShell, SWT.RADIO);
32 radioButton.setBounds(new Rectangle(70, 15, 84, 22));
33 radioButton.setText("Radio1");
34 radioButton1 = new Button(sShell, SWT.RADIO);
35 radioButton1.setBounds(new Rectangle(70, 46, 71, 22));
36 radioButton1.setText("Radio2");
37 text = new Text(sShell, SWT.BORDER | SWT.MULTI | SWT.WRAP);
38 text.setBounds(new Rectangle(163, 28, 124, 59));
39 text.setEditable(false);
```

Property	Value
background	Color {239, 235, 231}
border	<not set>
bounds	0,0,300,200
enabled	true
>field name	sShell
font	Sans,10

Visual Editor Projekt



GUI elemek palettája

```
24 sShell = new Shell();
25 sShell.setText("Shell");
26 sShell.setLayout(null);
27 sShell.setSize(new Point(300, 200));
28 button = new Button(sShell, SWT.NONE);
29 button.setText("Test Button");
30 button.setBounds(new Rectangle(66, 102, 121, 56));
31 radioButton = new Button(sShell, SWT.RADIO);
32 radioButton.setBounds(new Rectangle(70, 15, 84, 22));
33 radioButton.setText("Radio1");
34 radioButton1 = new Button(sShell, SWT.RADIO);
35 radioButton1.setBounds(new Rectangle(70, 46, 71, 22));
36 radioButton1.setText("Radio2");
37 text = new Text(sShell, SWT.BORDER | SWT.MULTI | SWT.WRAP);
38 text.setBounds(new Rectangle(163, 28, 124, 59));
39 text.setEditable(false);
```

Property	Value
background	Color {239, 235, 231}
border	<not set>
bounds	0,0,300,200
enabled	true
>field name	sShell
font	Sans,10

Visual Editor Projekt

Kód szerkesztő
terület

Visual Editor Preview:

Shell

Radio1

Radio2

Text box with long text content. This is really nice.

Test Button

```
1 Shell = new Shell();
2 Shell.setText("Shell");
3 Shell.setLayout(null);
4 Shell.setSize(new Point(300, 200));
5 Button = new Button(sShell, SWT.NONE);
6 button.setText("Test Button");
7 button.setBounds(new Rectangle(66, 102, 121, 56));
8 radioButton = new Button(sShell, SWT.RADIO);
9 radioButton.setBounds(new Rectangle(70, 15, 84, 22));
10 radioButton.setText("Radio1");
11 radioButton1 = new Button(sShell, SWT.RADIO);
12 radioButton1.setBounds(new Rectangle(70, 46, 71, 22));
13 radioButton1.setText("Radio2");
14 text = new Text(sShell, SWT.BORDER | SWT.MULTI | SWT.WRAP);
15 text.setBounds(new Rectangle(163, 28, 124, 59));
16 text.setEditable(false);
```

Properties View:

Property	Value
background	Color {239, 235, 231}
border	<not set>
bounds	0,0,300,200
enabled	true
>field name	sShell
font	Sans,10

Visual Editor Projekt

The screenshot displays the Eclipse IDE with a Java Swing application named "Shell". The application window contains two radio buttons labeled "Radio1" and "Radio2", a text box with the text "Text box with long text content. This is really nice.", and a button labeled "Test Button".

The code in the editor is as follows:

```
24 sShell = new Shell();
25 sShell.setText("Shell");
26 sShell.setLayout(null);
27 sShell.setSize(new Point(300, 200));
28 button = new Button(sShell, SWT.NONE);
29 button.setText("Test Button");
30 button.setBounds(new Rectangle(66, 102, 121, 56));
31 radioButton = new Button(sShell, SWT.RADIO);
32 radioButton.setBounds(new Rectangle(70, 15, 84, 22));
33 radioButton.setText("Radio1");
34 radioButton1 = new Button(sShell, SWT.RADIO);
35 radioButton1.setBounds(new Rectangle(70, 46, 71, 22));
36 radioButton1.setText("Radio2");
37 text = new Text(sShell, SWT.BORDER | SWT.MULTI | SWT.WRAP);
38 text.setBounds(new Rectangle(163, 28, 124, 59));
39 text.setEditable(false);
```

The Properties view at the bottom shows the following properties for the selected element:

Property	Value
border	Color {239, 235, 231}
bounds	<not set>
enabled	0,0,300,200
>field name	true
font	sShell
	Sans,10

Kijelölt elem
tulajdonságai

SWT dialógus ablakok

Dialógus ablakok

- Speciális ablak
 - Információközlés
 - Beavatkozási lehetőség
 - Gyakran modálisak
 - Amíg a felhasználó nem választ, a program letiltódik

Dialógus ablakok

■ Fajtái

- MessageBox- üzenetek
- ColorDialog – szín-kiválasztás
- DirectoryDialog - könyvtárfa
- FileDialog – fájl kiválasztás, mentés
- FontDialog – betűtípus-választás
- PrintDialog – nyomtatás
- Nem Widget alosztályok!

- Az SWT az operációs rendszer beépített dialógusait használja

Dialógusok

■ Metódusok

- `setText()` – címsor beállítása
- `open()` – megnyitás, visszatérési érték a kimenetről

■ Stílusbitek

- `APPLICATION_MODAL`
 - Az alkalmazást blokkolja
- `SYSTEM_MODAL`
 - Az egész rendszert blokkolja
- `SHEET`
 - MAC OSX specifikus, különleges megjelenés

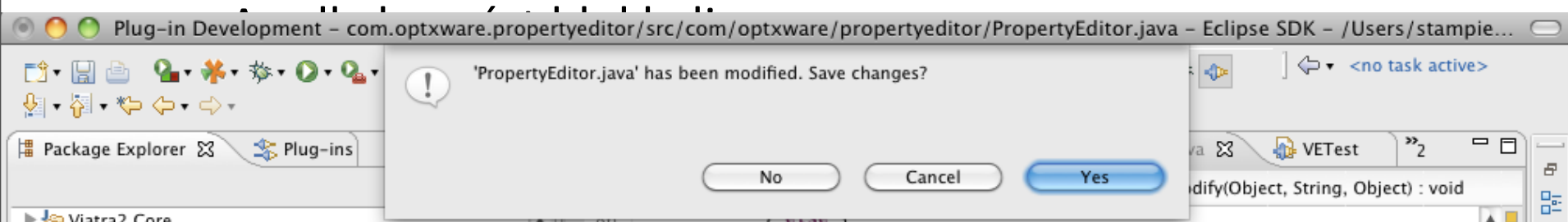
Dialogusok

■ Metódusok

- `setText()` – címsor beállítása
- `open()` – megnyitás, visszatérési érték a kimenetről

■ Stílusbitek

- `APPLICATION_MODAL`



- SHEET

- MAC OSX specifikus, különleges megjelenés

MessageBox

■ Metódusok

- `setMessage()` – Üzenet szöveg

■ Stílusok

- `ICON_ERROR`, `ICON_INFORMATION`,
`ICON_QUESTION`, `ICON_WARNING`,
`ICON_WORKING`
- `OK`, `OK | CANCEL`
- `YES | NO`, `YES | NO | CANCEL`
- `RETRY | CANCEL`, `ABORT | RETRY | IGNORE`

■ `open()`

- a lenyomott gomb kódjával tér vissza (pl. `OK`)

FileDialog

■ Metódusok

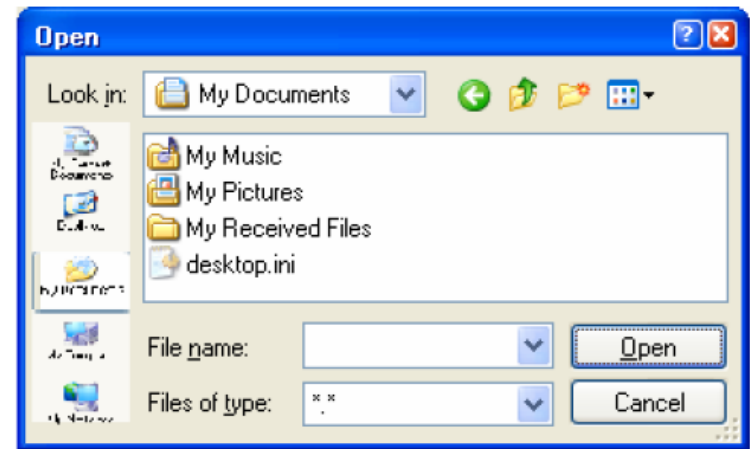
- `setFileName (String)`
- `setFileExtensions (String[])`
- `setFilterNames (String[])`
- `setFilterPath (String)`

■ Stílusok

- OPEN, SAVE, MULTI

■ `open ()` – fájlnev (String)

- Multi esetén: `getFileNames ()`



Grafika SWT-ben

Grafika

- Ha saját widget kell
- Ha rajzolni kell
- Grafika
 - alapok
 - eseménykezelés
 - nyomtatás

Koordináta-rendszer

- Origó a bal felső sarok (0,0)
- Minden távolság pixelben adott
- Jobbra és lefelé növekszik
- Jobbról-balra típusú locale esetén az x koordináta-tengely megfordulhat!
- A koordináta-rendszer általában control-relatív, azaz a kontrollon belül érvényes

Point és Rectangle osztályok

- Point – x,y koordináta-pár
 - Nincs OS erőforrás
- Rectangle – téglalap (koordináta-négyes)
 - Nincs OS erőforrás
 - API: contains(Point) intersects(Rectangle), union(Rectangle), intersection(Rectangle)

GraphicContext - GC

- Interfész a platform grafikus rendszeréhez
- Tartalmazza az aktuális rajzolási környezetet (Színek, stílusok, ...)
- Rajzolási környezetek
 - Control – általában egy kontrollra rajzolunk
 - Device – rajzolás egy eszközre (pl. Printer)
 - Image – rajzolás egy képre, az eredmény például kimenthető fájlba

Szálkezelés

- Nem muszáj az UI szálból rajzolni, de érdemes:
 - Jósolható: tudjuk, hogy senki sem nyúl a felülethez, amíg rajzolunk
 - Könnyen programozható: az UI-hoz való hozzáférés nem igényel szálak közötti hívásokat
 - Biztonságos: néhány operációs rendszer rosszul viseli a háttér-rajzolószálakat
 - Eclipse plug-inek fejlesztésekor szükséges is lehet

Rajzolás - vonalak

- `drawLine(int, int, int, int)` – egyszerű vonal
- `drawPolyLine(int[] xyArray)` – pontpár – sorozattal törtvonal rajzolása
- `drawPolygon(int[] xyArray)` – mint az előző, de a végén lezárja (utolsó párból az elsőig)

Rajzolás - alakzatok

- `drawRectangle(int, int, int, int)` – téglalap
- `drawRoundRectangle(int, int, int, int, int, int)` – Lekerekített téglalap
- `drawOval(int, int, int, int)` – ellipszis
- `drawArc(int, int, int, int, int, int)` – ellipszis-szelet
 - Szögek: fokban, a 0 jobb-vízszintes, óramutató járásával ellenkezően növekszik
- Kitöltött alakzatok: `draw` helyett `fill`

Színek

- RGB osztály
 - A szokásos számhármass
- Color osztály
 - Egy RGB és egy Device alapján számol színt
 - Specifikus színleképzést valósít meg (OS)
- SystemColor
 - OS szinten specifikált színek (pl. BLACK)
 - `Display.getSystemColor(int)` hívással megkapható

Region

- Több síkidom által alkotott terület
- `Region (Device)`
- **Metódusok:**
 - `add (Rectangle)` - hozzáadás
 - `add (Region)`
 - `add (int [])` - poligon
 - `subtract (Rectangle)` - kivonás
 - `subtract (Region)`
 - `subtract (int [])` - poligon
 - `getBounds ()` – határok (befoglaló téglalap)
 - `contains (x, y)` – tartalmazza-e a pontot
 - `intersects (Rectangle)` – van-e közös rész?
- OS erőforrásokat fog!

FontData

- `FontData(String name, int size, int style)`
 - Name: a font neve
 - Size: méret pont (1/72")-ben
 - Style: NORMAL, ITALIC, BOLD, (BOLD | ITALIC)
- **Megkapható:**
 - `FontDialog`
 - `Device.getFontList(String name, boolean scalable)`
 - Skálázható font: bármely méretben használható
 - Bitmap font: csak az előre definiált méretekben

Font

- `Font (Device, FontData)`
 - Adott eszközhöz kötött font
 - OS erőforrásokhoz kötődik!
- GC-ben:
 - `setFont (Font)`
 - `getFont ()`
 - `getSystemFont ()` – alapértelmezett

Szöveg rajzolás

- `drawString(String, int, int, boolean)`
 - Kiírja a szöveget az adott koordinátáktól kezdődően
 - A boolean paraméter azt adja meg, hogy átlátszó-e a háttér
 - Ha nem, akkor a szöveg körüli téglalapot kitölti a háttérszínnel
- `drawString(String, int, int, int style)`
 - Kiírja a szöveget az adott koordinátáktól kezdődően
 - Style paraméter
 - `SWT.DRAW_DELIMITER` – a sortörés karakterek hatására új sort kezd
 - `SWT.DRAW_TAB` - a tabulátor karaktereket értelmezi
 - `SWT.DRAW_MNEMONIC` – az & után aláhúzott betűk (gyorsbillentyű)
 - `SWT.DRAW_TRANSPARENT` – átlátszó háttér

FontMetrics

- Szöveg méreteinek meghatározása
 - Szöveg elrendezés, tördelés
- Miért nem `Font.getHeight()` ?
 - Nem pixel, hanem pont
 - Néhány platform hazudik
- Helyette: `gc.getFontMetrics()`
 - `getHeight()`
 - magasság (pixel)
 - `getAscent()`
 - betűmagasság
 - `getDescend()`
 - „lelógás”
 - `getLeading()`



FontMetrics

- Miért nem a Font tartalmazza a méretadatokat?
 - Minden GC-specifikus
 - Mindig az aktuális font metrikáit kapjuk meg a gc-től.
- Hátrány:
 - Amíg nincs GC, nem tudjuk megmérni a magasságot...
 - Hasonló: Word újratördeli a szöveget nyomtatóváltáskor

Képek - ImageData

- Egy kép leírása
- `ImageData (String)`
- `ImageData (InputStream)`
 - Betöltés fájlból
 - Gif, Jpeg, PNG, BMP, ICO, TIFF
- `ImageData (int, int, int, PaletteData)`
 - Új képet készít
- Nincs OS erőforrás

Képek - ImageData

■ Rajzoló metódusok

- `setPixel(int x, int y, int v)` – pixel beállítása
- `setPixels(int, int, int putW, int[] v)` – pixelek beállítása
- `getPixels(int, int, int getW, int[] v)` – pixelek lekérdezése
- `getRGB(int)` – RGB a színből
- `getPixel( RGB)` – szín az RGB-ből
- `scaleTo(int, int)` – adott méretre skálázás

Képek - PaletteData

- A színinformátum leírása egy képre
 - A színinformátum lehet direkt, vagy palettás
- `PaletteData (RGB [ ] )`
- `getPixel (RGB)`
- Nincs OS erőforrás

Képek - Image

- `Image(Device, ImageData)`
 - Kép az `ImageData` objektumból
- `Image(Device, int, int)`
 - Üres, meg nem jelenő kép rajzolásához
- `Image(Device, String)`
 - Kép betöltése közvetlenül fájlból
- `Image(Device, Image, int)`
 - transzformáció végrehajtás
 - `SWT.DISABLE`, `SWT.GRAY`
- OS szintű!

Képek - kirajzolás

- A GC-ben
- `drawImage (Image, int, int)`
 - Adott helyre
- `drawImage (Image, int, int, int, int, int, int, int, int)`
 - A kép egy részének kirajzolása az adott helyre

Kurzorok

- `Cursor(Device, int)`
 - Előre definiált kurzorok
 - `CURSOR_WAIT`, `CURSOR_HAND`, `CURSOR_HELP`, ...
 - `CURSOR_SIZExx` -> átméretező alakú kurzor
 - Többféle van(égtájak szerint)
- `Cursor(Device, ImageData, int, int)`
 - Saját kurzor
 - A két int a hotspot helye (az aktív pont a kurzorban)

Nyomtatás

- Grafikus nyomtatás az operációs rendszerben telepített nyomtatókra
- A legtöbb platformon támogatott
 - Windows, Linux, OSX rendszereken működik
 - Platformonként külön tesztelendő!
- `PrinterData` – a nyomtató adatai
 - `Printer.getPrinterList()` – telepített nyomtatók
 - `Printer.getDefaultPrinterData()` – alapértelmezett
 - `PrintDialog.open()` – felhasználó választhat

Nyomtatás

- A PrintData-ból egy Printer objektumot készítünk.
- `startJob (String) ... endJob ()` –
nyomtatás
- Grafikus erőforrások, mint a képernyő esetén
- Rajzolás a Printer GC-jére, a szokásos módon
- `startPage () ... endPage ()` - lapozás

SWT - Összefoglalás

- Natív grafikus keretrendszer
 - Gyors
 - Egyszerű
 - Nincsenek benne nagyon extra dolgok...
- Lehetőséget ad
 - Dialógusok
 - Menük
 - Komplet alkalmazások elkészítésére
 - Nyomtatási feladatok megoldására