Programozási technológia

Párbeszédablakok, vezérlő elemek
Rajzolás

Dr. Szendrei Rudolf
ELTE Informatikai Kar
2020.

Párbeszédablakok, vezérlőelemek

- Cél: párbeszédablakok használata, és a vezérlőelemek megismerése
- Az elkészítendő programban színekkel dolgozunk:
  - Nyilvántartjuk a színt és annak nevét is.
  - A konstruktorral hozhatjuk létre az adott nevű színt.
  - Lekérdezhetjük a szín nevét, magát a színt, és megadjuk a kiírási formátumot is (`toString`), ami a szín neve lesz.
  - A `toString` művelet akkor lehet hasznos, ha listákat, vagy comboboxokat akarunk ilyen elemekkel feltölteni, és nem adunk meg megjelenítési előírást. Ilyenkor ezt a műveletet használja a rendszer a megjelenítendő információ meghatározásához.

Az alkalmazás kerete

- A párbeszédablakokat egy keretből fogjuk vezérelni, amit a `Dialogs` osztály valósít meg.
- Ebbe kerülnek be a különböző dialógusok, és a kezelésükhöz szükséges elemek, továbbá a menüpontok eseménykezelői.
- A tesztelést menü segítségével végezzük.
- A színeket a szövegszerkesztő mező háttér-, illetve betűszínében használjuk.
- Ebben a mezőben fogjuk naplózni a párbeszédablakok használatát.

OKCancelDialog

- A vezérlő elemek használatát külön-külön modális párbeszédablakokban vizsgáljuk meg.
- A legtöbb párbeszédablak tartalmazza az `OK` és a `Mégsem` nyomógombokat.
- Létrehozzuk ehhez a `JDialog` osztályból származtatott `OKCancelDialog` osztályt, amely létrehozza a gombokat és elvégzi az alapvető eseménykezelést.
- Az OK és Cancel gombhoz két absztrakt műveletet ad, amiket a származtatáskor kell definiálnunk :
  - `OK` gomb megnyomásakor szükséges ellenőrzések elvégzése és az eredményének megadása. (Helyes eredmény esetén a párbeszédablak bezáródik.)
  - `Cancel` gomb megnyomásánál kellő teendők végrehajtása.
- A bezárást kiváltó gombot egy konstanssal azonosítjuk.

OKCancelDialog

- Az osztály tartalmazza a publikus osztályszintű konstansokat, és a származtatott osztályokban használható panelt, és annak gombjait.
- Konstruktor:
  - Megadjuk a keretet, ahonnan az ablakot aktivizáltuk, és az ablak címét.
  - Létrehozzuk a párbeszédablakot, és beállítjuk az alapértelmezett bezárási műveletet.
  - Létrehozzuk a gombokat, és elhelyezzük őket egy panelen.
  - Az `OK` gomb lesz az alapértelmezett. (`Enter`-el aktiválható.)
  - A `Cancel` gombhoz hozzárendeljük az `Escape` billentyűt.
- Esemény kezelése:
  - Megfelelő absztrakt művelet meghívása; gombkód beállítása; ablak bezárása (kivéve, ha `OK` esetén hamis értéket kapunk).

`OKCancelDialog` használata

- Implementálni kell a két absztrakt műveletet.
- A párbeszédablakot fel kell tölteni a megfelelő vezérlő elemekkel.
- A vezérlő elemekhez hozzá kell venni a két nyomógombot, pontosabban el kell helyeznünk a gombokat tartalmazó panelt.

Sorszerkesztő – `EditDlg`

- Egysoros szöveg bevitelét támogatja az `EditDlg`.
- Ebben egy `JTextField` objektummal valósítjuk meg a szerkesztőt.
  - A szerkesztett szöveget a `getText`-el kérdezhetjük le.
  - Kezdeti szöveget a `setText`-el lehet beállítani, mi ezt nem használjuk, így üres szöveg lesz kezdetben a tartalom, a későbbiekben pedig a párbeszédablak megelőző bezárásakor tartalmazott szöveg.
  - A sorszerkesztőt szokásos módon használhatjuk szerkesztésre, értelmezettek a vágólapot használó műveletek: kivágás, másolás, beillesztés.

Combobox – `ComboDlg`

- A `ComboDlg` párbeszédablak egy szöveget adhatunk meg combobox segítségével.
- A combobox előre definiált értékei a színek nevei.
- A neveket most a szerkeszthetőség (kiválaszthatóság miatt) `String` objektumokként helyezzük el a comboboxban az `addItem` művelettel.
- A combobox alapértelmezésben nem engedi az elemek szerkesztését. Ezt állíthatjuk a `setEditable` művelettel.
- A szerkesztett elemet a sorszerkesztő `getItem` műveletével kaphatjuk meg.
- A sorszerkesztőt a `getEditor` függvény adja meg.

Csúszka – `SliderDlg`

- Készítsünk egy csúszkát tartalmazó párbeszédablakot, amely felett megjelenik az aktuális értéke.
- A párbeszédablak létrehozásakor legyen beállítható:
  - a csúszka intervalluma,
  - az aktuális érték,
  - és a fő értékek (a fő értékeket az értékkel és hosszabb vonallal jelöli a csúszka).
- Megoldás:
  - Az `OK` gomb mindig elfogadható (`processOK`).
  - `Cancel` esetén nincs teendőnk (`processCancel`).
  - Új műveletek:
    - Érték lekérdezése az ablak bezárása után (`getValue`).
    - A csúszka értékének beállítása megjelenítés előtt (`setValue`).

Csúszka – `SliderDlg`

- Az érték megjelenítése miatt figyelni kell, hogy a csúszka értéke megváltozik-e.
- Ehhez az osztálynak meg kell valósítania a `ChangeListener` interfészt, és implementálnia kell annak `stateChanged` műveletét.
  - a csúszka értékének lekérdezése,
  - a címke feliratának módosítása.
- Ahhoz, hogy a művelet meghívásra kerüljön, az osztályt fel kell vennünk a csúszka állapotváltozásaira figyelők közé (a csúszka `addChangeListener` műveletével).

Gombok – `ButtonDlg`

- Az ablak checkbox és rádiógomb típusú gombokat tartalmaz, amelyek segítségével meghatározhatjuk a szövegmező háttérszínét.
  - Ha a checkbox-ot bekapcsoljuk, a háttér a szövegszín inverze lesz, egyébként a háttér színét a kiválasztott rádiógomb határozza meg.
- A rádiógombok megjelenítése speciális:
  - a szöveg az adott szín neve,
  - az ikon a színnel kitöltött kör, amelyben fekete pont jelzi a kiválasztottságot (fekete szín esetén a pont fehér).

Gombok – `ButtonDlg`

- A rádiógombok megjelenítési módját külön meg kell adnunk a gomb létrehozásakor az ikon, illetve a kiválasztottsági ikon specifikálásával.
- Erre szolgál a `button` függvény.
  - Ebben létrehozunk két képet (`BufferedImage`), amelyekben megrajzoljuk a kívánt ábrát, és a képekből létrehozzuk az ikonokat.
  - A rajzoláshoz szükséges a kép grafikus eszközkapcsolat leírója, ami a `getGraphics` függvény ad meg.
  - A rajzolás ezután értelemszerű.

Gombok – `ButtonDlg`

- A rádiógombokat egy csoportba foglaljuk, és egy kerettel ellátott panelra helyezzük.
- A gombokat nyilvántartjuk (`colorButtons`), hogy le tudjuk majd kérdezni a kiválasztottságukat.
- Az `OK` gomb akkor fogadható el, ha van kiválasztott háttérszín.
  - Ez lehet a szövegszín inverze, vagy ha ez nincs kijelölve, akkor
  - egy színt kellett kiválasztani.
- `Cancel` gomb lenyomásakor nincs teendő.
- Szükséges az értékek lekérdezése
  - inverzmód,
  - szín
- Továbbá szükséges a checkbox értékének megadása (`setValue`)

Lista – `ListDlg`

- A `ListDlg` párbeszédablak egy listában tartalmazza a választható szövegszíneket.
- A listában nem csak a szín nevét, hanem egy színnel kitöltött téglalapot is megjelenítünk.
  - Egy speciális megjelenítési forma szükséges (`elemforma`), amit a `setCellRenderer` művelettel rendelhetünk a listához.
  - (Ha csak szöveget akarunk a listában, erre nincs szükség, az elemek `toString` művelete szerinti szöveg jelenik meg.)
- Ha azt akarjuk, hogy a lista elemén duplán kattintva az elemet válasszuk ki és zárjuk be az ablakot, akkor a listához fel kell vennünk egy egér esemény figyelőt, amelyben a dupla kattintást figyeljük.
  (Az esemény megegyezik az `OK` gomb megnyomásával.)
- Az `OK` gomb elfogadható, ha van kiválasztott elem.

Rajzolás

- A rajzolást `JPanel` segítségével valósítjuk meg.
- Ebből származtatunk egy osztályt, amelynek a `paintComponent` művelete adja meg a megjelenítést.
- A `paintComponent` művelet minden esetben végrehajtódik, amikor a komponens területét újra kell rajzolni (átméretezés stb.).
- A végrehajtás kezdeményezhető a `repaint` művelettel.
- A `paintComponent` paramétere a grafikus eszközkapcsolat (`Graphics`), amire rajzolni lehet.

Rajzolás

- A `Graphics` eszközkapcsolaton adott rajzoló függvények eredményei nem túlságosan szépek (például hézag van a vonal és a kitöltés között).
- Jobb eredményhez jutunk, ha a később bevezetett `Graphics2D` eszközkapcsolatot használjuk.
- A `Graphics2D` bővebb funkcionalitás ad, és a műveletek eredménye is szebb.
- A `Graphics2D` kapcsolatot egyszerű átminősítéssel kaphatjuk meg az eredeti `Graphics` objektumból.
- Ezen ugyanúgy értelmezettek a korábbi műveletek.
- Az alakzatok rajzolásához, illetve fejlettebb műveletekhez a `java.awt.geom` csomag elemeire van szükség.

Rajzolás

- A `Graphics2D` főbb műveletei
  - `drawArc` `fillArc`
  - `drawOval` `fillOval`
  - `drawPolygon` `fillPolygon`
  - `drawRect` `fillRect`
  - `drawRoundRect` `fillRoundRect`
  - `drawLine`
  - `drawString`
  - `drawImage`
  - `getColor` / `setColor`
  - `getFont` / `setFont`

Feladat

- Készítsük el a Sokoban játékot
  - A játék lényege, hogy egy zárt szobában ládákat kell eltolni a megfelelő helyekre.
  - A játék nehézsége, hogy a ládákat csak tolni lehet, és csak akkor, ha van a láda mögött üres hely.
- További követelmények
  - Alkalmazzunk Model-View architektúrát
  - A játékpályákat fájlból olvassuk be
  - A pályát képekből rajzoljuk ki egy `JPanel`-re
  - Az irányítást a billentyűzet kurzor gombjaival végezhessük