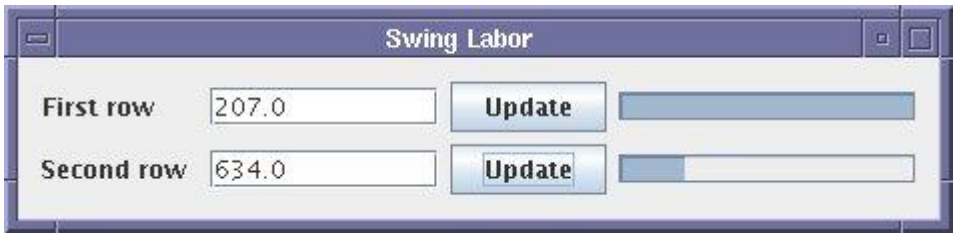
8. Java SWING MVC labor, iMSC

Készítette: Goldschmidt Balázs, BME IIT, 2020.

1. Bevezető

A feladat egy egyszerű Swing alkalmazás elkészítése. Az alkalmazás három osztályból áll: `Labor`, `Updater`, `Calculator`. Az első kettőhöz a forrás szkeletonja áll rendelkezésre, míg az utolsónak csak a bytecode alakja van meg (jar fájlban). A szkeletonokat csak a TODO-val jelölt megjegyzések helyén szabad kibővíteni, máshol módosítani őket, elvenni belőlük vagy hozzájuk adni nem kell.

2. Az alkalmazás



Az alkalmazás felületén két-két label, textfield, gomb és progressbar található. Ha a gombok egyikét megnyomjuk, akkor a háttérben elindul egy külön szál (`Updater` osztály implementálja, a ***SwingWorker*** leszármazottja). A szál futása során a megnyomott gomb melletti progressbar folyamatosan mutatja a futó szál aktuális állapotát. Mikor a szál véget ér, az eredmény a gomb melletti textfield-be kerül.

A gombok bármikor megnyomhatók. Ha egy korábban elindított szál már fut, akkor cancelláljuk, és új szálat indítunk. A két gomb egymástól teljesen függetlenül kell működjön.

3. Az osztályok leírása

labor.Labor

Az alkalmazás fő osztálya. Feladata, hogy felépítse a GUI-t, és kezelje a gomboktól és az `Updater`-ektől érkező eseményeket (***ActionEvent*** és ***PropertyChangeEvent***). Ehhez implementálja az ***ActionListener*** és a ***PropertyChangeListener*** interfészeket.

Az osztály kódjából hiányoznak a következő részek (a forrásban jelölve):

- `actionPerformed()` metódus. Feladata a gombnyomások kezelése. Ha a gombhoz tartozó szál már fut, akkor cancellálni kell és új Updater-t kell indítani.
- `propertyChange()` metódus. Feladata az Updater szálaktól jövő események (progress érték változása) feldolgozása. A megfelelő progressbar-t ennek megfelelően kell frissíteni. Ha a szál már megállt, akkor frissítésen kívül még a textfield-be is be kell írni a számítás végeredményét.

- *gombok létrehozása, beállítása*. A feladat a két gomb létrehozása, az *actionCommand* beállítása, az *actionlistener* regisztrálása.
- *layout megadása*. Az alkalmazásban **GroupLayout**-ot kell használni. Feladat, hogy az elemeket a fenti ábrán látható módon helyezzük el.

labor.Updater

Az alkalmazás háttérben futó szálainak osztálya, a **SwingWorker** leszármazottja . Ennek példányai futnak a gombnyomás hatására. A számításhoz mindig új `Calculator` objektumot kell használniuk. Implementálandó a **doInBackground()** metódus.

A metódus először létrehoz egy `Calculator` objektumot, majd ciklikusan kéri az objektumtól, hogy dolgozzon. Ha eközben cancellálják, akkor megszakad a ciklus. Egyébként a munka végeztével (a `Calculator` `work()` metódusa 100-zal tér vissza) visszaadja a `Calculator` által számított étéket (`get()` metódussal kérdezhető le).

calc.Calculator

Az értéket több lépésben számító objektum osztálya.

Metódusai:

- `public Calculator()`

  Publikus konstruktor.

- `public int work() throws java.lang.InterruptedException`

  A számítás egy részét végzi el. A visszatérési érték az elvégzett munka százalékban (0-100). Ha megszakad, kivételt dob.

- `public double get()`

  Visszaadja a kiszámított értéket.

4. Letölthető anyag

Letölthető a *swing.zip* zip-fájl. Tartalma:

- `labor/Labor.java:` a `Labor` osztály kitöltendő sablonja.
- `labor/Updater.java:` az `Updater` osztály kitöltendő sablonja.
- `calc.jar:`a `Calculator` osztály bytecode-ja *jar*-ban.

A szkeletonokat csak a TODO-val jelölt megjegyzések helyén szabad kibővíteni, máshol módosítani őket, elvenni belőlük vagy hozzájuk adni nem kell.