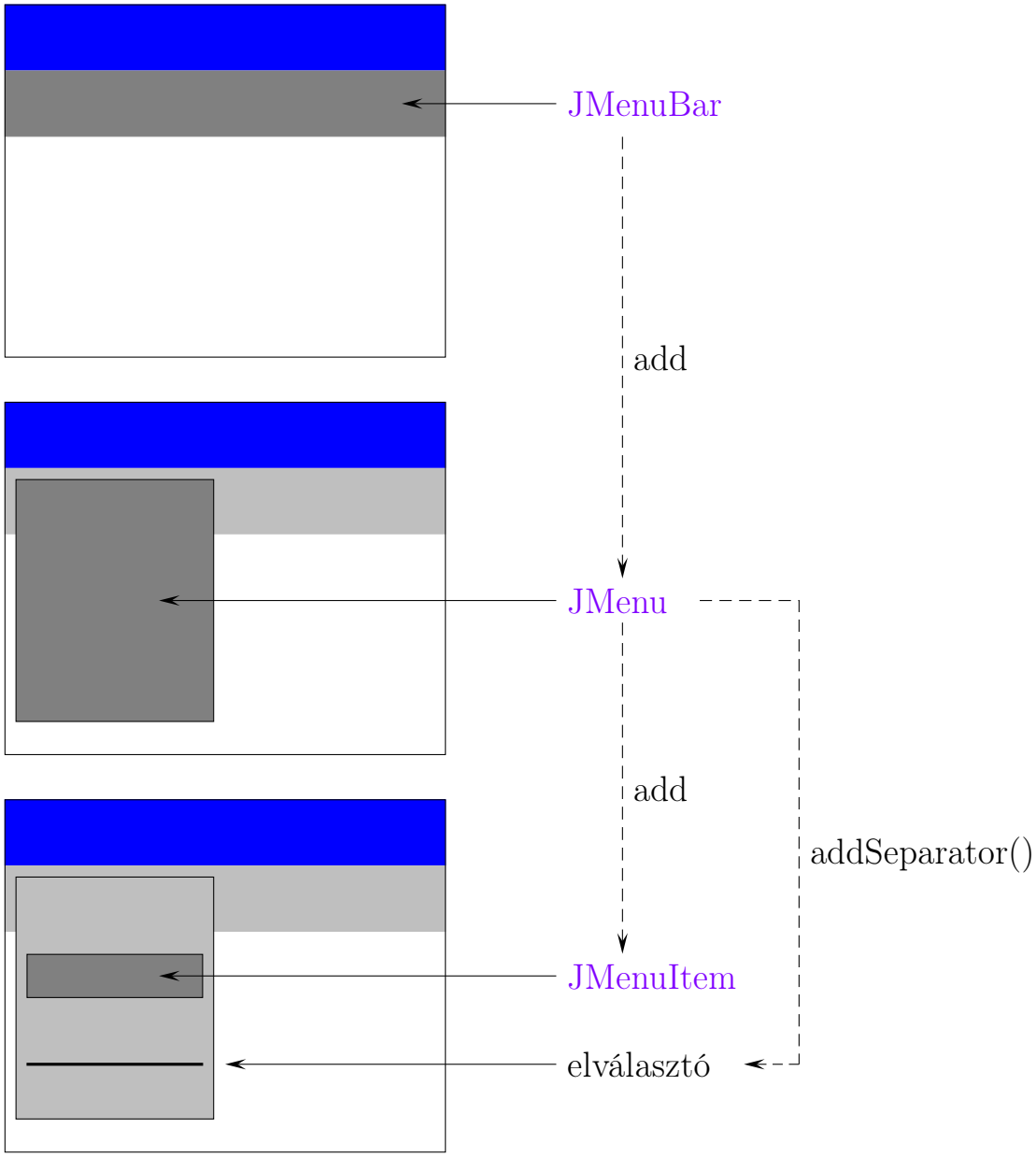
1. Menü

Egészítsük ki a kattintás számláló programot egy menüvel, amelyben nullázhatjuk a számláló értékét, illetve kiléphetünk a programból! A `Gomb4` megoldásból induljunk ki!

1.1. Menüsor felépítése



`JMenu`:

- konstruktor: paraméter a név
- `setMnemonic`: paraméter a kiválasztó billentyű

`JMenuItem`:

- konstruktor: paraméter `AbstractAction`-ből származtatott objektum (vagy név, ikon; de ekkor eseménykezelőt hozzá kell venni)
- `setMnemonic`: paraméter a menün belüli kiválasztó billenytű
- `setAccelerator`: a gyorsítóbillentyű kódja

Java program: `Menu1` projekt

Változtassuk meg a programot, hogy nullázni csak akkor lehessen, ha a kattintások száma nem nulla!

Ehhez az `AbstractAction` osztály `setEnabled` műveletét használhatjuk, amelynek paramétere egy logikai érték, ami a parancs végrehajthatóságát vezérli.

Kezdetben ezt hamisra kell állítani például a `Gomb` konstruktorban, kattintás esetén engedélyezni kell (`kattintásakció` eseménykezelője), nullázáskor pedig letiltani (`nullázás` eseménykezelője).

Java program: `Menu2` projekt

2. Rajzolás

Készítsünk egy programot, amelyik ellipsziseket (tojásokat) jelenít meg! Növelni, illetve csökkenteni lehet a legnagyobb tojás méretét egy beállítható értékkel. A többi tojás szélessége harmincasával csökken, amíg pozitív a méret. A legnagyobb tojást kell középen felülre elhelyezni, a többi tojást nagyság szerint csökkenően lefelé úgy, hogy illeszkedjenek! Egy tojás magassága a szélességének kétharmada.

- Az új mérték megadásához a már megismert `JOptionPane.showInputDialog` műveletet használjuk.
- A rajzolást `JPanel` segítségével valósítjuk meg. Ebből származtatunk egy osztályt, amelynek `paintComponent` művelete adja meg a megjelenítést.
- A `paintComponent` művelet minden esetben végrehajtódik, amikor a komponens területét újra kell rajzolni (átméretezés, ...). A végrehajtás kezdeményezhető a `repaint` művelettel.
- A `paintComponent` paramétere a grafikus eszközkapcsolat (`Graphics`), amire rajzolni lehet.
- Rajzolási lehetőségek `Graphics` objektumra: program, help.

Java program: `Tojások1` projekt

2.1. Fejlettebb grafika, eszköztár

A kirajzolt ellipszisek nem túlságosan szépek (például hézag van a vonal és a kitöltés között). Jobb eredményhez jutunk, ha a később bevezetett `Graphics2D` eszközkapcsolatot használjuk. Ehhez szebb eredményt létrehozó és sokkal bővebb funkcionalitást biztosító műveletek tartoznak.

A másik változtatás az eszköztár bevezetése lesz. Két gombot helyezünk el az eszköztáron a méret növelés, illetve csökkentés céljára.

2.1.1. `Graphics2D`

Először a rajzolással foglalkozunk. A `Graphics2D` kapcsolatot egyszerű átminősítéssel kaphatjuk meg az eredeti `Graphics` objektumból. Ezen értelemszerűen értelmezettek a korábbi műveletek. Az alakzatok rajzolásához, illetve fejlettebb műveletekhez a `java.awt.geom` csomag elemeire van szükség.

2.1.2. Eszköztár

A `JToolBar` osztály szolgál az eszközsorok megvalósítására. Ehhez az `add` művelettel vehetünk fel akciókat, a visszaadott érték a létrejött gomb.

A menüben és az eszköztáron ugyanazt az akciót kellene elhelyeznünk.

Ha az akció objektum rendelkezik névvel és ikonnal, akkor az eszköztáron az ikon, a emüben az ikon és a név jeleneik meg.

Ha az ikon zavaró a menüben, akkor

1. a létrehozott menüpontban az ikont `null` értékre állíthatjuk, vagy
2. az akciót név és ikon nélkül hozzuk létre, és menüpont esetén a szöveget, eszköztár gomb esetén az ikont állítjuk be.

A programban a második megoldást választottuk (tekintettel a későbbiekre).

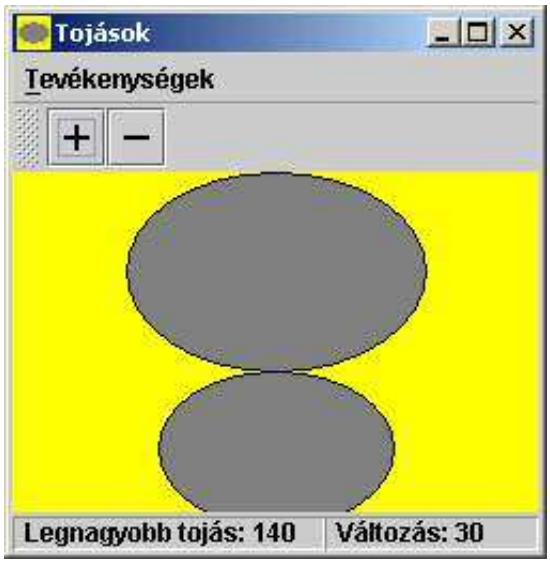
Java program: `Tojások2` projekt

2.2. Státuszsor

Az alkalmazások jelentős részében az ablak alsó része a program állapotával kapcsolatos információkat tartalmaz. Ezt nevezzük státuszsornak.

Az eddigi rajzoló programot kiegészítjük egy egyszerű státuszsorral. Feltesszük, hogy ebben csak szöveges információk jelenhetnek meg. A sor egyes mezőinek szélessége állítható, az első dinamikusan változhat az ablakkal együtt.

Elkészítünk a fentieknek megfelelő általános osztályt a státuszsor megvalósítására (`StatusBar`). Ezután ennek csak egy példányát kell a keret alsó részére elhelyeznünk, és azt megfelelően konfigurálni. Esetünkben két mező szerepel benne: a legnagyobb tojás aktuális mérete, és a változtatás aktuális mértéke.


Tojások
Tevékenységek
+
−
Legnagyobb tojás: 140
Változás: 30

A `StatusBar` osztályt a `JPanel` osztályból származtathatjuk, a mezői pedig a `JLabel` osztály példányai lesznek. Ezeket egy `GridBagLayout` elrendezésben jelenítjük meg, ahol az első elem szélessége változhat. A mezőket megfelelő kerettel kell ellátni.

(Lásd: projekt.)

- A státuszsorban megjelenítendő információ miatt a `RajzPanel` osztályt egy új művelettel kell bővítenünk (`maxMéret`), ami megadja a legnagyobb tojás méretét.
- A keret (`Tojások`) osztályban fel kell vennünk a státuszsort új adattagként, az eseményekhez rendelt akciók eljárásait ki kell bővíteni a státuszsor módosításával, a konstruktorban létre kell hoznunk és megfelelően beállítanunk a státuszsort, amit az adatterület aljára kell helyeznünk.

Java program: `Tojások3` projekt

2.3. Görgetés

A tojások nem férnek el a látható ablakban. A megoldás a rajzterület görgethetővé tétele.

1. A görgetés érdekében a rajzterületet egy `JScrollPane` objektumba kell ágyazni, és azt venni a kerethez. Ennek megfelelően a `Tojások` osztály konstruktora módosul.
2. A `RajzPanel` osztály `méretváltás` műveletét módosítani kell. Az újrarajzolás mellett tudatni kell a környezettel, hogy a szükséges rajzterület mérete megváltozott. Ennek alapján jelennek meg a görgetősávok. Erre szolgál az `updateUI` művelet.
3. A megfelelő görgetéshez a rajzterület alapértelmezett méretét is meg kell adni a `setPreferredSize` művelettel. (Konstruktor, méret váltás.) A magasság kiszámításához bevezetünk egy függvényt.

Java program: `Tojások4` projekt

2.4. Rajzolási paraméterek

A menüben és az eszköztáron elhelyezhetünk választható elemeket (rádiógombok) és kétállapotú elemeket (checkbox). Ezek használatát szemléltetjük a tojások rajzolási paramétereinek megadásával.

- Lehessen szabályozni, hogy a tojások kitöltöttek legyenek-e.
- A kitöltési szín is legyen választható előre megadott lehetőségek (szürke, fehér, zöld) közül.

2.4.1. `RajzPanel`

- Az osztályt ki kell egészíteni egy kitöltési színt megadó attribútummal (`töltés`). Ennek értéke legyen `null`, ha nincs kitöltés, különben a kitöltési szín.
- Be kell vezetni egy új műveletet, amellyel a kitöltési szín állítható (`tölt`).
- Egy tojás rajzolásakor (`rajzol`) meg kell vizsgálni, hogy kell-e kitöltés, és ha igen a megfelelő színnel kitölteni az alakzatot.

2.4.2. Tojások

Kitöltöttség:

- Szükségünk lesz a kitöltöttség logikai értékre
- Fel kell venni az eseménykezelő akciót: `kitöltés`.
- Kell egy új menüpont (`Attribútumok`), amelybe be kell tenni a megfelelő menüpontot.
- Az eszköztárhoz is fel kell venni az új elemet.
- Az eseménykezelőben kell gondoskodni arról, hogy a menü és az eszköztár elemek szinkronban legyenek.

Színek:

- Vegyünk fel két tömböt a színeknek és neveiknek. (Ezekből vesszük az adatokat.)
- Kell egy érték, amiben tárolhatjuk az utoljára kiválasztott szín indexét.
- Két tömbben tároljuk a menüpontokat, illetve eszköztár gombokat.
- Egy eseménykezelőt írunk az összes esethez. Ebben biztosítjuk a szín állítását, a menü és eszköztár szinkronizációját.
- A konstruktorban hozzuk létre a felületi elemeket, amelyket `ButtonGroup` objektumba kell ágyaznunk. (Ez biztosítja a csoport egyesített kezelését, azaz, ha az egyik változik, akkor a másik is.) Az elemeket a menübe, illetve az eszköztárba is elhelyezzük.
- Az eszköztár gombjai esetén egy-egy függvénnyel adjuk meg az ikont, illetve a kiválasztottsági ikont. (Nem lehet fájlból betölteni a konzisztencia miatt.)

Java program: `Tojások5` projekt

2.5. Menü és eszköztár elemeinek összevont kezelése

Az előző megoldásnak két „szépséghibája” van:

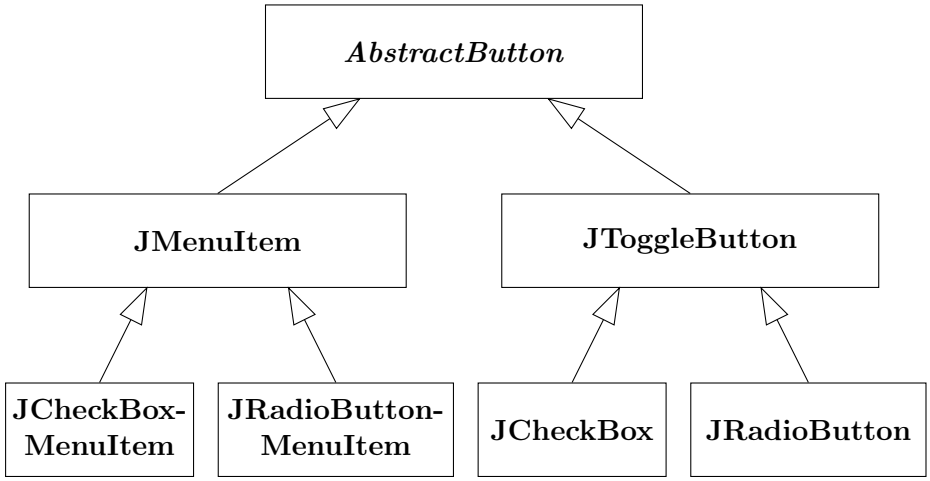
1. A színek kezelésekor külön szerepelnek a színek és a neveik, ami inkonzisztenciát eredményezhet.
2. Nehézkes a menüben és az eszköztáron szereplő kétállapotú (checkbox, rádiógomb) elemek kezelése.

Mindkét problémát egy-egy osztály bevezetésével oldjuk meg.

2.5.1. `Szín`

- Az osztály attribútumaiban tároljuk a színt és a nevét. (Lényegében egy rekord.)
- A konstruktorral lehet létrehozni egy ilyen párt.
- Egy-egy művelettel lehet lekérdezni az attribútumok értékét.
- Felüldefiniáljuk a `toString` műveletet, amelyben a nevet adjuk meg.

- Az attribútumok adják meg a menüpontot, az eszköztár elemet (ezek értéke `null`, ha nincs ilyen), illetve az eseménykezelést meghatározó akciót.



- A konstruktor paramétere az akció, a menüpontot és az eszköztár elemet „üresre” állítjuk.
- Az elemeket a `setMenuItem`, illetve a `setToolItem` műveletekkel rendelhetjük az objektumhoz.
- Az osztély a `ActionListener` interfész által megadott `actionPerformed` műveletben biztosítja a menü és eszköztár azonos állapotát, és meghívja a beállított akció eseménykezelőjét.
- A `setSelected` művelettel lehet állítani az elem(ek) kiválasztottságát.
- Az `isSeleceted` függvénnyel lehet lekérdezni a kiválasztottságot.
- Az `isSource` függvénnyel lekérdezhetjük, hogy az elem-e az esemény forrása, azaz a menüpont vagy az eszköztár elem-e az.

A keret ezek felhasználásával egyszerűsödik.

- Egyetlen tömbbe kerülnek a színek.
- A kitöltöttség és kitöltési szín vezérlésére `MenuToolToggleItem` típusú objektumot, illetve ilyen objektumok tömbjét használjuk.
- Az eseménykezelőkben csak a funkciókra kell figyelnünk, a konzisztencia fenntartására nem.
- A menü, illetve az eszköztár létrehozásakor kell a menüpontokat, illetve az eszköztár gombokat az objektumokhoz hozzárendelni.

Java program: `Tojások6` projekt

Ez a megoldás abból a szempontból is jó, hogy új kitöltési szín bevezetéséhez csak a kitöltési színek tömböt kell egy új `Szín` objektummal bővítenünk.

Ez azért lehetséges, mert a felületi elemeket a tömb alapján hoztuk létre, az eseménykezelés, pedig egységes minden elemre nézve (ciklus).

3. Felhasználói felület kialakítása

Az eddigiekből is látszik, hogy menü és eszköztár esetén hasonló módon kell eljárnunk:

- Meg kell adnunk az elem kinézetét, jellemzőit (szöveg, ikon, mnemonik, gyorsítóbillentyű, elhelyezkedés).
- Meg kell határoznunk az elemhez tartozó viselkedést, azaz az eseménykezelést (`AbstractAction`, illetve `Action`).

Egy elem kinézete, elhelyezkedése változhat (például nyelvet váltunk, vagy valamilyen oknál fogva ikont cserélünk, esetleg másik menübe helyezzük át) anélkül, hogy a viselkedése módosulna. Ezért jó lenne a kettőt szétválasztani egymástól.

A tervezőben valami hasonló történik, de ennél is tovább lehetne menni. A kinézet változása miatt ne kelljen újra fordítani a programot. Ez elérhető, ha a kinézetet ehy külön fájlban írjuk le, és az itt leírt elemekhez a fájl feldolgozása során rendeljük hozzá az eseménykezelőket.

Erre később térünk vissza.