# Titkosírás

## Ismétlés: Caesar kódolás

A Caesar kódolás lényege, hogy minden karaktert az utána következő valahányadik (például harmadik) karakterrel helyettesítünk.

Ha az eltolás túlmutat az ábécé végén, az ábécé elejéről vesszük a karaktereket. Ehhez a kapott kódból le kell vonni az ABC hosszát (26).

Például a +3-as eltolás:

ABCDEFGHIJKLMNOPQRSTUVWXYZ  
ABCDEFGHIJKLMNOPQRSTUVWXYZABC

A kódolásnál megkeressük a karaktert a felső sorban, és helyettesítjük az alatta lévővel.

Példa: ABLAK 🡪 DEODN

Mi lesz a kódja a POGANY szónak? Írd ide:

Az üzenet dekódolásakor ellenkező irányba kell eltolni a karaktereket ugyanannyival (vagyis ‑3-as eltolás). Ha az ABC elé kerülünk, a kódhoz hozzá kell adni az ABC hosszát.

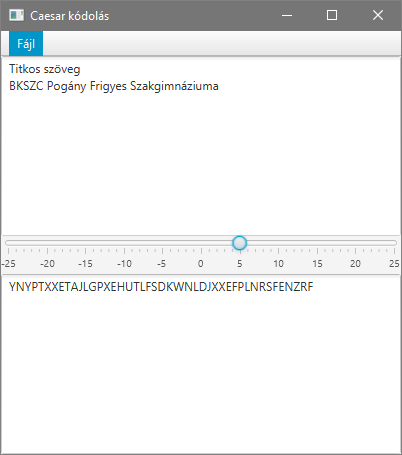
ABCDEFGHIJKLMNOPQRSTUVWXYZ  
XYZABCDEFGHIJKLMNOPQRSTUVWXYZ

Mit jelent a kódolt VLPRQBL szó? Írd ide:

A kódolás megfejtését nehezíti, hogy nem csak 3-mal lehet eltolni a karaktereket, hanem tetszőleges 1 és 25 közötti számmal. Így a megfejtőnek több lehetőséget kell végigpróbálnia.

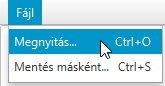
## Feladat

Ebben a leckében egy olyan programot készítünk, amellyel könnyen lehet változtatni az eltolás mértékén, és így gyorsan végig lehet próbálni a lehetőségeket:

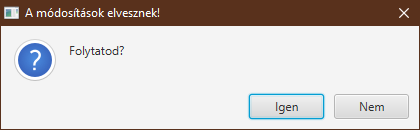


A fenti szövegdobozba (forrás) gépelve azonnal megjelenik a kódolt szöveg az alsóban (cél). Ha a csúszkán állítjuk az eltolás mértékét, akkor automatikusan újra elvégzi a kódolást az új eltolással.

A menüből kiválaszthatjuk a felső szöveg megnyitását vagy mentését:



Ha módosítottuk, és utána nem mentettük a forrást, megnyitáskor és kilépéskor figyelmeztet:



## A Caesar osztály

Nyisd meg a kapott *titkosiras* projektet! Nézd meg a Caesar osztály kódját!

A kódban újdonság a StringBuilder osztály használata. Ezt az osztályt akkor alkalmazzuk a String osztály helyett, ha sok elemet kell hozzáfűzni egy szöveghez.

Ez a String osztály esetén nem hatékony, mert minden hozzáfűzés után egy új String jön létre. Ezzel szemben a StringBuilder osztály módosítható, így nem kell minden hozzáfűzés után új példányt létrehozni belőle.

A StringBuilder osztálynál a létrehozáskor megadhatunk egy méretet, amely később még változhat (16. sor). Ezután új elemeket adhatunk hozzá az append metódussal (38. sor), a toString metódussal pedig normál String-gé alakíthatjuk (40. sor). Így a hozzáfűzések sokkal hatékonyabbak (gyorsabbak és kevesebb memóriát igényelnek).

Gondold végig a kodol metódus működését! Hogyan tudjuk meghívni egy másik osztályból?

## A Panel osztály

A programban szükségünk lesz kétféle párbeszédpanelre:

* Az egyik egy hibaüzenetet jelenít meg OK gombbal.
* A másik rákérdez valamire, amire az Igen vagy a Nem gombbal válaszolhatunk.

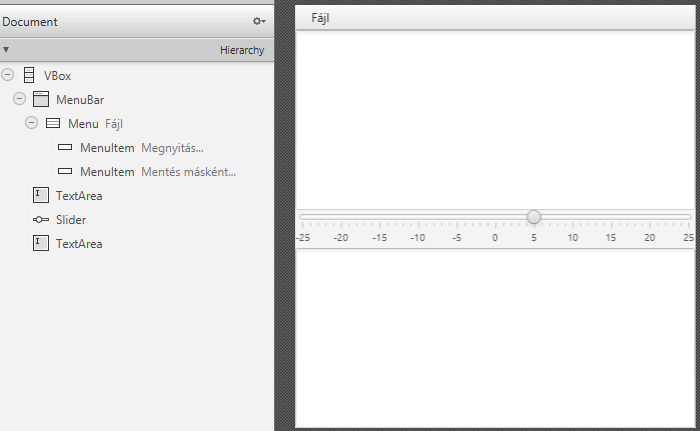
Ezeket a párbeszédpaneleket előre elkészítettük a Panel osztályban, amelyet megkaptál a forrásokban.

Figyeld meg, hogy a Panel osztályt külön panel nevű csomagba tettük!

Nézd végig a forráskódot, és gondold végig a metódusok működését! Hogyan lehet meghívni a metódusokat?

## Ablak elkészítése

Készítsd el a program ablakát a Scene Builderben az alábbi minta alapján:



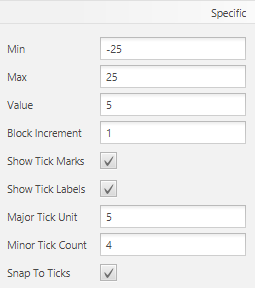
A VBox preferált szélessége 400 képpont legyen! Vízszintesen töltse ki a rendelkezésre álló helyet! (Layout, Fit Width)

A menüpontok az ábrának megfelelőek legyenek! A Fájl menüben a Megnyitás… választásakor a megnyitas, a Mentés másként… választásakor a mentes metódus fusson le!

Mindkét szövegterületre kapcsold be a sortörést (Wrap Text)! Az alsó ne legyen szerkeszthető!

A felső TextArea azonosítója: txaNyilt, az alsóé pedig txaTitkos legyen!

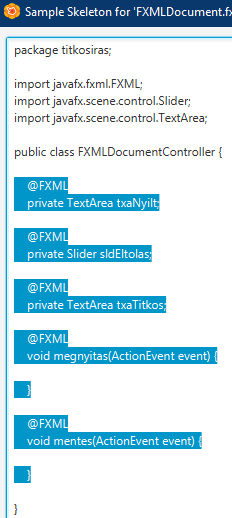
A Slider beállításai:



A Slider azonosítója sldEltolas legyen!

Add meg a controller osztály nevét a bal alsó sarokban (Document, Controller)!

Ezután nyisd meg a View, Show Sample Controller Skeleton paranccsal a controllerbe másolandó kódot, és másold a vágólapra a szükséges deklarációkat!



Mentsd a jelenetet, majd válts vissza a NetBeans-re!

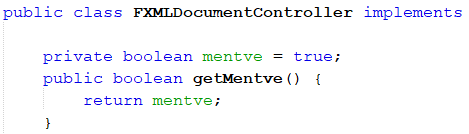
Nézd át a generált FXML dokumentumot!

## A Controller osztály

Másold be a kódot a vágólapról a controller osztályba! Töröld az eseménykezelők ActionEvent paramétereit! Javítsd az importálásokat!

Futtasd a programot!

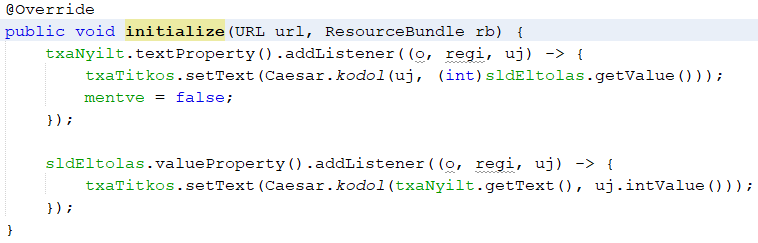
A programban mindig tudnunk kell, hogy el vannak-e mentve a forrás változásai. Ezért létrehozunk egy mentve nevű, logikai típusú változót, amely induláskor igaz lesz. Ha változik a forrás, hamisra fogjuk állítani. Ezt a Titkosiras osztályból is látnunk kell majd, ezért készítünk hozzá egy getter metódust is.



## Kódolás

Azt szeretnénk, hogy beírás közben, vagy az eltolás módosításakor egyből megtörténjen a kódolás. Ezért a szöveg és a csúszka változásához kell eseménykezelőket készítenünk.

Írd a következőket a controller osztályba:



Az első eseménykezelő a szöveg módosításakor fut le. Meghívja a Caesar osztály kodol metódusát, és a mentést hamisra álítja. A csúszka getValue metódusa double típusú értéket ad vissza, amit egész számmá kell konvertálni.

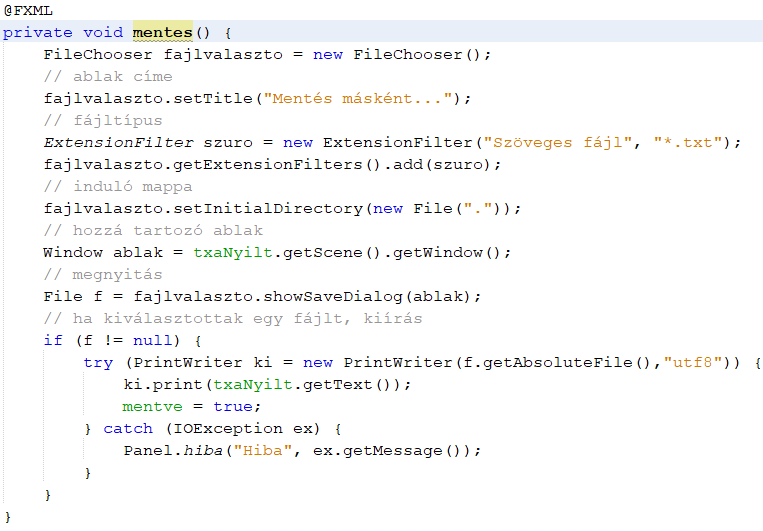
A második eseménykezelő a csúszka módosításakor fut le, és ez is a kodol metódust hívja meg. Most azonban az uj paraméter intValue metódusával határozzuk meg az eltolást, és a mentést nem kell módosítani.

Futtasd a programot, és próbáld ki néhány szöveggel! Figyeld meg, hogy a kódolt szöveg egyből megjelenik az alsó szövegdobozban! Változtasd az eltolást is!

## Mentés

A mentésnél az operációs rendszer mentés ablakát fogjuk használni.

Készítsd el a mentes metódust controller osztályba:



A mentés ablakot a FileChooser osztály segítségével használhatjuk. Először létrehozunk belőle egy új példányt.

Utána megadjuk az ablak címét a setTitle metódus segítségével.

A fájltípusnál azt adjuk meg, hogy milyen alapértelmezett kiterjesztése legyen a fájlnak, illetve milyen fájlokat mutasson az ablakban:



A mentés helyét az induló mappából indulva választhatjuk ki. A "." a projekt, vagy a JAR fájl mappáját jelenti.

A mentés párbeszédpanelt a showSaveDialog metódussal jelenítjük meg. Ha paraméterként megadunk egy ablakot, akkor az ablak elé kerül, ha null‑t adunk meg paraméterként, akkor pedig a képernyő közepére.

A fájlválasztás eredménye a File típusú f változóba kerül. Ha az f értéke null, akkor a felhasználó nem választott, hanem megnyomta a Mégse gombot vagy az Esc billentyűt.

Ha a felhasználó megadta a fájlt, akkor ennek teljes nevét és elérési útját a getAbsoluteFile metódussal kaphatjuk meg.

A fájlba egy PrintWriter objektummal írunk, majd a mentést igazra állítjuk.

Hiba esetén a Panel osztály statikus hiba metódusát hívjuk meg a megfelelő üzenetekkel. (Ezt nem fogjuk tudni kipróbálni, mert az operációs rendszer kezeli helyettünk a hibákat.)

Akkor is megengedjük a mentést, ha a szöveg már el van mentve. Vajon miért?

Próbáld ki a mentés működését! Ellenőrizd a mentett fájlokat!

## Megnyitás

Másik fájl megnyitása előtt meg kell vizsgálni, hogy nem vesznek-e el a módosítások a forrásszövegben. Ha elvesznének, rá kell kérdezni a folytatásra.

A megnyitás metódusban is a FileChooser osztályt használjuk, csak most a showOpenDialog metódussal nyitjuk meg (zöld kiemelés).

Készítsd el a megnyitas metódust:



Figyeld meg, hogyan fűzzük hozzá a beolvasott sorokat a szövegterülethez!

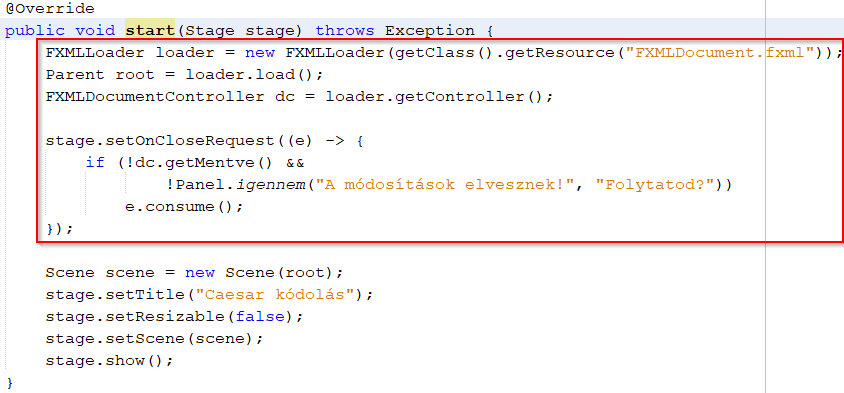
Próbáld ki a programot!

## Kilépés

Az ablak bezárásakor megvizsgáljuk, hogy el van-e mentve a szöveg, és ha nem, akkor rákérdezünk a folytatásra.

Mivel az eseménykezelőt a Titkosiras osztályban hozzuk létre, és a getMentve metódus a controller osztályban van, szükség lesz egy változóra, amely a controller osztályra mutat.

Ehhez sajnos át kell írnod a Titkosiras osztályban a start metódus elejét, és utána tudod elkészíteni az eseménykezelőt:



Az első két sor ugyanazt csinálja, mint az előző változat első sora: létrehoz egy FXMLLoader objektumot, és beolvassa belőle az elemeket.

A harmadik sorban létrehozott dc változóval hivatkozhatunk a controller osztályra.

Az ablak bezárás gombjának megnyomása egy CloseRequest eseményt generál. Ehhez adunk meg eseménykezelőt a bekeretezett rész alsó soraiban.

Az esemény consume metódusa megakadályozza az esemény továbbadását, vagyis a kilépést a programból.

Próbálj meg mentés nélkül kilépni a programból! Mi történik?

Állítsd be a címet és az átméretezhetőség kikapcsolását is!

## Kérdések

1. Mikor használjuk a StringBuilder osztályt?
2. Hogyan nyithatunk meg vagy menthetünk egy fájlt?
3. Milyen esemény következik be egy ablak bezárás gombjának megnyomásakor?