SZÉCHENYI ISTVÁN EGYETEM

GÉPÉSZMÉRNÖKI, INFORMATIKAI ÉS VILLAMOSMÉRNÖKI KAR

**MATEMATIKA ÉS SZÁMÍTÁSTUDOMÁNY TANSZÉK**

A képen tervezés látható

Automatikusan generált leírás

**FÉLÉVES FELADAT**

**Karóra**

**Hunyadvári Ádám Radó, Kazda Adrián Zsolt**

Programtervező informatikus BSc

**Győr, 2024**

Tartalom

[**1.** **Bevezetés** 2](#_Toc170222273)

[**1.1** **Korai tervek, ötletek** 2](#_Toc170222274)

[**1.2** **Az UI szerkezete** 3](#_Toc170222275)

[**1.3** **Egy objektum felépítése** 3](#_Toc170222276)

[**1.4** **JavaFX és FXML** 4](#_Toc170222277)

[**2.** **Tervezési dokumentáció** 5](#_Toc170222278)

[**2.1** **Fejlesztési környezet** 5](#_Toc170222279)

[**2.2** **Osztályok leírása** 5](#_Toc170222280)

[**2.3** **Main osztály bemutatása** 5](#_Toc170222281)

[**2.4** **Controller osztály bemutatása** 5](#_Toc170222282)

[**2.4.1 Adattagok** 6](#_Toc170222283)

[**2.4.2 Metódusok** 7](#_Toc170222284)

[**2.5** **Eseményfigyelők – az initialize metódus részletei** 7](#_Toc170222285)

[**3.** **Felhasználói dokumentáció** 8](#_Toc170222286)

[**3.1** **Rendszerkövetelmények** 8](#_Toc170222287)

[**3.2** **Programrendszer telepítése** 8](#_Toc170222288)

[**3.3** **A programrendszer működése** 8](#_Toc170222289)

[**3.4** **Véletlenszerű generálás** 8](#_Toc170222290)

[**3.5** **Fájlkezelés** 8](#_Toc170222291)

[**3.6** **Nézetek** 9](#_Toc170222292)

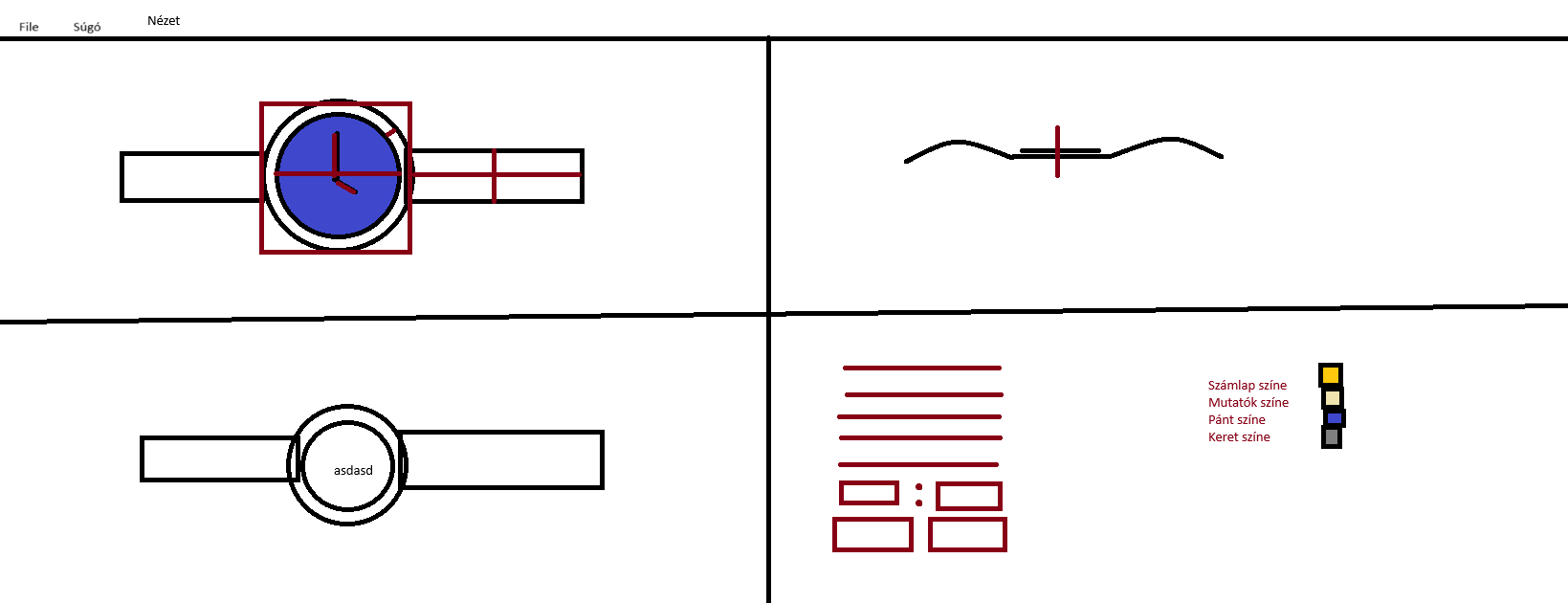
[**3.7** **Egyéb gyorsgombok** 9](#_Toc170222293)

[**4.** **Irodalomjegyzék** 10](#_Toc170222294)

# **Bevezetés**

Feladatunk egy grafikus alkalmazás megírása volt, ami megjelenít egy hagyományos karórát különböző paraméterek (számlap színe, szíj mérete, mutatók állása stb.) mellett. A kritériumok között szerepelt a példányok véletlenszerű generálása, személyre szabása, valamint fájlba mentése és betöltése. A program elkészítése alatt igyekeztünk a feltételek teljeskörű kielégítésére. Célunk volt egy modern, felhasználóbarát, letisztult kezelőfelület létrehozása, valamint egy új keretrendszer megtanulása.

* 1. **Korai tervek, ötletek**

A programunk fejlesztését gondos tervezgetés, egyeztetés előzte meg. Közösen megterveztük a felhasználói felületet Paintben, majd kiosztottuk, hogy ki mely részét fogja csinálni a programnak. Az óra dinamikus pozícióját (tehát, hogy reszponzív az ablak méretének változására), valamint a mutatók működését, képletét, csúszka-textfield szinkronizációját Ádám végezte. A menüpontok implementációját, fájlkezelést és a véletlenszerű generálást Adrián. Jelen dokumentációt Ádám, a bemutatandó prezentációt pedig Adrián készítette el.

1. ***ábra:*** *Kezdetleges UI terv*
   1. **Az UI szerkezete**

A képen képernyőkép, szöveg, diagram, szoftver látható

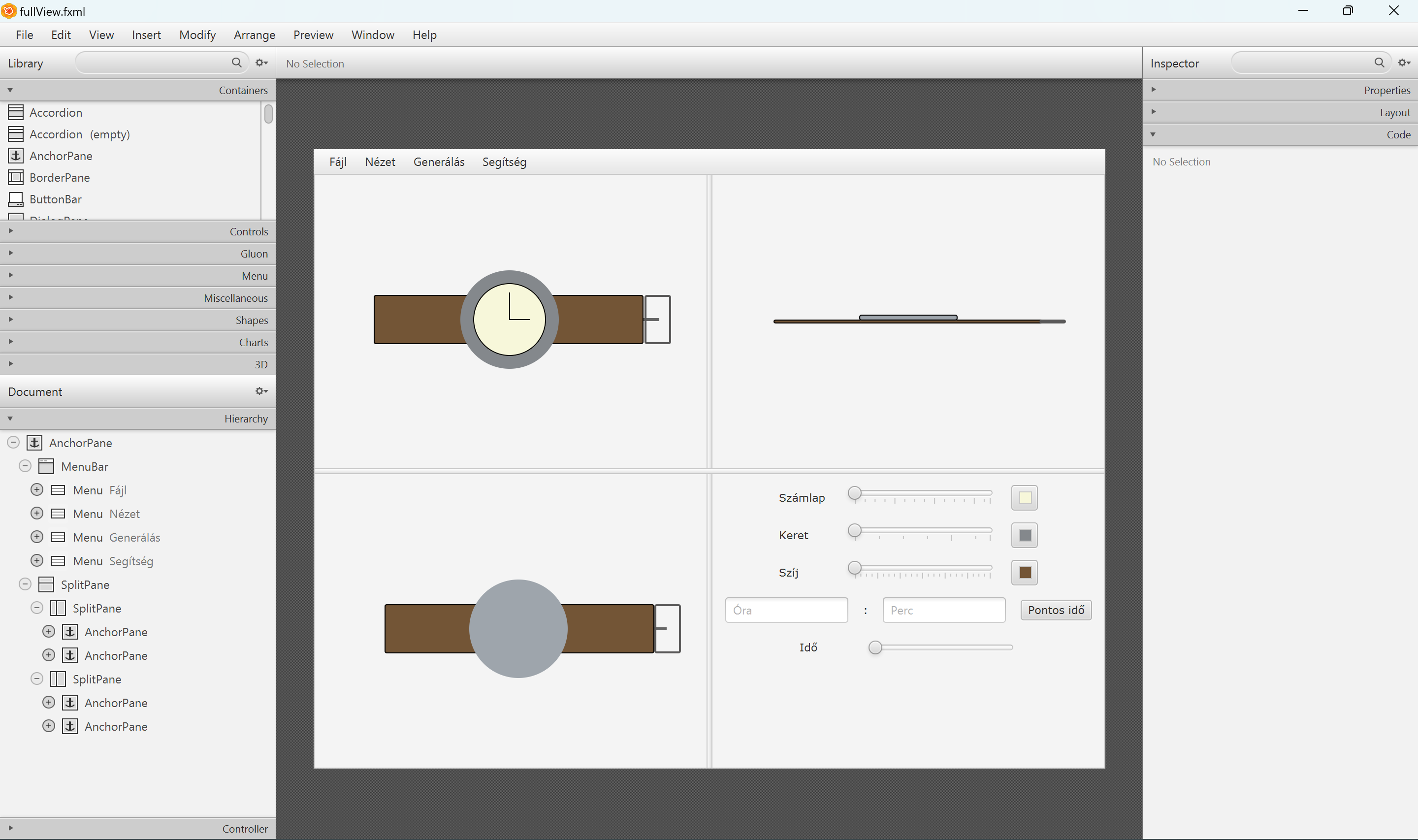
Automatikusan generált leírásA program (melyet *„JaWatch”* névre kereszteltünk) három nézetből jeleníti meg a karórát: elölnézetből, oldalnézetből és hátulnézetből. A jobb alsó sarokban változtathatjuk az aktuális példány tulajdonságait. (lásd 2. ábra) Beállíthatjuk a számlap, a keret és a szíj a színét és méretét. Az időt, amit szeretnénk, hogy mutasson, négyféleképpen állíthatjuk be. Inputmezőben, csúszkával, gyorsbillentyűvel, illetve a „Pontos idő” gombbal.

1. ***ábra:*** *UI elrendezése*
   1. **Egy objektum felépítése**

Az óra megjelenítésére a JavaFX által támogatott „primitív”-eket alkalmazzuk. A számlap és a keret egy kör, melynek a középpontja dinamikusan változik az ablak méretétől függően. A szíj és a csat egy téglalap, melynek pozíciója szintén dinamikusan változik. A mutatók, valamint a csat egy része pedig vonalként van kirajzolva. A mutatók kezdőpontja a számlap középpontja, végpontjaik pedig:

* 1. **JavaFX és FXML**

Szerettük volna a projektünket skálázhatóvá, átláthatóvá tenni, és mindenképpen újat is tanulni, ezért esett a választásunk a JavaFX-re. Segítségével modern, pehelysúlyú felhasználói felületet tudunk biztosítani. Előnye abban rejlik, hogy az egyes komponenseket nem a Java forráskódban kell létrehozni, hanem egy úgynevezett FXML fájlban kell deklarálni, amelyben a tulajdonságaikat is beállíthatjuk. A JavaFX FXML fájlokat sem kell kézzel szerkesztenünk. A JavaFX toolkit része egy program, melynek neve SceneBuilder, ahol egy grafikus felületen meg tudjuk tervezni a kívánt UI felépítést, amely utána pontosan úgy meg is valósítható. (lásd 3. ábra)



1. ***ábra:*** *SceneBuilder futás közben*

# **Tervezési dokumentáció**

* 1. **Fejlesztési környezet**

Az alkalmazás fejlesztése alatt használt környezet:

* Fejlesztői környezet:

IntelliJ IDEA 2023.3.3 tanulói licensszel

* Operációs rendszer:

64 bites Windows 11 Home (v23H2)

* Java:

java version "1.8.0\_411"

Java(TM) SE Runtime Environment (build 1.8.0\_411-b09)

Java HotSpot(TM) 64-Bit Server VM (build 25.411-b09, mixed mode)

* 1. **Osztályok leírása**

A program 2 osztályból áll: **Main** és **Controller**. A JavaFX könyvtár úgy lett megírva, hogy minden egyes *nézet*hez, ami egy FXML fájlban van definiálva, tartozik egy osztály, melynek példányváltozóiba „injektálhatjuk” a fájlban leírt objektumokat.

Például:

@FXML  
private Circle watchFace;

ami FXML fájlban:

<Circle fx:id="watchFace" centerX="198.0" centerY="147.0" fill="#f7f7da" radius="37.0" stroke="BLACK" strokeType="INSIDE" />

vagyis egy ID köti össze az FXML fájlt a Java kóddal.

Esetünkben a **Controller** osztály tartozik a *fullView.fxml* fájlhoz.

* 1. **Main osztály bemutatása**

A Main osztály az alkalmazás belépési pontja, amely az Application osztályt örökli. A main metódus a launch(args) hívással indítja az alkalmazást. A start metódusban betöltjük a felhasználói felületet a fullView.fxml fájlból, majd létrehozunk egy új jelenetet ezzel a felülettel. Betöltünk egy ikont a karoralogo.jpg fájlból, és beállítjuk a stage (alkalmazás ablaka) ikonját és címét. Végül a stage jelenetének beállítása után megjelenítjük az ablakot a felhasználó számára.

* 1. **Controller osztály bemutatása**

A Controller osztály tartalmazza az alkalmazásunk összes lényegi logikáját. A Main osztályban csak a JavaFX könyvtárat inicializáljuk. (root, scene)

### **Adattagok**

Az alábbi adattagok nagyrésze (kivétel: *hour, minute)* az FXML fájból való injektálással érhető el.

|  |  |
| --- | --- |
| RadioMenuItem **sizeShow** | A méretek megjelenítését szabályozó menüelem. |
| AnchorPane **topviewPane** | A felülnézetet tartalmazó panel. |
| AnchorPane **backviewPane** | A hátulnézetet tartalmazó panel. |
| AnchorPane **sideviewPane** | Az oldalnézetet tartalmazó panel. |
| SplitPane **felsoPane** | A felső elválasztó panel. |
| SplitPane **alsoPane** | Az alsó elválasztó panel. |
| SplitPane **mainPane** | A fő elválasztó panel. |
| Circle **watchFace** | Az óraszámlap. |
| Circle **frame** | Az órakeret. |
| Rectangle **strap** | Az óraszíj. |
| Rectangle **buckleRec** | Az óraszíj csatja. |
| Line **buckleLine** | Az óraszíj csatjának vonala. |
| Line **hourHand** | Az óramutató. |
| Line **minuteHand** | A percmutató. |
| Rectangle **backviewStrap** | A hátulnézet óraszíja. |
| Circle **backviewFrame** | A hátulnézet órakerete. |
| Rectangle **backviewBuckleRec** | A hátulnézet óraszíj csatja. |
| Line **backviewBuckleLine** | A hátulnézet óraszíj csatjának vonala. |
| Rectangle **sideviewStrap** | Az oldalnézet óraszíja. |
| Rectangle **sideviewBuckle** | Az oldalnézet óraszíj csatja. |
| Rectangle **sideviewFrame** | Az oldalnézet órakerete. |
| TextField **hourTextField** | Az óra beviteli mezője. |
| TextField **minuteTextField** | A perc beviteli mezője. |
| Button **currentTimeBtn** | A pontos idő gombja. |
| Slider **watchFaceSlider** | Az óraszámlap méretét szabályozó csúszka. |
| Slider **frameSlider** | Az órakeret méretét szabályozó csúszka. |
| Slider **strapSlider** | Az óraszíj méretét szabályozó csúszka. |
| Slider **timeSlider** | Az időt szabályozó csúszka. |
| ColorPicker **watchFaceColor** | Az óraszámlap színét szabályozó színpaletta. |
| ColorPicker **frameColor** | Az órakeret színét szabályozó színpaletta. |
| ColorPicker **strapColor** | Az óraszíj színét szabályozó színpaletta. |
| Stage **stage** | A színpad, ahol az alkalmazás fut. |
| int **hour** | Az óra értéke. |
| int **minute** | A perc értéke. |

**1. *táblázat:*** *Generáló gyorsgombok*

### **2.4.2 Metódusok**

**initialize(URL url, ResourceBundle resourceBundle):** Inicializálja a Controller osztályt, beállítja a csúszkák, szövegmezők és színpaletták értékeit, valamint hozzárendeli a megfelelő eseménykezelőket.

**save():** Mentési az aktuális óraállapotot egy fájlba.

**open():** Betölt egy korábban mentett óraállapotot egy fájlból.

**delete():** Töröl egy korábban mentett óraállapotot egy fájlból.

**generateColor():** Véletlenszerűen generál színeket az óra számára.

**generateShape():** Véletlenszerűen generál méreteket az óra számára.

**generateAll():** Véletlenszerűen generál színeket és méreteket az óra számára.

**help():** Megjeleníti a súgó ablakot.

**hotKeys():** Megjeleníti a gyorsbillentyűk listáját.

**exactTime():** Beállítja az órát a rendszer aktuális idejére.

**toggleSize():** Be- és kikapcsolja a méretek megjelenítését a csúszkákon.

**resetView():** Visszaállítja az összes nézetet az alapértelmezett állapotra.

**topFocus():** A felülnézetet állítja középpontba.

**backFocus():** A hátoldali nézetet állítja középpontba.

**sideFocus():** Az oldalnézetet állítja középpontba.

**wfColorPick():** Beállítja az óraszámlap színét a színpaletta aktuális értékére.

**frameColorPick():** Beállítja az órakeret színét a színpaletta aktuális értékére.

**strapColorPick():** Beállítja az óraszíj színét a színpaletta aktuális értékére.

**ChangeSliders():** Frissíti a csúszkákat az aktuális óra és perc értékekkel.

**ChangeSliders(double watchFaceRadius, double frameRadius, double strapWidth):** Frissíti a csúszkákat a megadott óraszámlap, keret és szíj értékekkel.

**ChangeHands():** Frissíti az óramutatók pozícióját az aktuális óra és perc értékek alapján.

**WindowChanged():** Frissíti az ablakot, amikor a felhasználó módosítja a nézeteket vagy a csúszkákat.

**Animate(int CycleCount):** Animációt hoz létre az óramutatók számára a megadott ciklusszám alapján.

* 1. **Eseményfigyelők – az initialize metódus részletei**

Az **initialize()** metódusban található listenerek (figyelők) segítségével tudunk reagálni a felhasználói interakciókra vagy a program állapotának változásaira. Az **initialize()** metódus a Controller osztály inicializálásakor fut le, amikor a felhasználói felület betöltődik. Ez a metódus ideális hely a listenerek beállításához, mivel biztosítja, hogy a listenerek készen állnak a működésre, amint a felhasználói felület elérhetővé válik.

**watchFaceSlider** és **frameSlider**: Ezek a listenerek figyelik a csúszkák értékének változását. Ha a felhasználó megváltoztatja a csúszka értékét, a listener frissíti a megfelelő óraelemek méretét, majd frissíti az óramutatókat és az ablakot.

**strapSlider**: Ez a listener figyeli a szíj csúszka értékének változását. Ha a felhasználó megváltoztatja a csúszka értékét, a listener frissíti a szíj szélességét az összes nézetben, majd frissíti az ablakot.

**timeSlider**: Ez a listener figyeli az idő csúszka értékének változását. Ha a felhasználó megváltoztatja a csúszka értékét, a listener frissíti az óra és perc értékeit, beállítja az óra és perc beviteli mezők értékét, majd frissíti az óramutatókat.

**hourTextField** és **minuteTextField**: Ezek a listenerek figyelik az óra és perc beviteli mezők értékének változását. Ha a felhasználó megváltoztatja a beviteli mezők értékét, a listener frissíti az óra és perc értékeit, majd frissíti a csúszkákat és az óramutatókat.

**felsoPane**, **alsoPane** és **mainPane**: Ezek a listenerek figyelik az elválasztó panelek pozíciójának változását. Ha a felhasználó megváltoztatja az elválasztó panelek pozícióját, a listener frissíti a többi elválasztó panel pozícióját, majd frissíti az ablakot.

# **Felhasználói dokumentáció**

* 1. **Rendszerkövetelmények**
* Operációs rendszer: Platformfüggetlen, bármely modern operációs rendszeren futtatható, amely képes a Java Virtual Machine futtatására, illetve telepítve van a JavaFX könyvtár
* Processzor: x86, x86-64, ARM architektúrájú processzorok
* Grafikus megjelenítési képesség
* >= Java 21
  1. **Programrendszer telepítése**

Ahhoz, hogy el tudjunk indítani a JaWatch programot, először meg kell bizonyosodnunk róla, hogy a megfelelő verziójú Java van a gépünkön. (>= Java 21). Ezt követően ki kell csomagolnunk a mellékelt .zip fájlt. (JaWatch.zip), majd a run.bat fájl segítségével már futtathatjuk is a programot. Fontos megjegyezni, hogy nekünk nem szükséges manuálisan telepíteni a JavaFX könyvtárcsomagot, mivel a .zip fájlunk ezt tartalmazza a kompatibilitás végett.

* 1. **A programrendszer működése**

Miután megnyitjuk a programot egy négy paneles kezelőfelület fog minket fogadni. A bal felső sarokban az aktuális órát láthatjuk felülnézetből, a jobb felső sarokban oldalnézetből, a bal alsó sarokban pedig hátulnézetből. Az aktuális példány tulajdonságait (méret, szín stb.) a jobb alsó sarokban található komponensekkel tudjuk módosítani. (lásd 1. ábra)

* 1. **Véletlenszerű generálás**

Lehetőségünk van egy példányt véletlenszerűen legenerálni. Három féle módja létezik:

* Színek generálása
* Forma generálása (Méretek)
* Színek és formák generálása egyszerre

Ezeket az opciókat a menüsorban a „Generálás” címszó alatt érjük el. Azonban definiáltunk gyorsgombokat is a gyorsabb használat érdekében (lásd: 1. táblázat)

|  |  |
| --- | --- |
| Funkció | Gyorsgomb |
| Színek generálása | F |
| Forma generálása | C |
| Együttes generálás | Space |

***2. táblázat:*** *Generáló gyorsgombok*

* 1. **Fájlkezelés**

Ha személyre szabtuk tetszés szerint az órát, akkor egy gyorsgomb segítségével (Ctrl+Alt+S), vagy a menü „Fájl > Mentés” elemével fájlba tudjuk menteni.

Lehetőségünk van egy fájl betöltésére is (FONTOS: .ora kiterjesztésű fájlok). Ezt a Ctrl+O gyorsgomb segítségével tehetjük meg, vagy a menüsor „Fájl > Megnyitás” elemével.

* 1. **Nézetek**

Négy nézet közül tudunk választani.

1. **Négypaneles**: A képernyő 4 részre van osztva, bal felül a felülnézetet, bal alul a hátulnézetet, jobb felül az oldalnézetet látjuk. A jobb alsó sarokban pedig a példány tulajdonságait szerkeszthetjük. Ebben a nézetben dupla kattintással nézetet tudunk váltani, majd az R gomb megnyomásával vissza tudunk lépni.
2. **Felülnézet**: Ebben a nézetben az aktuális órát láthatjuk felülnézetben, ám ilyenkor a szerkesztő panel eltűnik. Tehát ebben az állapotban az objektumot nem tudjuk változtatni, csak megtekinteni.
3. **Oldalnézet**: Ebben a nézetben az aktuális órát láthatjuk oldalnézetből. Szintén csak megtekinthető, szerkeszteni nem lehet.
4. **Alulnézet**: Ebben a nézetben az aktuális órát láthatjuk alulnézetből. Szintén csak megtekinthető, szerkeszteni nem lehet.

|  |  |
| --- | --- |
| **Nézet** | **Gyorsgomb** |
| Felsőnézet | F1 |
| Oldalnézet | F2 |
| Alsónézet | F3 |
| 4 paneles nézet/Nézet visszaállítása | R |

**3. *táblázat:*** *Nézetek gyorsgombjai*

* 1. **Egyéb gyorsgombok**

A program egyéb funkcióit is gyorsgombokkal láttuk el.

|  |  |
| --- | --- |
| **Funkció** | **Gyorsgomb** |
| Pontos idő visszaállítása | T |
| Méretek mutatása | M |
| Súgó megjelenítése | H |
| Gyorsgombok megjelenítése | K |

***4. táblázat:*** *Egyéb gyorsgombok*

# **Irodalomjegyzék**

* Angster E: Objektumorientált tervezés és programozás, Java, 1 és 2 kötet 4KÖR Bt., ISBN 963 00 6263 Ö, 2002.
* <https://openjfx.io/javadoc/22/>
* <https://youtu.be/9XJicRt_FaI>
* <https://docs.oracle.com/en/java/>