

Részletes dokumentáció

ECG Interpolation

Ez a JavaFX-alapú alkalmazás lehetővé teszi EKG jelek megjelenítését és feldolgozását különféle szűrési algoritmusokkal. A rendszer támogatja a szegmentált szűrést R csúcsok alapján, több szűrőtípust kínál, és JavaFX alapú interaktív felhasználói felületen jeleníti meg az eredményeket.

Fő funkciók

- EKG jelek beolvasása XML fájlokból
- Grafikus megjelenítés JavaFX segítségével
- Szűrési algoritmusok:
 - Gaussian Moving Average
 - Savitzky-Golay
 - LOESS
 - Cubic Spline
 - Wavelet (kísérleti, nincs implementálva)
- Szegmentált szűrés R-csúcsok körül
- Interaktív kezelőfelület:
 - Szűrők beállítása és láthatósága
 - Zoom és jel navigáció
 - R csúcsok vizualizálása

Rendszerkövetelmények

- Java JDK 17 vagy újabb
- Maven 3.6+

Telepítés

1. Projekt klónozása
2. Buildelés
3. Futtatás Mavenből

Projekt felépítése (lásd README)

```
` `` `text
```

```
|— .gitignore
```

|— README.md

| docs

└─ DOCUMENTATION.md

|— DOCUMENTATION_HU.md

- └ Detailed documentation.pdf

└─ Detailed_doc_hu.pdf

```
|_ pom.xml
```

```
|_ src
```

```
|— main
```

└─ java

L hu

— ujvari

|— ECGMenuApp.java

|— ecgmodel

└─ Signal.java

|— ecgplotter

$$\vdash \text{app}$$

```
└─ EcgPlotterApplication.java
```

└ controller

```
|— FilterController.java
```

```
└─ ViewController.java
```

 \vdash filter

```
|— FilterInterface.java
```

|— GaussianFilter.java

|— LoessFilter.java

|— SegmentedFilterAdapter.java

|— SgFilter.java

|— SplineFilter.java

└─ WaveletFilter.java

$$\vdash \text{model}$$

```
|— FilterParameters.java
```

```
└─ SignalData.java
```

└─ unused

```
|— CanvasEcgPlotter.java
```

|— EcgPlotter.java

FastEcgPlotter.java

L view

```
|— FilterControlPanel.java
```

|— FilterVisibilityPanel.java

|— NavigationPanel.java

|— SignalCanvas.java



Példafájlok

A `src/main/resources/xml/` mappában három minta EKG fájl található (`ecg1.xml`, `ecg2.xml`, `ecg3.xml`), amelyekkel kipróbálható az alkalmazás.

Főosztály

Az alkalmazás belépési pontja:

`hu.ujvari.ECGMenuApp`

Ez a JavaFX-alapú főmenü lehetővé teszi:

- EKG megjelenítő indítását
- XML fájlok elemzését (szerkezet, teljes XML megjelenítése)
- Kilépést

Készült Java 17+ és JavaFX 21 használatával.

ECG Interpolation Projekt Dokumentáció

Projekt Struktúra

Konfigurációs Fájlok

- `.vscode/launch.json`: Visual Studio Code indítási konfigurációk
- `.vscode/settings.json`: Projekt-specifikus VS Code beállítások
- `pom.xml`: Maven projektkezelő konfigurációs fájl

Erőforrások

- `resources/xml/`: ECG adatfájlok
 - `ecg1.xml`
 - `ecg2.xml`
 - `ecg3.xml`

Fájlok Részletes Ismertetése

Konfigurációs Fájlok

`.vscode/launch.json`

- Visual Studio Code futtatási konfigurációkat tartalmaz
- Meghatározza a projekt futtatásához és debugging-hoz szükséges beállításokat

`.vscode/settings.json`

- Projekt-specifikus VS Code beállításokat tárol
- Kódformázási, szerkesztési és egyéb IDE-vel kapcsolatos preferenciákat tartalmaz

`pom.xml`

- Maven projektkezelő konfigurációs fájl
- Tartalmazza a projekt függőségeit, verzióit és build beállításait

Fő Alkalmazás

`ECGMenuApp.java`

- A projekt fő belépési pontja

Általános leírás

Az `ECGMenuApp.java` egy JavaFX alapú grafikus felhasználói felületet (GUI) valósít meg az ECG alkalmazáshoz. A fő menüben három opció érhető el: ECG Megjelenítő, XML Elemző és Kilépés. Az XML Elemző almenüben lehet listázni az elérhető elvezetések vagy megtekinteni az XML struktúráját. A `startEcgPlotter()` metódus betölti az ECG adatokat egy XML fájlból (jelen esetben `ecg1.xml`), és megjeleníti azokat egy külön ablakban. A `startXmlAnalyzer()` metódus lehetővé teszi az XML fájl tartalmának vizsgálatát, amely képes kilistázni az elérhető elvezetések vagy megjeleníteni a teljes XML struktúrát egy szövegmezőben. A főbb metódusok: `showMainMenu()`, `showXmlAnalyzerMenu()`, `startEcgPlotter()` és `startXmlAnalyzer()`.

A `startEcgPlotter()` metódusban az `EcgPlotterApplication` osztály kerül meghívásra. Ez felelős az ECG jelek grafikus megjelenítéséért.

Az `EcgPlotterApplication.java` egy komplex ECG jel megjelenítő és elemző alkalmazás. A fő funkciói:

1. Betölti a kapott ECG jeleket egy statikus `setData()` metóduson keresztül.
2. Többféle szűrőt regisztrál és alkalmaz a jeleken (Gaussian, Savitzky-Golay, LOESS, Spline).
3. Létrehoz egy JavaFX alapú grafikus felületet, amely lehetővé teszi:
 - A jel megjelenítését (`SignalCanvas`)
 - Szűrők vezérlését (`FilterControlPanel`)
 - Navigációt és szűrő láthatóságot
 - Állapotjelzést

A `processData()` metódus aszinkron módon alkalmazza az összes regisztrált szűrőt, majd frissíti a canvas megjelenítését.

A `setData()` metódus egy statikus `signal` nevű `List<Double>`-be tölti a jelet:

Ezt a statikus `signal` listát használja fel később a `start()` metódusban a `SignalData` konstruálásakor:

```
signalData = new SignalData(signal);
```

Tehát a `setData()` egy statikus listába menti az adatokat, amelyet aztán a `SignalData` objektum fog feldolgozni az alkalmazás indulásakor.

A `SignalData.java` egy komplex modell osztály, amely az ECG jel adatainak kezelésére és manipulálására szolgál. Főbb jellemzői:

1. Tárolja az eredeti jelet (`originalSignal`) és a szűrt jeleket (`filteredSignals`)
2. Kezeli a jel minimális és maximális értékeit

3. Támogatja a viewport (nézet) kezelését:
 - Viewport mozgatása (`moveViewport()`)
 - Zoom szint beállítása (`setZoomLevel()`)
 - Nézet alaphelyzetbe állítása (`resetViewRange()`)

A kulcs metódusok:

- `addFilteredSignal()`: Új szűrt jel hozzáadása
- `updateMinMaxValues()`: Frissíti a jel min-max értékeit
- `setViewport()`: Beállítja a látható jelrészletet
- `setZoomLevel()`: Módosítja a nagyítási szintet

A tervezés thread-safe, szinkronizált metódusokkal védi az adatokat. Lehetőséget ad a jel különböző nézeteinek és feldolgozási módszereinek kezelésére.

Szűrők:

A `FilterParameters.java` egy absztrakt osztály, amely különböző szűrési algoritmusok paramétereinek tárolására és kezelésére szolgál. Fő jellemzői:

1. Absztrakt alaposztály `FilterParameters`, amelyből származnak a specifikus szűrő paraméter osztályok.
2. Minden szűrőtípushoz külön belső osztály létezik:
 - `GaussianParameters`: Gaussian szűrő (ablakméret)
 - `SavitzkyGolayParameters`: Savitzky-Golay szűrő (ablakméret, polinom rendje)
 - `LoessParameters`: LOESS szűrő (ablakméret, polinom rendje, sávszélesség)
 - `SplineParameters`: Spline interpoláció (downsampling mértéke)
 - `WaveletParameters`: Wavelet transzformáció (szint, küszöbérték)
 - `SegmentFilterParameters`: Szegmentált szűrőkhöz speciális paraméterek
3. Lehetőséget biztosít a szűrési paraméterek dinamikus beállítására és lekérdezésére.

A `FilterInterface.java` egy egyszerű, de kulcsfontosságú interfész, amely meghatározza a szűrő algoritmusok alapvető működését:

1. `getName()`: Visszaadja a szűrő nevét
2. `filter()`: Végrehajtja a tényleges szűrést egy bemeneti jelen
3. `getParameters()`: Lekéri a szűrő aktuális paramétereit
4. `setParameters()`: Lehetővé teszi a szűrő paramétereinek módosítását

Ez az interfész biztosítja, hogy minden szűrő implementáció egységes módon legyen definiálva a rendszerben, így könnyen lehet őket kezelni és alkalmazni.

A `SgFilter`, `LoessFilter`, `SplineFilter` osztályok valójában adapter osztályok, amely egy másik,

```
hu.ujvari.ecgprocessor.SavitzkyGolayFilter,  
hu.ujvari.ecgprocessor.LoessFilter,  
hu.ujvari.ecgprocessor.CubicSplineFilter
```

nevű implementációkat használnak fel. Ez egy köztes réteg, amely:

1. Átveszi a paramétereket
2. Létrehoz egy másik csomag szűrőjét
3. Meghívja annak `filter()` metódusát

Az adapter tervezési minta segítségével oldja meg, hogy a projekt különböző komponensei egységesen tudják kezelni a szűrőket, miközben a tényleges számítások egy másik, már meglévő implementációban történnek.

A jelet Teljes terjedelmükben vagy Az R csúcsok mentén szakaszokra bontva is lehet szűrni, ezeket a szűréseket összehasonlítni. A szegmentált szűrések az R csúcsok “megőrzése” érdekében kerülnek alkalmazásra

A szegmentált szűrések megvalósítása:

1. A `LoessFilter`, `CubicSplineFilter` és `SavitzkyGolayFilter` a `hu.ujvari.ecgprocessor` csomagban valósítják meg a tényleges szűrési algoritmusokat.
2. Az `ECGSegmenter` egy speciális segédosztály, amely lehetővé teszi a szűrők szegmentált alkalmazását az EKG jeleken. Főbb képességei:
 - R csúcsok detektálása (`detectRPeaks`)
 - Jel szűrése szegmensekre bontva, úgy, hogy az R csúcsok pozícióját megőrzi (`applyFilterBySegments`)
 - Finomított csúcspozíció meghatározás (`refineRPeakPosition`)

A szegmentált szűrés lényege, hogy:

- Felismeri az R csúcsokat a jelben
- A csúcsok körül külön kezeli a jel szűrését
- Biztosítja, hogy a fontos kardiológiai pontok (R csúcsok) ne torzuljanak

Ez egy olyan megközelítés, amely megőrzi a jel diagnosztikai jellegzetességeit a zajcsökkentés során.

Az **ECGSegmenter** egy komplex osztály, amely az EKG jelek speciális feldolgozására szolgál. Főbb funkciói:

1. R csúcsok detektálása (**detectRPeaks**):
 - Ablakozásos módszerrel keresi a jel csúcspontjait
 - Küszöbérték alapján dönt a csúcsok létezéséről
 - 400 ms-os ablakokat használ 200 ms-os átfedéssel
 - Kiszűri a közeli csúcsokat (minimális 100 ms távolság)
2. Szűrés szegmentáltan (**applyFilterBySegments**):
 - Két változata van: a) Automatikus R csúcs detektálással b) Előre megadott R csúcs indexekkel
 - Lépései:
 - R csúcsok detektálása
 - Jel felosztása szegmensekre a csúcsok körül
 - Minden szegmens külön szűrése
 - Csúcsok körüli sima átmenetek biztosítása
3. Átmenetek kezelése:
 - Lineáris interpoláció a csúcsok körül
 - Biztosítja a szűrt jel folytonosságát
 - Megőrzi a diagnosztikailag fontos pontokat

Alapvető célja, hogy zajcsökkentést végezzen úgy, hogy közben megőrzi a jel eredeti szerkezetét és fontos jellemzőit.

A **FilterController** egy komplex vezérlő osztály, amely az EKG jel szűrési folyamatát kezeli. Főbb jellemzői:

1. Szűrő regisztráció és kezelés:
 - Lehetővé teszi különböző szűrők (**FilterInterface**) regisztrálását
 - Tárolja a szűrőket egy térképben (**filters**)
 - Kezeli a szűrők közötti függőségeket (**dependencies**)
2. Szűrési folyamat:
 - **applyFilter()**: Egyetlen szűrő alkalmazása
 - Aszinkron módon hajtja végre a szűréseket (**CompletableFuture**)
 - Támogatja a szűrők láncolt alkalmazását a függőségek alapján
3. Speciális képességek:
 - Szegmentált szűrők speciális kezelése
 - Párhuzamos végrehajtás (több magos számítógépeken)
 - Rugalmas paraméter menedzsment

Főbb metódusok:

- **registerFilter()**: Új szűrő hozzáadása
- **addDependency()**: Szűrők közötti függőségek definiálása
- **applyFilter()**: Egyedi szűrő alkalmazása
- **applyAllFilters()**: Minden regisztrált szűrő alkalmazása

A szűrők nem csak önállóan, hanem egymásra épülve is alkalmazhatók, biztosítva a komplex jelfeldolgozási lehetőségeket. (Ez még nincs implementálva).

A **ViewController** az alkalmazás felhasználói felületének vezérlését és állapotkezelését végzi. Főbb funkciói:

1. Kezdeti beállítások:
 - Alapértelmezetten beállítja az eredeti jel láthatóságát
 - Inicializálja a szűrők láthatóságát egy alapértelmezett térképben
2. Komponensek összekapcsolása:
 - **setSignalCanvas()**: Beállítja a jel megjelenítő vásznat
 - **setVisibilityPanel()**: Konfigurálja a szűrő láthatóság panelt
 - **setNavigationPanel()**: Beállítja a navigációs panelt
 - **setStatusPanel()**: Állapotjelző panel beállítása
3. Interakció kezelése:
 - Szűrők láthatóságának dinamikus módosítása
 - Nézet frissítése (viewport változás, újrarajzolás)
 - Állapotüzenetek kezelése

Főbb metódusok:

- **redrawCanvas()**: Vászon újrarajzolása
- **resetView()**: Nézet alaphelyzetbe állítása
- **updateStatus()**: Állapotüzenet frissítése

Lényegében egy koordináló réteg, amely összekösszi a modellt (SignalData) a nézettel (SignalCanvas, NavigationPanel stb.), és kezeli a felhasználói interakciókat.

A **SegmentedFilterAdapter** egy speciális adapter osztály, amely lehetővé teszi az alapszűrők szegmentált alkalmazását EKG jeleken. Főbb jellemzői:

1. Működési elv:
 - Egy alapvető szűrőt kap (pl. Loess, Savitzky-Golay, CubicSpline)
 - R csúcsokat detektál az eredeti jelen
 - A jelet szegmensekre bontja a csúcsok körül
 - Minden szegmenst külön szűr
 - Megőrzi a diagnosztikailag fontos pontokat
2. Kulcs funkciók:
 - Dinamikus R csúcs detektálás
 - Küszöbérték alapú csúskeresés
 - Szűrés szegmensekre bontva
 - Paraméterek rugalmas kezelése
3. Speciális mechanizmusok:
 - Csúcsok egyszer történő detektálása

- Védett másolatok kezelése
- Alapszűrő paramétereinek frissítése

A tervezési minta lehetővé teszi, hogy bármilyen alap szűrőt szegmentáltan lehessen alkalmazni, megőrizve a jel diagnosztikai tulajdonságait.

A **SignalCanvas** egy JavaFX Canvas komponens, amely az EKG jelek megjelenítéséért felelős. Főbb jellemzői:

1. Jel megjelenítés:
 - Több szűrő egyidejű ábrázolása
 - Külön szín és átlátszóság minden szűrőhöz
 - Dinamikus láthatóság beállítás
2. Interaktív vezérlők:
 - Egér húzással viewport mozgatás
 - Görgetéssel zoom funkció
 - Koordináta transzformáció
3. Speciális megjelenítési elemek:
 - Eredeti és szűrt jelek ábrázolása
 - R csúcsok megjelölése
 - Tengelycímkék és információs szövegek

Főbb metódusok:

- **redrawChart()**: Teljes diagram újrarajzolása
- **drawSignal()**: Egyedi jel kirajzolása
- **drawRPeaks()**: R csúcsok megjelölése
- **setupCanvasEvents()**: Egér és görgetés események kezelése

Tervezési célok:

- Rugalmas jel vizualizáció
- Felhasználóbarát interakció
- Részletes diagnosztikai információk megjelenítése

A **drawSignal()** metódus minden egyes híváskor kirajzolja az adott jelet, míg a **redrawChart()** valóban minden jelentősebb esemény után újrarajzolja a teljes diagramot.

A **redrawChart()** akkor kerül meghívásra, amikor:

- Változik a viewport (nézetablak)
- Módosul a zoom szint
- Új szűrő kerül alkalmazásra
- Megváltozik egy szűrő láthatósága
- Egyéb felhasználói interakciók történnek
- Törli a vásznat (**clearCanvas()**)
- Újrarajzolja az összes szűrőt, amelyek láthatóságra vannak állítva
- Frissíti a tengelycímkéket és az információs szöveget

Tehát minden alkalommal, amikor valami megváltozik a jelben vagy a megjelenítésben, a `redrawChart()` újrarajzolja a teljes diagramot.

Ezek a komponensek mind hozzájárulnak a felhasználói felület és a jel megjelenítésének vezérléséhez:

1. `FilterControlPanel`:
 - Szűrő beállítások vezérlőfelülete
 - Lehetővé teszi a különböző szűrők paramétereinek módosítását
 - Alkalmazza a kiválasztott szűrőket
 - Frissíti a `SignalCanvas` megjelenítését minden szűrő alkalmazásakor
2. `FilterVisibilityPanel`:
 - Szabályozza, hogy melyik szűrt jel látható a diagramon
 - Kapcsolók (checkboxok) az egyes szűrők be- és kikapcsolásához
 - Közvetlenül befolyásolja a `SignalCanvas` megjelenítését
3. `NavigationPanel`:
 - Zoom vezérlés
 - Viewport mozgatás (balra-jobbra navigálás a jelben)
 - Teljes nézet visszaállítása
 - Befolyásolja a megjelenített jel tartományát
4. `StatusPanel`:
 - Állapotüzenetek megjelenítése
 - Folyamatjelző sáv
 - Nem közvetlen hatással van a jel megjelenítésére, inkább tájékoztató jellegű

Mindegyik komponens szorosan együttműködik a `SignalCanvas`-szal és a `SignalData` modellel, így komplex módon irányítják a jel megjelenítését és feldolgozását.

Szegmentált szűrés példa:

1. Indítás az `ECGMenuApp`-ból:
 - Betölti az XML fájlból a jelet
 - Átadja az adatokat az `EcgPlotterApplication`-nek
2. `EcgPlotterApplication` inicializálása:
 - Létrehozza a `SignalData` objektumot
 - Regisztrálja a szűrőket a `FilterController`-ben
 - Létrehoz egy szegmentált Savitzky-Golay szűrőt és regisztrálja
3. Felhasználói interakció a `FilterControlPanel`-on:
 - A felhasználó kiválasztja a "Segmented Savitzky-Golay" fület
 - Beállíthatja az R csúcs küszöbértéket
4. Szűrési folyamat (`SegmentedFilterAdapter`):

- **ECGSegmenter** detektálja az R csúcsokat az eredeti jelben
- A jelet szegmensekre bontja a csúcsok körül
- Minden szegmenst külön alkalmazza a Savitzky-Golay szűrőt
- Finoman illeszti a szegmenseket és megőrzi az R csúcsokat

5. Megjelenítés (**SignalCanvas**):

- Kirajzolja a szűrt jelet
- Külön megjelöli az R csúcsokat
- Lehetővé teszi a különböző szűrők közötti váltást

A kulcs komponensek:

- **ECGSegmenter**: R csúcsok detektálása és szegmentálás
- **SegmentedFilterAdapter**: Szűrési logika
- **FilterController**: Szűrők vezérlése

A **SegmentedFilterAdapter** generikus módon képes kezelni bármilyen alapszűrőt, így könnyen létrehozhatók szegmentált változatok más szűrőkhöz is.