11. Grafikus felület specifikációja

54 – *Override*

Konzulens:
Dr. László Zoltán

Csapattagok:

Kriván Bálint	CBVOEN	balint@krivan.hu
Jákli Gábor	ONZ5G1	j_gab666@hotmail.com
Dévényi Attila	L1YRH0	devenyiat@gmail.com
Apagyi Gábor	X8SG3T	apagyi.gabooo@gmail.com
Péter Tamás Pál	N5ZLEG	falconsaglevlist@gmail.com

2011. április 24.

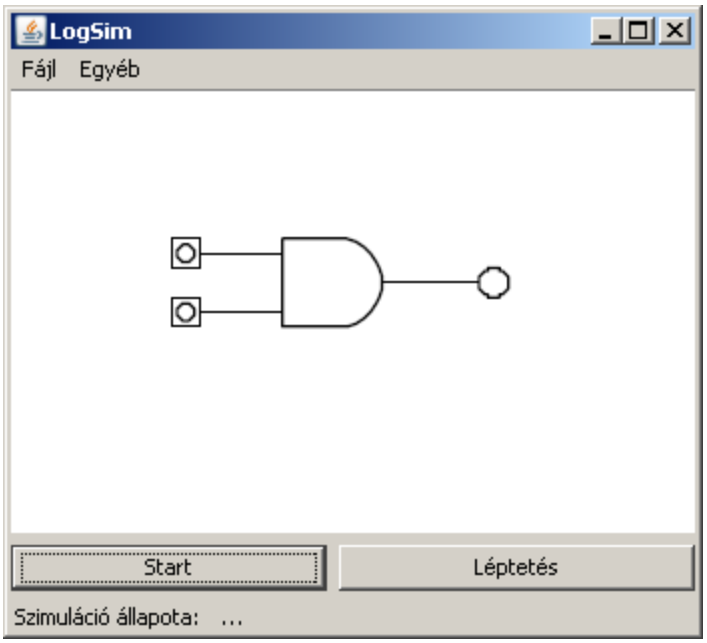
Tartalomjegyzék

Ábrák jegyzéke

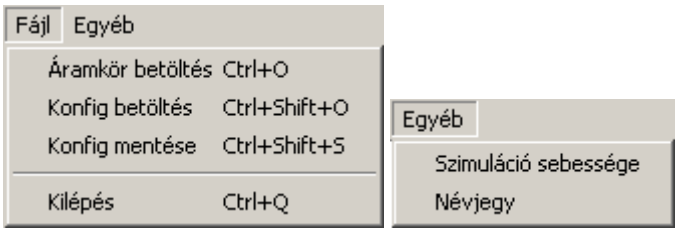
11. Grafikus felület specifikációja

11.1. A grafikus interfész

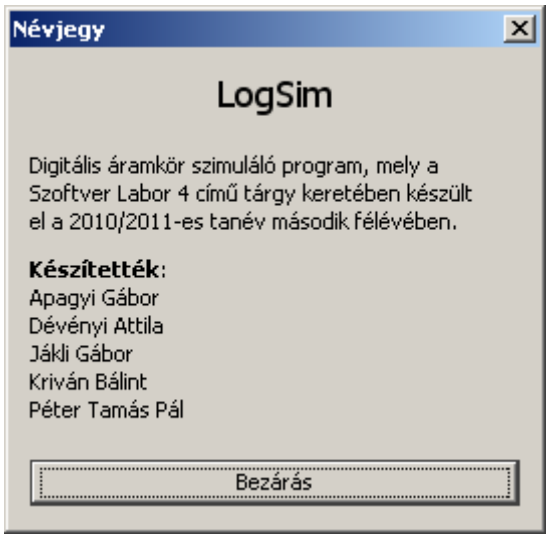
A 11.1. ábra mutatja a főablakot, a benne lévő áramkör csak illusztráció. A két menü almenüi a 11.2. ábrán látszódnak. A Fájl menü almenüi beszédesek, a felső három menüpontra megnyílik egy fájlválasztó ablak, ahol megadható egy fájl, majd az adott akció lefut. A Kilépés menüpont segítségével kiléphetünk az alkalmazásból. Az Egyéb menü Néjegy menüpontjára kapcsolva pedig megnyílik a 11.3. ábrán látható ablak.



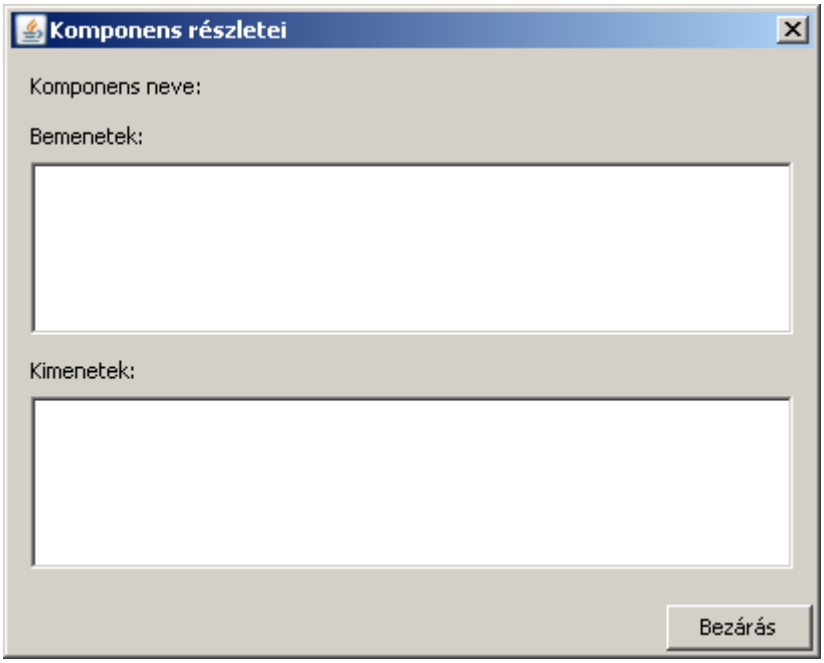
11.1. ábra. Főablak



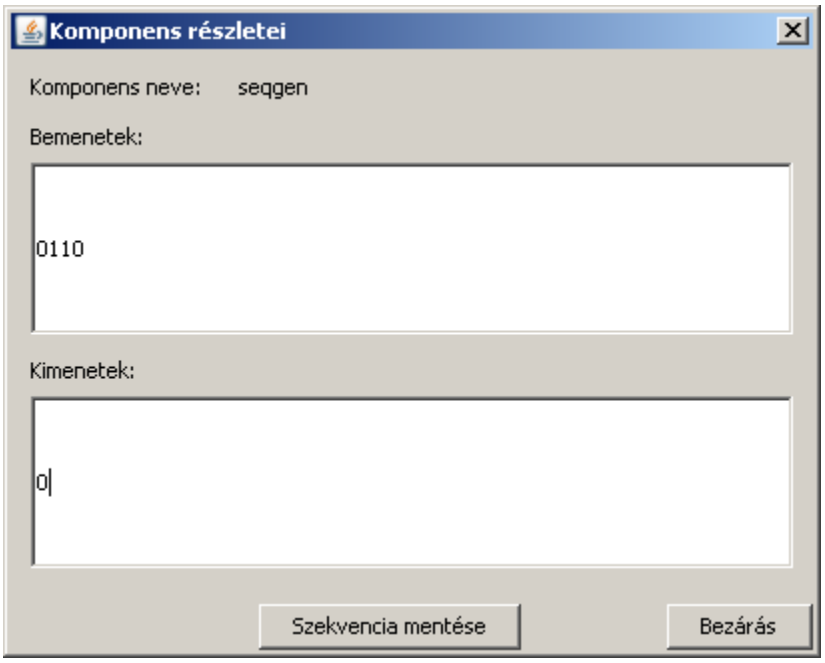
11.2. ábra. Fájl és az Egyéb menü almenüi



11.3. ábra. Névjegy



11.4. ábra. Komponens részletei



11.5. ábra. Szekvencia generátor beállítása

11.2. A grafikus rendszer architektúrája

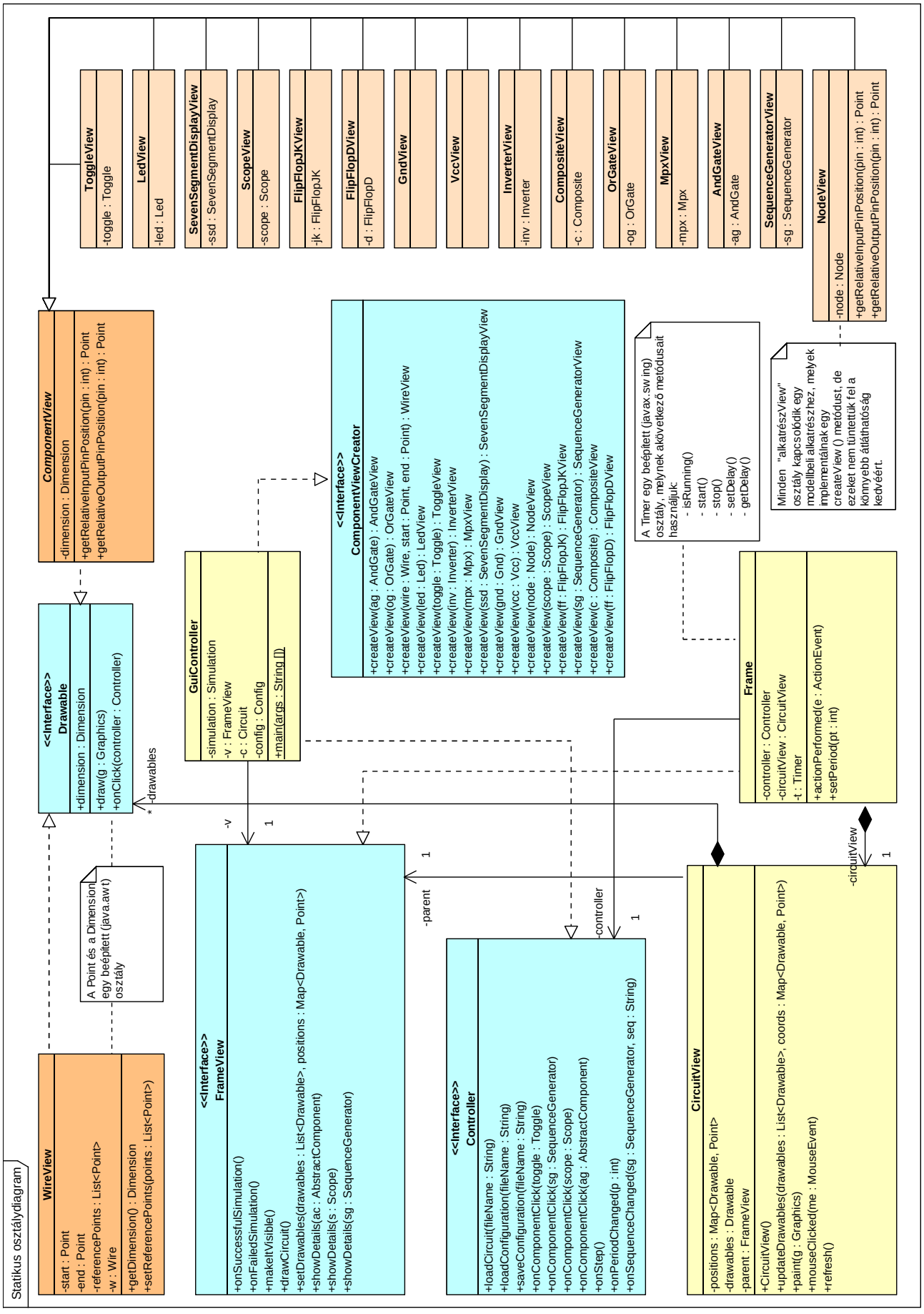
[A felület működésének elve, a grafikus rendszer architektúrája (struktúra diagramok). A struktúra diagramokon a prototípus azon és csak azon osztályainak is szerepelnie kell, amelyekhez a grafikus felületet létrehozó osztályok kapcsolódnak.]

11.2.1. A felület működési elve

[Le kell írni, hogy a grafikai megjelenésért felelős osztályok, objektumok hogyan kapcsolódnak a meglevő rendszerhez, a megjelenítés során mi volt az alapelv. Törekedni kell az MVC megvalósításra. Alapelvek lehetnek: ***push*** *alapú: a modell értesíti a felületet, hogy változott;* ***pull*** *alapú: a felület kérdezi le a modellt, hogy változott-e;* ***kevert****: a kettő kombinációja.]*

Az általunk elkészített grafikus felület "pull" típusú, vagyis a grafikus rendszer kérdezi le a modell objektumoktól az aktuális állapotukat. Azokhoz a modellobjektumokhoz, melyeket megjelenítünk, elkészítettünk egy-egy wrapper osztályt, mely a megjelenítésért és a megjelenítéshez szükséges információk tárolásáért felel. Az áramkört egy JPanel-ra rajzoljuk, mely biztosítja számunkra, hogy az elhelyezhető legyen bármilyen ablakon. Áramkör újrarajzoláskor, az eltárolt objektumok egyenként rajzolják ki magukat az előzőleg megadott koordináták alapján. Bármilyen felhasználói interakciónál, melynél változhat az áramkör állapota, az egész áramkört újrarajzoljuk, biztosítva ezzel, hogy a kirajzolt áramkör mindig az aktuális állapotban legyen megjelenítve.

11.2.2. A felület osztály-struktúrája



11.6. ábra. Statikus struktúra nézet

11.3. A grafikus objektumok felsorolása

11.3.1. ComponentViewCreator

Interfész.

- Felelősség
  Az egyes alkatrészekhez létrehozza a "megjeleníthető" wrapper objektumokat.
- Ősosztályok ComponentViewCreator.
- Interfészek (nincs)
- Metódusok

  + `AndGateView createView(AndGate ag)`: Megjeleníthető ÉS kapu létrehozása

  + `CompositeView createView(Composite c)`: Megjeleníthető Kompozit létrehozása

  + `FlipFlopDView createView(FlipFlopD ff)`: Megjeleníthető D flip-flop létrehozása

  + `FlipFlopJKView createView(FlipFlopJK ff)`: Megjeleníthető JK flip-flop létrehozása

  + `GndView createView(Gnd gnd)`: Megjeleníthető GND komponens létrehozása

  + `InverterView createView(Inverter inv)`: Megjeleníthető Inverter komponens létrehozása

  + `LedView createView(Led led)`: Megjeleníthető LED komponens létrehozása

  + `MpxView createView(Mpx mpx)`: Megjeleníthető Multiplexer komponens létrehozása

  + `NodeView createView(Node node)`: Megjeleníthető Node komponens létrehozása

  + `OrGateView createView(OrGate og)`: Megjeleníthető VAGY kapu létrehozása

  + `ScopeView createView(Scope scope)`: Megjeleníthető Scope komponens létrehozása

  + `SequenceGeneratorView createView(SequenceGenerator sg)`: Megjeleníthető jelgenerátor létrehozása

  + `SevenSegmentDisplayView createView(SevenSegmentDisplay ssd)`: Megjeleníthető Hétszegmenses komponens létrehozása

  + `ToggleView createView(Toggle toggle)`: Megjeleníthető Kapcsoló komponens létrehozása

  + `VccView createView(Vcc vcc)`: Megjeleníthető VCC komponens létrehozása

  + `WireView createView(Wire wire, Point start, Point end)`: Megjeleníthető vezeték létrehozása

11.3.2. Controller

Interfész.

- Felelősség
  A program ezeket a szolgáltatásokat nyújta a grafikus felület felé
- Ősosztályok Controller.
- Interfészek (nincs)
- Metódusok

+ `void loadCircuit(String fileName)`: Áramkör betöltése

+ `void loadConfiguration(String fileName)`: Áromkör konfigurációs fájl betöltése

+ `void onComponentClick(AbstractComponent ag)`: Általános komponens információ megjelenítés (név, bemenet, kimenet)

+ `void onComponentClick(Scope scope)`: Scope megjelenítés (eddig eltárolt értékek)

+ `void onComponentClick(SequenceGenerator sg)`: Jelgenerátor megjelenítése és konfigurálása

+ `void onComponentClick(Toggle toggle)`: Kapcsoló változtatása

+ `void onPeriodChanged(int p)`: Szimuláció sebességének megváltoztatása

+ `void onSequenceChanged(SequenceGenerator sg, String seq)`: Új szekvencia mentése

+ `void onStep()`: Áramkör léptetése

+ `void saveConfiguration(String fileName)`: Konfigurációs fájl mentése

11.3.3. GuiController

- Felelősség
  Az alkalmazás vezérlője
- Ősosztályok Object → GuiController.
- Interfészek ComponentViewCreator, Controller.
- Attribútumok
  - `Circuit c`: vezérelt áramkör
  - `Config config`: vezérelt áramkörhöz tartozó konfiguráció
  - `Simulation simulation`: szimuláció
  - `FrameView v`: alkalmazás főablaka
- Metódusok
  + `GuiController()`: Konstruktor
  + `AndGateView createView(AndGate ag)`: Megjeleníthető ÉS kapu létrehozása
  + `CompositeView createView(Composite c)`: Megjeleníthető Kompozit létrehozása
  + `FlipFlopDView createView(FlipFlopD ff)`: Megjeleníthető D flip-flop létrehozása
  + `FlipFlopJKView createView(FlipFlopJK ff)`: Megjeleníthető JK flip-flop létrehozása
  + `GndView createView(Gnd gnd)`: Megjeleníthető GND komponens létrehozása
  + `InverterView createView(Inverter inv)`: Megjeleníthető Inverter komponens létrehozása
  + `LedView createView(Led led)`: Megjeleníthető LED komponens létrehozása
  + `MpxView createView(Mpx mpx)`: Megjeleníthető Multiplexer komponens létrehozása
  + `NodeView createView(Node node)`: Megjeleníthető Node komponens létrehozása
  + `OrGateView createView(OrGate og)`: Megjeleníthető VAGY kapu létrehozása
  + `ScopeView createView(Scope scope)`: Megjeleníthető Scope komponens létrehozása

`+ SequenceGeneratorView createView(SequenceGenerator sg)`: Megjeleníthető jelgenerátor létrehozása

`+ SevenSegmentDisplayView createView(SevenSegmentDisplay ssd)`: Megjeleníthető Hétszegmenses komponens létrehozása

`+ ToggleView createView(Toggle toggle)`: Megjeleníthető Kapcsoló komponens létrehozása

`+ VccView createView(Vcc vcc)`: Megjeleníthető VCC komponens létrehozása

`+ WireView createView(Wire wire, Point start, Point end)`: Megjeleníthető vezeték létrehozása

`+ void loadCircuit(String fileName)`: Áramkör betöltése

`+ void loadConfiguration(String fileName)`: Áromkör konfigurációs fájl betöltése

`+ static void main(String[] args)`: Program belépési pontja

`+ void onComponentClick(AbstractComponent ag)`: Általános komponens információ megjelenítés (név, bemenet, kimenet)

`+ void onComponentClick(Scope scope)`: Scope megjelenítés (eddig eltárolt értékek)

`+ void onComponentClick(SequenceGenerator sg)`: Jelgenerátor megjelenítése és konfigurálása

`+ void onComponentClick(Toggle toggle)`: Kapcsoló változtatása

`+ void onPeriodChanged(int p)`: Szimuláció sebességének megváltoztatása

`+ void onSequenceChanged(SequenceGenerator sg, String seq)`: Szekvenciagenerátor értékének változtatása

`+ void onStep()`: Áramkör léptetése

`− void run()`: főablakot kirajzoljuk

`+ void saveConfiguration(String fileName)`: Konfigurációs fájl mentése

11.3.4. Parser (vált.)

- Felelősség
  Áramkör értelmező objektum, feladata, hogy a paraméterként átadott, illetve fájlban elhelyezett komponenseket értelmezze, a kapcsolatokat feltérképezze, elvégezze az összeköttetéseket, és ezáltal felépítse az áramkört.
- Ősosztályok Object → Parser.
- Interfészek (nincs)
- Attribútumok
- Metódusok

  `+ Point getPosition(AbstractComponent ac)`: Komponens pozíciójának a lekérdezése

  `− AbstractComponent parseComponent(String variableName, String componentNa String argumentsStr, Composite composite)`:

  `− AbstractComponent parseComponentFromLine(Matcher matcher, Composite composite)`: Egy komponens-sor feldolgozása a fájlban

  `− AbstractComponent parseTopLevelComponentFromLine(Matcher matcher, Circuit circuit)`: Egy olyan komponens-sor feldolgozása a fájlban, ami a legfelső szinten szerepel, azaz a kompozit amiben szerepel az az áramkör. Itt a pozíció információt is feldolgozzuk!

11.3.5. AbstractComponent (vált.)

Absztrakt osztály.

- Felelősség
  Egy komponens absztrakt megvalósítása, ebből származik az összes többi komponens. A közös logikát valósítja meg. A gyakran használt dolgokra ad alapértelmezett implementációt (kimenetekre és bemenetekre kötés, kiértékelés stb.)
- Ősosztályok Object → AbstractComponent.
- Interfészek (nincs)
- Attribútumok
- Metódusok

  `+ ComponentView createView(ComponentViewCreator cvc)`: Lekérjük a komponenst ábrázoló viewt, de a tényleges rajzolást nem mi végezzük, hanem a ComponentViewCreator, kihasználva a Visitor tervezési mintát.

11.3.6. Composite (vált.)

- Felelősség
  Kompozit elem leírása, kiértékelésnél a tartalmazott komponenseket kiértékeli, lépteti a jelgenerátorokat stb. Ha nem áll be stacionárius állapotba a kiértékelésnél, akkor ezt jelzi kifelé.
- Ősosztályok Object → AbstractComponent → Composite.
- Interfészek (nincs)
- Attribútumok
- Metódusok

  `+ ComponentView createView(ComponentViewCreator cvc):`

11.3.7. Wire (vált.)

- Felelősség
  Vezeték osztály. Két komponens-lábat köt össze. A rajta lévő érték lekérdezhető és beállítható.
- Ősosztályok Object → Wire.
- Interfészek (nincs)
- Attribútumok
- Metódusok

  `+ WireView createView(ComponentViewCreator cvc, Point start, Point end)`: Lekérjük a vezetéket ábrázoló viewt, de a tényleges rajzolást nem mi végezzük, hanem a ComponentViewCreator, kihasználva a Visitor tervezési mintát.

11.3.8. AndGate (vált.)

- Felelősség
  ÉS kapu, az áramkör egyik alapeleme. Bemeneteire kötött komponensek kiértékelését kezdeményezi, s a kapott értékek logikai ÉS kapcsolatát valósítja meg, amit a kimenetén kiad.
- Ősosztályok Object → AbstractComponent → AndGate.
- Interfészek (nincs)
- Attribútumok
  - (nincs)
- Metódusok

```
+ ComponentView createView(ComponentViewCreator cvc):
```

11.3.9. FlipFlopD (vált.)

- Felelősség
  D flipflop, mely felfutó órajelnél beírja a belső memóriába az adatbemeneten (D) lévő értéket.
- Ősosztályok Object → AbstractComponent → FlipFlop → FlipFlopD.
- Interfészek (nincs)
- Attribútumok
- Metódusok

```
+ ComponentView createView(ComponentViewCreator cvc):
```

11.3.10. FlipFlopJK (vált.)

- Felelősség
  JK flipflop, mely a belső memóriáját a Követelmények résznél leírt módon a J és K bemenetektől függően változtatja.
- Ősosztályok Object → AbstractComponent → FlipFlop → FlipFlopJK.
- Interfészek (nincs)
- Attribútumok
- Metódusok

```
+ ComponentView createView(ComponentViewCreator cvc):
```

11.3.11. Gnd (vált.)

- Felelősség
  A "föld" komponens, mely állandóan a hamis értéket adja ki. Nincs bemenete.
- Ősosztályok Object → AbstractComponent → Gnd.
- Interfészek (nincs)
- Attribútumok
  - (nincs)
- Metódusok

```
+ ComponentView createView(ComponentViewCreator cvc):
```

11.3.12. Inverter (vált.)

- Felelősség
  Inverter alkatrész, mely invertálva adja ki a kimenetén a bemenetén érkező jelet.
- Ősosztályok Object → AbstractComponent → Inverter.
- Interfészek (nincs)
- Attribútumok
  - (nincs)
- Metódusok

```
+ ComponentView createView(ComponentViewCreator cvc):
```

11.3.13. Led (vált.)

- Felelősség
  Egy LED-et reprezentál, mely világít, ha bemenetén igaz érték van.
- Ősosztályok Object → AbstractComponent → DisplayComponent → Led.
- Interfészek (nincs)
- Attribútumok
  - (nincs)
- Metódusok

```
+ ComponentView createView(ComponentViewCreator cvc):
```

11.3.14. Mpx (vált.)

- Felelősség
  4-1-es multiplexer, melynek a bemeneti lábak sorrendje a következő: D0, D1, D2, D3, S0, S1. Ahol Dx az adatbemenetek, Sy a kiválasztóbemenetek. Kimenetén a kiválasztóbemenetektől függően valamelyik adatbemenet kerül kiadásra.
- Ősosztályok Object → AbstractComponent → Mpx.
- Interfészek (nincs)
- Attribútumok
- Metódusok

```
+ ComponentView createView(ComponentViewCreator cvc):
```

11.3.15. Node (vált.)

- Felelősség
  Csomópont elem. Az egyetlen bemenetére kötött értéket kiadja az összes kimeneti lábán.
- Ősosztályok Object → AbstractComponent → Node.
- Interfészek (nincs)
- Attribútumok
  - (nincs)
- Metódusok

```
+ ComponentView createView(ComponentViewCreator cvc):
```

11.3.16. OrGate (vált.)

- Felelősség
  VAGY kapu, az áramkör egyik alapeleme. Bemenetein lévő értékek logikai VAGY kapcsolatát valósítja meg, amit a kimenetén kiad.
- Ősosztályok Object → AbstractComponent → OrGate.
- Interfészek (nincs)
- Attribútumok
  - (nincs)
- Metódusok

```
+ ComponentView createView(ComponentViewCreator cvc):
```

11.3.17. Scope (vált.)

- Felelősség
  Egy oszcilloszkópot reprezentál. Eltárolt értékek egy sorba kerülnek bele, mely fix méretű.
- Ősosztályok Object → AbstractComponent → DisplayComponent → Led → Scope.
- Interfészek (nincs)
- Attribútumok
- Metódusok

```
+ ComponentView createView(ComponentViewCreator cvc):
```

11.3.18. SequenceGenerator (vált.)

- Felelősség
  Jelgenerátort reprezentál, amely a beállított bitsorozatot adja ki. Alapértelmezetten (amíg a felhasználó nem állítja be, vagy tölt be másikat) a 0,1-es szekvenciát tárolja.
- Ősosztályok Object → AbstractComponent → SourceComponent → SequenceGenerator.
- Interfészek (nincs)
- Attribútumok
- Metódusok

```
+ ComponentView createView(ComponentViewCreator cvc):
```

11.3.19. SevenSegmentDisplay (vált.)

- Felelősség
  7-szegmenses kijelzőt reprezentál, melynek 7 bemenete vezérli a megfelelő szegmenseket, ezek világítanak, ha az adott bemenetre logikai igaz van kötve.
- Ősosztályok Object → AbstractComponent → DisplayComponent → SevenSegmentDisplay.
- Interfészek (nincs)
- Attribútumok
  - (nincs)
- Metódusok

```
+ ComponentView createView(ComponentViewCreator cvc):
```

11.3.20. Toggle (vált.)

- Felelősség
  Kapcsoló jelforrás, melyet a felhasználó szimuláció közben kapcsolgathat.
- Ősosztályok Object → AbstractComponent → SourceComponent → Toggle.
- Interfészek (nincs)
- Attribútumok
- Metódusok

  `+ ComponentView createView(ComponentViewCreator cvc):`

11.3.21. Vcc (vált.)

- Felelősség
  A tápfeszültés komponens, ami konstans igaz értéket ad. Nincs bemenete.
- Ősosztályok Object → AbstractComponent → Vcc.
- Interfészek (nincs)
- Attribútumok
  - (nincs)
- Metódusok

  `+ ComponentView createView(ComponentViewCreator cvc):`

11.3.22. CircuitView

- Felelősség
  Áramkört kirajzoló panel.
- Ősosztályok JPanel → CircuitView.
- Interfészek MouseListener.
- Attribútumok
  - `List drawables`: kirajzolandók listája
  - `FrameView parent`: főablak
  - `Map positions`: kirajzolandók pozíciója
- Metódusok
  - `+ CircuitView()`: Áramkört kirajzoló panel
  - `+ void mouseClicked(MouseEvent me)`: Egérkattintás kezelő
  - `+ void paint(Graphics g)`: Áramkör kirajzolása
  - `+ void refresh()`: Áramkör újrarajzolása.
  - `+ void setParent(FrameView parent)`: Szülő beállítása
  - `+ void updateDrawables(List drawables, Map coords)`: Kirajzolandó objektumok és koordinátáik beállítása

11.3.23. Drawable

Interfész.

- Felelősség
  Áramköri panelre rajzolható objektum.
- Ősosztályok Drawable.
- Interfészek (nincs)
- Metódusok

  + `void draw(Graphics g)`: Kirajzolási logika

  + `Dimension getDimension()`: Lekérhetjük az objektumtól a méretét, ha beszélhetünk ilyenről.

  + `void onClick(Controller controller)`: Komponensre kapcsolás logikája (visszahívhat a vezérlőre)

11.3.24. Frame

- Felelősség
  Alkalmazás főablaka. Ő tartalmazza a CircuitView-t és a menüsort valamint a gombokat.
- Ősosztályok JFrame → Frame.
- Interfészek ActionListener, FrameView.
- Attribútumok

  − `CircuitView circuitView`

  − `Controller controller`: vezérlő

  − `Timer t`: időzítő

- Metódusok

  + `Frame(Controller controller)`: Kontruktor

  − `void aboutCloseBtnActionPerformed(ActionEvent evt)`: Névjegy ablak bezárása

  − `void aboutMIActionPerformed(ActionEvent evt)`: Névjegy menüpont eseményvezérlője

  + `void actionPerformed(ActionEvent e)`: Timer tick eventje

  − `void closeDetailedBTNActionPerformed(ActionEvent evt)`: Komponens részletei ablak bezárása

  + `void drawCircuit()`: Áramkör kirajzolása

  − `void exitMIActionPerformed(ActionEvent evt)`: Kilépés menüpont eseményvezérlője

  + `Controller getController()`: Lekérdezhető a vezérlő

  − `void loadCircuitMIActionPerformed(ActionEvent evt)`: Áramkör betöltése menüpont eseményvezérlője

  − `void loadConfigMIActionPerformed(ActionEvent evt)`: Konfig fájl betöltése menüpont eseményvezérlője

  + `void makeItVisible()`: Megjelenítés

  + `void onFailedSimulation()`: Áramkör szimulációja nem sikerült

+ `void onSuccessfulSimulation()`: Áramkör szimulációja sikeres

− `void saveConfigMIActionPerformed(ActionEvent evt)`: Konfig fájl mentése menüpont eseményvezérlője

− `void saveSeqBTNActionPerformed(ActionEvent evt)`: Új szekvencia elmentése

+ `void setDrawables(List drawables, Map positions)`: Megjelenítendő objektumok és koordinátáik átadása a megjelenítőnek

+ `void setPeriod(int pt)`: Szimuláció sebességének beállítása

+ `void showDetails(AbstractComponent ac)`: Általános komponens részleteinek megjelenítése

+ `void showDetails(Scope s)`: Scope részleteinek megjelenítése

+ `void showDetails(SequenceGenerator sg)`: Szekvenciagenerátor részleteinek megjelenítése

− `void simSpeedSaveBtnActionPerformed(ActionEvent evt)`: Új sebesség mentése

− `void simulationDelayActionPerformed(ActionEvent evt)`: Szimuláció sebességének beállítására szolgáló ablak megjelenítése

− `void StartStopActionPerformed(ActionEvent evt)`: Szimuláció start/stop

− `void stepBtnActionPerformed(ActionEvent evt)`: Léptetés gomb eseményvezérlője

11.3.25. FrameView

Interfész.

- Felelősség
  Főablak interfésze
- Ősosztályok FrameView.
- Interfészek (nincs)
- Metódusok

  + `void drawCircuit()`: Kirajzoljuk az áramkört.

  + `Controller getController()`: Lekérdezzük a vezérlőt

  + `void makeItVisible()`: Itt kell megadni, hogy a főablak, hogy tehető láthatóvá.

  + `void onFailedSimulation()`: Itt adható meg, hogy mi történjen, ha nem stabil az áramkör

  + `void onSuccessfulSimulation()`: Itt adható meg, hogy mi történjen, ha sikeres egy szimulációs lépés

  + `void setDrawables(List drawables, Map positions)`: Beállítjuk a kirajzolandó objektumokat és azok pozícióját.

  + `void setPeriod(int pt)`: Szimuláció sebességének beállítása

  + `void showDetails(AbstractComponent ac)`: Általános komponens részleteinek megjelenítése

  + `void showDetails(Scope s)`: Scope részleteinek megjelenítése

  + `void showDetails(SequenceGenerator sg)`: Szekvenciagenerátor részleteinek megjelenítése

11.3.26. ComponentView

Absztrakt osztály.

- Felelősség
  A kirajzoló wrapper objektumok ősosztálya. Itt irhatjuk le a közös kirajzoló logikákat (pl. kábelek pinjei).
- Ősosztályok Object → ComponentView.
- Interfészek Drawable.
- Attribútumok
  - `Dimension dimension` Szélesség-magasság
- Metódusok
  + `ComponentView(int w, int h)`: Konstruktor a méretek megadásával.
  + `void draw(Graphics g)`: Kirajzolási logika
  + `Dimension getDimension()`: Lekérhetjük az objektumtól a méretét.
  # `abstract int getInputPinsCount()`: Bemeneti pinek száma.
  # `abstract int getOutputPinsCount()`: Kimeneti pinek száma
  + `Point getRelativeInputPinPosition(int pin)`: Visszaadja a bemeneti pin relatív pozícióját.
  + `Point getRelativeOutputPinPosition(int pin)`: Visszaadja a kimeneti pin relatív pozícióját.
  # `abstract void onDraw(Graphics g)`: Komponens kirajzolásának egyedi logikája.

11.3.27. WireView

- Felelősség
  Egy vezeték megjelenítéséért felelős, amit törött vonallal jelenítünk meg.
- Ősosztályok Object → WireView.
- Interfészek Drawable.
- Attribútumok
  - `Point end` Vezeték vége
  - `List referencePoints` Vezeték referenciapontjai, ahol a vezeték "törik".
  - `Point start` Vezeték kezdete
  - `Wire w` Vezeték, aminek a megjelenítéséért felel.
- Metódusok
  + `WireView(Wire w, Point start, Point end)`: Konstruktor
  + `void draw(Graphics g)`: Kirajzolási logika
  + `Dimension getDimension()`:
  + `void setReferencePoints(List referencePoints)`: Vezeték referenciapontjainak a beállítása

11.3.28. AndGateView

- Felelősség
  ÉS kaput kirajzoló osztály
- Ősosztályok Object → ComponentView → AndGateView.
- Interfészek (nincs)
- Attribútumok
  - `AndGate ag` Becsomagolt ÉS kapu
- Metódusok
  - `+ AndGateView(AndGate ag)`: Konstruktor
  - `# int getInputPinsCount()`: Bemeneti pinek száma
  - `# int getOutputPinsCount()`: Kimeneti pinek száma
  - `+ void onClick(Controller controller)`: ÉS kapura kattintás
  - `# void onDraw(Graphics g)`: Kirajzolási logika

11.3.29. CompositeView

- Felelősség
  Kompozitot kirajzoló osztály
- Ősosztályok Object → ComponentView → CompositeView.
- Interfészek (nincs)
- Attribútumok
  - `Composite c` Becsomagolt kompozit
- Metódusok
  - `+ CompositeView(Composite c)`: Konstruktor
  - `# int getInputPinsCount()`: Bemeneti pinek száma
  - `# int getOutputPinsCount()`: Kimeneti pinek száma
  - `+ void onClick(Controller controller)`: Komponensre kapcsolás
  - `# void onDraw(Graphics g)`: Kirajzolási logika

11.3.30. FlipFlopDView

- Felelősség
  D flip-flopot kirajzoló osztály
- Ősosztályok Object → ComponentView → FlipFlopDView.
- Interfészek (nincs)
- Attribútumok
  - `FlipFlopD d` Becsomagolt D flip-flop
- Metódusok
  - `+ FlipFlopDView(FlipFlopD d)`: Konstruktor
  - `# int getInputPinsCount()`: Bemeneti pinek száma

`int getOutputPinsCount()`: Kimeneti pinek száma

\+ `void onClick(Controller controller)`: D flip-flopra kattintás

`void onDraw(Graphics g)`: Kirajzolási logika

11.3.31. FlipFlopJKView

- Felelősség
  JK flip-flopot kirajzoló osztály
- Ősosztályok Object → ComponentView → FlipFlopJKView.
- Interfészek (nincs)
- Attribútumok

  − `FlipFlopJK jk` Becsomagolt JK flip-flop
- Metódusok

  \+ `FlipFlopJKView(FlipFlopJK jk)`: Konstruktor

  # `int getInputPinsCount()`: Bemeneti pinek száma

  # `int getOutputPinsCount()`: Kimeneti pinek száma

  \+ `void onClick(Controller controller)`: JK flip-flopra kattintás

  # `void onDraw(Graphics g)`: Kirajzolási logika

11.3.32. GndView

- Felelősség
  GND-t kirajzoló osztály
- Ősosztályok Object → ComponentView → GndView.
- Interfészek (nincs)
- Attribútumok
- Metódusok

  \+ `GndView()`: Konstruktor

  # `int getInputPinsCount()`: Bemeneti pinek száma

  # `int getOutputPinsCount()`: Kimeneti pinek száma

  \+ `void onClick(Controller controller)`: Komponensre kapcsolás

  # `void onDraw(Graphics g)`: Kirajzolási logika

11.3.33. InverterView

- Felelősség
  Invertert kirajzoló osztály
- Ősosztályok Object → ComponentView → InverterView.
- Interfészek (nincs)
- Attribútumok

  − `Inverter inv` Becsomagolt inverter

- Metódusok
  - `+ InverterView(Inverter inv)`: Konstruktor
  - `# int getInputPinsCount()`: Bemeneti pinek száma
  - `# int getOutputPinsCount()`: Kimeneti pinek száma
  - `+ void onClick(Controller controller)`: Komponensre kapcsolás
  - `# void onDraw(Graphics g)`: Kirajzolási logika

11.3.34. LedView

- Felelősség
  LED-et kirajzoló osztály.
- Ősosztályok Object → ComponentView → LedView.
- Interfészek (nincs)
- Attribútumok
  - `- Led led` Becsomagolt Led.
- Metódusok
  - `+ LedView(Led led)`: Konstruktor
  - `# int getInputPinsCount()`: Bemeneti pinek száma
  - `# int getOutputPinsCount()`: Kimeneti pinek száma
  - `+ void onClick(Controller controller)`: Komponensre kacsolás
  - `# void onDraw(Graphics g)`: Kirajzolási logika

11.3.35. MpxView

- Felelősség
  Multiplexert kirajzoló osztály
- Ősosztályok Object → ComponentView → MpxView.
- Interfészek (nincs)
- Attribútumok
  - `- Mpx mpx` Becsomagolt multiplexer.
- Metódusok
  - `+ MpxView(Mpx mpx)`: Konstruktor
  - `# int getInputPinsCount()`: Bemeneti pinek száma
  - `# int getOutputPinsCount()`: Kimeneti pinek száma
  - `+ void onClick(Controller controller)`: Komponensre kapcsolás
  - `# void onDraw(Graphics g)`: Kirajzolási logika

11.3.36. NodeView

- Felelősség
  Node-ot kirajzoló osztály.
- Ősosztályok Object → ComponentView → NodeView.
- Interfészek (nincs)
- Attribútumok
  - `Node node` Becsomagolt csomópont
- Metódusok
  + `NodeView(Node node)`: Konstruktor
  + `void draw(Graphics g)`: Kirajzolási logika
  # `int getInputPinsCount()`: Bemeneti pinek száma
  # `int getOutputPinsCount()`: Kimeneti pinek száma
  + `Point getRelativeInputPinPosition(int pin)`: Megadott bemeneti pin relatív pozicióját adja vissza
  + `Point getRelativeOutputPinPosition(int pin)`: Megadott kimeneti pin relatív pozicióját adja vissza
  + `void onClick(Controller controller)`: Komponensre kapcsolás
  # `void onDraw(Graphics g)`: Kirajzolási logika

11.3.37. OrGateView

- Felelősség
  VAGY kaput kirajzoló osztály
- Ősosztályok Object → ComponentView → OrGateView.
- Interfészek (nincs)
- Attribútumok
  - `OrGate og` Becsomagolt VAGY kapu
- Metódusok
  + `OrGateView(OrGate og)`: Konstruktor
  # `int getInputPinsCount()`: Bemeneti pinek száma
  # `int getOutputPinsCount()`: Kimeneti pinek száma
  + `void onClick(Controller controller)`: Komponensre kapcsolás
  # `void onDraw(Graphics g)`: Kirajzolási logika

11.3.38. ScopeView

- Felelősség
  Scope-ot kirajzoló osztály.
- Ősosztályok Object → ComponentView → ScopeView.
- Interfészek (nincs)
- Attribútumok

- `Scope scope` Becsomagolt oszcilloszkóp

- Metódusok
  - `+ ScopeView(Scope scope)`: Konstruktor
  - `# int getInputPinsCount()`: Bemeneti pinek száma
  - `# int getOutputPinsCount()`: Kimeneti pinek száma
  - `+ void onClick(Controller controller)`: Komponensre kacsolás
  - `# void onDraw(Graphics g)`: Kirajzolási logika

11.3.39. SequenceGeneratorView

- Felelősség
  Jelgenerátort kirajzoló osztály
- Ősosztályok Object → ComponentView → SequenceGeneratorView.
- Interfészek (nincs)
- Attribútumok
  - `SequenceGenerator sg` Becsomagolt szekvenciagenerátor
- Metódusok
  - `+ SequenceGeneratorView(SequenceGenerator sg)`: Konstruktor
  - `# int getInputPinsCount()`: Bemeneti pinek száma
  - `# int getOutputPinsCount()`: Kimeneti pinek száma
  - `+ void onClick(Controller controller)`: Komponensre kapcsolás
  - `# void onDraw(Graphics g)`: Kirajzolási logika

11.3.40. SevenSegmentDisplayView

- Felelősség
  Hétszegmenses kijelzőt kirajzoló osztály.
- Ősosztályok Object → ComponentView → SevenSegmentDisplayView.
- Interfészek (nincs)
- Attribútumok
  - `SevenSegmentDisplay ssd` Becsomagolt 7-szegmenses kijelző
- Metódusok
  - `+ SevenSegmentDisplayView(SevenSegmentDisplay ssd)`: Konstriktor
  - `# int getInputPinsCount()`: Bemeneti pinek száma
  - `# int getOutputPinsCount()`: Kimeneti pinek száma
  - `+ void onClick(Controller controller)`: Komponensre kapcsolás
  - `# void onDraw(Graphics g)`: Kirajzolási logika

11.3.41. ToggleView

- Felelősség
  Kapcsolót kirajzoló osztály
- Ősosztályok Object → ComponentView → ToggleView.
- Interfészek (nincs)
- Attribútumok
  - `Toggle toggle` Becsomagolt kapcsoló
- Metódusok
  - `+ ToggleView(Toggle toggle)`: Konstruktor
  - `# int getInputPinsCount()`: Bemeneti pinek száma
  - `# int getOutputPinsCount()`: Kimeneti pinek száma
  - `+ void onClick(Controller controller)`: Komponensre kapcsolás
  - `# void onDraw(Graphics g)`: Kirajzolási logika

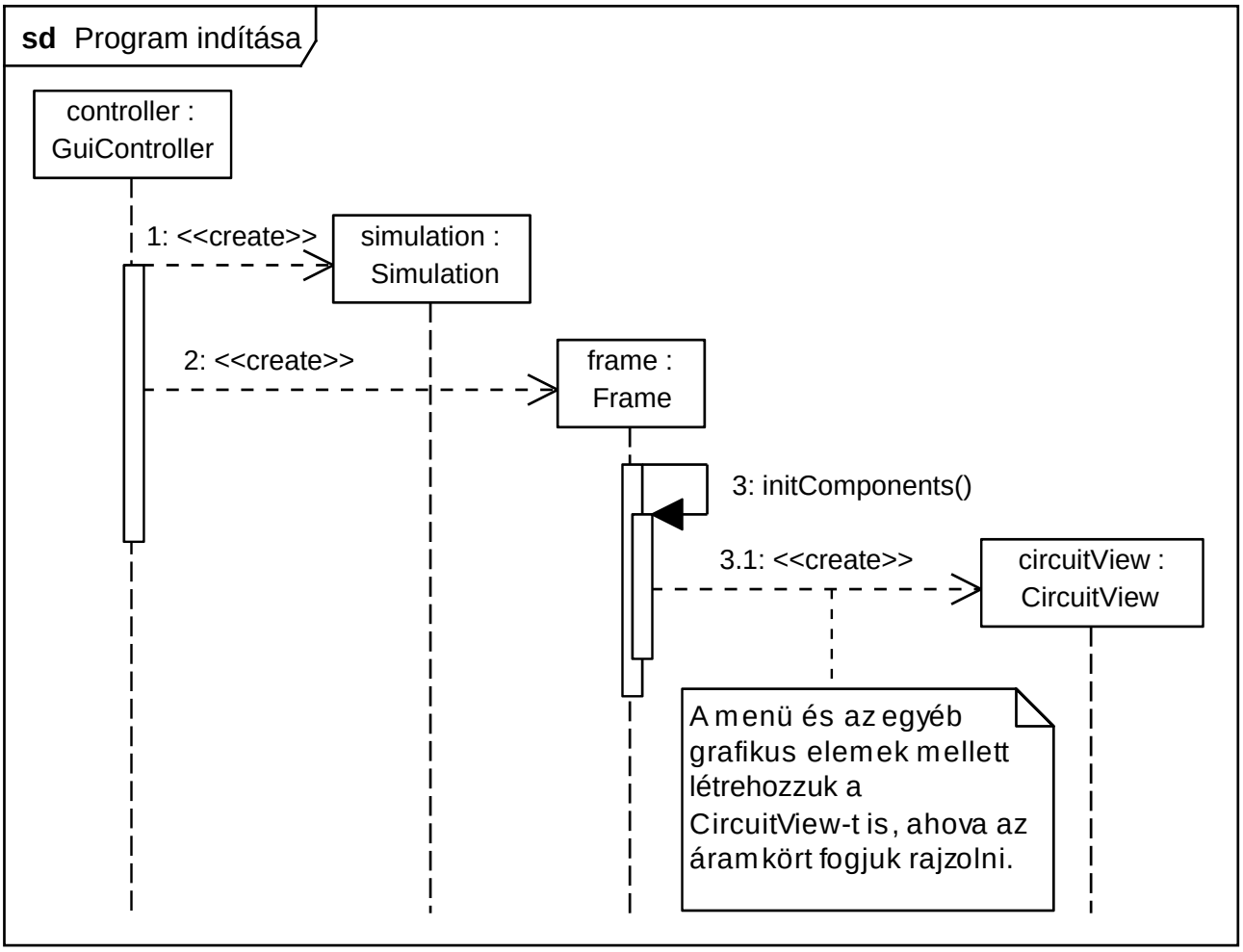
11.3.42. VccView

- Felelősség
  VCC-t kirajzoló osztály
- Ősosztályok Object → ComponentView → VccView.
- Interfészek (nincs)
- Attribútumok
- Metódusok
  - `+ VccView()`: Konstruktor
  - `# int getInputPinsCount()`: Bemneti pinek száma
  - `# int getOutputPinsCount()`: Kimeneti pinek száma
  - `+ void onClick(Controller controller)`: Komponensre kapcsolás
  - `# void onDraw(Graphics g)`: Kirajzolási logika

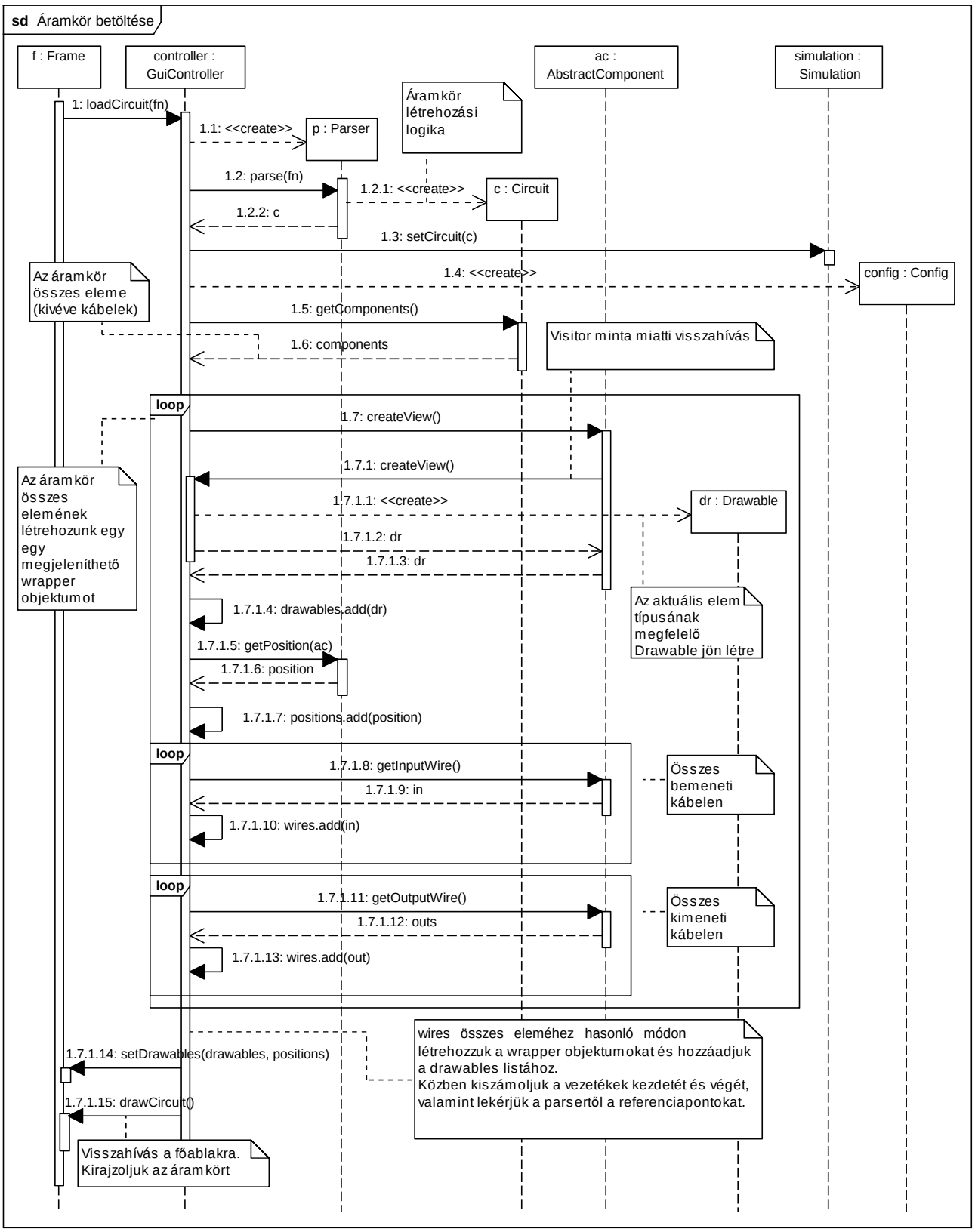
11.4. Kapcsolat az alkalmazói rendszerrel

11.5. Napló

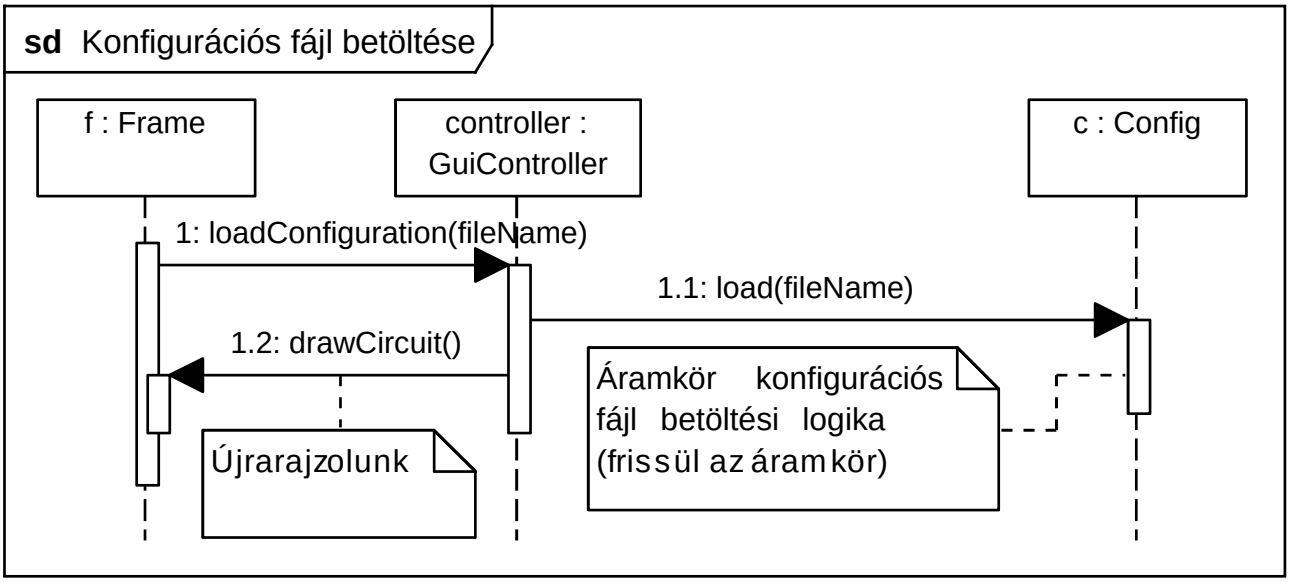
Kezdet	Időtartam	Résztvevők	Leírás
2011.04.20. 14:00	3 óra	**Dévényi A.** **Jákli G.** **Kriván B.**	Értekezlet. Megbeszéltük a grafikus felületet, osztályokat és a szekvenciákat. Döntés: szétosztottuk a diagramokat
2011.04.23. 11:30	45 perc	**Kriván B.**	A grafikus interfész c. fejezet elkészítése



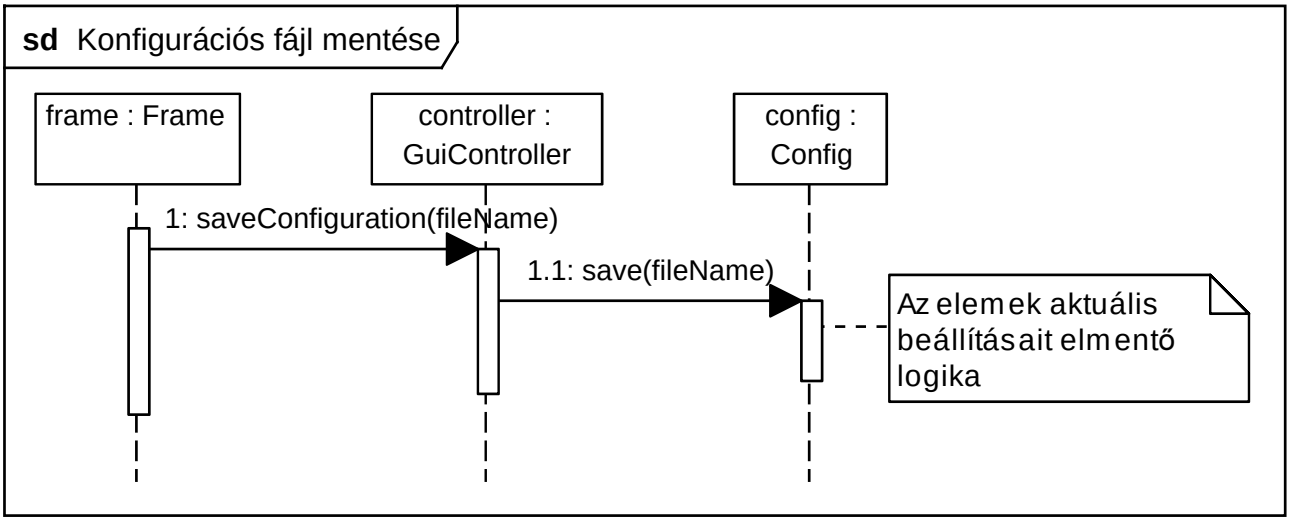
11.7. ábra. Program indítása



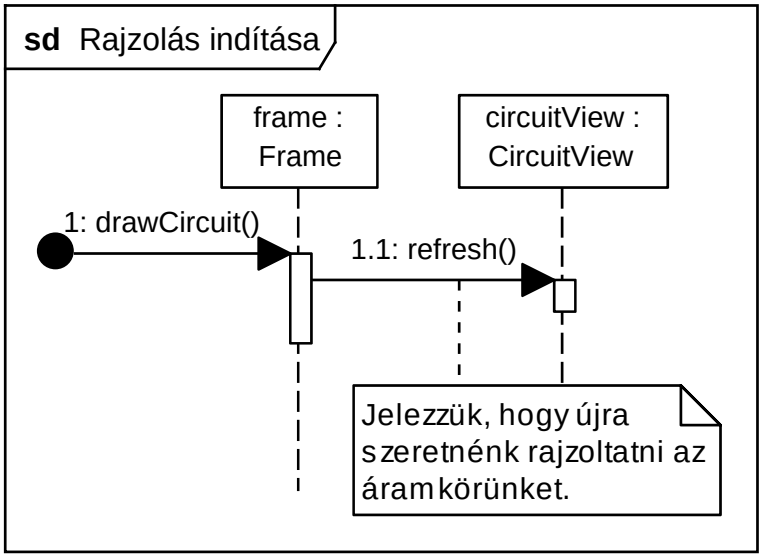
11.8. ábra. Áramkör betöltése



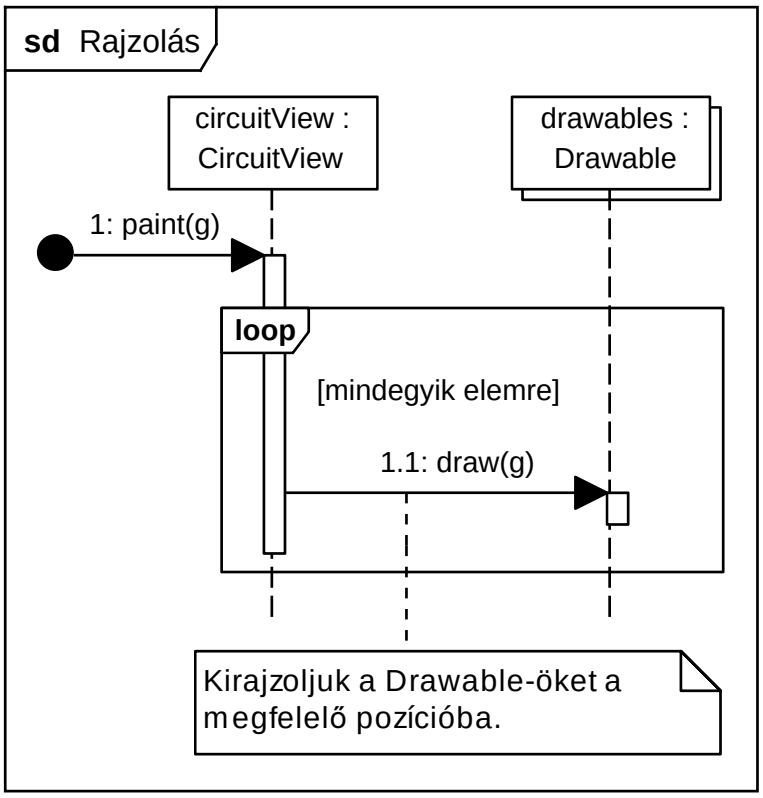
11.9. ábra. Konfigurációs fájl betöltése



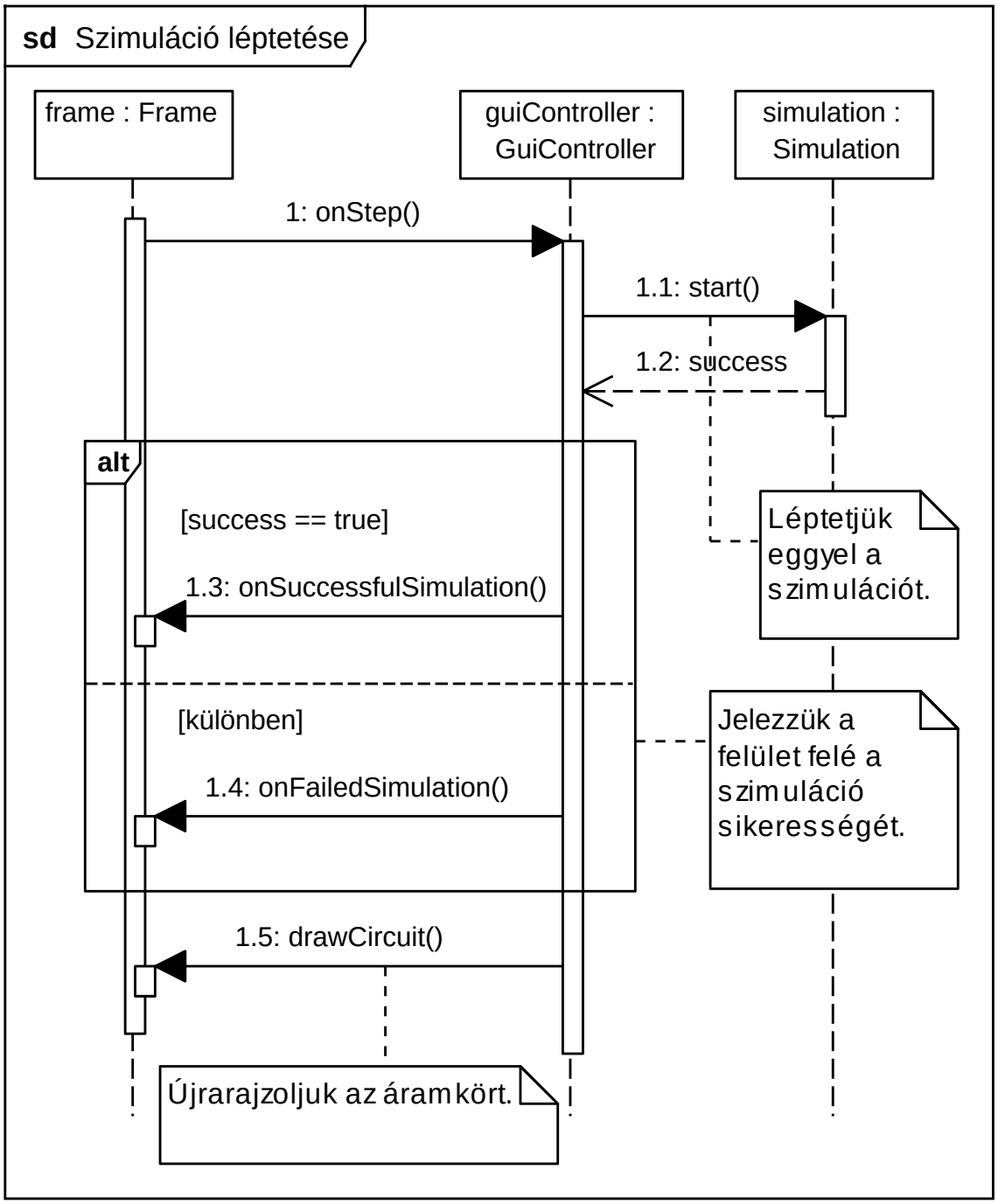
11.10. ábra. Konfigurációs fájl mentése



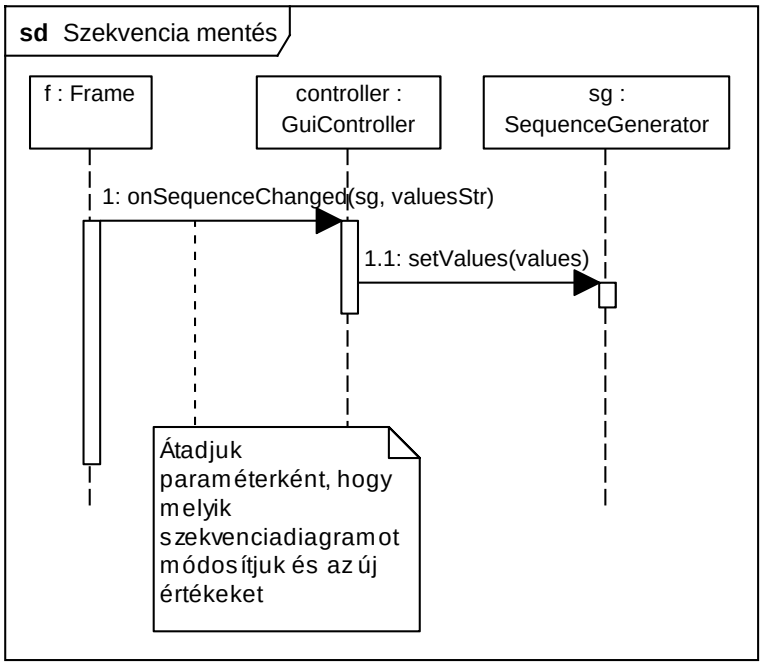
11.11. ábra. Rajzolás indítása



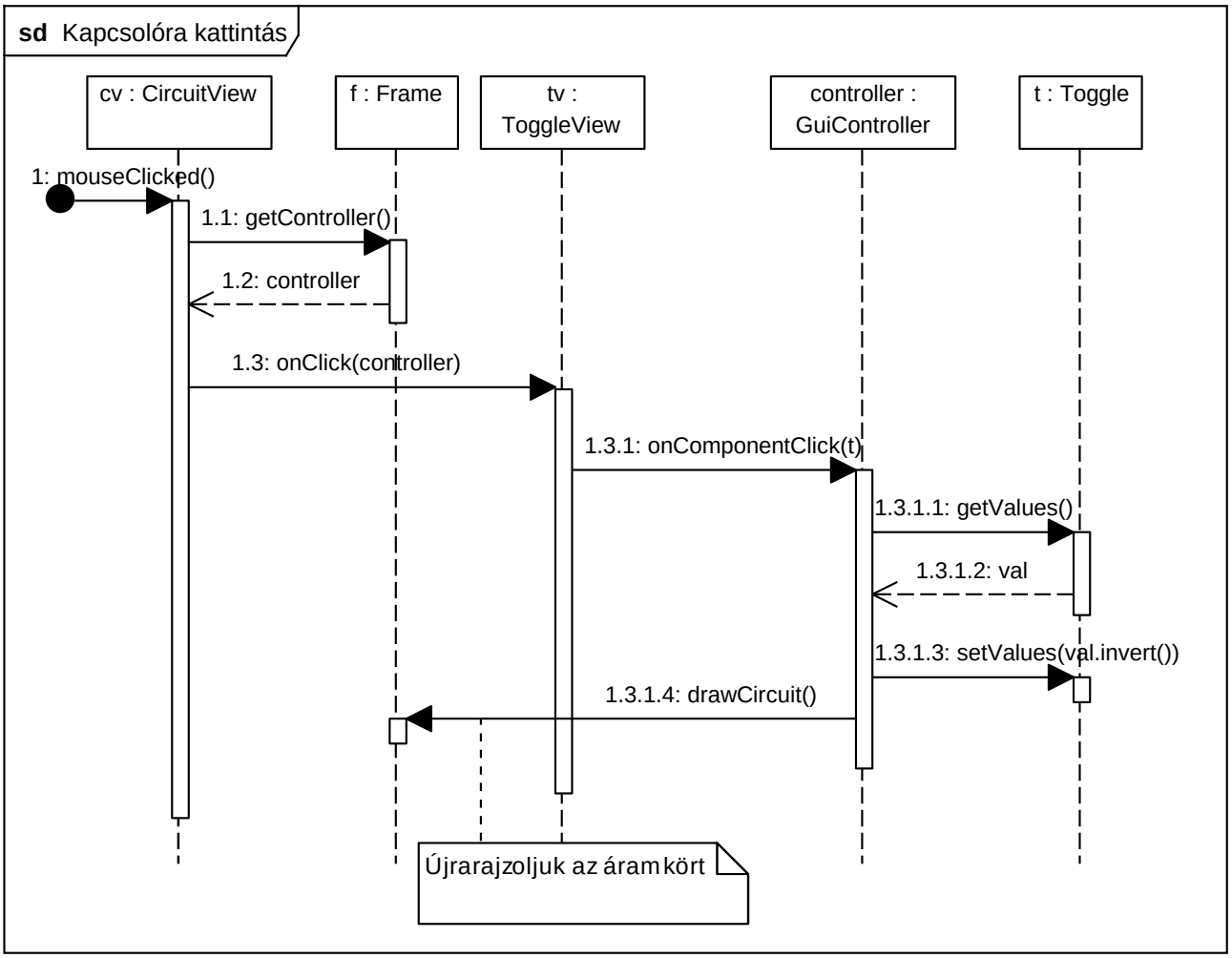
11.12. ábra. Rajzolás



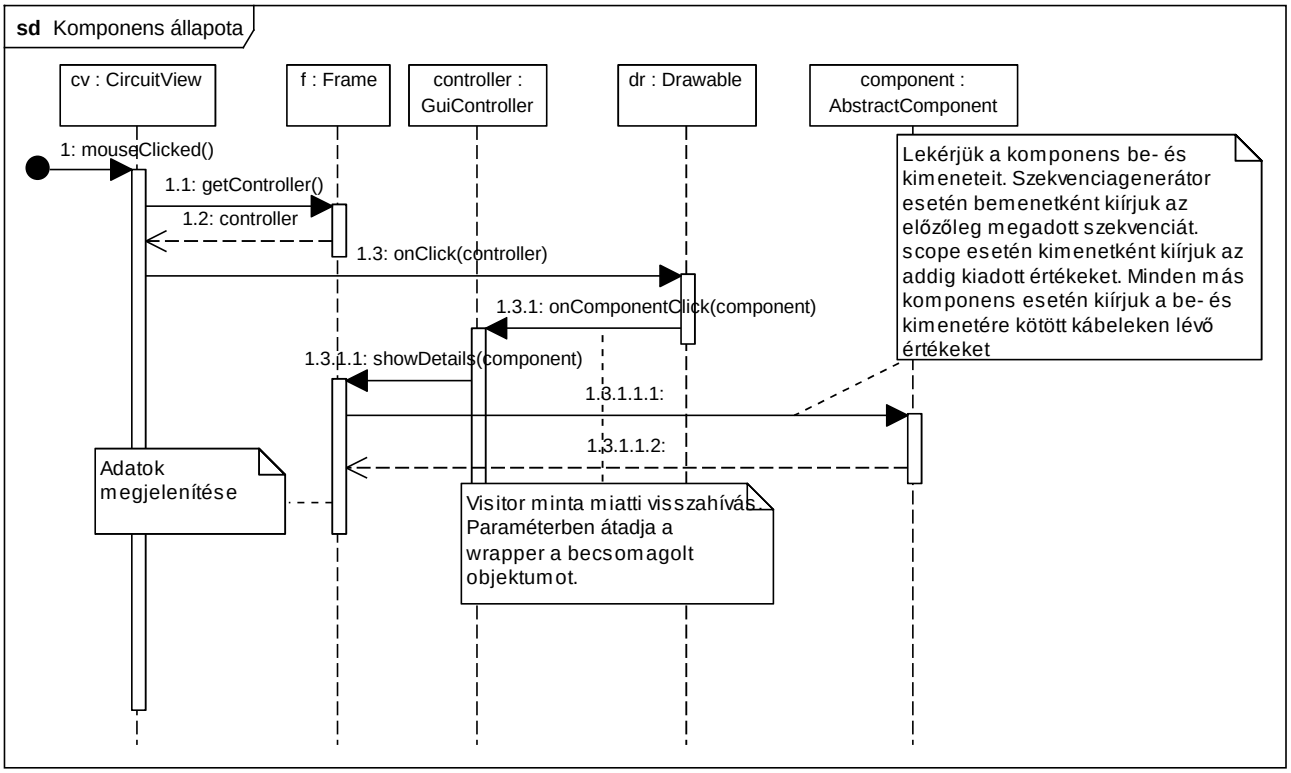
11.13. ábra. Szimuláció léptetése



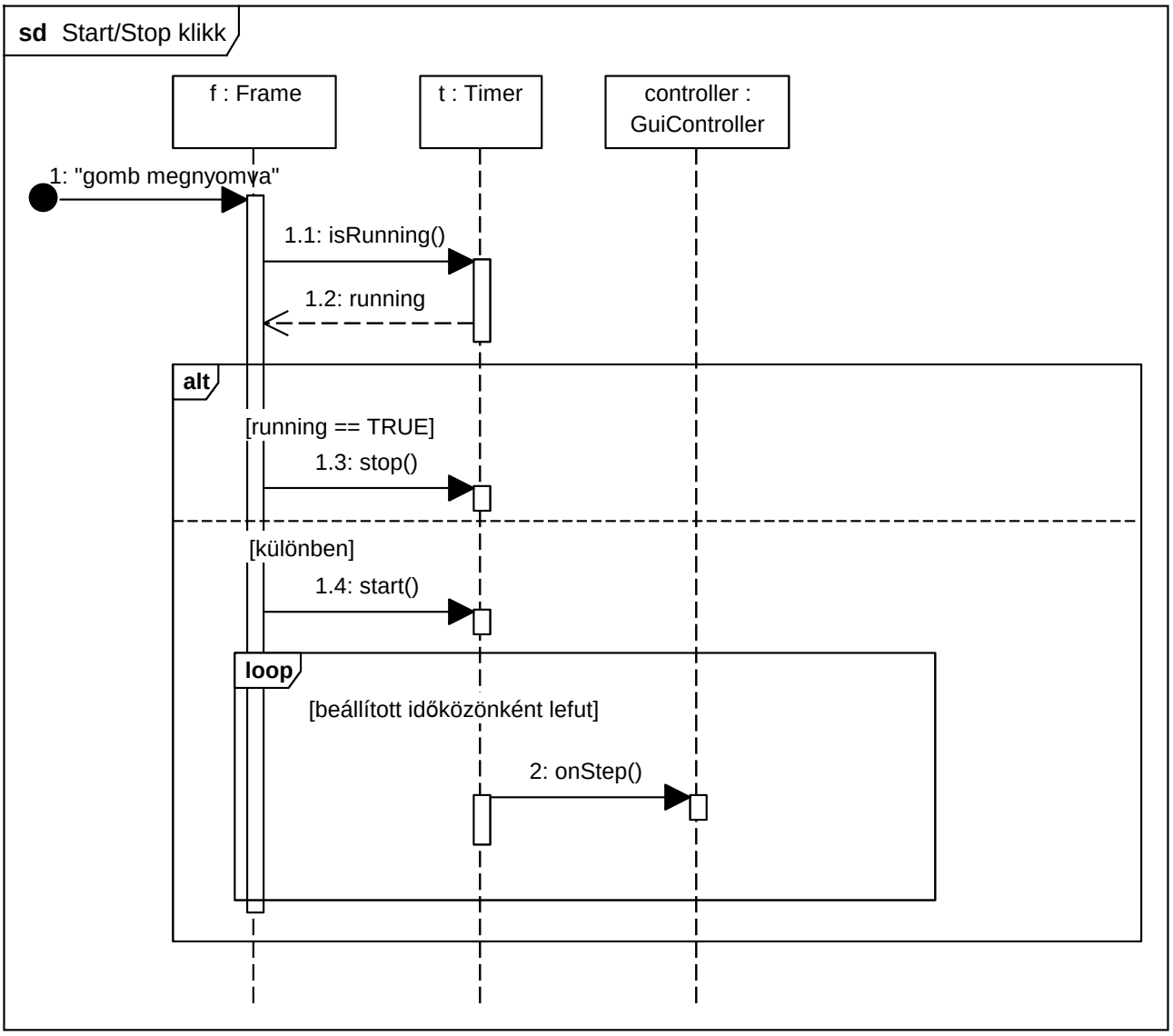
11.14. ábra. Szekvencia mentése



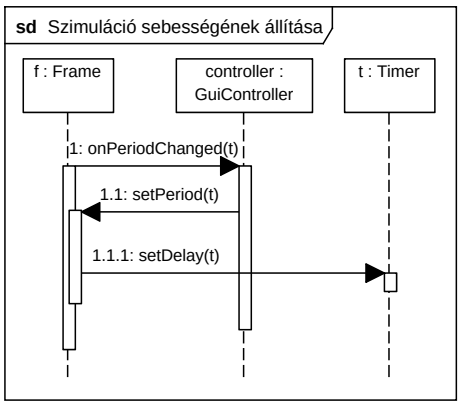
11.15. ábra. Kapcsolóra kattintás



11.16. ábra. Komponens állapotának kijelzése



11.17. ábra. Start/Stop klikk



11.18. ábra. Szimuláció sebességének állítása