Modellek programozott feldolgozása

Rendszertervezés laboratórium 1

Jegyzőkönyv

2021. tavasz

|  |  |
| --- | --- |
| Készítette | Somogyi Bence (Q79IBL) |
| Dátum | 2021. március 05. |
| GitHub hivatkozás | <https://github.com/Bence56/RETElab1MIT2> |
| Bónusz feladat | elkészült |

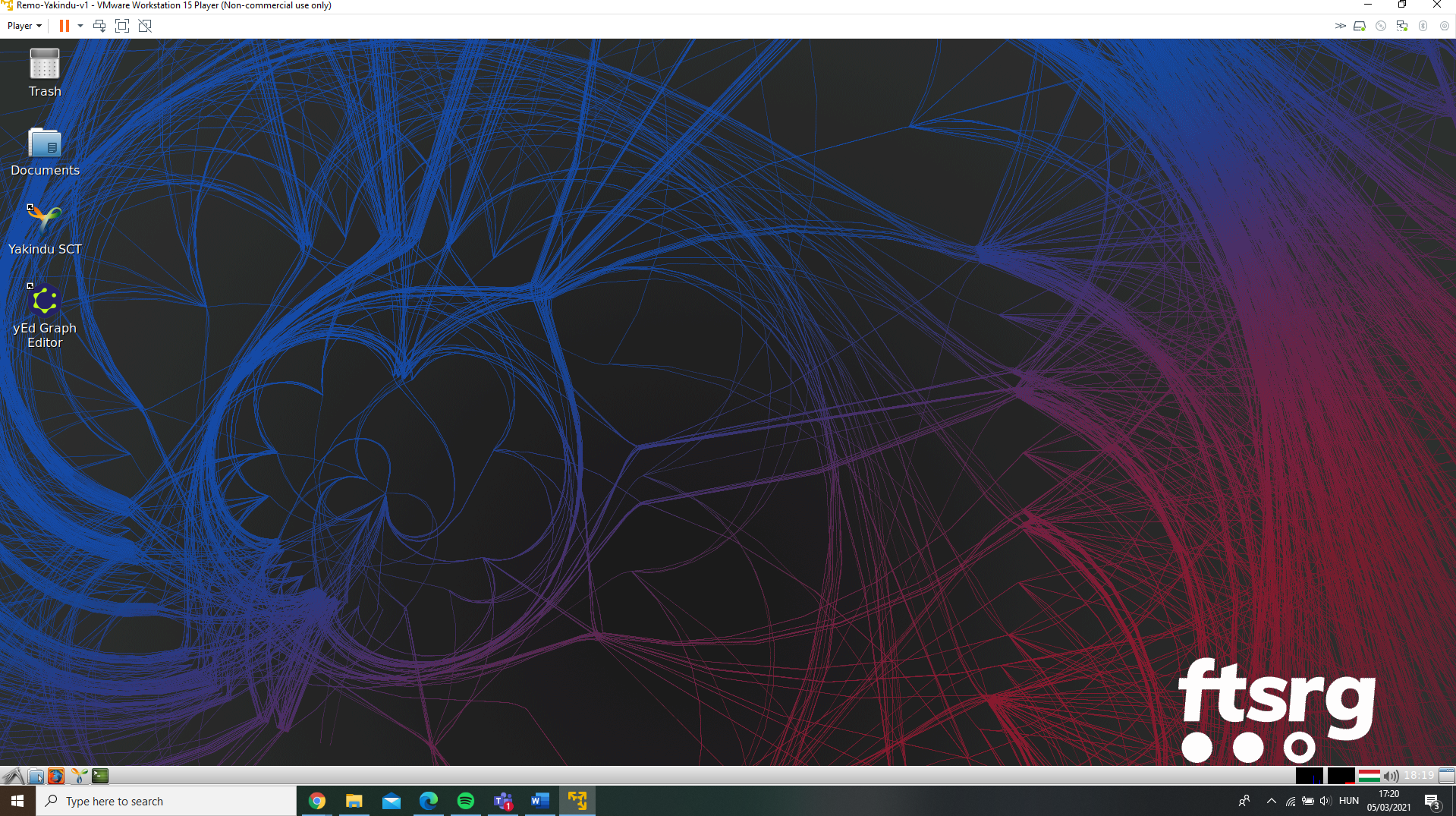
Budapesti Műszaki és Gazdaságtudományi Egyetem

Méréstechnika és Információs Rendszerek Tanszék

# Bevezetés

## Virtuális gép elérése

Letöltöttem a virtuális gép fileját és VMWare Player 15-öt használok a feladatok megoldására?

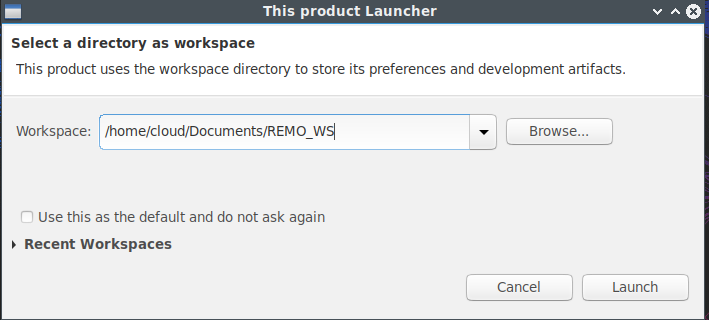


## Kiindulási fájlok letöltése

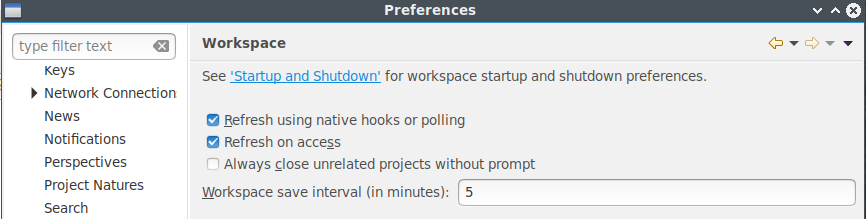
Volt telepítve egy tool már a VMWare playeremhez, ami megengedi, hogy a host gépről CTRL+C CTRL+V kombinációkkal fileokat másoljak a host gépről a guest gépre. Nem tudom miért, de a guest gép nem érte el az internetet, ezért így volt a legegyszerűbb átmásolni az állományokat.



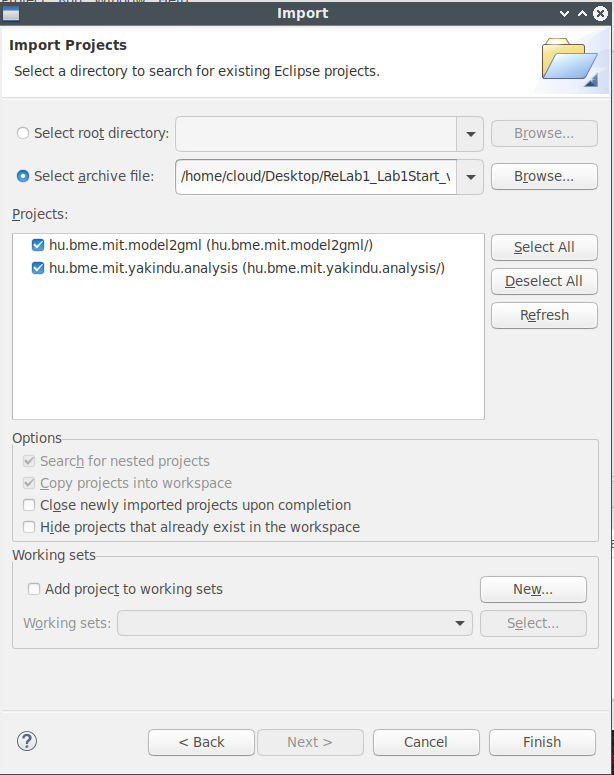
## Yakindu SCT megnyitása az alapértelmezett workspace-szel



## Tipp alkalmazása, az autómatikus frissítésre



## Projekt importálása

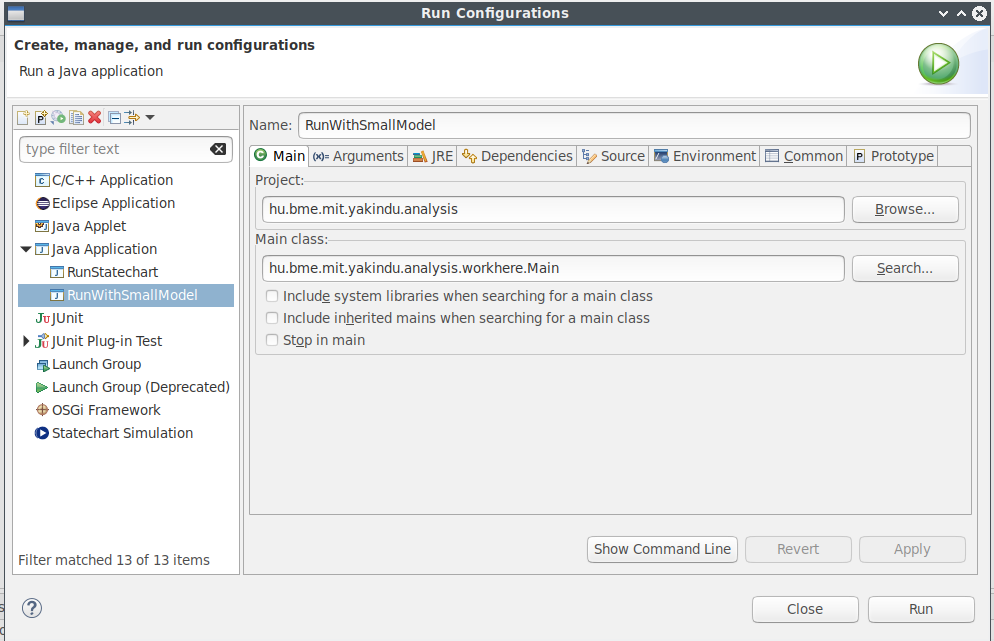


## Projekt átnézése

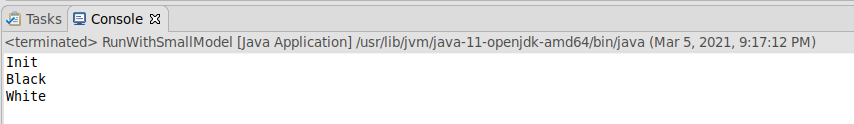
Elolvastam, megnéztem a leírtakat.

# Modell bejárása

## Futtatás RunWithSmallModel konfigurációval

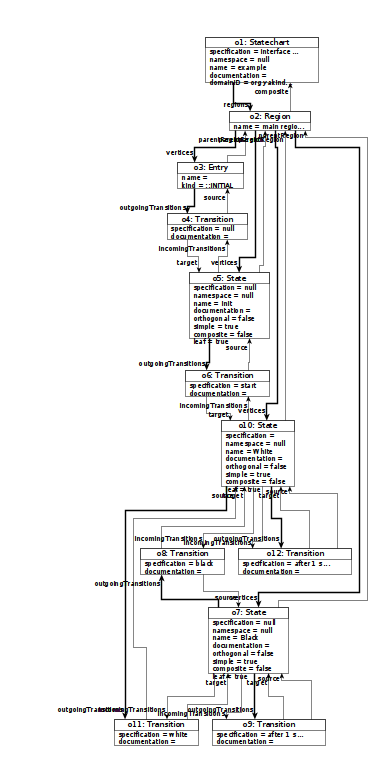


A kimenet a következő:



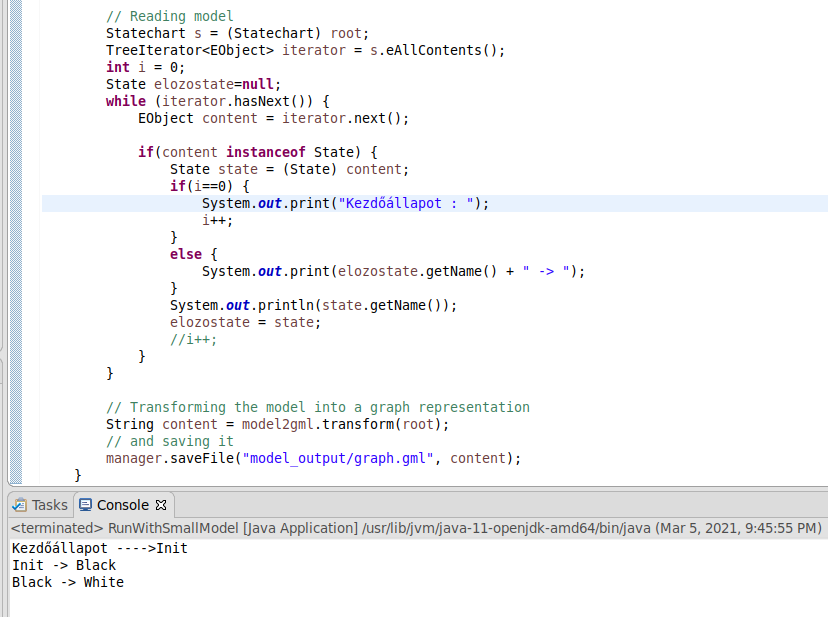
## Generált vizuális fájl megtekintése

A fájl yEd alkalmazással valló megnyitása után, alkalmaztam a Layout🡪 Hierarchical 🡪 Ok műveletsort. Sajnos a kép nem a legszebb a felbontás miatt. A vizuális modell a következő:



## Tranzíciók kiírása

A feladatot kiegészíteném annyival, hogy a kezdőállapot elött nincs állapot (Example állapotgép esetén Init állapot). Ezért az elé a következőt írom: „Kezdőállapot : ”. A kiegészített forráskód a következő képen látható.

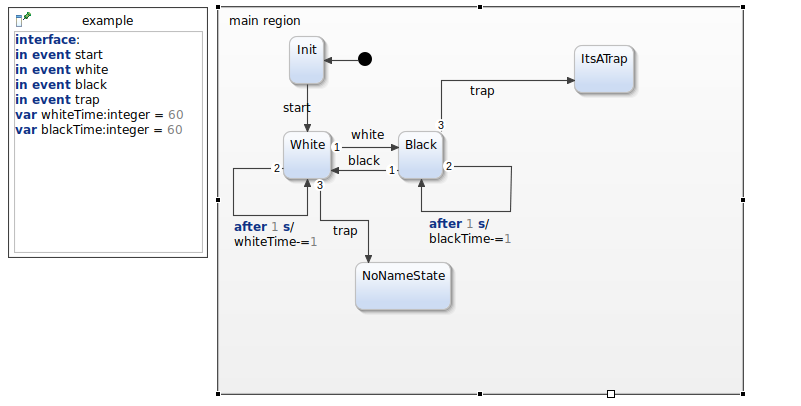


## Csapdaállapotok kiírása

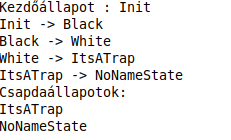
A kód amivel kiegészíthetem a következő képen látható. Sajnos nem fér el az egész képernyőre és csak a fontos, és új részeket szeretném mutatni. Ezen kívül egy listát csináltam a csapdaállapotoknak. Ez a kódban feljebb látható. Ennek a kódja: *ArrayList<State> csapdaallapotok = new ArrayList<State>();*



A módosított modell:



A kimenet a módosított modellre:

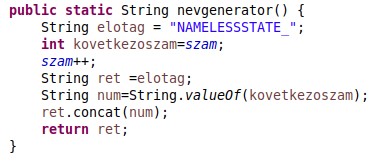


Mint az látható, nem az igazi tranzíció kiírás…

Tudom, hogy fa (gráf) bejárási algoritmust kellene implementálni, de úgy gondolom nem ez a labor célja. Idő hiányában nem oldom meg helyesen a feladatot.

## Névgenerálás névnélküli állapotoknak

Készítettem egy egyszerű függvényt, ami konkatenál egy állandó stringet és egy stringesített, statikus, állandóan változó(növekvő) számot. Elsőre, nem tűnik túl beszédesnek az előtag, de szerintem kellően felhívja a figyelmet a felhasználó számára, hogy nevezze át az állapotot.

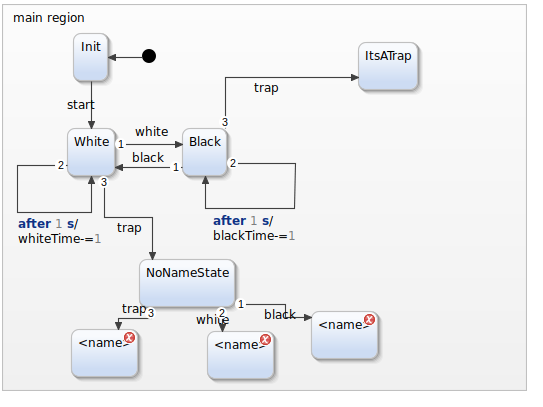


## Ellenőrzés, esetleges névegyezésre.

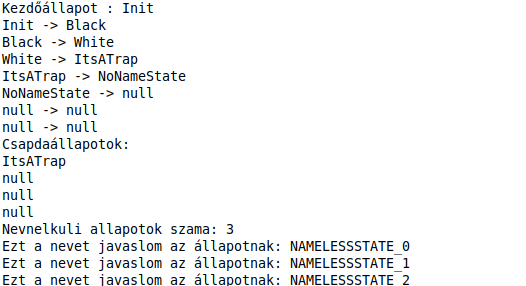
A következő képen látható a névegyezés ellenőrzése, és az átnevezés is. Ez igazából az előző feladathoz tartozik, de így több, releváns kód látható egy képen.



Módosítottam a modellt:



Azért a trap átmenetet használtam, mert nem akartam jobban komplikálni a modellt. A lényeges részek látszódnak a részfeladat szempontjából. A lefutás a modellre:

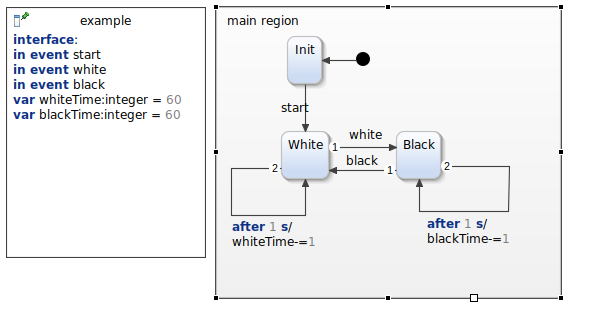


Sajnos mégjobban kijönnek a tranzíciós rész hibái ☹

# Yakindu kódgenerátor használata

## Modell visszaállítása

A modellt visszaállítottam:



## Kódgenerátor kiegészítése

A kód bemásolása után a következőket tudtam meg: A *GeneralFeatures* megengedi, hogy konfiguráljunk különböző szolgáltatásokat, amik az állapotgépen kívül különböző dolgokat generálnak. Alapvetően az összes érték *false*, azaz nem generál semmit. Opcionális funkciói:

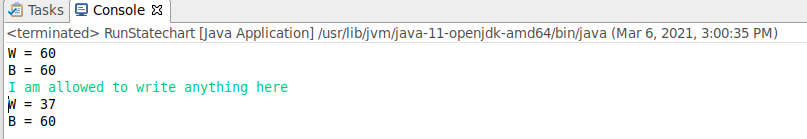
* *InterfaceObserverSupport:* Engedélyezi, vagy tiltja (nem engedélyezi) az állapotgép *ListenerInterface*-einek generálását.
* *RuntimeService:* Engefélyezi, vagy tiltja (nem engedélyezi) a futásidejű szolgáltatások generálását, ami *triggereli* az életciklus eseményeket az életciklus-alapú állapotgépekben.
* *TimerService:* Enegedélyezi, vagy tiltja(nem engedélyezi) az időzítéssel kapcsolatos események generálását, ami a *java.util.Timer* könyvtárat használja.

## RunStateChar.java file áttekintése

Miután átnéztem a kódot bemásoltam az importokat, és kikommenteztem a kódot.

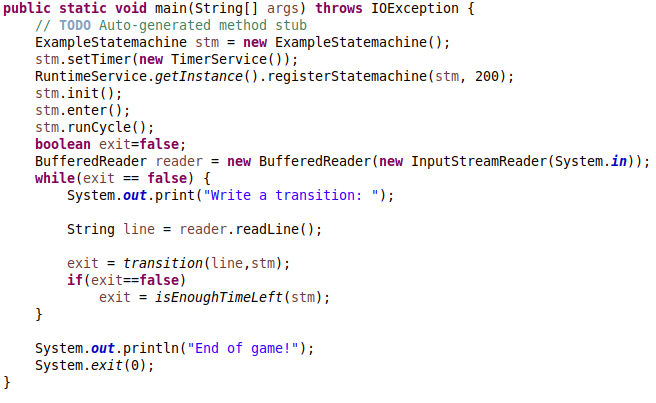
## Futtatás RunStatechart.launch konfigurációval

A futtatás kimenete:

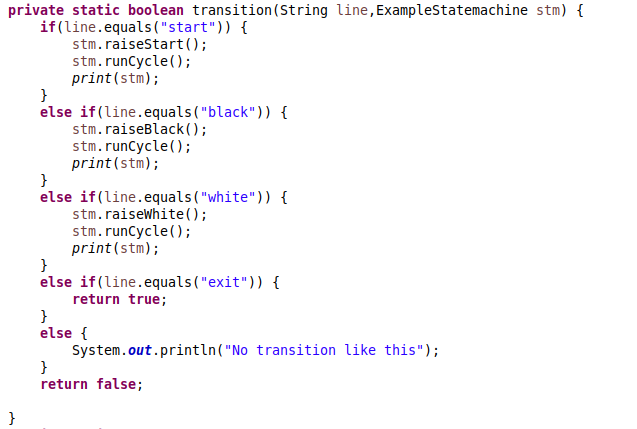


## Alkalmazás készítése

Az alkalmazást a *hu.bme.mit.yakindu.analysis* projektben, azon belül a *hu/bme/mit/yakindu/analysis/workhere* packageben készítettem el *ExcerciseToWriteAnApp* néven. A main függvényben inicializálom az állapotgépet, a kezdeti változókat, belépek a ciklusba. A tranzíció beolvasása után meghívom a *transition* függvényt, paraméterei az állapotgép és a beolvasot sor.

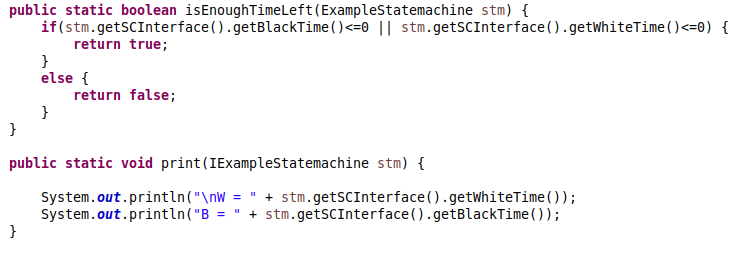


A függvény ellenőrzi, hogy melyik tranzíciót hívta tüzelte a felhasználó, azt tüzeli és meghívja a *runCycle()* függvényt. Ellenőrzi, hogy ki akar-e lépni a felhasználó az *exit* paranccsal. Ha ki akar lépni, akkor igaz ellenkező esetben hamis értékkel tér vissza. Ha nem talál ilyen parancsot a függvény, akkor hibaüzenetet küld, de nem dob kivételt.



A függvény amivel kiírom a játékosok hátralevő idejét a korábban tanulmányozott programból másoltam ki.

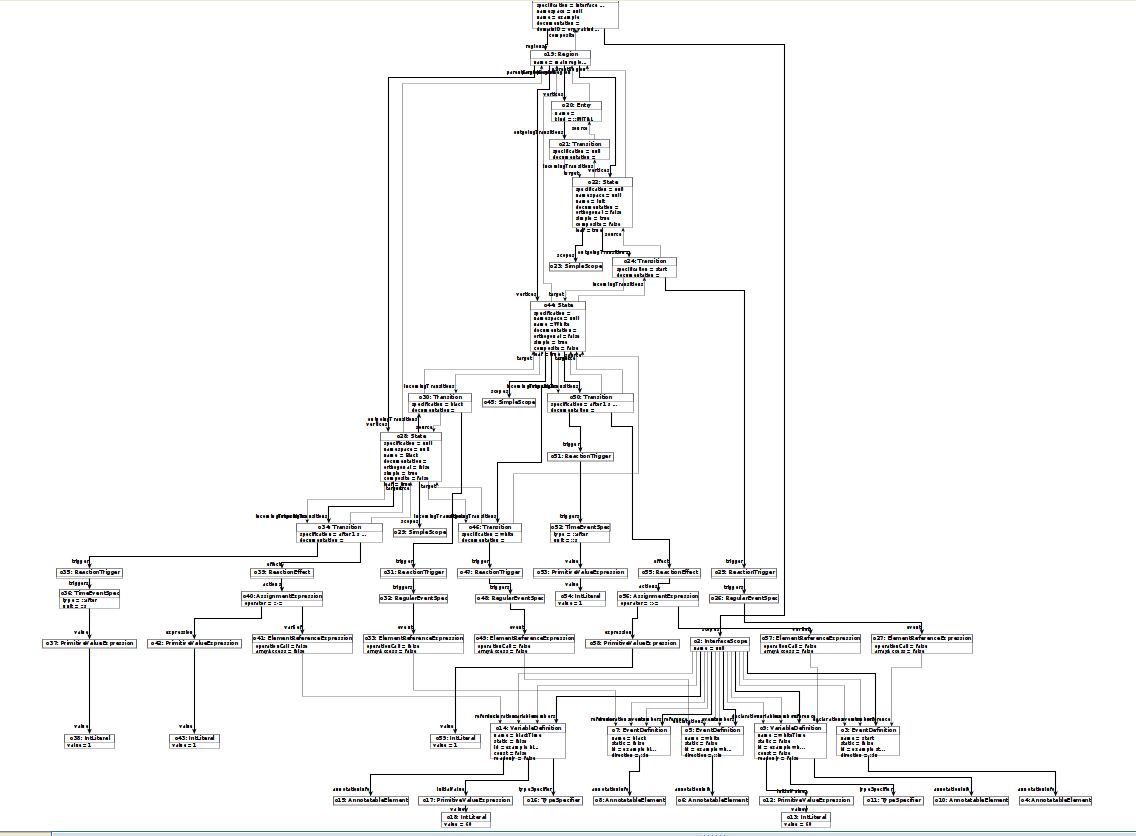
Végül fontosnak tartottam ellenőrizni, hogy vége van-e a játéknak. Nincs értelme bekérni tranzíciókat, ha valamelyik játékosnak elfogyott az ideje. A képen a kiírást végző függvény is látható.



# Saját kódgenerátor készítése

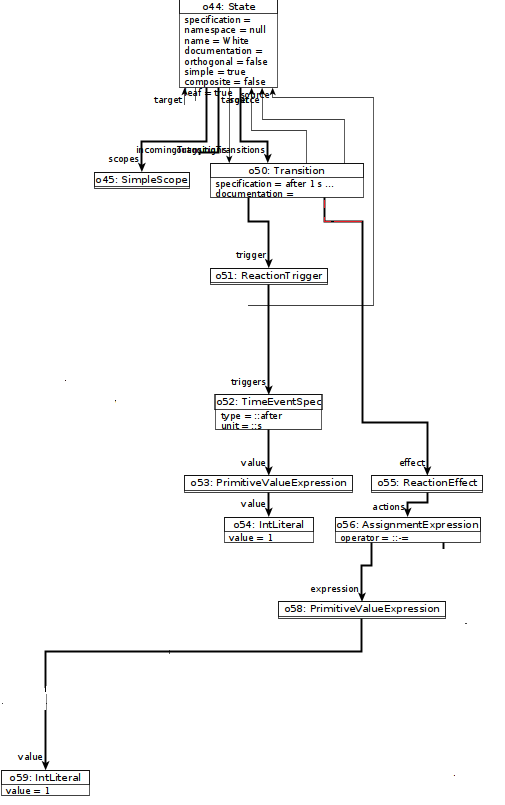
## Generált gráf áttekintése

A launch file első sorából kiderül, hogy ez egy JUnitos konfiguráció. A generált gráf a következő képen látható, sokkal bővebb mint az előzőleg generált gráf.



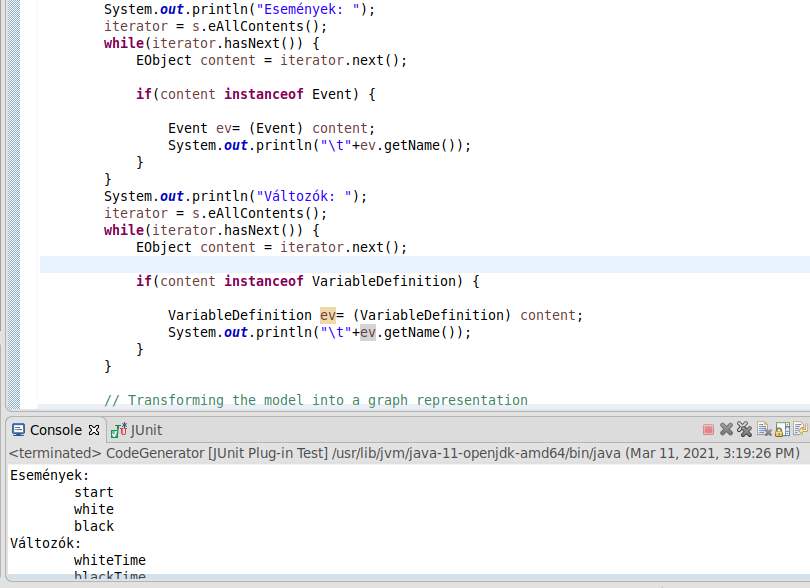
## whiteTime -= 1 az új gráfon

Az átláthatóság kedvéért próbáltam kitörölni az irreleváns részeket, nagyjából csak azt megtartani, ami a feladat megoldása szempontjából fontos.



## Belső változók és bemenő események kiírása

Megtrükközött a running configuration, de sikerült:

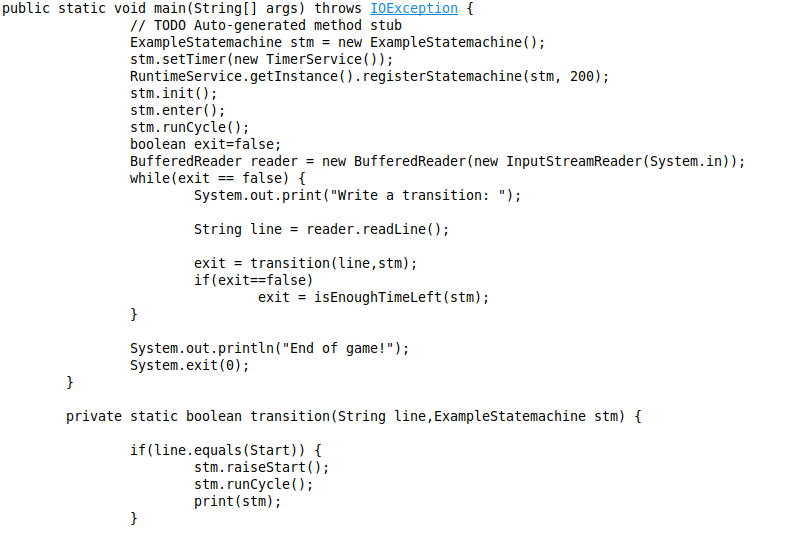


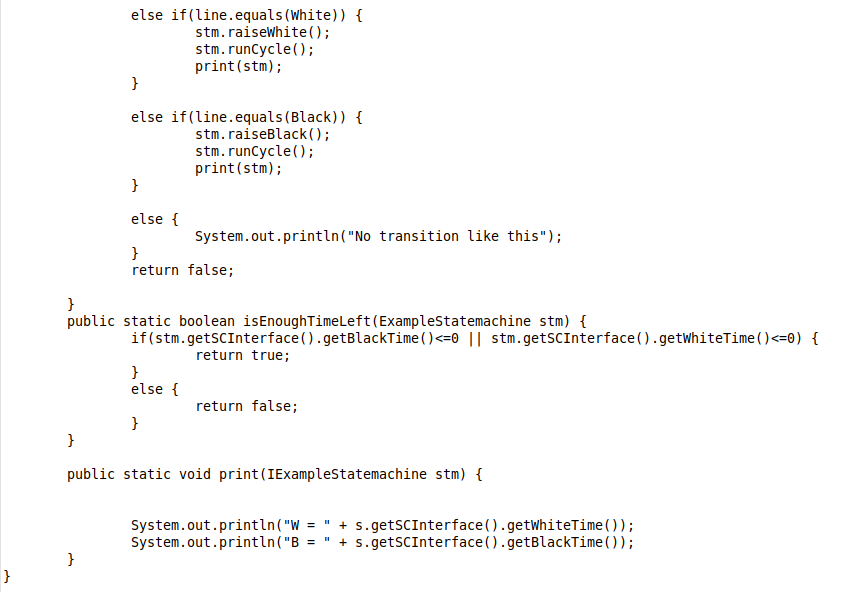
## Programkód kiírása

A kiegészített kód sajnos nem fér rá egy kijelzőre, de a *CodeGenerator.java* fileban megtalálható.

## Kód kiírása

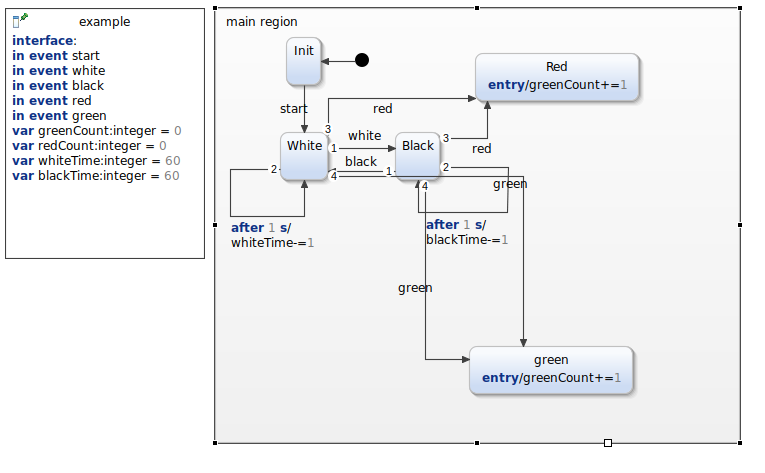
Az alábbi két képen a konzolról készített képernyőképek láthatóak:



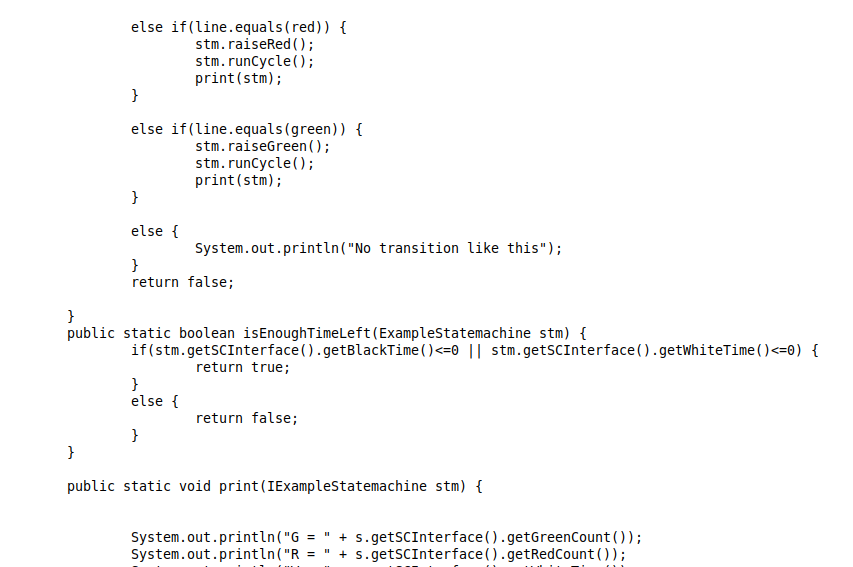


## Modell módosítása, majd kód futtatása

A módosított modell:



Releváns kódrészletek arról, hogy a generator nem beégetett:



## Futtatható kód készítése

Készítettem egy filebaírót, ami beírja a *GeneratedCode.java* fileba ami a workspace gyökér könyvtárában, ahol vannak a launch fileok. A következő kép már ennek a file-nak a futtatásának az eredménye. A konfiguráció amivel futtattam a BigModel konfiguráció duplikálásával készült. Az indentálás nem szép, de ez egy nagyon jó feladat volt! Különösen izgalmas és élvezetes volt számomra.

