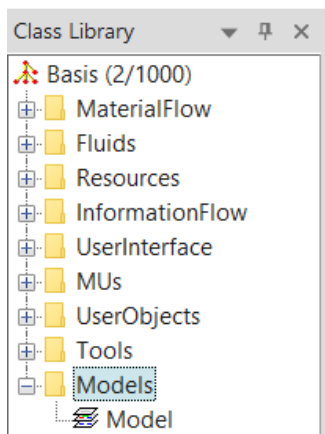




Plant Simulation

Objektovo-orientovaný prístup

Objekty v Plant Simulation (PLS) sú usporiadané v stromovej štruktúre, ktorú vidíme v Class Library. Dajú sa triediť podľa priečinkov, a jednotlivé priečinky je možné ukladať na disk.



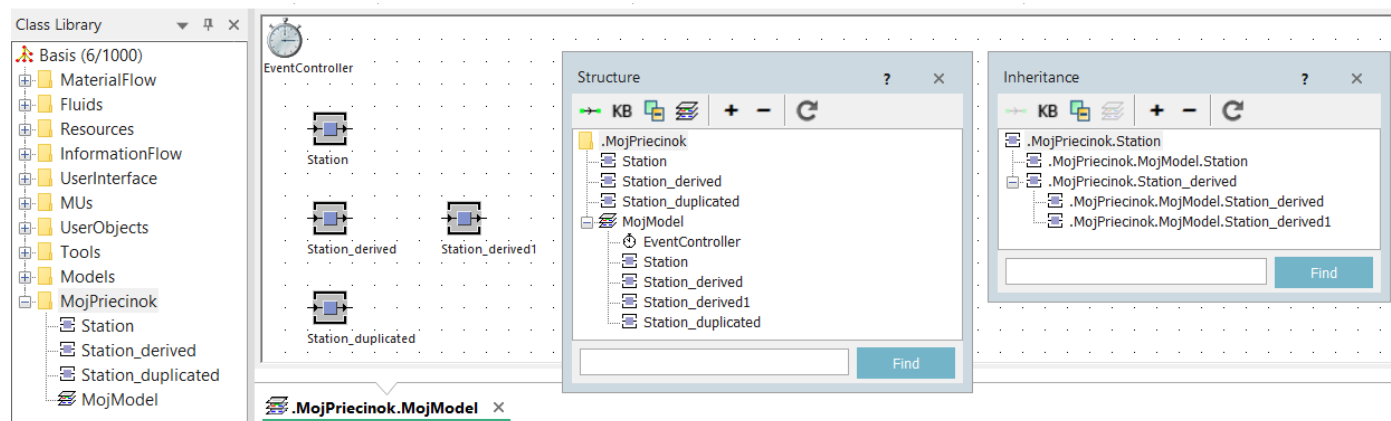
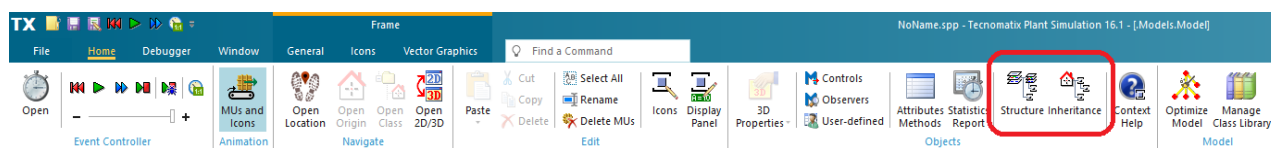
Práca s objektami sa riadi objektovo-orientovaným prístupom tak, ako ho poznáme z programovacích jazykov: V Class Library sa nachádza **trieda** objektu, v modelovacej ploche je **inštancia** danej triedy (inštancie vytvárame presunutím triedy do modelovacej plochy pomocou myši).

Pri modelovaní je tiež možné využiť dedičnosť – vybrané atribúty môže od rodičovskej triedy zdediť trieda, ktorá bola z rodičovskej vytvorená pomocou možnosti **Derive**. Trieda vytvorená pomocou **Duplicate** je samostatná bez dedičného vzťahu. (Funkcie **Derive** a **Duplicate** sa nachádzajú v kontextovom menu praveho kliku pre triedu objektu). Hodnota atribútu, ktorá má byť zdedená od rodiča je v dcérskom objekte označená . Hodnota atribútu, ktorá sa nemá dediť a bude nami špecifikovaná sa označuje . (Uvidíme to neskôr vo vlastnostiach príslušného objektu).

V okne **Structure** vidíme triedy v rámci priečinku a inštancie tried v rámci modelu.

V okne **Inheritance** vidíme dedičné vzťahy. *Station_duplicated* tam nevidíme, lebo nededí žiadne atribúty. Inštancia *MojModel.Station* dedí atribúty z triedy *Station*. Inštancie *MojModel.Station_derived* a *MojModel.Station_derived1* dedia atribúty z triedy *Station_derived*, ktorá dedí atribúty z triedy *Station*.

Vidíme, že k jednotlivým hlbším položkám stromu sa dostávame cez bodku.



Práca s atribútmi

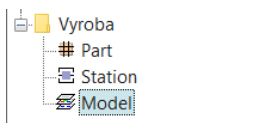
Najprv si pripravíme základnú zostavu, s ktorou budeme pracovať.

Vytvorte si vlastný priečinok nazvaný Vyroba v Class Library: Pravým tlačidlom kliknite na Basis, zvolíte New > Folder.

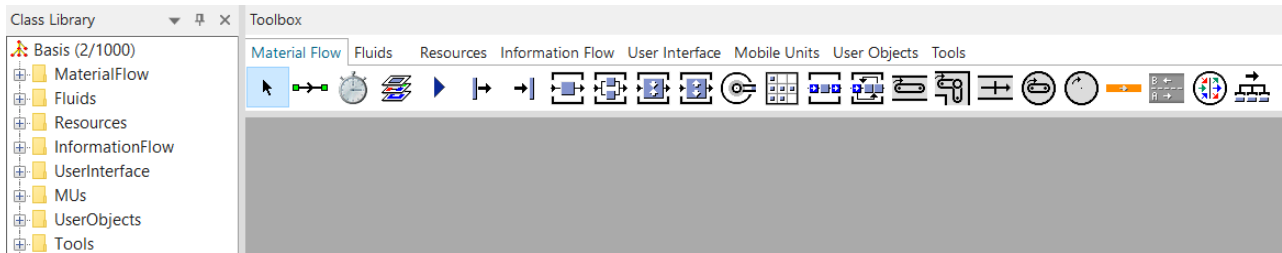
Z priečinku MUs duplikujte Part do priečinku Vyroba takto:

- Možnosť 1 – Pravý klik na Part, zvolíte Duplicate. Myšou presunúť za súčasného držania klávesy **Shift** novovytvorený Part z **UserObjects** do Vyroba.
- Možnosť 2 – Myšou za súčasného držania klávesy **Ctrl** chytiť Part v priečinku **MUs** a umiestniť kópiu do priečinku Vyroba.

Rovnako si do Vyroby zduplikujte aj Station z priečinku MaterialFlow a tiež Model z priečinku Models.

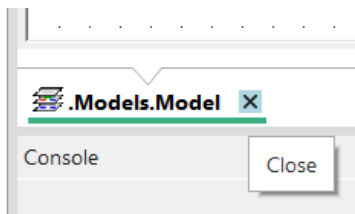


Robíme to preto, aby sme nemenili vlastnosti defaultných tried. Defaultné triedy sú prístupné aj cez Toolbox vo forme ikon.

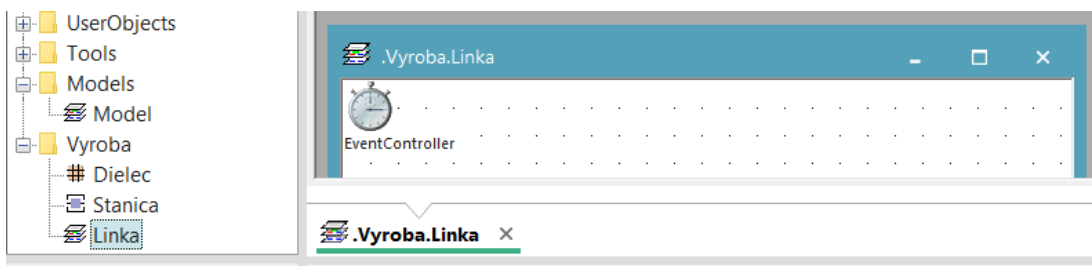


V našej Vyrobe: premenujte Part na Dielec, Station na Stanica a Model na Linka. (Pomocou dvoch samostatných klikov na tú istú položku, alebo po dvojkliku vo vlastnostiach - Name).

Ak máte otvorenú nejakú modelovaciu plochu, pravdepodobne defaultnú .Models.Model, zavrite ju. Nie je to model, ktorý chceme upravovať.

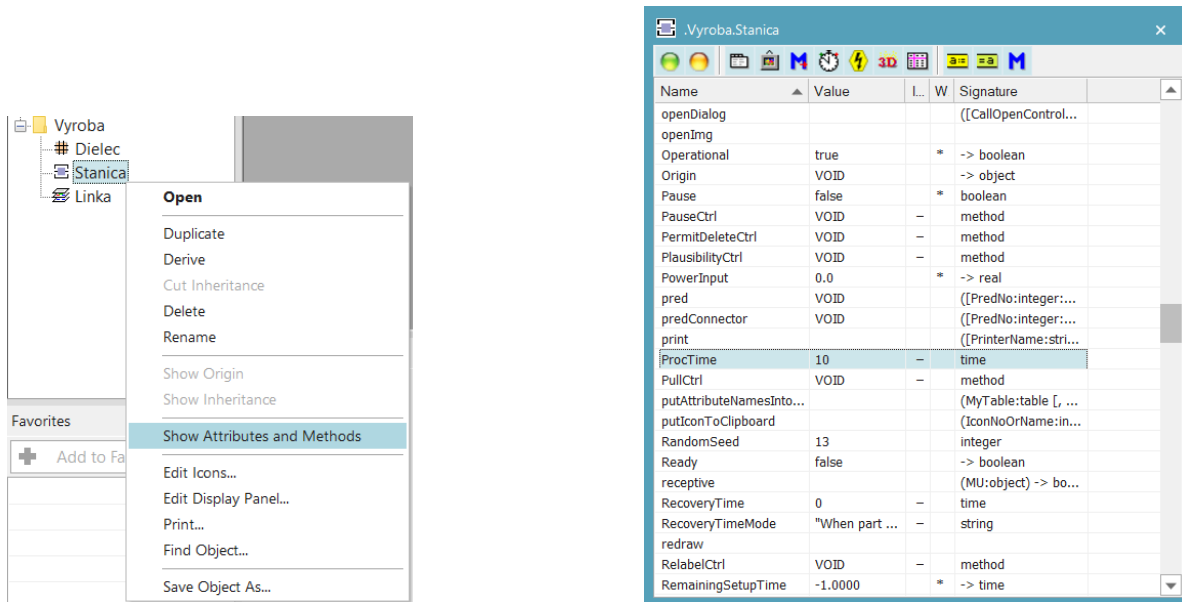


Dvojklikom na model Linka vo vašej Vyrobe otvoríte modelovaciu plochu tohto modelu.



Teraz môžeme preskúmať niektoré atribúty.

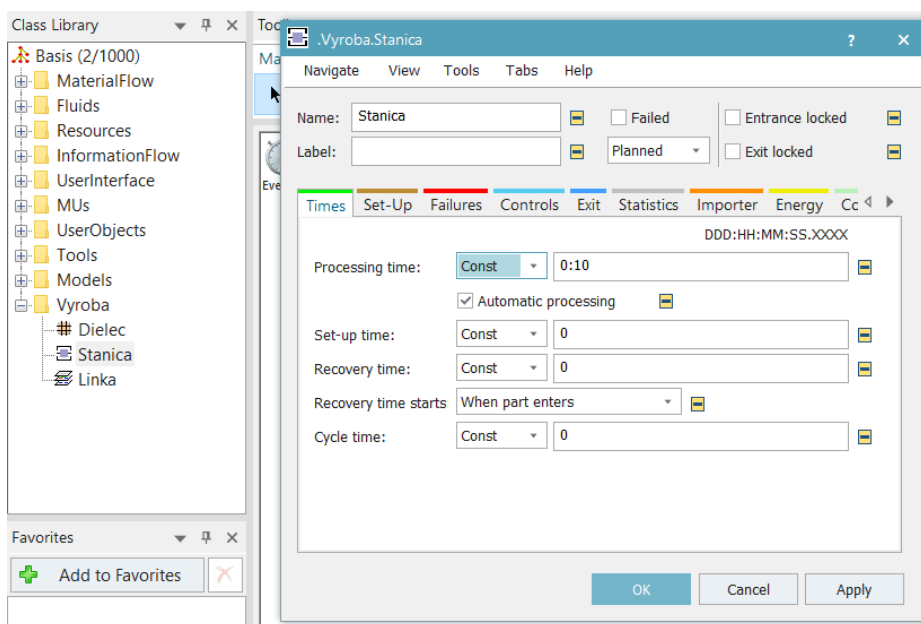
Každá trieda obsahuje svoje atribúty a metódy (podobne ako pri objektovo orientovanom programovaní). Ich zoznam nájdeme po pravom kliknutí na triedu objektu v kontextovom menu: Show Attributes and Methods. Alebo tiež stlačením klávesy **F8** po kliknutí na požadovanú triedu.



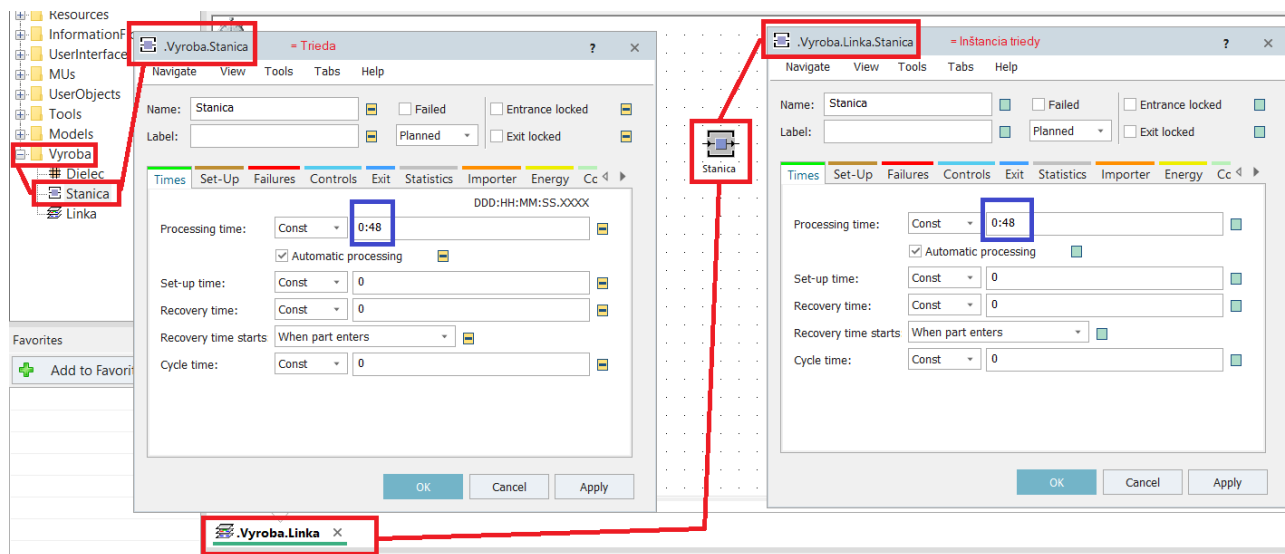
Napravo hore je zoznam atribútov a metód pre Stanicu. Vidíme tam aktuálne nastavenie Processing time (ProcTime) = 10s. V tomto okne je možné meniteľné položky upravovať a meniť im hodnoty.

Na každej položke je možné stlačiť **F1** a vyvolať help k danej položke.

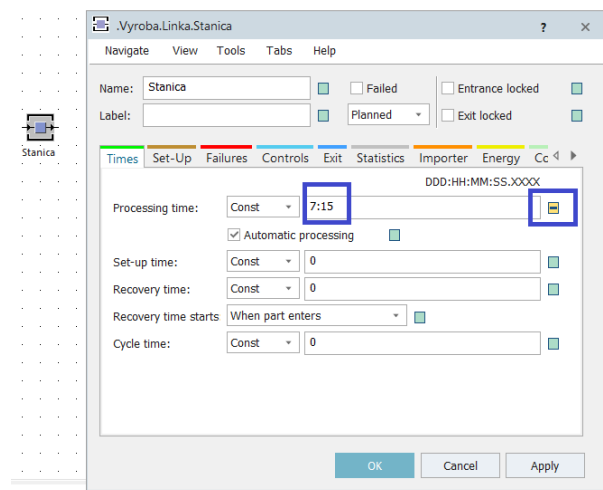
Niektoré z týchto atribútov je možné ovládať aj pomocou formulárov vo vlastnostiach príslušného objektu, ktoré nájdeme po dvojkliku na triedu alebo inštanciu. (Nasledujúci obrázok je pre dvojklik na triedu Stanica). Vidíme tam napríklad Processing time, ktorý je nastavený na 10s, čo sme videli už v predošlom zozname.



Ak meníme vlastnosti triedy, potom aj jej inštancie budú mať rovnako zmenené príslušné parametre. Napríklad, keď do modelu Linka vložíme objekt Stanica a až následne zmeníme vlastnosť triedy Stanica, tak sa táto zmena prejaví aj na už vloženom objekte (zmenili sme v triede Processing time na 48s):

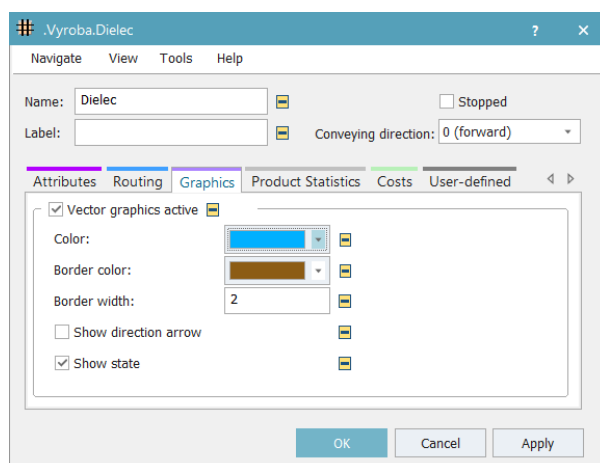
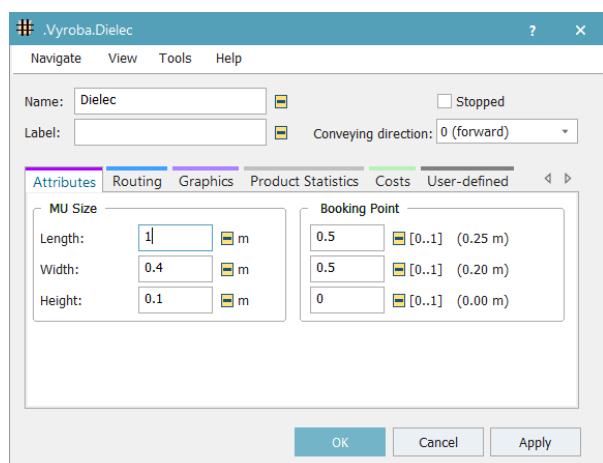


Na obrázku hore vidieť, že trieda Stanica nedeľí žiadne vlastnosti (☐) a naopak objekt Stanica v modeli Linka dedí všetky atribúty od svojho rodiča (od triedy Stanica) (☑). Ak zmeníme niektorý parameter v objekte Stanica, príslušný riadok sa označí ☐ a stáva sa nezávislý od vlastností rodiča (obrázok dolu). Následná zmena tohto parametra v triede sa už na tomto konkrétnom objekte Stanica neprejaví.

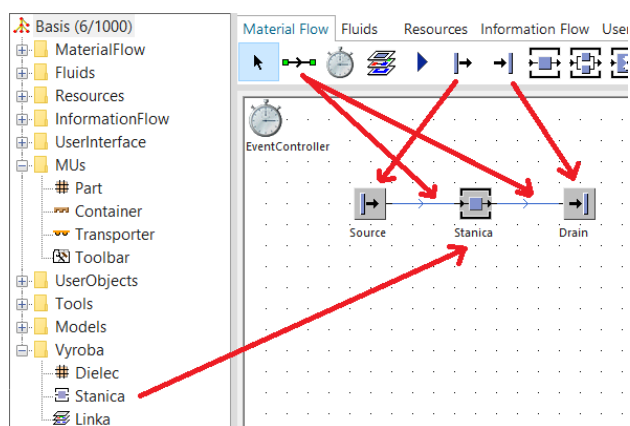


V modelovacej ploche Linka je možné objekty kopírovať pomocou Ctrl+C a Ctrl+V a tiež pomocou drag&drop so stlačeným Ctrl. Tieto kópie si zachovávajú zmeny vykonané na kopírovanom objekte. V našom prípade aj kópie budú mať Processing time nastavitelný nezávisle od rodiča v triede. Po kliknutí na ikonu ☐ sa ikona zmení na ☑ a pôvodne nezávislá hodnota sa načíta z rodičovskej triedy a položka sa znova stáva predmetom dedenia.

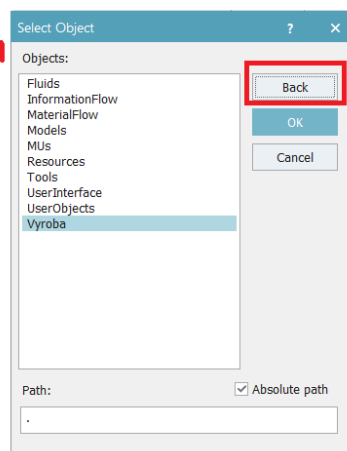
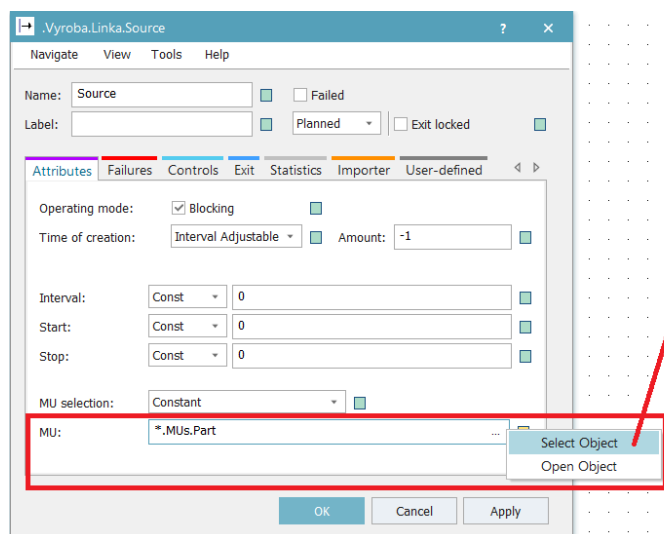
V triede Dielec si upravme vlastnosti dielca takto: Dĺžku dielca zmeníme na 1m a zmeníme aj farbu dielca.



Do modelu Linka urobíme nasledujúcu zostavu. Source a Drain prevezmeme z default knižnice materiálového toku, Stanicu použijeme z našej Vyroby a spojíme to connectormi tiež z default knižnice. Spustením simulácie cez vidíme, že systémom prúdia hnedé prvky (nie sú to naše dielce), je to Part z priečinka MUs. Simuláciu zastavíme opätovným stlačením a resetneme cez .



Otvorme si vlastnosti objektu Source a zmeňme prvok, ktorý bude vstupovať do systému:



V okne Select Object v časti Objects vidíme triedy aktuálneho priečinka (MUs).

Stlačením **Back** sa dostaneme o úroveň vyššie, čiže vidíme priečinky v koreňovom priečinku. Vyberieme Vyroba a v ňom zvolíme náš Dielec. Upravená hodnota vo vlastnostiach Source vyzerá nasledovne:

MU:

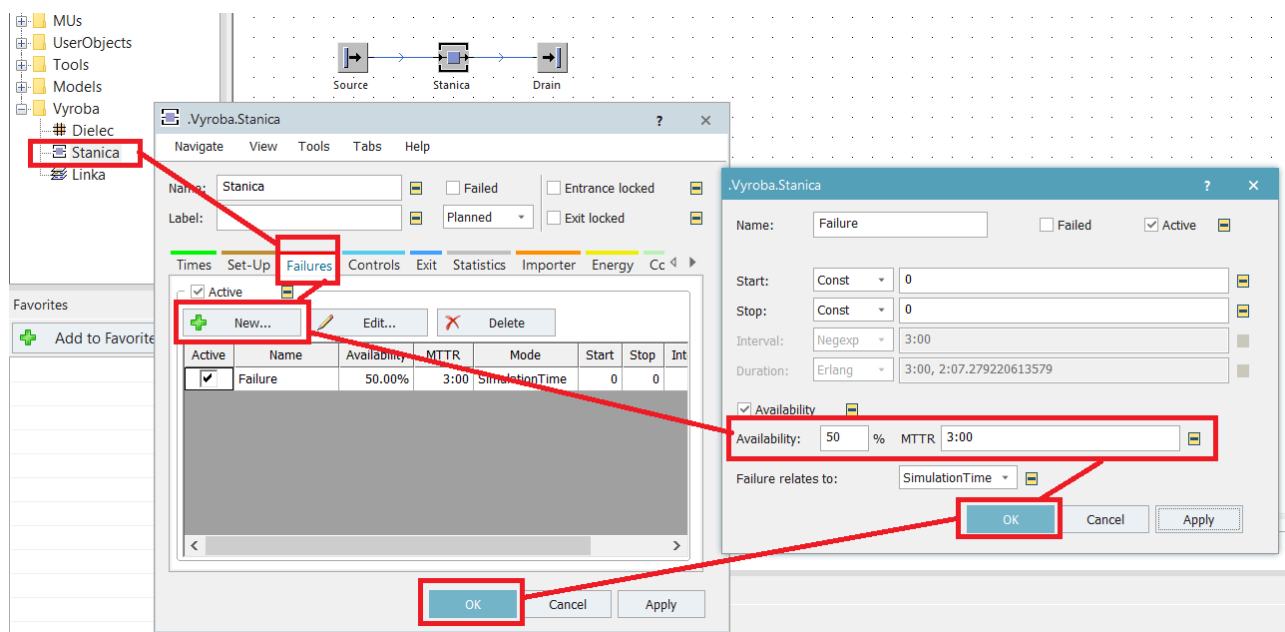
Po spustení simulácie už systémom prúdi naše zmenené dielce.

Atribút objektu Source, ktorý uchováva adresu nášho dielca sa nazýva Path. Vieme to skontrolovať v zozname atribútov a metód po kliknutí na ikonu Source a následným stlačením klávesy F8.

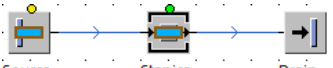
Preskúmame ďalšie atribúty Stanice.

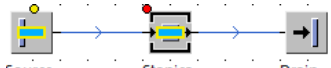
Poruchy:

Dvojkliknite na triedu Stanica a v záložke Failures kliknite na New, čím zadefinujete tejto triede staníc nejaký nový profil poruchy. V špecifikácii nastavíme, že Stanica bude dostupná 50% simulačného času, a keď nastane porucha, tak stredná doba na opravu stanice bude 3min (MTTR – mean time to repair).



Spustíte simuláciu a spomaľte si ju , aby ste videli, kedy nastáva porucha:

Stanica pracuje: 

Stanica je v poruche: 

Signalizuje to farebný bod nad ikonou. Ďalej nad objektom Source vidíme aj žltý bod, čo signalizuje, že objekt čaká na Stanicu, kým jej môže odovzdať pripravený Dielec.

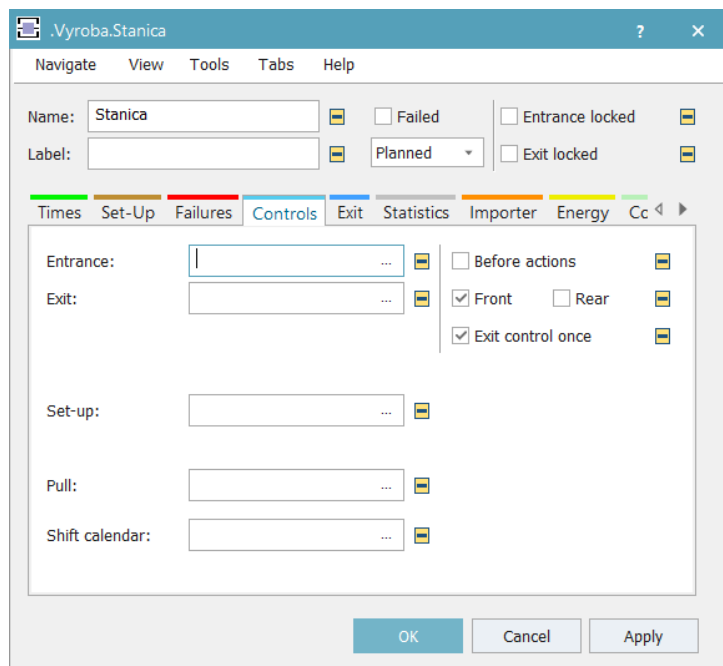
Pre deaktiváciu tohto profilu poruchy zrušte označenie v stĺpci Active

Active	Name	Av
<input checked="" type="checkbox"/>	Failure	

Stanica bude opäť fungovať 100% času.

Riadiace programy:

Vo vlastnostiach triedy Stanica kliknite na záložku Controls:



Vidíme tu riadky, do ktorých vieme vložiť nejaké riadiace metódy. **Metóda** je program, ktorý zabezpečí požadovanú funkcionálnosť. Programovací jazyk, ktorý sa používa na písanie metód je **SimTalk**.

Predstavme si, že každé zariadenie má nejaký vstupný a výstupný senzor. Ak do zariadenia vojde nejaký prvok, tak môže byť aktivovaná určitá obslužná metóda. Takúto metódu vkladáme do poľa **Entrance**. Ak chceme, aby bola spustená nejaká metóda, keď prvok opúšťa zariadenie, túto metódu vkladáme do poľa **Exit**.

Pri výstupe prvku máme možnosť rozlišovať medzi 2 prípadmi:

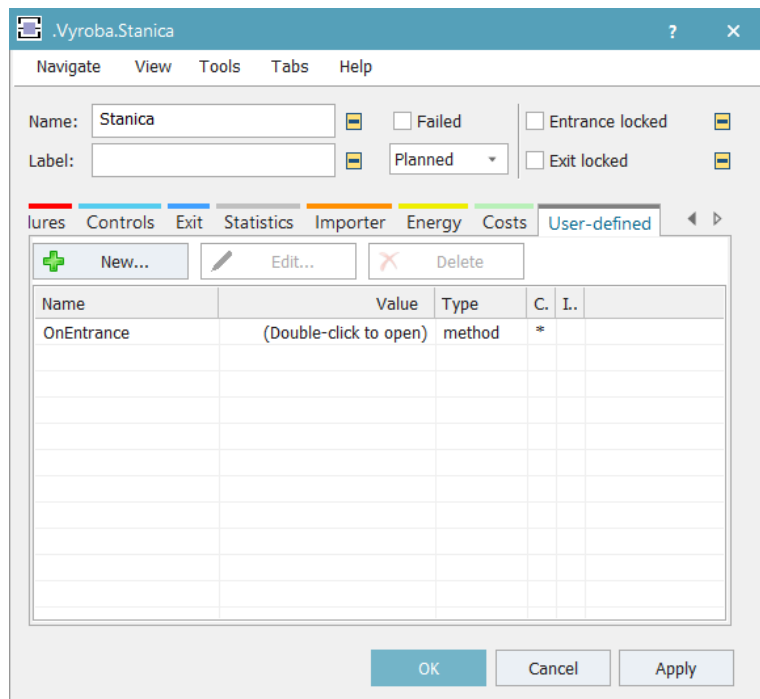
1. Keď sa prvok ešte len pokúša vyjsť zo zariadenia – čiže svojou prednou časťou aktivuje pomyselný výstupný senzor. Pre takýto prípad označíme možnosť **Front**.
2. Keď prvok kompletne vyjde zo zariadenia – čiže svojou zadnou časťou aktivuje pomyselný výstupný senzor. Pre takýto prípad označíme možnosť **Rear**. V tomto prípade sa prvok po opustení tohto zariadenia dostal pozdĺž connectora už do iného zariadenia.

Pozor! Použitím **Exit** metódy s možnosťou **Front** spôsobí, že prvok sa v tomto zariadení prestane sám presúvať pozdĺž connectora do ďalšieho zariadenia, ale očakáva, že naša obslužná metóda okrem iných potrebných vecí takýto presun zabezpečí v programe.

Vytvorme si vstupnú metódu, ktorá sa bude spúšťať pri každom vstupe Dielca do Stanice. Kliknite na bodky ... na konci poľa **Entrance** a vyberte **Create Control**:



Automaticky sa vytvorí metóda s názvom **OnEntrance**. Táto metóda je uložená v našej triede Stanica. Nájde ju v oblasti „používateľsky-definovaných atribútov a metód“, čiže pod záložkou **User-defined** (viď obr. nižšie).



Keďže do poľa Entrance v záložke Controls je nutné vkladať adresu metódy v stromovej štruktúre objektov využije sa slovíčko **self**, ktoré je zástupným symbolom adresy aktuálneho objektu, čiže našej Stanice, pretože metóda je uložená v Stanici.

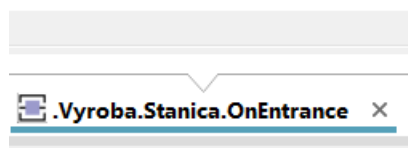
Entrance: ...

V tomto prípade, keď sa jedná o **triedu Stanica**, je teda **self = .Vyroba.Stanica**

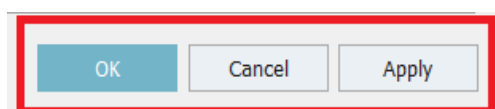
Inštancia triedy Stanica bude mať **self = .Vyroba.Linka.Stanica**

Každý objekt pozná svoju adresu, a teda má vlastné self. V tejto adrese je zahrnuté aj meno tohto objektu. Metóda je tiež nejaký objekt v stromovej štruktúre a teda tiež má vlastné self.

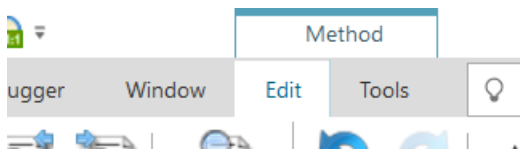
Po vytvorení metódy sa v pracovnej ploche táto metóda aj otvorila, čo je vidno v paneli okienok:



Pozor! Vo vlastnostiach Stanice nezbudnime potvrdiť vykonané zmeny (vytvorenie metódy) pomocou tlačidla **Apply** alebo **OK**.



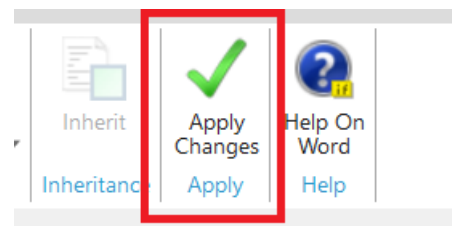
Všimnite si, že po otvorení metódy v hornom menu pribudli panely **Edit** a **Tools** určené na prácu s metódou.



Programovanie metód

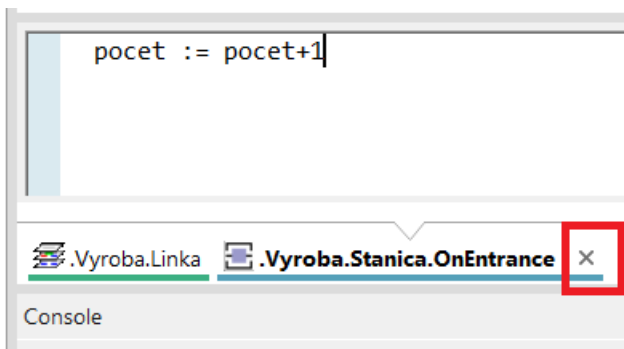
Teraz môžeme editovať našu metódu. Jej účelom bude pre jednoduchosť iba počítanie Dielcov, ktoré vošli do Stanice. Preto do editačnej plochy napíšeme iba:

```
pocet := pocet+1
```

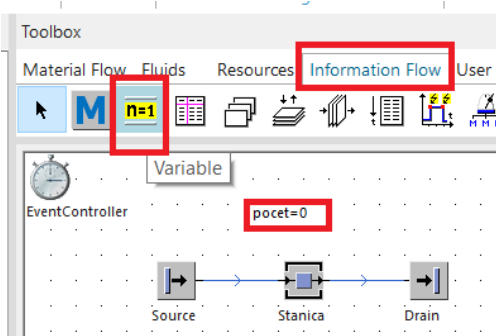


Z menu panela **Method > Edit** vyberieme položku **Apply Changes**

Teraz máme metódu uloženú a môžeme ju zavrieť.



Zatiaľ sme nikde nedefinovali premennú `pocet`, ale chceme, aby to bola **globálna premenná**. Preto do plochy modelu vložíme z Toolboxu pod priečinkom Information Flow prvok **Variable** a premenujeme ho na **pocet**. Týmto máme zadefinovanú globálnu premennú, ktorú poznajú všetky metódy definované v rámci modelu Linka.

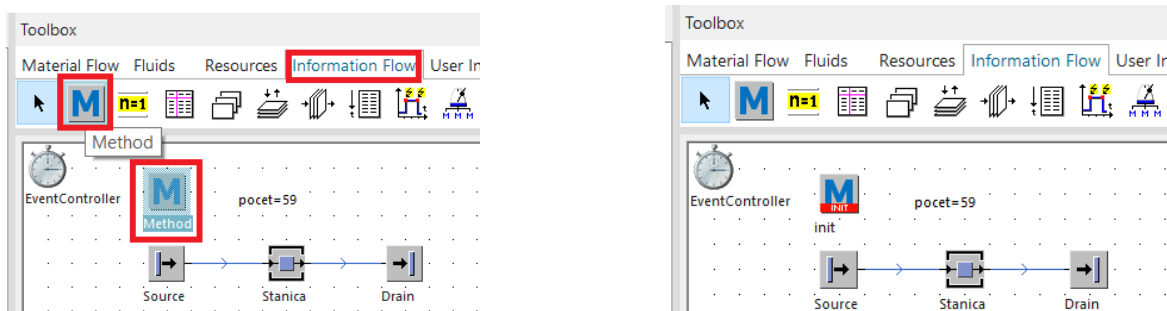



Spustením simulácie vidíme, že hodnota premennej `pocet` narastá po každom vstupe Dielca do Stanice. Zastavte simuláciu, resetnite ju a znova spustite. Premenná `pocet` sa nevynuluje, pretože na to nemáme spravený žiaden príkaz.

Na tento účel si predstavíme špeciálne metódy, ktoré majú rezervované názvy:

- **Init** - Metóda, ktorá sa spustí hneď po spustení simulácie ako prvá.
- **Reset** - Metóda, ktorá sa spustí po kliknutí na tlačidlo reset simulácie.
- **EndSim** - Metóda, ktorá sa spustí po ukončení definovaného času simulácie.

Na vynulovanie premennej použijeme metódu `init`. Metódy nemusia byť uložené iba v rámci nejakého zariadenia, ako sme mali našu `OnEntrance` v Stanici. Metódy môžeme mať uložené aj v rámci nášho modelu Linka. Presuňte si teda do modelovacej plochy ikonu **Method** z Toolboxu Information Flow. Premenujte metódu na **init** (taktiež sa zmení vizuál ikony).

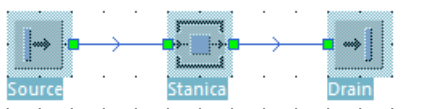


Dvojklikom na ikonu metódy **init** ju otvoríme, napíšeme `pocet := 0`, uložíme cez  **Apply Changes** a môžeme zavrieť.

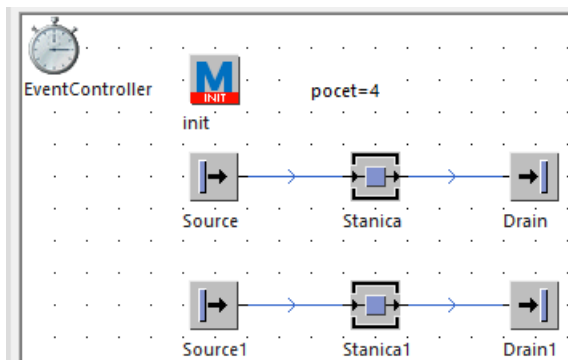
Po resete a novom spustení simulácie sa spustí metóda `init` a vynuluje našu premennú.

Zduplikujme našu zostavu troch zariadení takto:

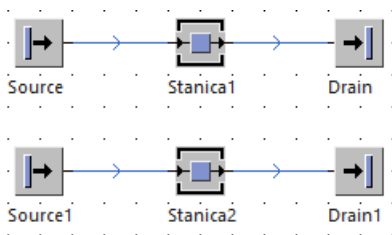
Označte všetky 3 zariadenia aj s konektormi (označením bloku pomocou myši).



Chyťte niektorý z objektov, **stlačte Ctrl**, **potiahnite** a **pustite** na inom mieste. Výsledok je na obrázku nižšie:



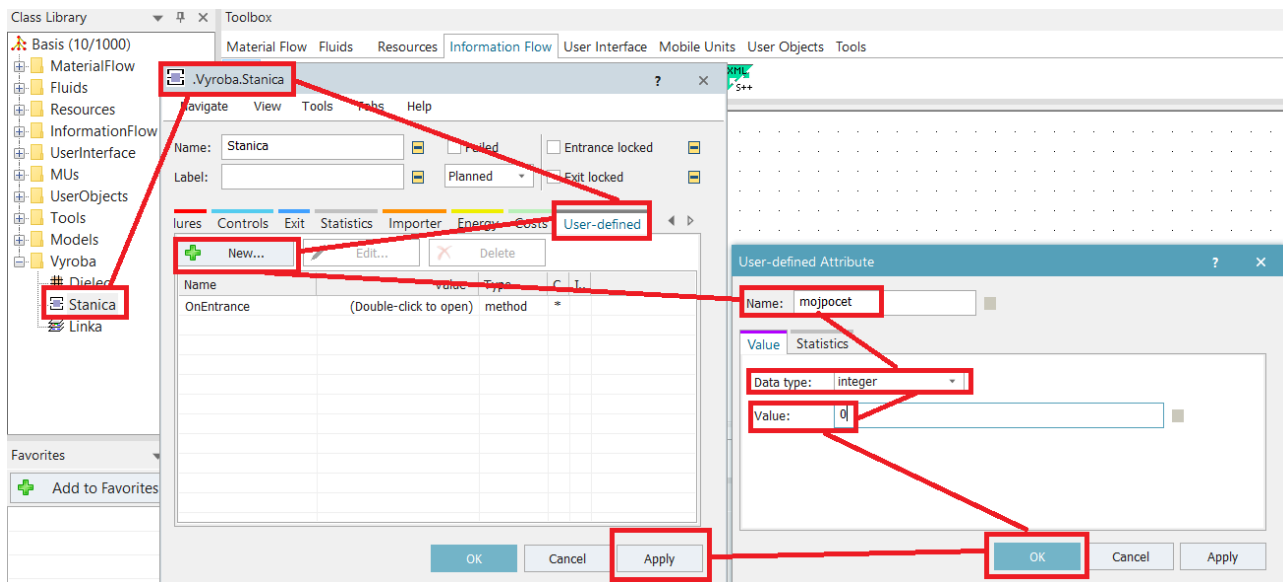
Pre prehľadnosť si **premenujme stanice** nasledovne:



Spomaľte si simuláciu natoľko, aby ste videli ako po jednom vchádzajú dielce do Stanice1 aj do Stanice2. Všimnite si, že premenná `pocet` narastá dvojnásobnou rýchlosťou. Je to preto, že metóda `OnEntrance` je definovaná v triede `Stanica` a teda všetky inštancie tejto triedy ju tiež obsahujú a prístupujú k rovnakej globálnej premennej.

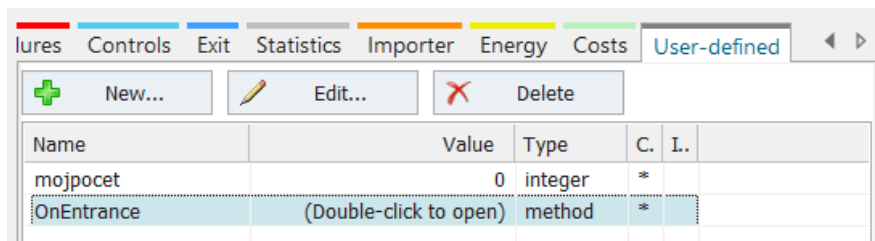
V **triede Stanica** pridáme ďalší atribút, ktorým bude lokálna premenná **mojpocet**.

Otvorte vlastnosti triedy `Stanica`, a prekliknite sa do záložky **User-defined**. Kliknite na **New**, zadajte názov nového atribútu **mojpocet**, dátový typ bude **integer** a počiatočná hodnota bude **0**. Potvrďte **OK** a v Stanici potvrdíte **Apply**.



V **User-defined** pribudol užívateľsky definovaný atribút **mojpocet**.

Dvojklikom na text **(Double-click to open)** otvoríme metódu `OnEntrance` a ideme ju editovať. Okno vlastností Stanice môžeme zavrieť cez **OK**.



V metóde OnEntrance doplníme druhý riadok tak, aby vyzerala nasledovne:

```
pocet := pocet+1
?.mojpocet := ?.mojpocet + 1
```

Otáznik ? je ďalší zástupný symbol a predstavuje **objekt, ktorý spustil metódu**. Vieme, že **mojpocet** je premená užívateľsky definovaná v Stanici a vieme tiež, že Stanica spúšťa metódu OnEntrance.

(V tomto prípade nemôžeme použiť zástupný symbol self, pretože ten predstavuje adresu a názov tejto metódy, ale mojpocet nie je definovaný v tejto metóde, ale v Stanici.)

Keďže zmeny vykonané v triede sa prenesú do jednotlivých inštancií, každá bude mať definovanú premennú mojpocet, a tiež v rámci metódy bude pre každú inštanciu zástupný symbol (otáznik) obsahovať inú hodnotu:

V metóde OnEntrance inštancie Stanica1: ? = **.Vyroba.Linka.Stanica1**

V metóde OnEntrance inštancie Stanica2: ? = **.Vyroba.Linka.Stanica2**

Z toho dôvodu môžeme povedať, že sa lokálne premenné jednotlivých staníc budú meniť nezávisle.

Výpis hodnôt môžeme vykonať napríklad do konzoly použitím príkazu **print**.

V metóde OnEntrance doplníme tretí riadok tak, aby vyzerala nasledovne:

```
pocet := pocet+1
?.mojpocet := ?.mojpocet + 1
print ?.Name + " " + ?.mojpocet
```

Atribút **Name** uchováva meno objektu ako string. Cez + tento reťazec spojíme s medzerou " " a cez ďalšie + spojíme s našou premennou (tu dôjde k automatickému pretypovaniu na string). Aj pri atribúte Name sa potrebujeme dostať k objektu, ktorý volá túto metódu, aby sme dostali správne meno. Preto tiež použijeme zástupný symbol otáznik.



Uložíme zmeny cez  **Apply Changes** a môžeme metódu zavrieť.

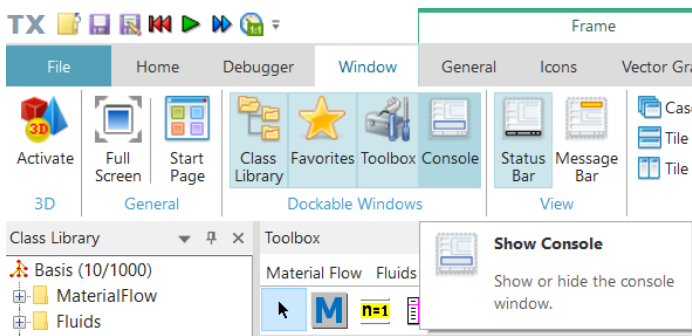
Tieto lokálne premenné môžeme ovplyvňovať aj z vonka s absolútnymi adresami, čiže napríklad z našej metódy init. **Editujme metódu init** tak, aby vyzerala nasledovne:

```
pocet := 0
.Vyroba.Linka.Stanica1.mojpocet := 0
.Vyroba.Linka.Stanica2.mojpocet := 0
```

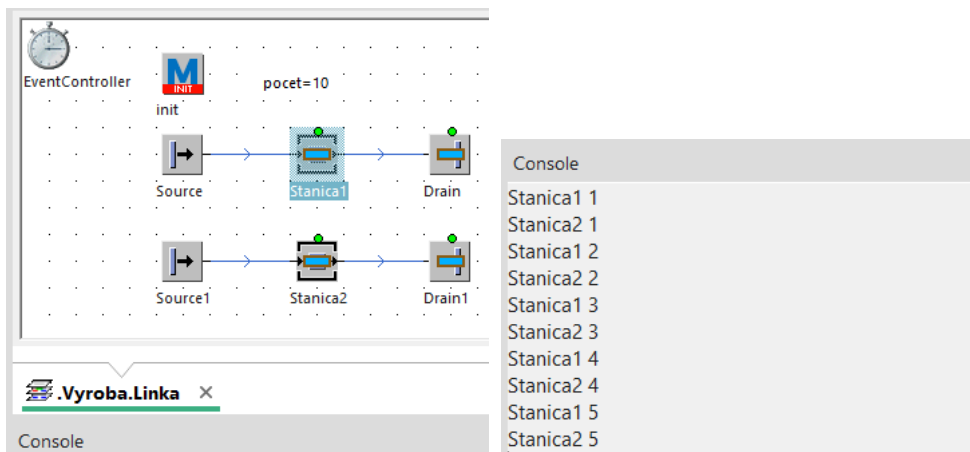


Uložíme zmeny cez  **Apply Changes** a môžeme metódu zavrieť.

V hornom menu Window si zapnite konzolu.

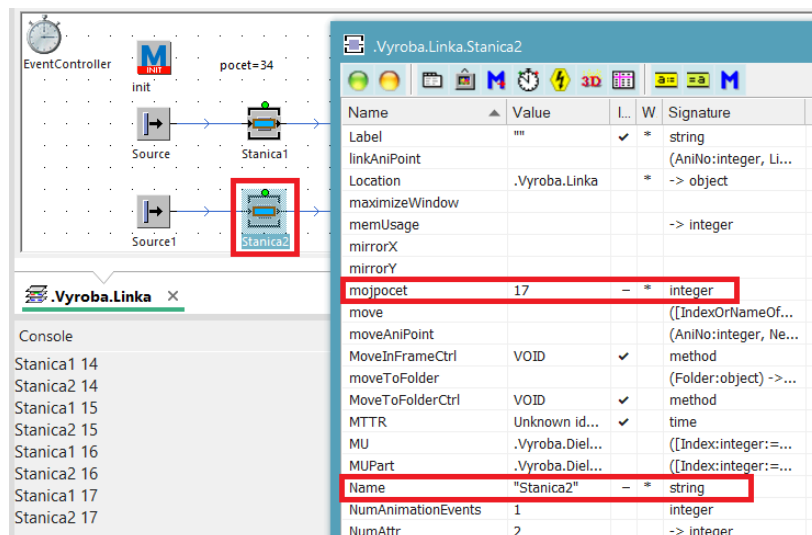


Spustíte simuláciu a uvidíte nasledovné: Globálna premenná stále počíta celkový počet. Hodnoty lokálnych premenných pre jednotlivé stanice vidíme v konzole.

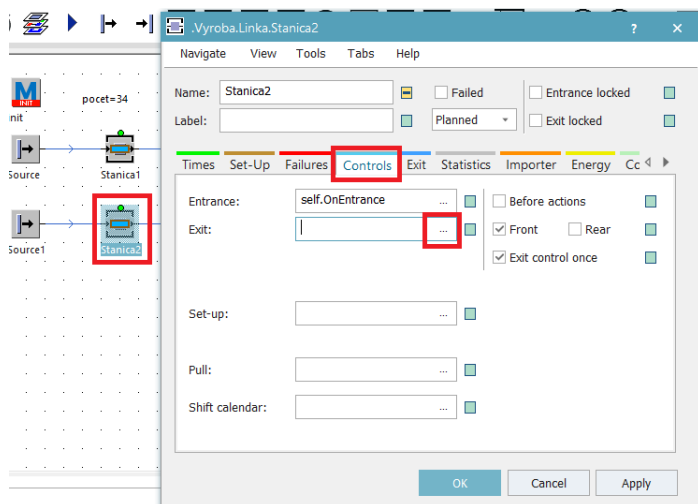


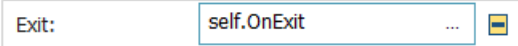
Zastavte simuláciu. Kliknite napríklad na Stanicu2 a stlačte **F8**. Medzi atribútmi si všimnite našu lokálnu premennú **mojpocet** a tiež atribút **Name**, ktorý sme využili na výpis do konzoly.

Podobne je v metódach možné využívať a nastavovať aj iné atribúty. Namiesto nášeho počítadla by sme mohli využiť zabudovaný atribút **StatNumIn** alebo **StatNumOut**. Vyhľadajte ich a stlačte na nich **F1**. V help pre tento atribút zvolte nápoedu týkajúcu sa Material flow objectu.



V inštancii triedy Stanica, konkrétne v **Stanica2** si skúsime nadefinovať aj **výstupnú metódu**, ktorá sa bude spúšťať ešte predtým, než Dielec opustí stanicu (označená možnosť ☒ **Front**). Po kliknutí na bodky v poli Exit vytvoríme metódu (cez Create Control).




Automaticky sa nám vytvorila metóda s názvom **OnExit**, na pozadí sa otvorila a do poľa Exit sa doplnila adresa k tejto metóde:  . Navyše, keďže upravujeme **inštanciu** triedy Stanica, **odstránila sa dedičná závislosť** tohto poľa od rodičovskej triedy. V tejto chvíli ma jedine inštancia **Stanica2** priradenú výstupnú riadiacu metódu.

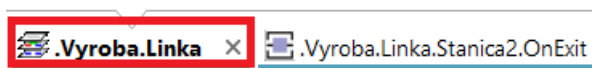
Zavrite vlastnosti Stanice2 cez **OK**, aby sa zachovali vykonané zmeny (t.j. vytvorená metóda)!

V editačnej oblasti metódy máme predpripravený minimálny kód na to, aby Dielec bol schopný zo stanice odísť. Spomínali sme, že ak definujeme výstupnú metódu s možnosťou ☒ **Front**, tak pohyb pozdĺž konektora k nasledujúcej stanici sa už nevykoná automaticky, ale musíme ho riadiť my cez metódu.

@.move -- Remove this, if the exit control is rear triggered!

Tento príkaz spôsobí práve to, že metóda **pošle Dielec** zo Stanice2 **pozdĺž konektora** do nasledujúceho objektu, teda do Drainu.

Uložíme zmeny cez  **Apply Changes** ale nechajme si metódu ešte otvorenú, vrátime sa k nej. **Prekliknite** sa do modelu Linka.



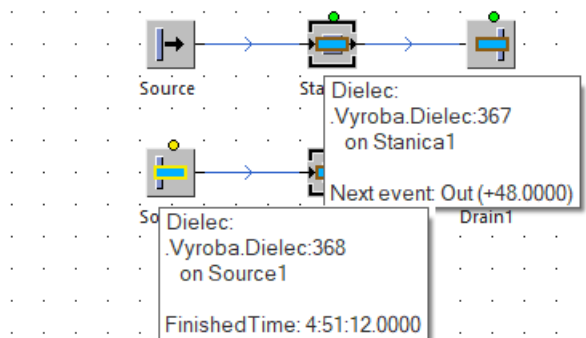
Spustite simuláciu.

Vidíme, že **žiadna zmena** z pohľadu správania sa systému sa neudiala.

Prekliknite sa **naspäť do metódy OnExit.**

Zavináč @ je v metóde zástupný symbol **inštancií pohyblivých jednotiek** (Mobile Units - MUs), **ktoré vošli do zariadenia spúšťajúceho metódu**. Pohyblivá jednotka môže byť: Part, Container alebo Transporter. V našom prípade používame Part premenovaný na Dielec.

Keďže Source automaticky generuje inštancie triedy Dielec, tak tieto inštancie majú špeciálnu adresu, kde za názvom triedy je za dvojbodkou poradové číslo vygenerovanej inštancie (napríklad .Vyroba.Dielec:367 na obrázku). Keď teda **367. inštancia** triedy Dielec **vojde do Stanice2** tak v metóde **OnExit** bude pre zavináč platiť rovnosť: **@ = .Vyroba.Dielec:367**



Atribúty a metódy, ktoré vieme v metóde použiť cez bodku za zavináčom nájdeme v „Atribútoch a metódach“ triedy Dielec. Kliknite na Dielec v Class Library a stlačte **F8**. Nájdite v zozname metódu **move**, kliknite na ňu a stlačte **F1** pre help. V helpe si môžeme všimnúť, že metóda move ma 4 verzie podľa toho, aké parametre použijeme. Ak použijeme metódu bez parametrov, Dielec pošleme pozdĺž existujúceho konektora. Ak použijeme metódu, ktorej jediný parameter bude cieľový objekt, tak ignorujeme existujúce konektory a Dielec pošleme priamo do zvoleného objektu.

Riadok v našej metóde OnExit prepíšeme tak, aby vyzeral nasledovne:

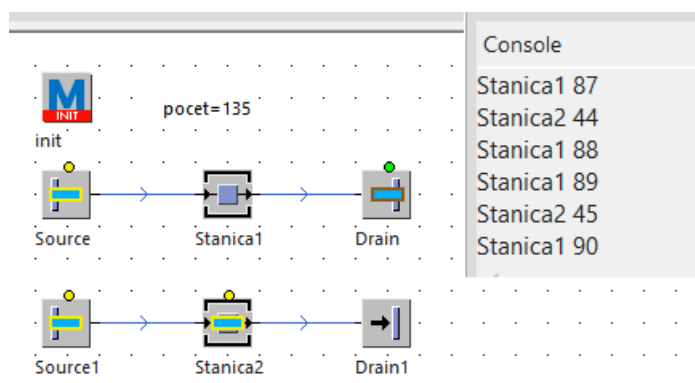
```
@.move(Stanica1)
```



Apply Changes

Uložíme zmeny cez **Apply Changes** a môžeme metódu zavrieť.

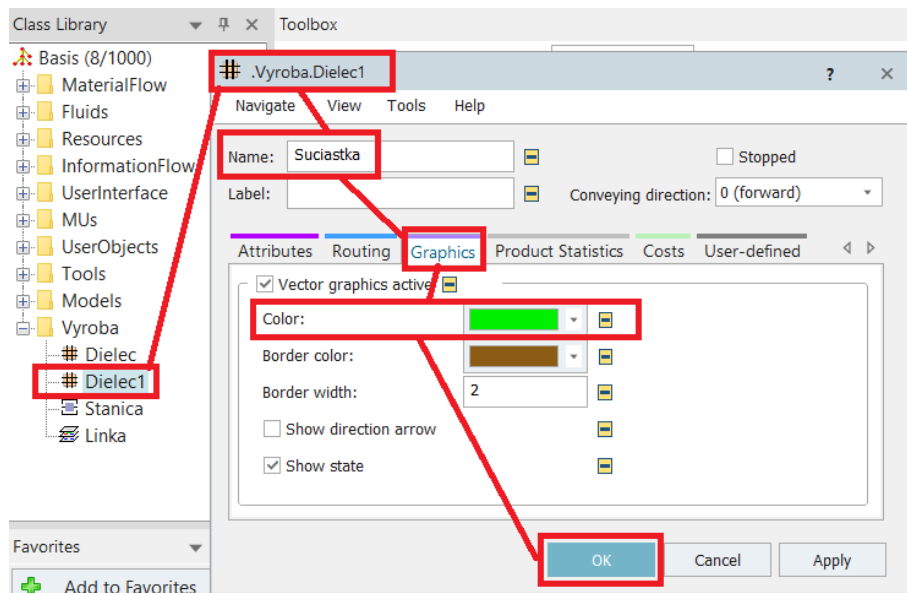
Spustíme simuláciu a všimneme si, že dielce zo Stanice2 sú posielané do Stanice1 ignorujúc existujúce konektory. Dôkaz vidíme aj v našich výpisoch v konzole, kedy Stanica1 spracúva 2-krát viac dielcov než Stanica2. (Poznamenajme, že dielce zo Source1 budú spracované 2-krát, a toto spracovanie sa pripočíta do globálnej premennej pocet.)



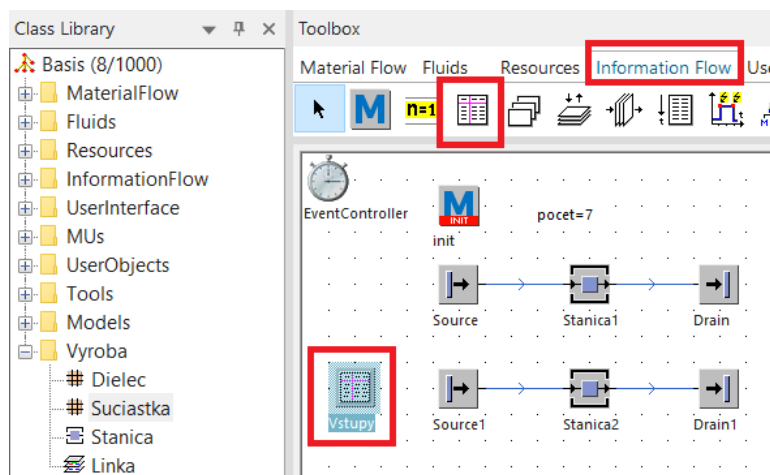
Tabuľky

Veľmi silným nástrojom v Plant Simulation sú tabuľky, na základe ktorých sa môže simulácia nastaviť, riadiť jej priebeh a tiež môžeme tabuľky počas behu simulácie programovo vytvárať, robiť v nich záznamy či upravovať.

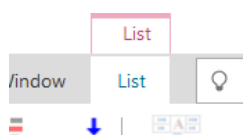
Využijeme tabuľku ako predpis pre spôsob generovania rôznych vstupných dielcov v Source1. V Class Library si vytvoríme **duplikát** triedy Dielec a nazveme ho **Suciastka**. Zmeňme jej **farbu**, aby sme ju odlíšili od Dielea.



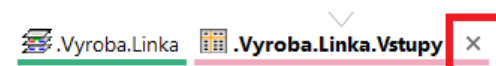
Z default Toolboxu Information Flow si do modelu preneste **DataTable** a premenujte ju na **Vstupy**.



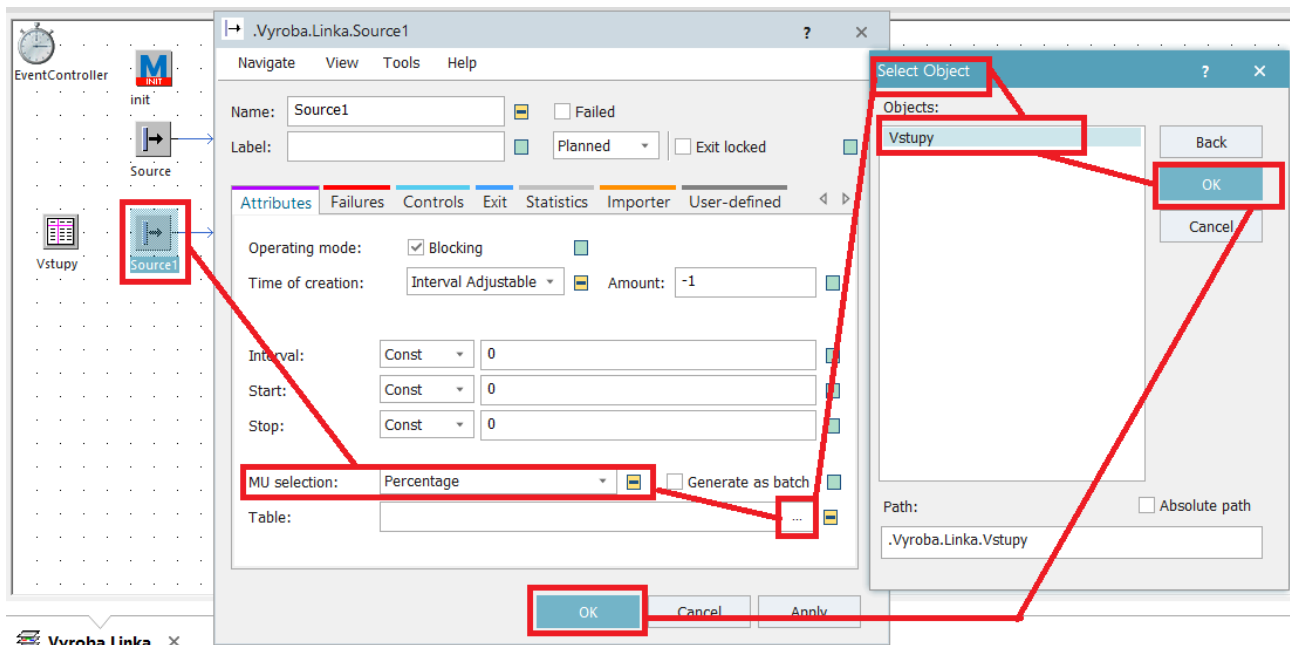
Dvojklikom otvoríte **prázdnu tabuľku**. Všimnite si, že v hornej lište pribudlo **menu List** pre prácu s tabuľkou.



Zatvorte tabuľku bez zmien:



Otvorte **Source1**, zmeňte **MU selection** na **Percentage**, v poli **Tables** kliknite na bodky ... a otvoríte Select Object. Zvoľte **tabuľku Vstupy**, potvrdte **OK** a vlastnosti Source1 tiež potvrdte **OK**



Po opätovnom otvorení tabuľky **Vstupy** vidíme, že Source1 nám ju **naformátoval** tak, ako to potrebuje.

	object 1	real 2	integer 3	string 4	table 5
string	MU	Portion	Number	Name	Attributes
1					
2					
3					

Budeme chcieť aby Source1 produkoval prvky tak, že 90% z nich bude Dielec a 10% bude Suciastka. Tabuľku upravíme nasledovne:

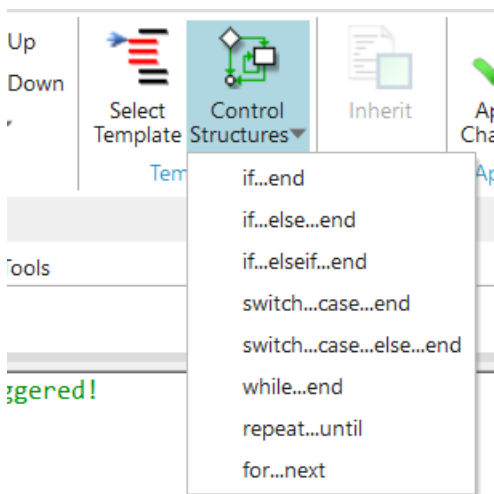
	object 1	real 2	integer 3	string 4	table 5
string	MU	Portion	Number	Name	Attributes
1	.Vyroba.Dielec	0.90			
2	.Vyroba.Suciastka	0.10			
3					

Zavretím sa tabuľka automaticky ukladá.

Spustíme simuláciu a vidíme, že 10% (každý desiaty) z prvkov Source1 sú Súčiastky – majú inú farbu.

Riadiace štruktúry programu

SimTalk ako aj iné programovacie jazyky pozná riadiace štruktúry. Základné vzory týchto štruktúr nájdete po otvorení nejakej metódy v hornom paneli v menu Method>Edit:



Bližší popis ako aj iné konštrukcie nájdete v helpe (pozrite si teraz):

Help: SimTalk Reference > SimTalk Reference > Control Flow Statements > Control Flow Statements

Takisto v helpe nájdete aj ďalšie potrebné informácie k programovaniu metód aj s príkladmi.

Pre ilustráciu využijeme if...else...end štruktúru. Z inštancie stanice **Stanica2** si otvorte zo záložky **User-defined** metódu **OnExit** a upravte ju nasledovne:

```
if @.name = "Suciastka"
    @.move
else
    @.move(Stanica1)
end
```

Kód spôsobí to, že ak je meno prvku, ktorý chce opustiť **Stanicu2** „Suciastka“, tak metóda pošle tento prvok pozdĺž konektora do **Drain1**. Inak ju pošle do **Stanice1**.



Uložíme zmeny cez **Apply Changes** a môžeme metódu zavrieť.

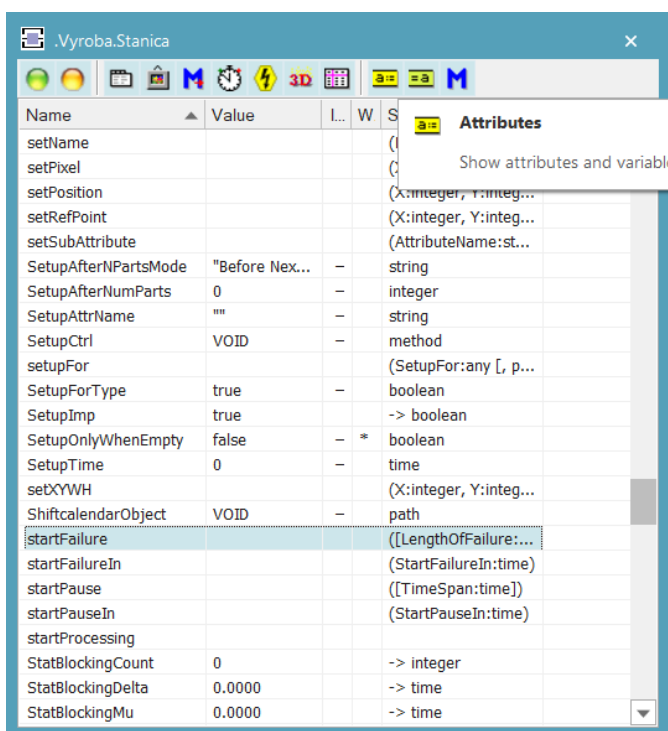
Pri programovaní metód často využijete klávesovú kombináciu **Ctrl + medzerník**, čo spôsobí **doplnenie** známych názvov, atribútov, metód, kľúčových slov...

Zástupné symboly

- self – cesta k objektu, v ktorom je self napísané (objektom môže byť napr. metóda, stanica...)
- ? – cesta k objektu, ktorý spustil metódu, v ktorej je ? napísaný
- @ – cesta k inštancii pohyblivej jednotky, ktorá sa nachádza práve v objekte materiálového toku a vyvolala spustenie aktuálnej metódy.
- ~ – cesta k objektu, ktorý je v hierarchii objektov o úroveň vyššie než volajúci objekt
- self.~ – cesta k objektu, v ktorom je definovaný aktuálny objekt, v ktorom je napísané self.~
- @.~ – cesta k objektu, v ktorom sa práve nachádza inštancia pohyblivej jednotky, ktorá vyvolala spustenie aktuálnej metódy

Ďalší atribút

Pri riešení nasledovných úloh budete ešte potrebovať preskúmať metódu Stanice **StartFailure** (viď atribúty a metódy Stanice F8). Stlačte na nej F1.



Metóda okamžite spustí poruchu stanice, ktorá bude trvať zadaný čas.

Použitie pri programovaní:

<Cesta_ku_stanici>.startFailure(Trvanie_poruchy)

Cesta môže byť **absolútna**, alebo aj **zástupný symbol** (tak ako sme používali metódu move).

Trvanie sa udáva v **sekundách**.

Úlohy

Všetky úlohy môžete robiť do jedného projektu, pretože budú rozdelené do zvlášť priečinkov a modelov. Do AIS odovzdáte iba tento jeden projektový súbor.

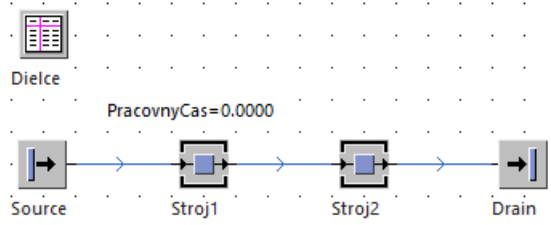
1. V Class Library si vytvorte novú zložku Uloha1, do ktorej si zduplikujete potrebné objekty. Použite vlastný duplikát Part (premenujte na Dielec1), Station (premenujte na Stroj), Model (premenujte na Model1). Trieda Dielec1 má mať červenú farbu. Trieda Stroj má mať pracovný čas 50s. Do modelovacej plochy Modelu1 vytvorte zostavu:



Zabezpečte aby sa Dielec1 presúval medzi stanicami zo Source až do Drain bez konektorov.

2. V Class Library si vytvorte novú zložku Uloha2, do ktorej si zduplikujete potrebné objekty. Potrebujeme Dielec1 červenej farby, Dielec2 zelenej farby a Dielec3 modrej farby. Ďalej potrebujeme triedu Stroj s rovnakým nastavením ako v 1. úlohe a tiež Model2.

Do modelovacej plochy Modelu2 vytvorte zostavu:



V tejto úlohe na konektoroch nezáleží. Môžete ich použiť. Pomocou tabuľky zadefinujte, že Source bude generovať prvky v nasledujúcom pomere: Dielec1 20%, Dielec2 30%, Dielec3 50%.

Ak bude v Stroji1 Dielec1, stroj bude mať pracovný čas 20s.

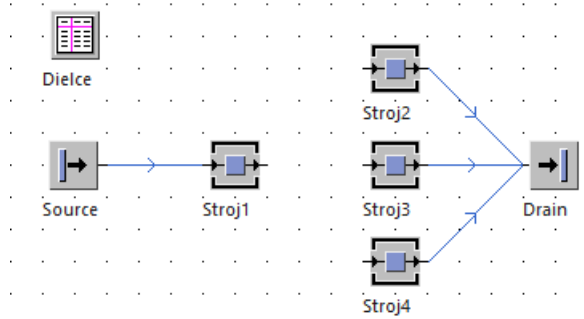
Ak bude v Stroji1 Dielec2, stroj bude mať pracovný čas 30s.

Ak bude v Stroji1 Dielec3, stroj bude mať pracovný čas 50s

Aktuálny pracovný čas zobrazte v globálnej premennej dátového typu time nad Strojom1.

3. V Class Library si vytvorte novú zložku Uloha3, do ktorej si zduplikujete potrebné objekty. Potrebujeme 3 dielce s takými istými farbami ako bolo definované v úlohe 2. Ďalej potrebujeme triedu Stroj s rovnakým nastavením ako v predošlých úlohách a tiež Model3.

Do modelovacej plochy Modelu3 vytvorte zostavu:



Pomocou tabuľky zadefinujte, že Source bude generovať prvky v takom pomere ako v predošlej úlohe. Stroj1 bude triediť Dielce do Strojov2, 3 a 4. Ak v Stroji1 bude Dielec1, tak hneď po jeho opustení stroja, sa Stroj1 dostane na 30s do poruchy.