

# Modelovanie a simulácia

## Dynamická simulácia

AnyLogic

7. cvičenie

# Ciele cvičenia

- Použitie objektov modelu v AnyLogic-u pre modelovanie:
  - Obmedzenie počtu čakajúcich zákazníkov vo fronte, s následkom predčasného odchodu z modelu
  - Spúšťanie udalostí, s cieľom riadiť interné premenné modelu
  - Zmeny stavu zdroja cez stavový diagram
  - Stanovenie 90% intervalu spoľahlivosti
  - Export hodnôt výstupného údaju do textového súboru
- Použitie nástroja Output Analyzer pre vykonanie párového t-testu
  - Pomocou tohto testu je možné zistiť, či sú výsledky dvoch experimentov naozaj odlišné na štatisticky významnej hladine (Praktický úvod do modelovania a simulácie, str. 261)

# Zadanie

- Modelovaný systém – doplnenie a úprava modelu z cvičenia 5
  - Vstupný prúd toku zákazníkov – zmena intenzity pomocou posuvníka
  - Návštevníci nie sú ochotní čakať v rade pred pokladňou, ak v ňom stojí už 50 návštevníkov, preto odchádzajú z múzea bez prehliadky múzea
  - Text s informáciou o % návštevníkov, ktorí odišli
  - Upratovanie vonkajšej expozície
    - Po každej tridsiatke návštevníkov
    - Trvanie 5 minút
    - Počas upratovania návštevníci čakajú pred turniketom

# Zadanie

- Modelovaný systém – doplnenie a úprava modelu z cvičenia 5
  - Turniket pred vnútornou expozíciou sa kazí
    - Priemerný čas medzi poruchami je 3 hodiny (exponenciálne rozdelenie)
    - Priemerný čas trvania opravy je 20 minút (exponenciálne rozdelenie)
    - Po oprave je testovaná funkčnosť – trvá 5 minút
  - Pre čas strávený návštevníkom pri čakaní v rade na vstupenky treba stanoviť 90% interval spoľahlivosti
  - Hodnoty priemerného času čakania treba po skončení replikácií zapísať do textového súboru
  - Experiment so zvýšeným vstupným tokom návštevníkov
  - Experiment so zvýšeným počtom sprievodkýň ovládajúcich cudzí jazyk

# Nové prvky simulačného modelu

- Posuvník pre riadenie vstupného toku zákazníkov – objekt *sliderPrichod*
- Popis k posuvníku – objekt *textVstupnyPrud*
- Premenná súvisiaca s riadením vstupného prúdu – objekt *varVstupnyTok*
- Text s informáciou o percente návštevníkov, ktorí odišli kvôli dlhému fronku – objekt *textOdisli1*
- Udalosti zatvorenia a otvorenia vonkajšej expozície – objekty *udalostZatvor*, *udalostOtvor*

# Nové prvky simulačného modelu

- Premenná pre pamätanie počtu návštevníkov vo vonkajšej expozícii – objekt *varVonkPrehliadkaPocet*
- Stavy turniketu do vnútornej prehliadky – objekty *stavFunguje*, *stavPokazeny*, *stavTest*
- Prechody medzi stavmi turniketu – objekty *transition1*, *transition2*, *transition3*
- Premenné pre výpočet polovičnej šírky intervalu spoľahlivosti, dolnej a hornej hranice intervalu spoľahlivosti pre čas čakania pred pokladňou – objekty *varPolSirkaIntSpolahlivosti*, *varDolnaHranicaCasCakania*, *varHornaHranicaCasCakania*

# Nové prvky simulačného modelu

- Rozhranie pre zápis výsledkov do textového súboru – objekt *suborVystupCasCakania*

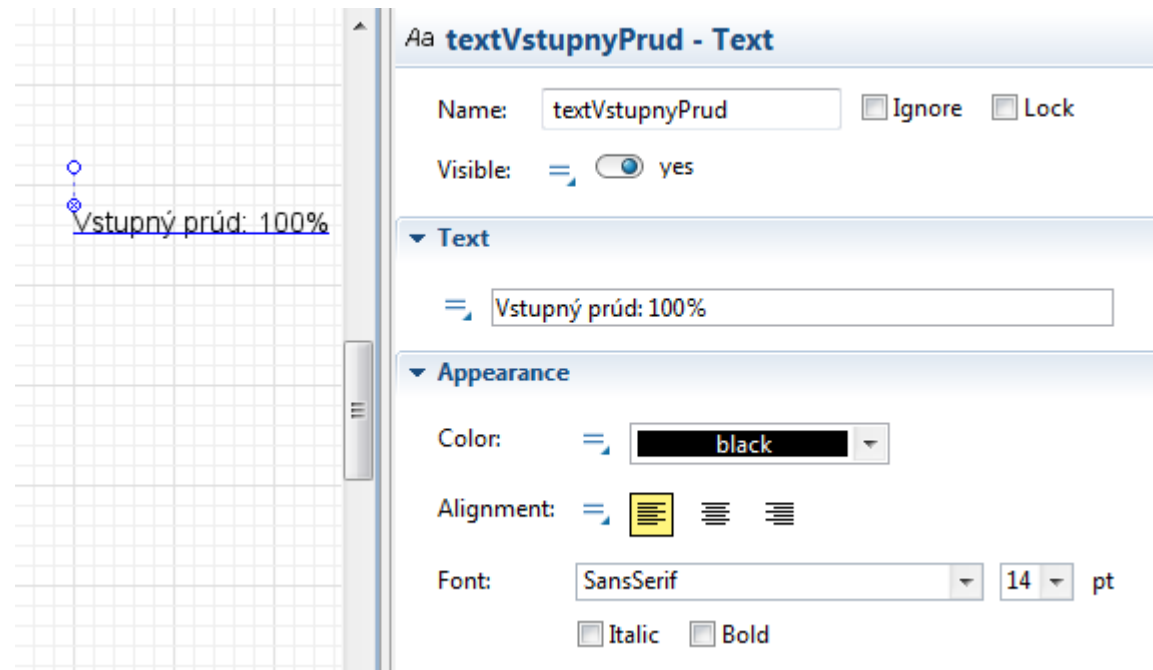
# Modifikované prvky simulačního modelu

- Objekt Obsluha



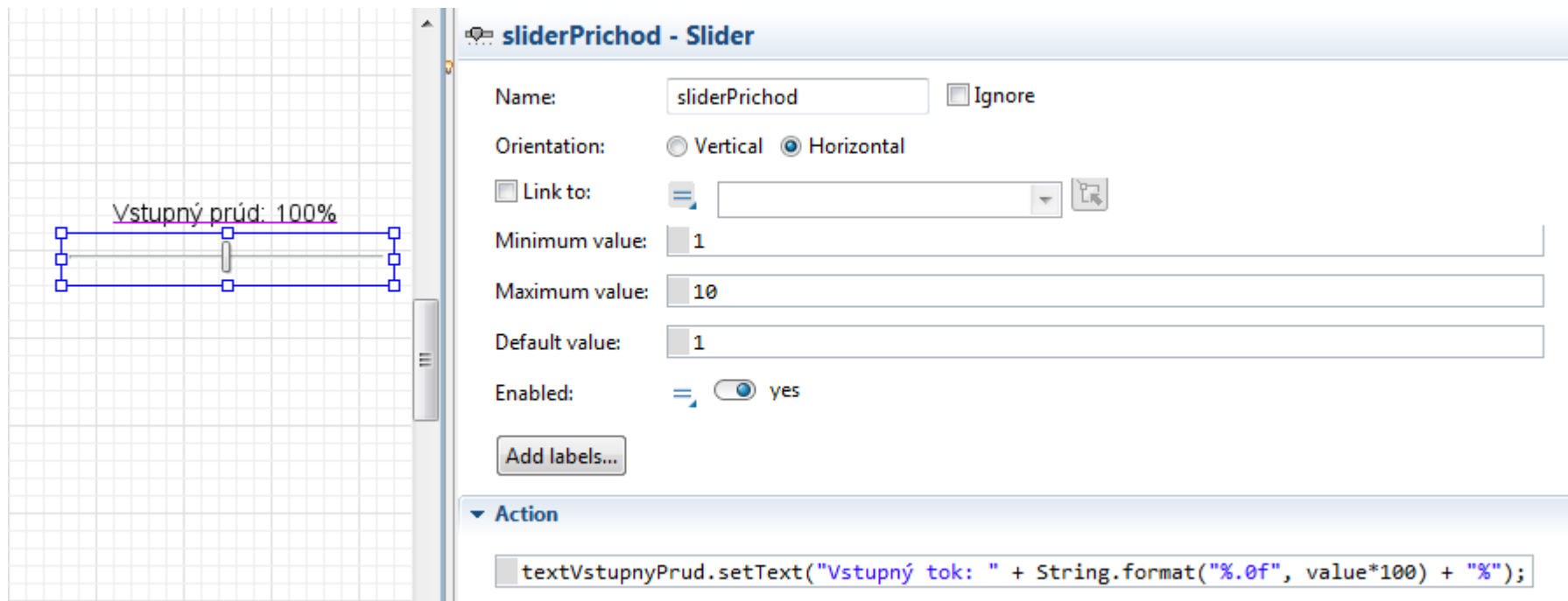
# Riadenie vstupného toku zákazníkov posuvníkom

- Trieda *Replikacie: Main*
- Vedľa tlačidla *Run* treba vložiť objekt **Text** z knižnice **Presentation**



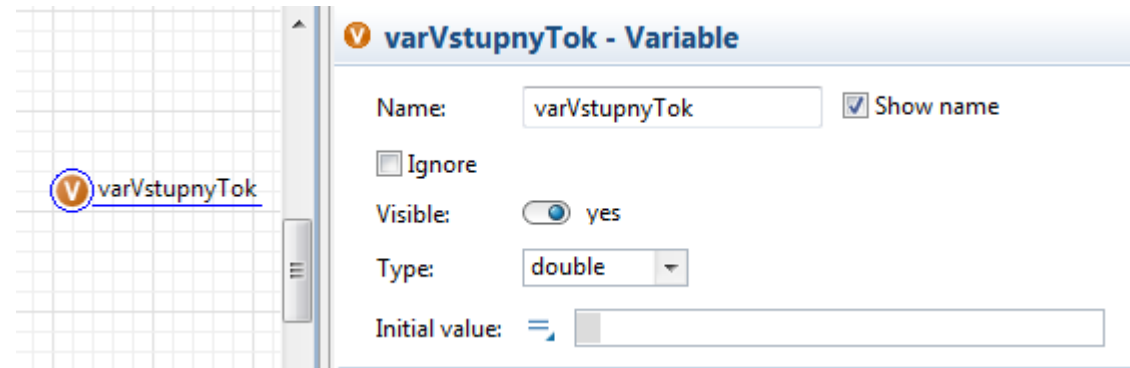
# Riadenie vstupného toku zákazníkov posuvníkom

- Trieda *Replikacie: Main*
- Objekt **Slider** z knižnice **Controls** - pod objekt *textVstupnyPrud*



# Riadenie vstupného toku zákazníkov posuvníkom

- Trieda *Main* – nový objekt **Variable** (knihnica **Agent**)



- Objekt *Prichod* – úprava nastavenia Interarrival time:  
 $\text{exponential}((\text{double})1/(100/(\text{varVstupnyTok})))$
- Trieda *Replikacie: Main* – Java actions – Before simulation run  
`root.varVstupnyTok = sliderPrichod.getValue();`

# Riadenie vstupného toku zákazníkov posuvníkom

- Posuvník mení vstupný tok zákazníkov iba pri spustení replikácií cez triedu *Replikacie: Main*
- Aby to podobne fungovalo aj pri spúšťaní simulácie cez triedu *Simulation: Main*, objekty **Slider**, **Text** a akciu **Before simulation run** treba definovať aj v triede *Simulation: Main*
- Alternatívou je ručné nastavovanie hodnoty premennej *varVstupnyTok* v triede *Main*



# Odchod zákazníkov pri prekročení kapacity frontu

- Úprava objektu *Obsluha*

**Obsluha - Service**

Name:  ☒ Show name ☐ Ignore

Seize: ☐ (alternative) resource sets  
☒ units of the same pool

Resource pool:   

Number of units:

Queue capacity:

Maximum queue capacity:

Delay time:

**Advanced**

Customize resource choice:

---

Queue: exit on timeout: ☒

Timeout:

Queue: enable preemption: ☒

Restore agent location on exit: ☒

---

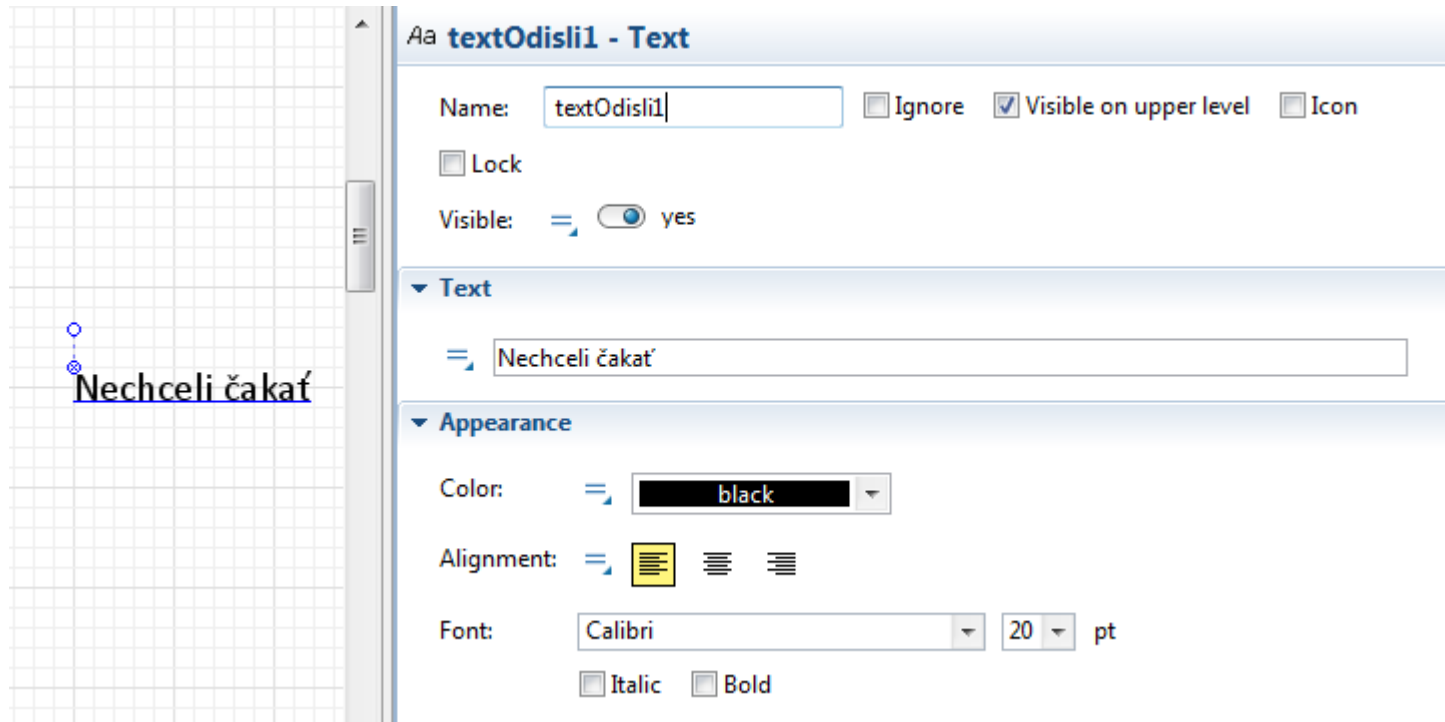
Force statistics collection:

# Odchod zákazníkov pri prekročení kapacity frontu

- Priorities/preemption – úprava Task priority:  
*(agent == null) || (agent.parTZP == true) ? 1 : 0*

# Text s aktualizovanou informáciou

- Objekt **Text** – knižnica **Presentation**



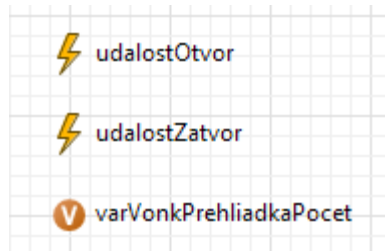
# Prepojenie textu a premennej

- Kód v sekcii **Actions** objektu *Obsluha*, z ktorého odchádzajú návštevníci pri prekročení kapacity frontu
  - On enter, On exit (preempted)

```
textOdisli1.setText(String.format("%.1f",  
((double)Obsluha.outPreempted.count()/Obsluha.in.count()) * 100) +  
"% zákazníkov\n odišlo pre dlhý rad");
```



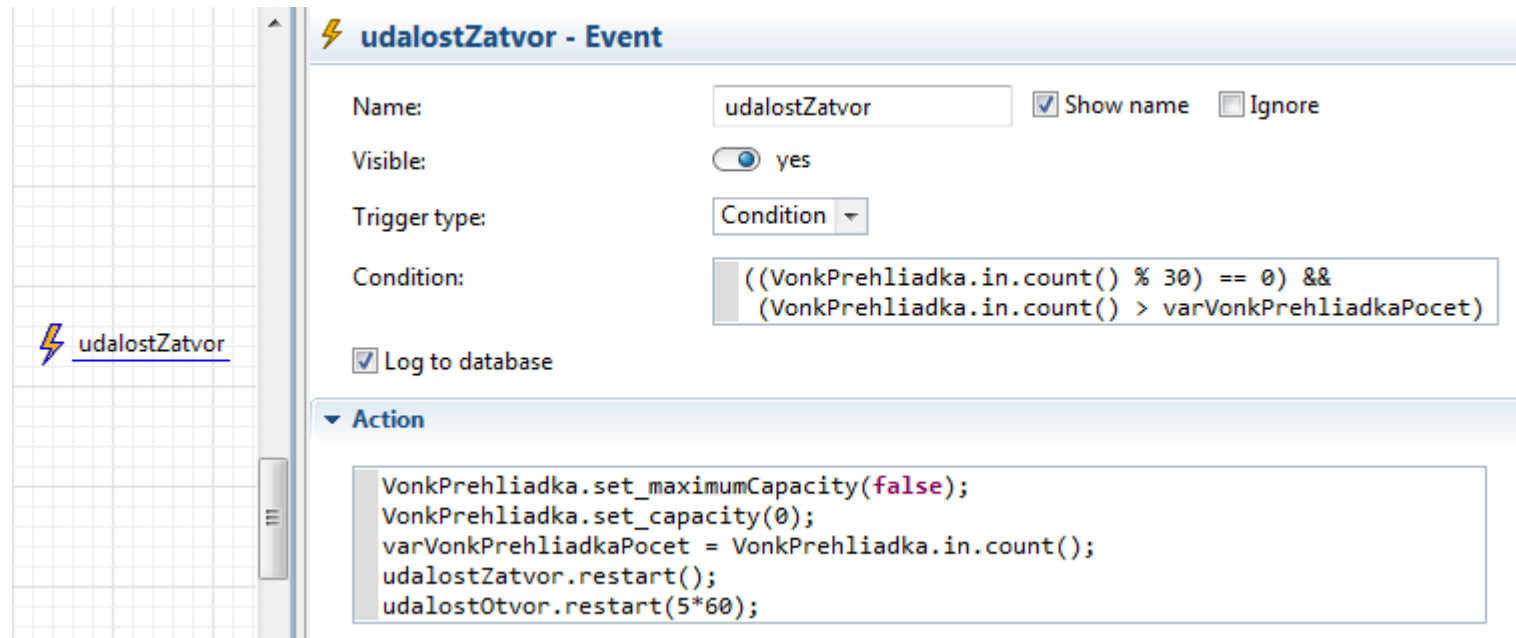
# Čistenie vonkajšej prehliadky pomocou objektu Event



- Objekt **Event** – knižnica **Agent**
- Jeden objekt – zatvorenie na začiatku čistenia
- Jeden objekt – otvorenie po skončení čistenia
- Premenná pre sledovanie počtu návštevníkov
- Objekty **Event** treba reštartovať, inak nastane udalosť iba raz!!!

# Čistenie vonkajšej prehliadky pomocou objektu Event

- Objekt *udalostZatvor*



**udalostZatvor - Event**

Name:  ☒ Show name ☐ Ignore

Visible: ☒ yes

Trigger type:

Condition:

☒ Log to database

**▼ Action**

```
VonkPrehliadka.set_maximumCapacity(false);
VonkPrehliadka.set_capacity(0);
varVonkPrehliadkaPocet = VonkPrehliadka.in.count();
udalostZatvor.restart();
udalostOtvor.restart(5*60);
```

# Čistenie vonkajšej prehliadky pomocou objektu Event

- Objekt *udalostOtvor*

**udalostOtvor - Event**

Name:  ☒ Show name ☐ Ignore

Visible: ☒ yes

Trigger type:

Mode:

☒ Use model time ☐ Use calendar dates

First occurrence time (absolute):

Occurrence date:

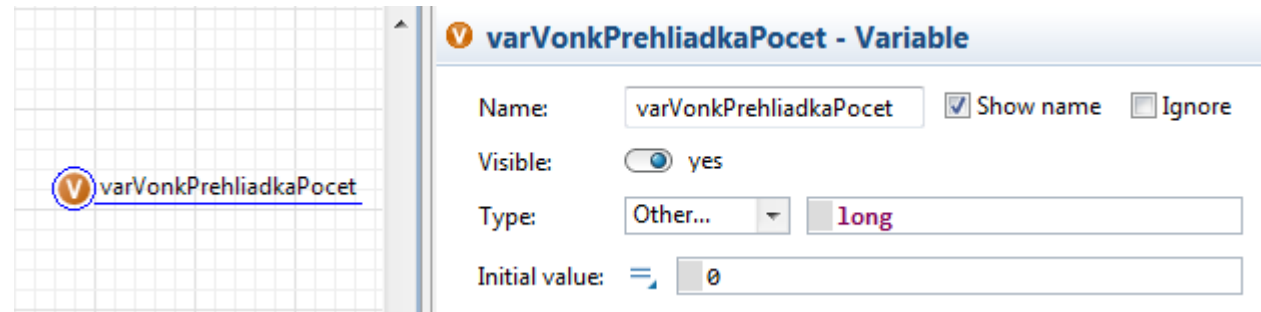
☒ Log to database

**Action**

```
VonkPrehliadka.set_maximumCapacity(true);
```

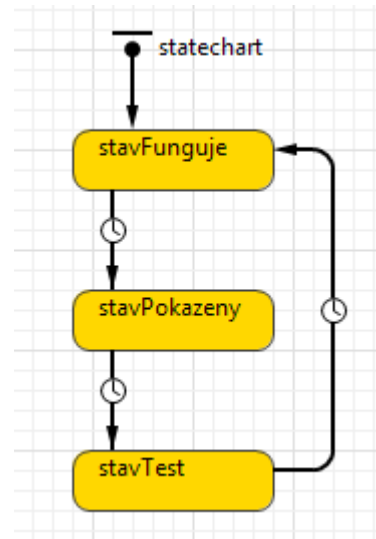
# Čistenie vonkajšej prehliadky pomocou objektu Event

- Objekt *varVonkPrehliadkaPocet*

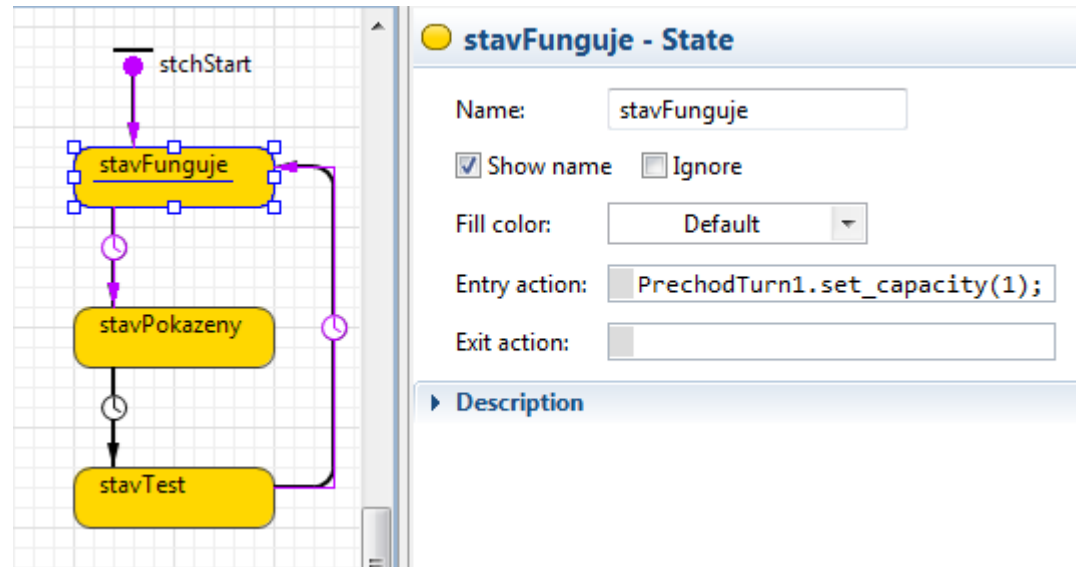


# Stavový diagram – knižnica Statechart

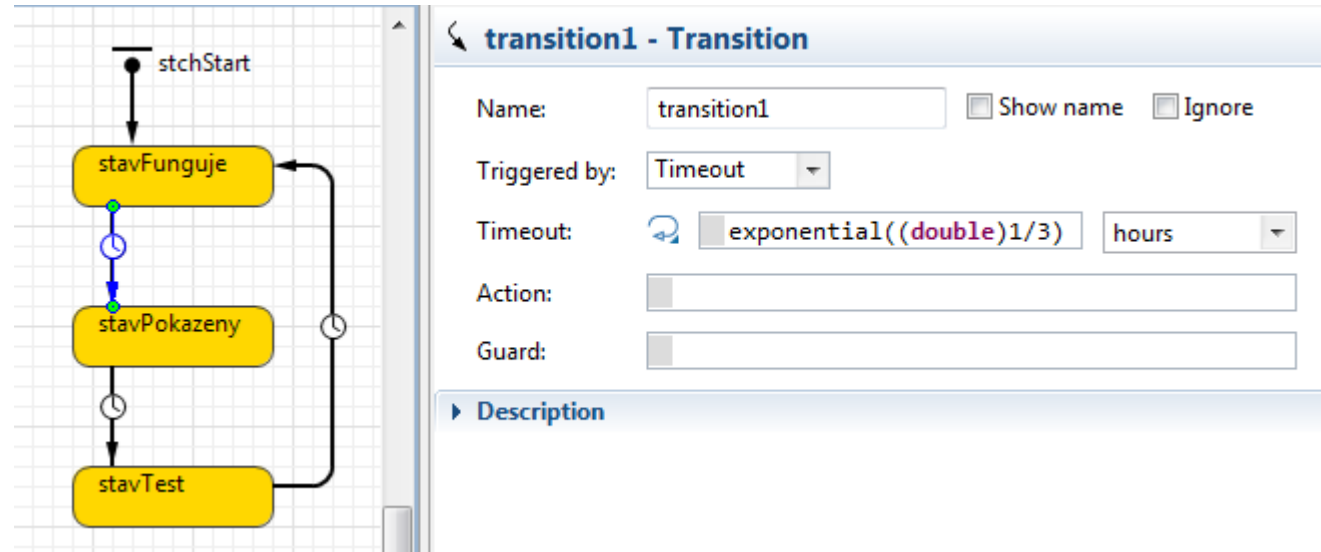
- Pomocou stavového diagramu môžeme meniť stavy zdroja (jeden z niekoľkých spôsobov)
  - v tomto prípade modelovanie náhodných porúch turniketu 1



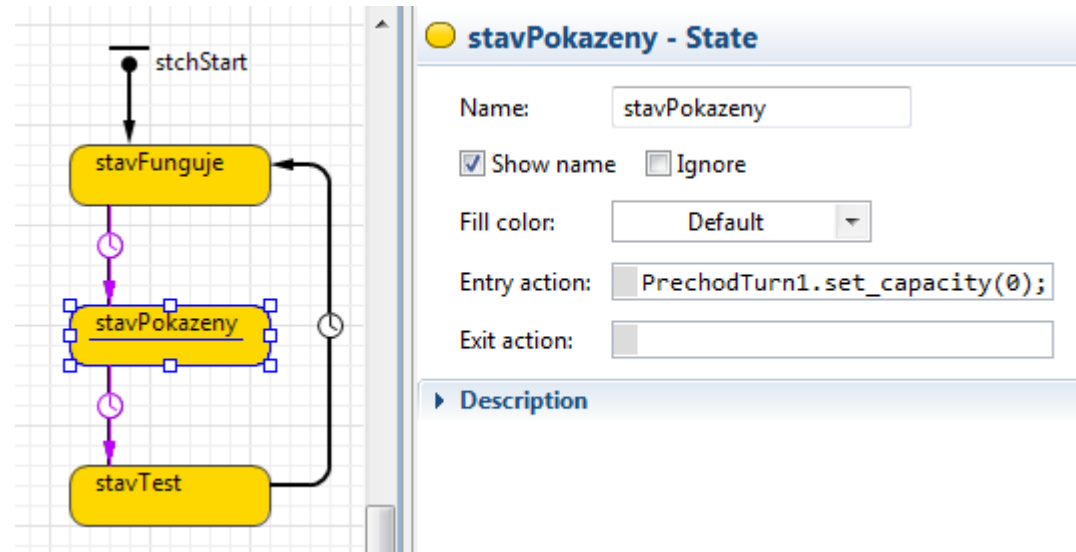
# Stavový diagram – stavFunguje



# Stavový diagram – transition1

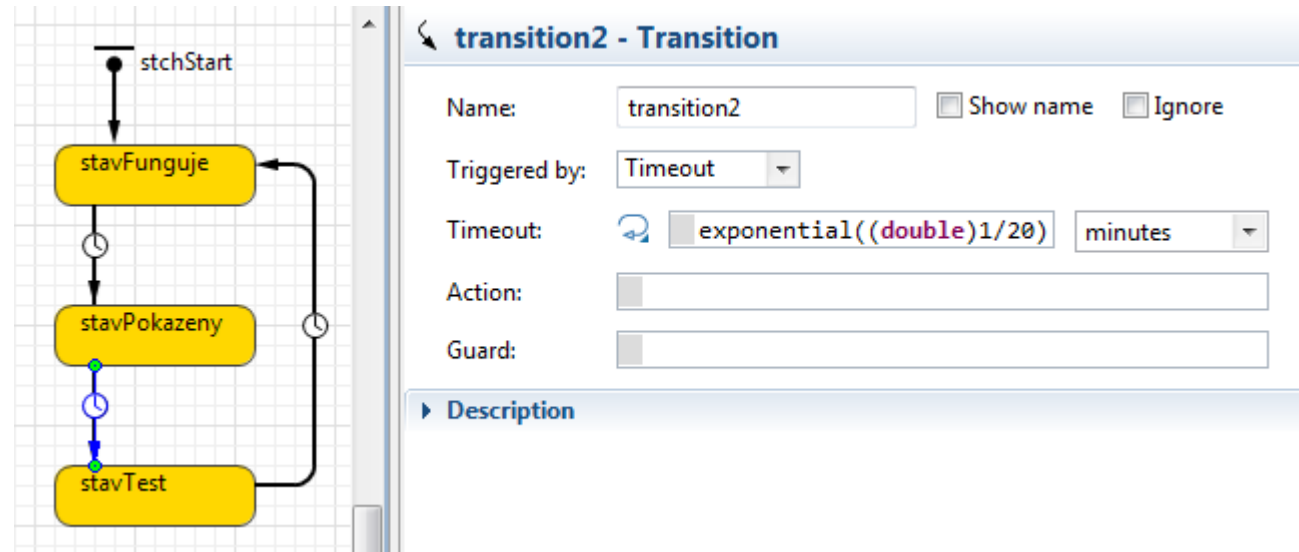


# Stavový diagram – stavPokazeny

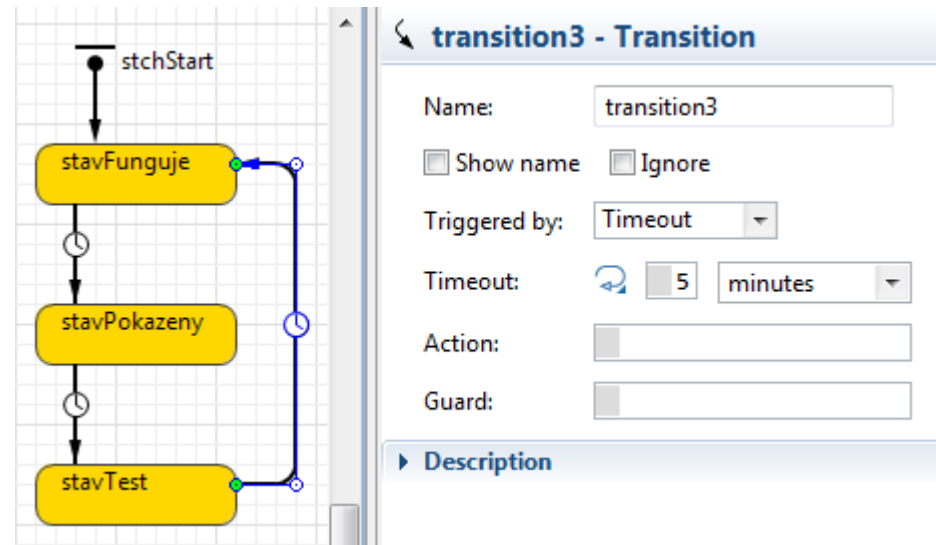




# Stavový diagram – transition2



# Stavový diagram – transition3

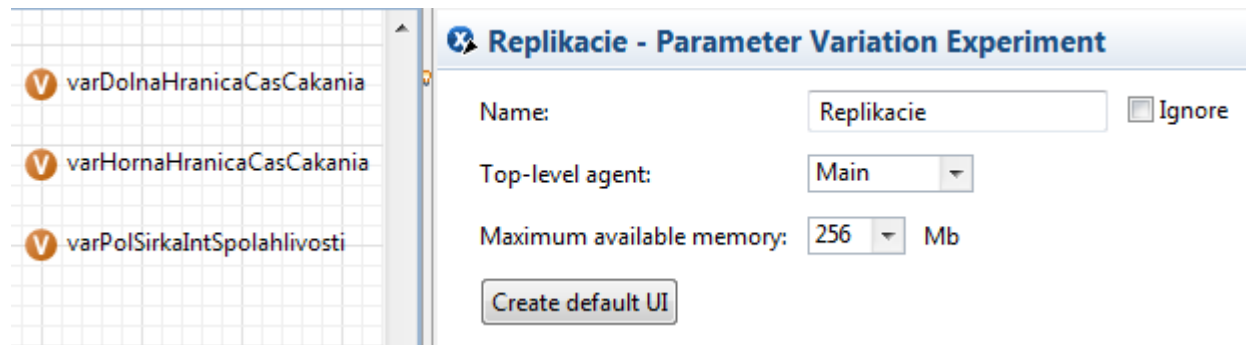


# Vrátenie štatistiky o priemernom čase čakania

- Do modelu treba vrátiť štatistiku pre sledovanie priemerného času čakania vo fronte pred pokladňou
- Objekt *hdCasCakania* – objekt typu **Histogram Data**
- Kód *hdCasCakania.add(time() – agent.parZacCakania)* do objektu *Obsluha*, **Actions – On enter delay**:
- Objekt *statPriemCasCakania* – v triede *Replikacie: Main*, pre výpočet odhadu strednej hodnoty času čakania za všetky replikácie
- Na niektorom z predchádzajúcich cvičení sme tieto objekty z modelu vymazali

# Stanovenie 90% intervalu spoľahlivosti

- Trieda *Replikacie: Main* – tri nové objekty typu **Variable** (knižnica **Agent**)

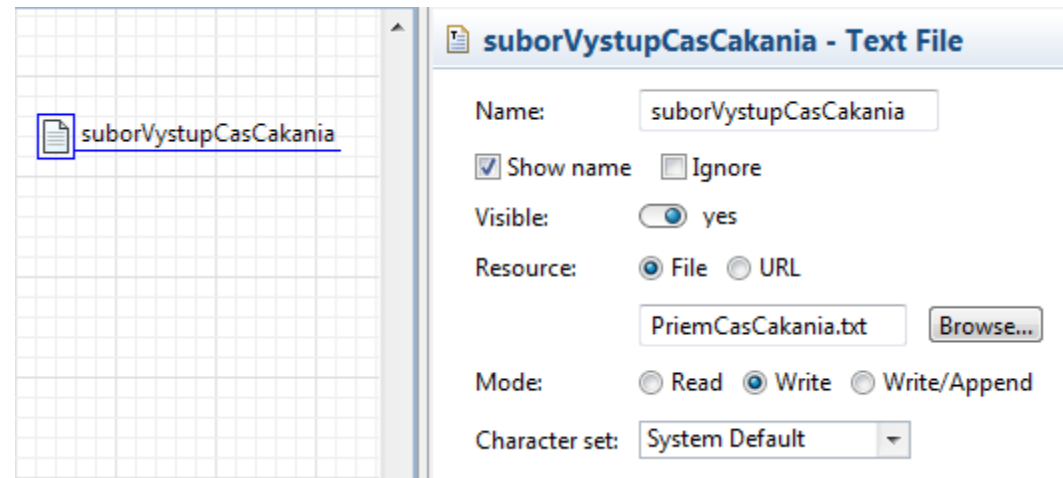


- Trieda **Replikacie: Main** – **After simulation run**

```
varPolSirkalntSpolahlivosti = (1.6449 *  
statPriemCasCakania.deviation()/sqrt(statPriemCasCakania.count()));  
varDolnaHranicaCasCakania = statPriemCasCakania.mean() -  
varPolSirkalntSpolahlivosti;  
varHornaHranicaCasCakania = statPriemCasCakania.mean() +  
varPolSirkalntSpolahlivosti;
```

# Zapisovanie výstupných údajov do textového súboru

- Objekt **Text File** – knižnica **Connectivity**



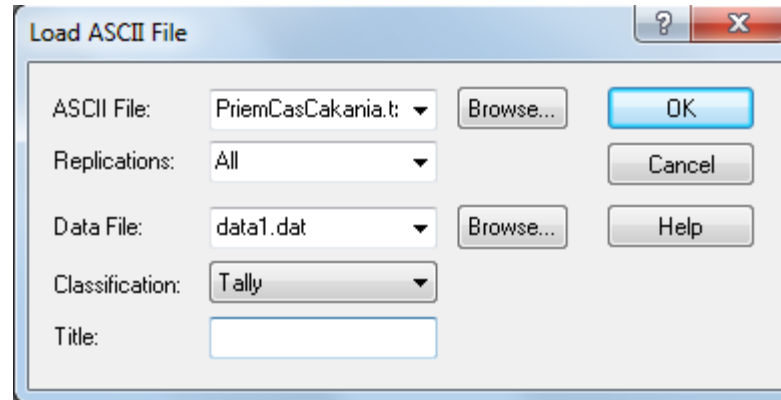
- Trieda **Replikacie: Main** – **After simulation run**  
*suborVystupCasCakania.println(statPriemCasCakania.count() + " " +  
root.hdCasCakania.mean());*

## Párový t-test – porovnanie výsledku dvoch experimentov

- Pomocou párového t-testu je možné zistiť, či sú výsledky dvoch experimentov naozaj odlišné na štatisticky významnej hladine
  - Experiment 1 – so vstupným tokom návštevníkov väčším o 30%
  - Experiment 2 – s troma sprievodkyňami ovládajúcich cudzí jazyk
  - Oba experimenty porovnané s pôvodným modelom (stav po strane 29 tejto prezentácie)
- 
- Test sa vykoná pomocou nástroja **Output Analyzer**
  - Je potrebné uložiť tri súbory s výstupnými údajmi – pre každý experiment jeden a jeden pre referenčný model

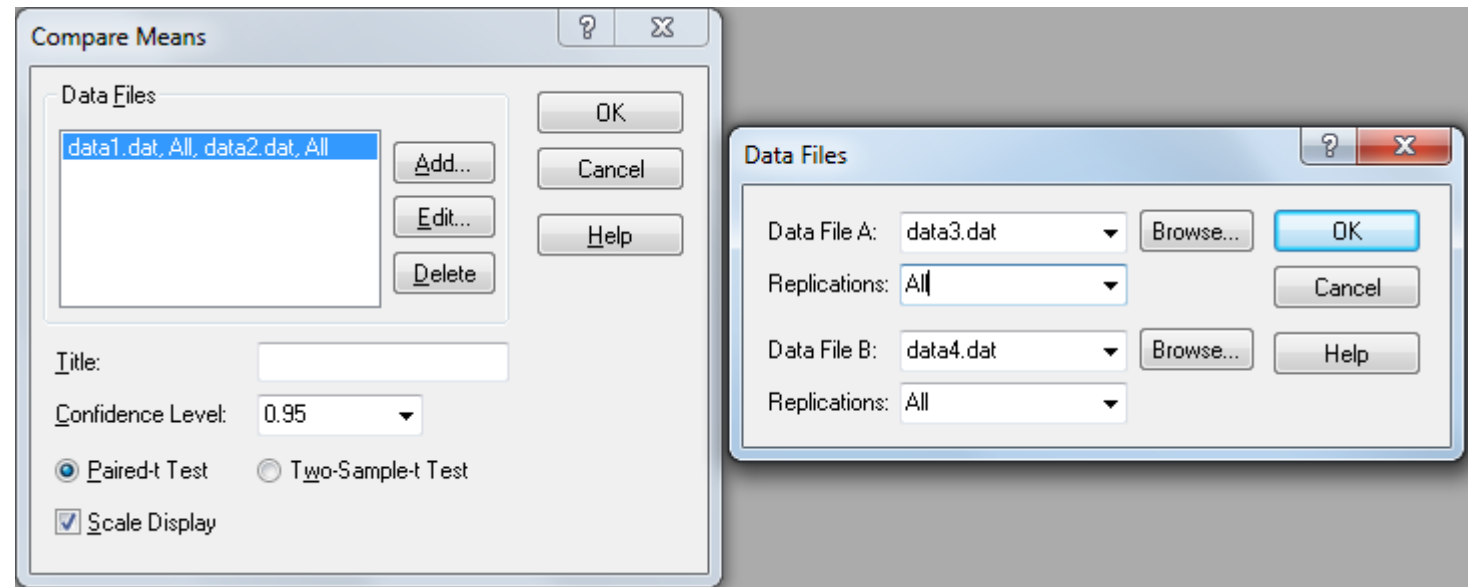
# Nástroj Output Analyzer – párový t-test

- Pripravené textové súbory treba konvertovať na „dátové“ súbory (s koncovkou \*.dat)
- **File – Data File – Load ASCII File...**



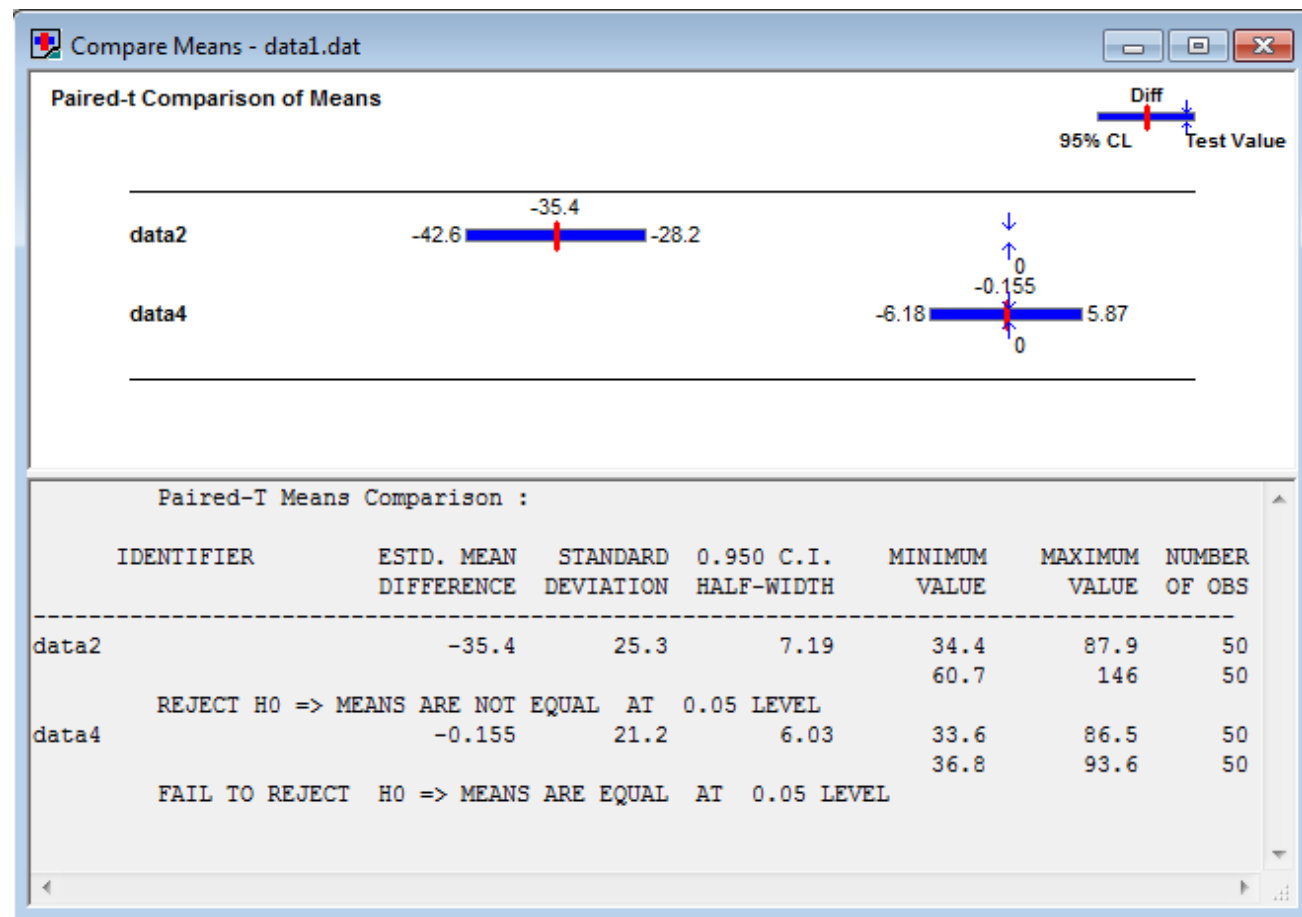
# Nástroj Output Analyzer – párový t-test

- **File – Analyze – Compare Means...**
- V dialógovom okne definovať dve dvojice porovnávaných experimentov
  - Vstupný prúd 100% a 130%
  - Sprievodkyne so znalosťou cudzieho jazyka – 2 a 3





# Výsledok párového t-testu



# Výsledok párového t-testu

- Čas čakania pri pokladni pri vstupnom prúde 100% je odlišný od času čakania pri vstupnom prúde 130% na štatisticky významnej hladine
- Čas čakania pri pokladni pri dvoch sprievodkyniach so znalosťou cudzích jazykov nie je odlišný na štatisticky významnej hladine od času čakania pri troch takých sprievodkyniach

**Koniec**