# Dokumentácia k zápočtovému programu

**Autor: Marek Bajtoš**
Predmet: NMIN201, Objektově orientované programování
Semester: Zimný semester 2016/2017
Dátum: 28.3.2017
Matematicko-Fyzikálna fakulta, Univerzita Karlova, Praha

**Názov práce: Diskrétna simulácia reštaurácie.**

## 1. Zadanie úlohy

Program, ktorý na základe vstupných parametrov bude simulovať jeden deň (8 hodín) chodu reštaurácie.

Na výstup vráti rôzne informácie o zákazníkoch, obsluhe a kuchároch danej reštaurácie. Z týchto informácií sa dá vyhodnotiť, či je pomer obsluha:kuchary:pocet stolov v reštaurácií ideálny, alebo naopak je potrebné v závislosti na možnostiach upraviť nejakú z týchto hodnôt. Napríklad ak máme pevne daný nemenný počet stolov, ku ktorým chceme určiť potrebný počet ľudí v obsluhe a v kuchyni (tak, aby hostia ktorý do reštaurácie prišli, ju neopustili nespokojný), budeme meniť iba hodnoty obsluhy a kuchárov.

## 2. Terminológia

**Zákazník / Hosť** – skupina ľudí (aj jeden človek), ktorí prišli spoločne do reštaurácie a na jedlo budú dokým nebude hotové pre každého človeka v skupine.

**Ľudia** – spoločne v skupinách prichádzajú do reštaurácie. Každý človek za seba má určené, aké jedlo si objedná.

## 3. Vstupné parametre

Program má na vstupe 3 hlavné vstupné parametre:

- Počet stolov
- Počet ľudí v obsluhe (počet čašníkov)
- Počet ľudí v kuchyni (počet kuchárov)

**Počet stolov:**
Presný počet stolov v reštaurácií, ktoré sú k dispozícií zákazníkom.

**Počet čašníkov:**
Počet ľudí, ktorí budú obsluhovať, teda prijímať objednávky, pripravovať nápoje, prinášať hotové jedlo z kuchyne, prijímať platbu za jedlo a upratovať stoly po odchode zákazníka.

**Počet kuchárov:**
Počet ľudí, ktorí budú pripravovať jedlá. Čašník im odovzdá informáciu, pre akého zákazníka potrebujú pripraviť koľko a akých jedál. Keď budú všetky tieto jedlá hotové, čašník ich zanesie hosťom.

### 3.1. Pevné predpoklady programu

**Predpoklady pre stoly a usádzanie:**
Každý stôl má, kvôli zjednodušeniu výpočtu, práve 4 miesta na sedenie. Pri jednom stole sedia vždy iba ľudia, ktorí prišli spolu. To znamená, že ak pri jednom stole sedia 2 ľudia, neprisadnú si k ním ďalší dvaja. Naopak, ak je ľudí v skupine viac ako 4, obsadia potrebný počet stolov tak, že jeden stôl môže ostať obsadený neúplne. Napríklad skupina 5 ľudí potrebuje 2 stoly. Druhý stôl sa už ale nebude doobsadzovať ľudmi z inej skupiny.

**Predpoklady pre zákazníka:**
Simulácia je nastavená tak, že každý zákazník, ktorý príde do reštaurácie, si najprv objedná jedlo a následne po dojedení si ešte môže objednať nápoj.

**Predpoklady pre obsluhu:**
Pre zjednodušenie predpokladáme, že všetci čašníci sú rovnako šikovní a teda všetky úkony im trvajú rovnako dlho, sú teda nerozoznateľní. Zároveň nerozlišujeme, či rovnaký čašník pre daného zákazníka prijme objednávku, donesie jedlo, prijme peniaze aj uprace stôl. Pre simuláciu je hlavné, aby tieto úkony boli urobené. Je nám jedlo, ktorý čašník ich vykoná.

**Predpoklady pre kuchárov:**
Rovnako ako pri čašníkoch, predpokladáme, že všetci kuchári sú od seba nerozoznateľný a všetci dokážu pripraviť všetky jedlá rovnako rýchlo.
Zároveň platí, že jedno jedlo pripraví jeden kuchár. Takže na prípravu objednávky pre napr. 5 ľudí, musí každý chod pripraviť jeden kuchár. Celkovo sa postupne zamestná 5 kuchárov. Po tom, čo posledný kuchár pripraví posledné jedlo pre daného zákazníka dá pokyn obsluhe na odnesenie zákazníkovi.

## 4. Použité triedy a ich vlastnosti

### 4.1. Class Model

Diskrétna simulácia bude prebiehať v triede **Model**.

**Premenné modelu:**

- **Čas** – riadiaci čas diskrétnej simulácie. Podľa tohto času zaraďujeme udalosti do kalendára a je aktualizovaný vyberaním udalostí z kalendára.
- **MaxCas** – maximálny čas, do kedy sa simuluje. Nastavený v konštruktore modelu na 8*60 = 480 minút (8 hodín).
- **MaxPocStol, MaxPocObsl, MaxPocKuch** - maximálne dostupné voľné hodnoty s ktorými pracujeme. Nastavené v konštruktore modelu. Berieme ich zo vstupných parametrov zadaných užívateľom.
- **MaxPotrebStolov, MaxPotrebObsluha, MaxPotrebKuchar** – premenné na udržiavanie maximálnych využitých kapacít počas výpočtu. Napríklad maximálny počet súčasne obsadených stolov, maximálny počet súčasne zamestnaných čašníkov... Na začiatku inicializované 0.
- **CisloZakaznika** – poradové číslo novo vygenerovaného zákazníka, po generovaní zákazníka sa zvýši o jedna. Na začiatku inicializované 1.
- **PoceVolStoly** – v danom stave výpočtu aktuálny počet voľných stolov v reštaurácií. Na začiatku inicializované počtom stolov na vstupe.
- **ZakazniciOdisli** – počet zákazníkov, ktorí odišli z reštaurácie nespokojný, kvôli dlhému čakaniu.
- **LudiaOdisli** – počet ľudí odišli nespokojný

- **ObsluzenychZakaznikov** – počet zákazníkov, ktorí odišli z reštaurácie spokojný. Teda zaplatili za jedlo.
- **ObsluzenychLudi** - počet ľudí, ktorí odišli spokojný
- **KalendarRestaurace** - kalendár udalostí v reštaurácií
- **VRestauraci** - zoznam zákazníkov, ktorí sú usadený v reštaurácií
- **FrontaNaVstupe** – fronta zákazníkov, ktorí ešte nesedia v reštaurácií, ale čakajú na uvoľnenie stola. Z tejto fronty sa po uvoľnení stola presunie zákazník do zoznamu *VRestauraci*
- **Kuchyna** – trieda spracúvajúca činnosť kuchárov. Podrobnejšie popísané nižšie.
- **Obsluha** – trieda spracúvajúca činnosť čašníkov. Podrobnejšie popísané nižšie.
- **Rnd** - Random, pomocou ktorého budeme generovať všetky náhodne generované parametre výpočtu.

### 4.2. Class Kalendar

Trieda na udržovanie zoznamu udalostí, z ktorého sa budú vyberať udalosti odohrávajúce sa v simulácií. Vyberá sa vždy udalosť, ktorá má najnižší čas. Tento čas sa použije na aktualizáciu času v modely.

### 4.3. Class Udalost

Udalosť, ktorú zadávame do kalendára reštaurácie. Obsahuje informácie o:

- Čase, kedy daná udalosť nastane
- Osobe, ktorá má danú udalosť spracovať
- Stav hosťa, podľa ktorého osoba, ktorá udalosť spracuje bude pri jej spracovaní postupovať.
- A číslo hosťa, ktorého sa daná udalosť týka.

Tieto informácie sú postupne v premenných udalosti: *kdy*, *kdo*, *akce*, *host*.

**Osoba:**
Osoba, ktorá môže spracúvať udalosť je hosť, čašník alebo kuchár **(enum Osoba).**

**Stav hosťa:**
Stav, v akom sa hosť počas svojho pobytu v reštaurácií môže nachádzať **(enum StavHosta).** Máme stavy:

- NovyHost
- Usadeny
- Objednane
- DoneseneJedlo
- CakaNaObjednanieNapoja
- DonesenyNapoj
- Zaplatim
- Odchod

Podľa týchto stavov sa budú diať jednotlivé kroky simulácie.

### 4.4. Class zakaznik

Zákazník, ktorý príde do reštaurácie.
Informácie, ktoré obsahuje:

- **Cislo** – číslo daného zákazníka. Pri generovaní nového zákazníka sa berie z čítača v premennej *CisloZakaznika* v modely.

- **PocetLudi** – počet ľudí v tejto skupine. Pri generovaní zákazníka sa náhodne vygeneruje v rozsahu 1 až 15.
- [] **jedlo** – pole s informáciou, aké jedlo si táto skupina zákazníkov objedná. Jedlo si vyberajú z 3 druhov. Reprezentujeme tak 3 druhy jedál, ktoré majú rôznu časovú náročnosť na prípravu. Každý človek si vyberie práve jedno jedlo. Súčet v poly jedlo sa teda rovná počtu ľudí daného zákazníka. Koľko ľudí si dá ktoré jedlo sa generuje náhodne pri generovaní nového zákazníka.
- **PocetStolov** – počet stolov, ktoré daný zákazník obsadí v reštaurácií. Je dopočítaná z počtu ľudí.
- **Stav** – aktuálny stav zákazníka v priebehu simulácie. Po generovaní má hodnotu NovyHost. Po usadení do reštaurácie sa zmení na Usadeny a ďalej na všetky stavy, ktoré môžu nastať.
- **Spokojnost** – indikátor toho, či bol daný zákazník spokojný s návštevou reštaurácie. Pri generovaní hosťa nastavený na 1, čo indikuje nespokojného zákazníka. Na spokojného zákazníka (na hodnotu 0) sa zmení v momente zaplatenia za jedlo. Ak sa k zaplateniu nedostane, teda odíde bez zaplatenia, pripočíta sa počet ľudí k počtu ľudí, ktorí odišli, inak sa pripočíta k obslúženým ľuďom.

### 4.5. Class Kuchyna

Trieda reprezentujúca kuchyňu v reštaurácií. Nachádzajú sa v nej kuchári, ktorí pripravujú jedlo pre zákazníkov.

Samotná trieda obsahuje nasledovné informácie:

- **PocetVolKuchar** – aktuálny počet voľných kuchárov. Na začiatku inicializovaný počtom kuchárov zadaných na vstupe.
- **FrontaObjednavokVKuchyni** – fronta objednávok, ktoré čašníci prebrali od zákazníkov a zadali ich kuchyni na prípravu, ale kuchyňa ich ešte nezačala pripravovať.
- **ZoznamPripravovanychObjednavok** – zoznam objednávok, ktoré sa aktuálne pripravujú.

Fronta aj zoznam sú zložené z **ObjednavkaVKuchyni.** Tá obsahuje tieto prvky:

- **Cisloobjednavky** – číslo zákazníka, pre ktorého je daná objednávka.
- **Jedlo** – pole s informáciou, aké jedlá sa majú pripraviť. Skopírované od zákazníka s daným číslom.
- **HotoveJedla** – počítač hotových jedál v danej objednávke. Vieme, že je objednávka hotová, keď CelkovoObjednanych sa rovná HovoteJedla.
- **CelkovoObjednanych** – súčet počtov jedál, ktoré sa majú v danej objednávke pripraviť**.**
- **Cas** – čas, v ktorom prišla daná objednávka na jedlo do kuchyne. Kuchyňa ich bude spracúvať v takom poradí, v akom prišli do kuchyne.

### 4.6. Class Obsluha

Trieda reprezentujúca čašníkov. Riadi prácu čašníkov.
Obsahuje tieto informácie:

- **PocetVolObsluha** – aktuálny počet voľných čašníkov. Na začiatku výpočtu inicializovaný počtom čašníkov zo vstupu programu.
- **ZoznamUlohPreObsluhu** – úlohy, ktoré čakajú na spracovanie čašníkom.

Úlohy pre čašníka tvoria nasledujúce informácie:

- **Cas** – čas, v ktorom prišla požiadavka na vykonanie danej činnosti na čašníka. Čašníci ich budú spracúvať od tých, ktoré prišli prvé.

- **cisloHosta** – číslo hosťa, ktorého sa daná úloha týka.

## 5. Algoritmus

Popíšeme chod algoritmu.

Ako v každej reštaurácií, čašníci a kuchári sa správajú podľa želania zákazníka. Preto sa celá simulácia riadi tým, v akom stave sa nachádza daný hosť a podľa toho sa zachová obsluha, v prípade objednania jedla aj kuchyňa.

V triede Program inicializujeme vstupné parametre do nového modelu pomocou konštruktora. Následne na danom modely spustíme funkciu Vypocet.

**Funkcia Vypocet:**

Výpočet sa začína tým, že do kalendára vložíme prvú udalosť, udalosť generovania nového zákazníka. Túto udalosť dáme do kalendára na prvú minútu.

Funkcia Vypocet je hlavná funkcia programu. Riadi celú simuláciu tak, že vyberá udalosti z kalendára podľa času spracovania, vždy vyberá udalosť s najnižším časom. Z kalendára vyberá, dokým nie je prázdny.

Po výbere udalosti z kalendára aktualizuje čas modelu časom tejto novej udalosti.

Podľa osoby, ktorá je zodpovedná za danú udalosť sa udalosť spracuje.

### 5.1. Generovanie hosťa

Vypocet sme začali vložením prvej udalosti. Keďže v kalendári inej niet, vyberie sa ako prvá z kalendára a spracuje. Osoba, ktorej udalosť patrí je Host, jeho stav v udalosti je NovyHost. Vo funckii **Vypocet** sa dostaneme do if vetvy pre spracovanie udalosti hosťom.

Zavolá sa funkcia **GenerujNovehoHosta**. Tá pridá do *FrontyNaVstupe* nového zákazníka. Ten sa náhodne generuje, teda sa generuje koľko ľudí a aké jedlá si dajú. Táto funkcia ešte novo pridaného zákazníka do fronty usadí do reštaurácie pomocou funkcie **UsadHosta**.

Vo funkcií Vypocet sa ešte pridá do kalendára reštaurácie udalosť na ďalšie generovanie zákazníka o niekoľko minút po poslednom generovaní. Do kalendára už nepridáme generovanie ďalších hostí 30 minút pred koncom pracovnej doby reštaurácie, teda 7 hodín a 30 minút od začiatku simulácie.

### 5.2. Usadenie (funkcia UsadHosta v Class Model)

Usadenie hosťa prebieha tak, že pokiaľ je počet stolov, ktoré potrebuje hosť na vrchu fronty, menší alebo rovný počtu voľných stolov, tak tieto stoly obsadí. Počet voľných stolov sa teda zníži o príslušnú hodnotu a zákazník sa presunie do zoznamu *VRestauraci*. Pri usadení si kontrolujeme aj to, či sa nezvýšil maximálny počet použitých stolov v jednej chvíli. To bude jedna z výstupných hodnôt.

Ak sa zákazník usadí, zmení sa jeho stav na Usadeny a zadá sa do kalendára udalosť pre čašníka (usadený hosť si bude chcieť objednať) ale aj udalosť pre hosťa (nastavenie trpezlivosti na pridlhé čakanie na čašníka).

!! Poznámka (viac zákazníkov usadených v rovnakom čase): V priebehu simulácie môže nastať prípad, že sa uvoľní viac stolov, keď odíde väčšia skupina a na vrchu fronty bude viacero malých skupín, ktoré sa chcú usadiť. Preto sa pri usádzaní pokúšame usádzať, dokým na vrchu fronty nebude zákazník, ktorý vyžaduje viac stolov, ako je aktuálne voľných.

### 5.3. Objednanie (Spracovanie úlohy pre čašníka pomocou funkcie ZamestnajObsluhu v triede Obsluha)

Z kalendára sa vyberie udalosť patriaca čašníkovi so stavom zákazníka Usadeny.

!! Poznámka (spracovanie udalosti obsluhou): Spracovanie udalosti patriacich obsluhe vždy spracuje funkcia **SpracujUlohu** v triede Obsluha. Tá v prípade, že číslo hosťa bolo -1, uvoľní jedného čašníka. -1 ako číslo hosťa nám bude indikovať, že udalosť v kalendári bola na ukončenie činnosti čašníka a ten sa stáva po jej skončení voľným. Druhá možnosť je, číslo hosťa je kladné. V takom prípade pridáme úlohu obsluhe do *zoznamUlohPreObsluhu,* ktorá sa týka zákazníka s daným číslom, ktoré bolo v pôvodnej udalosti vybranej z kalendára reštaurácie. Následne sa zavolá funkcia **ZamestnajObsluhu** v triede obsluha, ktorá v prípade, že máme voľného čašníka, vyberie zo zoznamu úloh pre čašníka tú najstaršiu a čašník ju spracuje podľa toho, v akom stave sa nachádza hosť, ktorého číslo bolo v úlohe pre čašníka.

Podľa predchádzajúcej poznámky teda udalosť z kalendára patriaca čašníkovi, kde je stav hosťa Usadený, najprv zaradíme do *zoznamUlohPreObsluhu,* a čakáme, kým sa nájde voľný čašník, ktorý danú úlohu spracuje. Pre prvého zákazníka tam samozrejme voľného čašníka máme hneď a tak sa hneď začne úloha vykonávať.

Vo funkcii **ZamestnajObsluhu** v triede Obsluha zo zoznamu úloh pre čašníka sa vybrala úloha pre čašníka, kde hosť s číslom uvedeným v danej úlohe v tomto zozname má v zozname *VRestauracii* v triede Model stav Usadeny. Do kuchyne zadáme objednávku na jedlo. Na to použijeme funkciu **PridajObjednavkuDoKuchyneVRestauraci** v triede Kuchyna. Do kuchyne sa pošlú informácie o čísle hosťa a aké jedlo si objednáva z poľa *jedlo*. Zmeníme stav daného hosťa na Objednane. Na pár minút obsadíme čašníka (teda pridáme do kalendára udalosť s číslom hosťa -1 o pár minút, kedy sa čašník uvoľní). Do kalendára ešte pridáme trpezlivosť zákazníka na čakanie na jedlo.

!! Poznámka: (zrušenie trpezlivosti zákazníka z kalendára): V prípade, že sa vyberie zo zoznamu úloh pre čašníka úloha, ktorú ide čašník spracovať, zrušíme z kalendára trpezlivosť hosťa na čakanie na čašníka, pretože čašník už k nemu prišiel. To robíme pre všetky prípady, kedy hosťovi nastavujeme trpezlivosť na čakanie (na objednanie, na jedlo, na objednanie napoja, na zaplatenie).

Na konci objednania je teda objednávka v kuchyni, čašník je na pár minút obsadený (čo predstavuje čas kým prijme objednávku + pripraví a donesie nápoje hosťovi) a zákazník má nastavenú trpezlivosť na čakanie na jedlo.

### 5.4. Príprava jedla (funkcie v Class Kuchyna)

Príprava jedla sa vykonáva v kuchyni. Preto všetky potrebné funkcie obsahuje samotná trieda kuchyňa.

Ako bolo spomenuté pri objednaní, čašník zavolá funkciu **PridajObjednavkuDoKuchyne** a tým sa do kuchyne dostane objednávka, ktorú treba pripraviť. Táto funkcia pridá objednávku najprv do *FrontaObjednavokVKuchyni.* Nasleduje cyklus, ktorý opakovane volá funkciu **ZamestnajKuchara**. Cyklus sa skončí v momente, keď sa po zavolaní funkcie **ZamestnajKuchara** nezmení počet kuchárov.

Samotná funkcia **ZamestnajKuchara** funguje nasledovne. Ide o to určiť, aké jedlo pre akú objednávku má voľný kuchár pripraviť v danej chvíli. Preto táto funkcia začína kontrolou, či máme voľného kuchára. Ak nie je voľný kuchár, tak sa nič neudeje. Objednávka, ktorá prišla do kuchyne ostáva vo fronte, kam bola umiestnená, a čaká, kým sa uvoľní kuchár.

Ak je *PocetVolKuchar* aspoň 1, prejdeme zoznam *ZoznamPripravovanychObjednavok*, či sa tam nenachádza objednávka, ktorá ma v poly *jedlo* nejakú nenulovú hodnotu. To by

znamenalo, že z danej objednávky už iný kuchár začal pripravovať nejaké jedlo a to znamená, že je potrebné ju dokončiť, aby mohla byť odnesená zákazníkovi.

Ak sa tam takáto objednávka nájde (čo zistíme tak, že premenná *Pom* zmenila hodnotu z -1 na číslo nezáporné označujúce pozíciu nájdenej objednávky v zozname *ZoznamPripravovanychObjednavok*), tak sa v danej objednávke pozrieme na to, ktoré z troch druhov jedál je ešte potrebné pripraviť. Hľadanie prebieha tak, že ideme od indexu 2 a pozrieme sa, či je hodnota na tomto indexe nulová. Ak je nulová ideme na ďalší index. Ak je nenulová ideme pripraviť jedlo na indexe 2. Podobne pre index 1. Ak na indexe 2 aj 1 je nulová hodnota, tak na indexe 0 už musí byť nenulová, pretože to bola objednávka, ktorej súčet na týchto troch miestach bol nenulový.

Na príslušnom indexe znížime počet objednávok o jedna.

Nakoniec pridáme do kalendára udalosť ukončenia prípravy daného jedla pre danú objednávku. Ak pripravujeme jedlo na indexe 2 bude jeho príprava trvať najdlhšie zo všetkých troch, teda 10 minút. Preto začíname výber jedla od indexu 2, aby sa príprava jedla začala s jedlom, ktorého príprava trvá najdlhšie. Pre jedlo na indexe 1 potrebujeme 7 minút na prípravu a pre jedlo na indexe 0 – 4 minúty.

Nakoniec sa zníži počet voľných kuchárov o 1.

Ak sa v zozname pripravovaných objednávok nenájde rozpracovaná objednávka, tak môžeme začať pripravovať novú objednávku. Teda sa overí, či je počet vo *FrontaObjednavokVKuchyni* nenulový. V takom prípade máme vo fronte objednávku, ktorú môžeme začať pripravovať.

Z fronty vyberieme tú objednávku, ktorá prišla ako prvá z tých, ktoré tam sú. Ďalej sa postupuje podobne, ako pri priradení práce kuchárovi v prípade rozpracovanej objednávky v *ZoznamPripravovanychObjednavok*. S tým rozdielom, že sa po pridaní udalosti na dokončenie jedla daná objednávka upravená o zníženie počtu jedál na indexe v poly *jedlo* podľa toho, ktoré jedlo sme začali pripravovať, ešte pridá do *ZoznamPripravovanychObjednavok.* Ak bolo v tejto objednávke viac ako 1 jedlo, tak ďalší kuchári, ktorí si budú brať prácu, budú pracovať na tejto novej objednávke, dokým sa všetky indexy v poly *jedlo* nevynulujú.

### 5.5. Dokončenie prípravy jedla (funkcia Vypocet v Modely, spracovanie udalosti kalendára pre kuchára + funkcia SpracujHotoveJedlo v triede Kuchyna)

Všetky jedlá, ktoré kuchári pripravujú sa postupne pripravia. Ukončenie prípravy jedla nám ohlási kalendár tým, že vyberieme udalosť pre kuchára, ktorá nám hovorí, že jedno jedlo pre hosťa s daným číslom bolo hotové. Tu už nerozlišujeme, ktoré jedlo z troch možných je hotové.

Takto sa postupne vyberajú udalosti pre kuchára, ktoré spracuje funkcia **SpracujHotoveJedlo** v triede Kuchyna. Táto funkcia cyklom prejde zoznam objednávok, ktoré sa v kuchyni pripravujú, teda v zozname *ZoznamPripravovanychObjednavok,* a nájde hosťa s číslom, ktoré prišlo v udalosti z kalendára. Tomuto hosťovi v tomto zozname zvýšime o jedna počet hotových jedál v premennej *HotoveJedla*. Tým sme označili, že sa danému zákazníkovi dokončilo jedno jedlo.

Tým, že je jedlo dokončené, sa nám uvoľní kuchár. Zvýšime o jedna počet kuchárov a zavoláme funkciu **ZamestnajKuchara**, ktorú sme už popísali.

Nakoniec overíme, či počet *HotoveJedla* a *CelkovoObjednanych* u daného zákazníka je rovnaký. Ak to nastane, znamená to, že všetky jedlá, ktoré pre daného zákazníka majú byť pripravené sú hotové. Pridáme teda do kalendára úlohu pre čašníka. Tým mu dáme vedieť, že môže celú objednávku odovzdať hosťovi. Daného hosťa vyradíme zo *ZoznamPripravovanychObjednavok*.

### 5.6. Prinesenie hotového jedla zákazníkovi (funkcia ZamestnajObsluhu v triede Obsluha, spracovanie úlohy kedy stav zákazníka je Objednane)

Správa od kuchára sa štandardne dostane do kalendára, kde udalosť spracuje čašník tak, že sa volá funkcia **SpracujUlohu**. Tá pridá úlohu do *ZoznamUlohPreObsluhu*.

Keď sa nájde voľný čašník a je v poradí úloha zo zoznamu úloh pre obsluhu taká, že stav zákazníka je Objednané, tak to znamená, že kuchyňa dala pokyn, že je objednávka pre daného hosťa hotová a môžeme mu ju doniesť.

V takom prípade sa už pohybujeme vo funkcií **ZamestnajObsluhu** v triede obsluha. V switch vetve sa dostaneme na StavHosta.Objednane a to vykonáme. Teda stav hosťa sa zmení na DoneseneJedlo. Čašníkovi sa uvoľní o 2 minúty, pridáme teda do kalendára udalosť s číslom hosťa -1 o 2 minúty. Tento čas potrebuje na prinesenie jedla zákazníkovi.

Nakoniec sa nastaví zákazníkovi čas, kedy je. Nie je to teda trpezlivosť.

### 5.7. Dojedenie jedla (funkcia Vypocet v triede Model, osoba Hosť, stav DoneseneJedlo)

Po čase, ktorí zákazník potreboval na jedenie sa môžu stať 2 veci. Buď je čašník spokojný s jedlom a odíde z reštaurácie alebo si ešte objedná ďalší nápoj a ostane sedieť.

O tom, ktorá situácia nastane sa rozhodne náhodne, pomocou generovania náhodného čísla do premennej *OdchodAleboNapoj*. S 20% pravdepodobnosťou si objedná ešte nápoj, inak odchádza. Objednať nápoj si však môže iba v prípade, že čas v simulácií ešte nedosiahol maximálneho času. V opačnom prípade musí odísť, aj keby si ešte chcel objednať.

Objednanie nápoja bude popísané o chvíľu. Odchod zákazníka necháme na poslednú kapitolu algoritmu až po zaplatení.

### 5.8. Objednanie nápoja

Pokračujeme na mieste, kde sme skončili pri Dojedení jedla. Objednanie nápoja prebieha tak, že sa stav hosťa zmení na *CakaNaObjednanieNapoja*. Do kalendára sa pridá udalosť čašníkovi a trpezlivosť hosťa na čakanie na čašníka.

Keď dôjde na rad táto úloha pre čašníka, spracujeme ju vo funkcií **ZamestnajObsluhu** v stave hosťa *CakaNaObjednanieNapoja*. Zmení stav hosťa na DonesenyNapoj, čašníkovi dá do kalendára udalosť na uvoľnenie o pár minút, kým pripraví a donesie nápoje zákazníkovi. Tento čas sa počíta ako minúta za každého človeka v danom zákazníkovi plus minúta. K tomuto času sa pridá ešte 15 minút na vypitie nápojov a s týmto časom sa do kalendára pridá ešte udalosť pre zákazníka. Táto udalosť znamená dopitie nápoja.

### 5.9. Vypitie nápoja

Po uplynutí času na pitie, teda keď sa z kalendára vyberie udalosť pre zákazníka so stavom zákazníka DonesenyNapoj, nasleduje rovnaký postup ako v prípade stavu DoneseneJedlo. Zákazník ma opäť rovnakú možnosť si objednať ďalší nápoj, ak ešte má čas, alebo odísť z reštaurácie. Popisuje to odsek 5.7.

### 5.10. Zaplatenie

V prípade, že zákazník v odseku 5.7 neobjednal nápoj, tak reštauráciu chce opustiť, a teda musí zaplatiť. V triede Model vo funkcii **Vypocet** sa na mieste spracovania udalosti pre hosťa so stavom DoneseneJedlo a DonesenyNapoj dostaneme do else klauzule.

Stav hosťa sa zmení na Zaplatím. Do kalendára pridáme udalosť pre čašníka s číslom daného hosťa. Keďže ešte musíme čakať kým sa nájde voľný čašník, ktorý prijme platbu, zadáme aj trpezlivosť na čakanie.

Keď sa opäť nájde voľný čašník, ktorý spracuje úlohu so zákazníkom, ktorého stav je Zaplatim (opäť v triede obsluha, funkcia **ZamestnajObsluhu**, stav hosta Zaplatim) zmení stav zákazníka na Odchod. Jeho spokojnosť ale nastaví na 0, čo znamená, že je spokojný. Do kalendára sa pridá udalosť pre čašníka o 4 minúty, ide o uvoľnenie čašníka, ktorý upratal stoly po hosťoch, ktorí odchádzajú. Do kalendára pridáme aj udalosť pre hosťa, pretože potrebujeme, aby sa uvoľnili stoly.

### 5.11. Odchod hosťa (spokojného = odišiel po zaplatení aj nespokojného = odišiel pred zaplatením kvôli vypršaniu trpezlivosti pri čakaní)

Vo funkcii Vypocet v triede Model sa vybrala z kalendára udalosť pre hosťa so stavom hosťa Odchod. Na spracovanie tejto udalosti zavoláme funkciu **ZakaznikOdchadzaZRestaurace**.

Máme ale 2 možnosti, ako sme sa k stavu hosťa Odchod dostali. Obe možnosti rieši už spomenutá funkcia **ZakaznikOdchadzaZRestaurace**.

**Prvá možnosť** je, že sa táto funkcia volá so stavom zákazníka pred zaplatením, takže v momente, keď sa z kalendára vyberie skôr trpezlivosť zákazníka ako to, že danú úlohu spracuje čašník.

To sa ešte delí na dve možnosti, prvá možnosť je, že hosť odišiel pred objednaním, alebo pred donesením jedla, v takom prípade nepotrebujeme samostatné zamestnanie čašníka na upratanie stolu a hosť sám odchádza. V tomto prípade, sa vybranie udalosti z kalendára so stavom hosťa Odchod ani nekoná. Hosť sám odchádza a započíta sa k hosťom, ktorí odišli (nespokojný hosť).

To, či treba stôl upratať je pri volaní danej funkcie dané premennou *Uklid*. Ak je 0 netreba upratať, ak je 1 treba upratať.

Druhá možnosť pri odchode pred zaplatením je, že odišiel až po jedle a teda stôl je potrebné upratať. Táto možnosť nastane pri vypršaní trpezlivosti v prípade, že čaká na objednanie nápoja, alebo čaká na zaplatenie. V tomto prípade ešte musíme volať čašníka. Toho už zavolá funkcia **ZakaznikOdchadzaZRestaurace**, pretože nastavíme vstupnú premennú *Uklid* na 1. Táto funkcia v tom prípade zmení stav hosťa na Odchod a zadá do kalendára udalosť pre čašníka. Čašník pri spracovaní úlohy, kde má hosť stav odchod, zadá do kalendára uvoľnenie čašníka o 4 minúty + o 4 minúty odchod hosťa. Spracovanie udalosti pre hosťa už po upratаní, teda spracovanie udalosti so stavom hosťa Odchod, je taký, že sa opäť volá funkcia **ZakaznikOdchadzaZRestaurace** ale tentokrát už bez upratania. V tomto prípade je zákazník nespokojný, takže zvýšime počet nespokojných ľudí a zákazníkov. Uvoľníme počet stolov, ktoré tento zákazník obsadil a odstránime ho zo zoznamu hostí *VRestauraci*. Ak je ešte čas do záverečnej, pokúsime sa tieto uvoľnené stoly znova obsadiť funkciou **UsadHosta** v triede Model.

!! Poznámka: (odchod hosťa, pred zaplatením) Je potrebné upozorniť, že v prípade, že hosť odchádza pred zaplatením, sa odstránia aj všetky „stopy“ po tomto zákazníkovi v simulácií. To znamená, že v závislosti na tom, v akom stave hosť pred zaplatením odchádza, odstraňujeme tohto hosťa aj zo zoznamov u obsluhy a v kuchyni, a odstránime aj všetky budúce udalosti s číslom tohto zákazníka z kalendára. Na to máme samostatné funkcie v triedach Obsluha a Kuchyňa, ktoré odstraňujú dané záznamy v zoznamoch. Tie podrobnejšie nebudeme popisovať. Ich funkčnosť je zrejmá.

**Druhá možnosť** je, že túto funkciu voláme po zaplatení. V tomto prípade je zákazník spokojný, teda jeho premenná *Spokojnost* je na 0. Tento prípad ďalej spracuje (po výbere udalosti z kalendára pre hosťa so stavom hosťa Odchod) funkciou **ZakaznikOdchadzaZRestaurace**. *Uklid* je na 0, teda už nevoláme čašníka (čašník uprace hneď, ako prijme platbu). Zvýšime počet spokojných zákazníkov. Uvoľníme stoly, ktoré obsadil daný zákazník a zákazníka odstránime zo zoznamu *VRestauraci*. Na záver sa pokúsime tieto uvoľnené stoly znova obsadiť funkciou **UsadHosta** v triede Model.

### 5.12. Záverečné slovo

Celá kapitola 5 je venovaná tomu, ako sa spracuje jeden zákazník počas behu simulácie. Takto nám postupne pribúdajú do reštaurácie hostia, ktorí sú spracovaný súčasne, pokiaľ máme voľného čašníka a voľné stoly. Takýmto spôsobom beží simulácia, dokým nedosiahneme maximálneho času simulácie, kedy už neprijímame objednávky a negenerujú sa noví hostia.

## 6. Výstupné hodnoty

Výstupnými hodnotami simulácie sú pre nás:

- **Počet obslúžených ľudí** (premenná *ObsluzenychLudi* v Modely) – počet ľudí, ktorí odišli z reštaurácie až po zaplatení.
- **Počet ľudí, ktorí odišli pred zaplatením** (premenná *LudiaOdisli* v Modely).
- **Maximálny počet využitých stolov** (premenná *MaxPotrebStolov* v Modely).
- **Maximálny počet súčasne zamestnaných čašníkov** (premenná *MaxPotrebObsluha* v Modely).
- **Maximálny počet súčasne zamestnaných kuchárov** (premenná *MaxPotrebKuchar* v Modely).

### 6.1. Čo chceme dosiahnuť?

Ako správny majiteľ reštaurácie chceme dosiahnuť, aby žiaden hosť, ktorí príde do reštaurácie neodišiel pred zaplatením. Chceme aby všetci čašníci, ktorých máme zamestnaných, boli naplno vyťažený a nemali sme tam žiadneho nadbytočného. To isté platí o kuchároch. Chceme zamestnávať minimálne nutné množstvo čašníkov a kuchárov, ktoré je nevyhnuté na to, aby sa všetci hostia obslúžili.

Naopak, ak máme daný pevný počet čašníkov alebo kuchárov, môžeme testovať, aký počet stolov je maximálne možné mať, aby sa žiaden čašník ani hosť zbytočne neflákal, ale rovnako aj, aby sme nemali navyše stoly, ktoré sa nikdy neobsadia.

Z týchto hodnôt následne získame približný počet hostí, ktorých je pri danom nastavení reštaurácia schopná obslúžiť. Keďže je veľká časť parametrov generovaná náhodne, tak pri opakovanom spustení simulácie s rovnakými vstupmi dostaneme rozdielne počty obslúžených ľudí. Rovnako ale môžeme pri viacerých spusteniach programu získať rozdielny počet potrebných kuchárov, čašníkov a stolov. To je tiež spôsobené náhodným generovaním určitých parametrov. Výsledkom sú teda približné hodnoty, podľa ktorých sa vieme riadiť, pri zamestnaní nových zamestnancov, alebo zväčšenia priestoru reštaurácie na väčšiu kapacitu. Z tohto dôvodu je možné, že pri zmene o jedného čašníka pri veľkom počte stolov nemusí byť výrazná zmena v počte obslúžených zákazníkov.

To, že dostaneme približné hodnoty nám samozrejme nijak nevadí, pretože ani v reálnom svete nedokážeme mať rovnaké hodnoty každý deň a preto nám to takto stačí.