Žilinská Univerzita v Žiline, Fakulta Riadenia a Informatiky

**Algoritmy a údajové štruktúry 1**

Semestrálna práca 1

2018/2019 Jozef Kubík

5ZY022

# Návrh aplikácie

## Použité údajové štruktúry

* Array
* ArrayList
* LinkedList
* ExplicitQueue

Prípady a dôvody použitia práve týchto údajových štruktúr:

**Array** – použitie na uchovávanie pointrov na jednotlivé prekladiská v triede *System*, a na uchovanie trasy v triede *Vozidlo*

*Dôvod použitia poľa:* v oboch prípadoch je pole vhodné, pretože počet regiónov Slovenska sa nemení a zároveň, v oboch prípadoch potrebujeme veľmi rýchly prístup na daný prvok, pretože pristupovanie k Prekladiskám a trase vozidla podľa indexu (číslo regiónu - 1) je veľmi časté

**ArrayList** – použitie ako zoznam vozidiel v triede *System*

*Dôvod použitia ArrayList-u namiesto LinkedList-u:* pri vozidlách je predpoklad, že pridávanie vozidiel bude veľmi málo časté a celkovo vozidiel bude málo, naopak pristupovanie bude veľmi časté

**LinkedList** – v triede *Prekladisko* uchováva zoznam *Dron-*ov a v triede *System* uchováva *Zásielky*

*Dôvod použitia LinkedList-u a nie ArrayListu:* predpokladom je, že zásielok bude rádovo viac než napr. *Vozidiel* aj *Dronov* a budú pribúdať veľmi často, pričom nebude potrebný prístup k jednotlivým zásielkam podľa indexu

V prípade *Dronov* sú predpoklady podobné – nepristupujem k jednotlivým dronom podľa indexu (zväčša prehľadám celý zoznam dronov a vyberiem najvhodnejší) a pridávanie môže byť častejšie (pravdepodobne častejšie taká firma kúpi nové drony než nové kamióny)

**ExplicitQueue** – použitie v triede *Dron* na uchovávanie časového rozvrhu – roznášanie zásielok / nabíjanie

*Dôvod explicitnej implementácie*: explicitná implementácia je vhodná, pretože môžu nastať 2 extrémy – dron celý deň nebude nič roznášať a naopak – dron bude musieť mať v rozvrhu veľa časových rámcov v prípade, že zásielky bude roznášať na krátke vzdialenosti a rovnako nabíjacie časy budú krátke

## Zložitosti všetkých operácií

**1. Pridanie nového vozidla** – výsledná zložitosť: **O(n)**

kde n – počet vozidiel v podniku

Pri vytváraní nového vozidla sa najskôr prejde celý ArrayList vozidiel v triede System – O(n), a hľadá sa zhoda ŠPZ, v prípade, že zhoda nie je, nové vozidlo sa priradí na koniec O(1)

*Toto neplatí v prípade, že sa ArrayList zväčšuje, vtedy je zložitosť O(2n)*

**2. Vypísanie zoznamu vozidiel podľa dátumu pridania** – výsledná zložitosť: **O(n)**

kde n – počet vozidiel v podniku

Vypísanie jedného vozidla má zložitosť O(1), vypísanie celého zoznamu je potom O(n)\*O(1)

**3. Pridanie nového dronu do prekladiska** – výsledná zložitosť: **O(n)**

kde n – počet dronov v prekladisku

Po zadaní všetkých parametrov dronu, pristúpime do správneho prekladiska – O(1), keďže pristupujeme do poľa podľa indexu, následne prejdeme LinkedList dronov O(n) a hľadáme, či sériové číslo už neexistuje, ak neexistuje pridáme dron na koniec LinkedListu – tj. O(1) – keďže náš LinkedList má pointer na posledný prvok

**4. Vypísanie všetkých dronov zo zvoleného prekladiska** – výsledná zložitosť: **O(n)**

kde n – počet dronov v prekladisku

Pristúpime do správneho prekladiska O(1), prejdeme celý zoznam dronov O(n), pričom vypisujeme jednotlivé drony O(1)

**5. Vytvorenie novej objednávky – výsledná zložitosť: O((n + N + 2\*m) + (2\*n + 2\*N)), kde:**

n – počet dronov v prekladisku odosielateľa

N – počet dronov v prekladisku prijímateľa

m – počet vozidiel firmy

Najskôr sa skontroluje, či nie je po 20:00 – O(1), následne sa prejdú všetky drony v prekladisku odosielateľa, či je aspoň jeden schopný zásielku transportovať – O(n), rovnako sa to urobí v prekladisku prijímateľa O(N)

Následne prejdeme zoznam vozidiel, či je nejaké vozidlo schopné transportovať zásielku do centrálneho skladu O(m), v prípade, že áno prejdeme týmto zoznamom ešte raz, či existuje vozidlo schopné transportovať zásielku z centrálneho skladu do prekladiska prijímateľa

Ak toto platí, prejdeme celý list dronov v prekladisku odosielateľa a hľadáme najvhodnejší voľný dron O(n), ak nenájdeme prejdeme tento list znova, ale už vyberáme podľa iných pravidiel – opäť O(n)

V prípade, že doba vyzdvihnutia zásielky neprekračuje 1 hodinu alebo prekračuje, ale zákazník ju nezruší zopakujeme tento proces v koncovom prekladisku – O(2\*N)

Ak dokážeme zásielku doručť do 18:00 (v cieľovom prekladisku), naložíme zásielku do už vybraných vozidiel 2\*O(1), zaradíme ju do rozvrhov dronov (na dnes aj na zajtra) – 2\*O(1)

*Výsledná zložitosť je dosiahnutá, iba ak prejde celý proces vytvárania novej objednávky. Napr. v prípade, že objednávku zrušíme hneď na podmienke, že je po 20:00 a teda ju nemôžeme transportovať, bude zložitosť O(1) – iba ju zoradíme na posledné miesto LinkedListu O(1) a nastavíme jej atribúty ako zamietnutá O(1)*

**6. Naplánovanie vyzdvihnutia zásielky u odosielateľa**

Táto funkcionalita je obsiahnutá v bode 5

**7. Návrat vozidiel do centrálneho skladu – výsledná zložitosť: O(n)**

kde n – počet vozidiel

*Trasu majú vozidlá určenú na pevno, nemení sa,* preto v tomto kroku iba prejdem všetky vozidlá v ArrayListe O(n) a každému aktualizujem prevádzkové náklady O(1)

**8. Naplnenie vozidiel v centrálnom sklade – výsledná zložitosť: O(2\*n)**

*Táto funkcionalita opäť počíta s premenlivými trasami, ktoré však nemám implementované (podľa M. Vargu to nie je potrebné)*, zvyšok funkcionality je už implementovaný v bode 5 a jeho zložitosť by mala byť O(2\*n)

**9. Transport zásielok z centrálneho skladiska do lokálnych prekladísk**

Rovnako aj táto funkcionalita je obsiahnutá v bode 5

**10. Odovzdanie zásielky adresátovi**

Jej časť je už v bode 5 (výber dronu), samotné odovzdanie by malo zložitosť **O(n)**, kde n je počet časových rámcov v drone, za predpokladu, že všetky časové rámce by mali čas <= 60 minútam

**11. Štatistiky**

**A) Región s najviac doručenými zásielkami za daný čas – výsledná zložitosť: O(n)**

kde n – počet všetkých zásielok

Musím prejsť všetky zásielky O(n) pričom kontrolujem, či sú doručené, či spadajú do časového obdobia a následne pripočítavam ich množstvo podľa prekladísk (všetko O(1)), potom ešte hľadám v poli vypočítaných výskytov maximum, čo je O(25 – počet regiónov), to zanedbávame, nakoniec vypíšem počet zásielok a prislúchajúci región, ktorý som práve našiel, obe O(1)

**B) Región s najviac odoslanými zásielkami za daný čas – výsledná zložitosť: O(n)**

kde n – počet všetkých zásielok

Princíp je úplne rovnaký, ako v predchádzajúcom bode, jediná zmena je samozrejme, že kontrolujem odoslané zásielky

**C) Výpis všetkých zrušených zásielok z prekladiska za daný čas – výsledná zložitosť: O(n)**

kde n – počet všetkých zásielok

Prechádzam všetkými zásielkami O(n) a zvyšujem počítadlo O(1), nakoniec ho vypíšem O(1) a pripíšem región O(1), toto zopakujem pre všetky regióny O(25)

**D) Vypísanie počtu zrušených zásielok zo všetkých prekladísk za daný čas – výsledná zložitosť: O(n)**

kde n – počet všetkých zásielok

Opäť musím prechádzať všetkými zásielkami, pričom kontrolujem či spadá do daného časového rámca, či je zamietnutá a či je z daného prekladiska – tieto kontroly majú zložitosť O(1), a výpis výskytov podľa regiónov je vždy O(25), preto je výsledná zložitosť iba O(n)

**E) Vypísanie celkového počtu doručených zásielok – výsledná zložitosť: O(n)**

kde n – počet všetkých zásielok

Prechádzam všetkými zásielkami O(n) a zvyšujem počítadlo O(1), nakoniec ho vypíšem O(1)

**G) Nalietané hodiny dronov podľa typu, zo všetkých prekladísk – výsledná zložitosť: O(n)**

V každom prekladisku prechádzam jeho list dronov O(n) a zvyšujem 2 počítadlá nalietaných hodín, nakoniec ich vypíšem O(1), toto zopakujem 25 krát – pre každé prekladisko

# Diagram tried

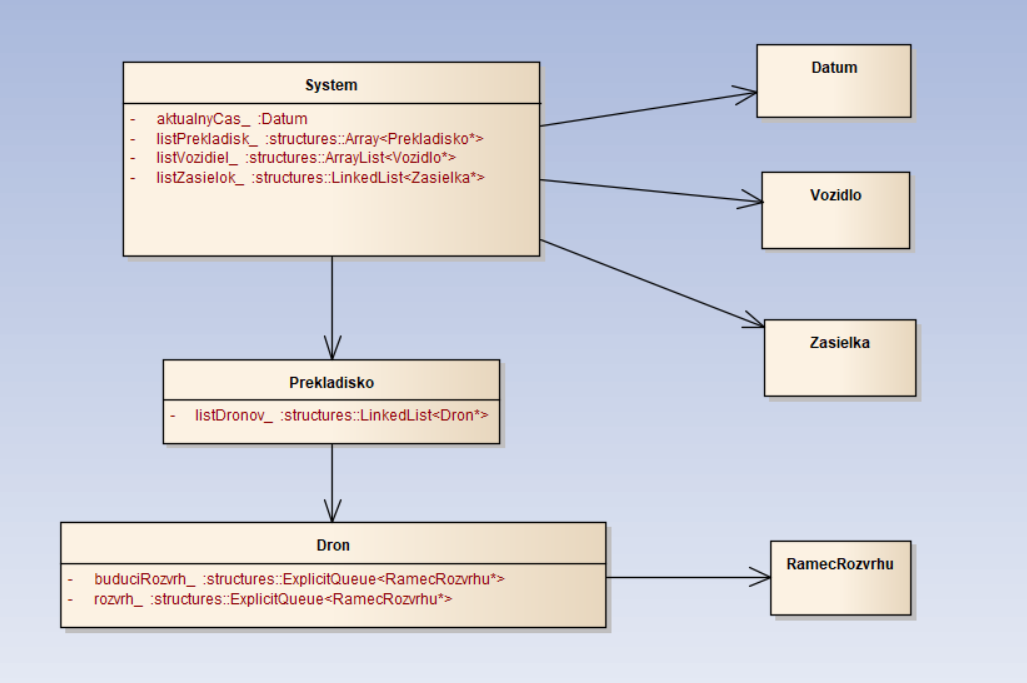


Diagram najdôležitejších súčastí systému (obsahuje všetky súčasti hlavnej triedy *System*)

# Používateľská príručka

Po spustení aplikácie si môže užívateľ zvoliť, či chce načítať predtým vytvorený a uložený systém (stlačením klávesy A) alebo chce vytvoriť nový systém (klávesa N).

Následne užívateľ obsluhuje program zadávaním číslic 0 až 9, v prípade desatinných čísiel sa používa desatinná bodka ‘**.**‘ (nie čiarka – ‘**,**‘). Na potvrdenie vstupu slúži klávesa Enter. Ovládanie je veľmi jednoduché a užívateľ vždy dostane hlásenie, aký vstup od neho aplikácia očakáva.