Západočeská univerzita v Plzni

Fakulta aplikovaných věd

Semestrální práce z předmětu Programovací techniky

Vypracovali: Runt Lukáš (A20B0226P, lrunt@students.zcu.cz) a Mlezivová Martina (A20B0038P, martimle@students.zcu.cz)

1 Obsah

1 Zadání

Mezinárodní přepravní společnost *Hlavně rychle s. r. o.* je vyhlášena ve své schopnosti co nejrychleji letecky přepravit jakýkoliv náklad odkudkoliv a kamkoliv. Tento rok tato společnost vyhrála výběrové řízení pro přepravu koní z celého světa do Paříže na nadcházející olympijské hry ve Francii. Jedná se však o poměrně komplexní záležitost, je třeba přepravit nejen koně, také krmivo a vodu. Koně musejí být také přepravovány s nadměrnou opatrností a každý má jiné potřeby a jiné vybavení s sebou k přepravě. Petr Rychlý, aktuální šéf společnosti, právě převzal vedení po svém otci. Zatím však netuší, že vlastní pouze omezený počet letounů, do kterých je možné uložit speciální boxy pro přepravu koní, přičemž letouny mají různou nosnost. Protože Petr (na rozdíl od svého otce) je impulzivní a vše řeší na poslední chvíli, myslí si, že koně jen nahází do letadla a vše stihne den před začátkem olympijských her. Pojďme za něj udělat tu tvrdou práci a spočítat vhodné trasy letadel tak, aby se vše stihlo včas a čest firmy byla i nadále zachována za takového předpokladu, že první letadlo vzlétlo se nejpozději. Poslední kůň musí být přepraven před samotným započetím olympijských her, budeme tedy uvažovat, že olympijské hry začínají dokončením přepravy všech koní. Známe:

- zeměpisné souřadnice Paříže *a* a *b,*
- zeměpisná souřadnice *x* a *y*, kde má přepravní společnost každého koně naložit, spolu s celkovou hmotnosti *m* (koně + vybavení) k naložení a dobu naložení a vyložení (každé trvá dobu *n*),
- počáteční zeměpisné souřadnice *X* a *Y* každého letounu spolu s maximálním povoleným zatížením letounu *M* a rychlostí letu *V.*

Uvedené souřadnice považujeme za kartézské, sférické souřadnice jsou velmi komplexní.

V rámci výpisu použijte jednu z následujících variant (formát je závazný, indexace od nuly):

- Letoun startuje:
  CAS: <T>, LETOUN: <1>, START Z MISTA: <X>, <Y>
- Letoun nakládá koně a bude nakládat dalšího:
  CAS: <T>, LETOUN: <1>, NAKLAD KONE: <K1>, ODLET V: <T+N>, LET KE KONI: <K2>
- Letoun naložil posledního koně a letí na olympiádu:
  CAS: <T>, LETOUN: <1>, NAKLAD KONE: <K>, ODLET V: <T+N>, LET DO FRANCIE
- Letoun koně vyložil a letí znovu:
  CAS: <T>, LETOUN: <1>, PRISTANI VE FRANCII, ODLET V: <T+∑N>, LET KE KONI: <K>
- Letoun přistál ve Francii a již dále neletí:
  CAS: <T>, LETOUN: <1>, PRISTANI VE FRANCII, VYLOZENO V: <T+ ∑N>

Výstup Vašeho programu bude do standardního výstupu. Dále čas ve výpisu bude zaokrouhlen dle pravidel zaokrouhlení na celé číslo.

1.1 Vytvoření funkčního programu

- Seznamte se se strukturou vstupních dat (souřadnicemi, hmotnostmi, rychlostmi,…) a načtěte je do vytvořeného programu. Formát souborů je popsán přímo v záhlaví vstupních souborů.
- Navrhněte vhodné datové struktury pro reprezentaci vstupních dat, zvažujte časovou a paměťovou náročnost algoritmů pracujících s danými strukturami.
- Proveďte základní simulaci jednoho letu, na závěr simulace vypište celkový počet přepravených koní během tohoto letu > 0. Let musí být smysluplný.
- Vytvořte prostředí pro snadnou obsluhu (menu, ošetření vstupů) – nemusí být grafické. Umožněte manuální zadání nového koně k přepravě nebo jeho odstranění jiného během simulace.
- Umožněte sledování (za běhu simulace) aktuálního stavu přepravy. Program bude možné pozastavit, vypsat stav přepravy, krokovat vpřed a nechat doběhnout do konce, podobně jako je tomu v debuggeru.
- Proveďte celkovou simulaci a vygenerujte do souborů následující statistiky (v průběhu simulace ukládejte data do vhodných struktur a následně uložte statistiky do vhodných souborů):
  - Přehled jednotlivých letounů – rozpis jaké koně kdy a kudy rozvážely, statistiku zatížení, dobu letu a prostojů (čas, jak dlouho trvalo koně nakládat).
  - Přehled jednotlivých koní – kudy a jak dlouho cestovali, jak dlouho před začátkem olympiády byli již na místě.
  - Celková doba všech letů, všech prostojů a celková doba přepravy.
- Vytvořte generátor vlastních dat. Generátor bude generovat vstupní data pomocí rovnoměrného rozdělení, rychlosti letu budou generovány ořízntutým normálním či extremálním rozdělením (zvolte jedno z uvedených rozdělení) s vhodnými parametry. Data budou generována do souboru (nebudou přímo použita programem) o stejném formátu jako již dodané vstupní soubory. Při odevzdání přiložte jeden dataset s méně než 100 koňmi a jeden s více než 1000 koňmi.
- Vytvořte dokumentační komentáře ve zdrojovém programu a vygenerujte programovou dokumentaci (Javadoc).
- Vytvořte kvalitní dále rozšiřitelný kód – pro kontrolu použijte softwarový nástroj PMD.
- Vytvořte dokumentaci daného programu.

2 Analýza problému

Ze zadání vyplývá, že se jedná o NP složitý problém. Můžeme zde vidět určitou podobnost s tzv. problémem obchodního cestujícího. Při řešení dané úlohy budeme vycházet ze znalosti, že tzv. problém obchodního cestujícího nemá jednoznačně dané řešení, tudíž se jedná o heuristiku. Existuje tedy více možností řešení daného problému. Budeme se snažit vytvořit algoritmus k optimálnímu řešení.

Hned na úvod nás čeká úkol, jak vyřešit načítání vstupních dat a jejich následné uložení do vhodných datových struktur. V úvahu připadá buď reprezentace polem nebo ArrayListem.

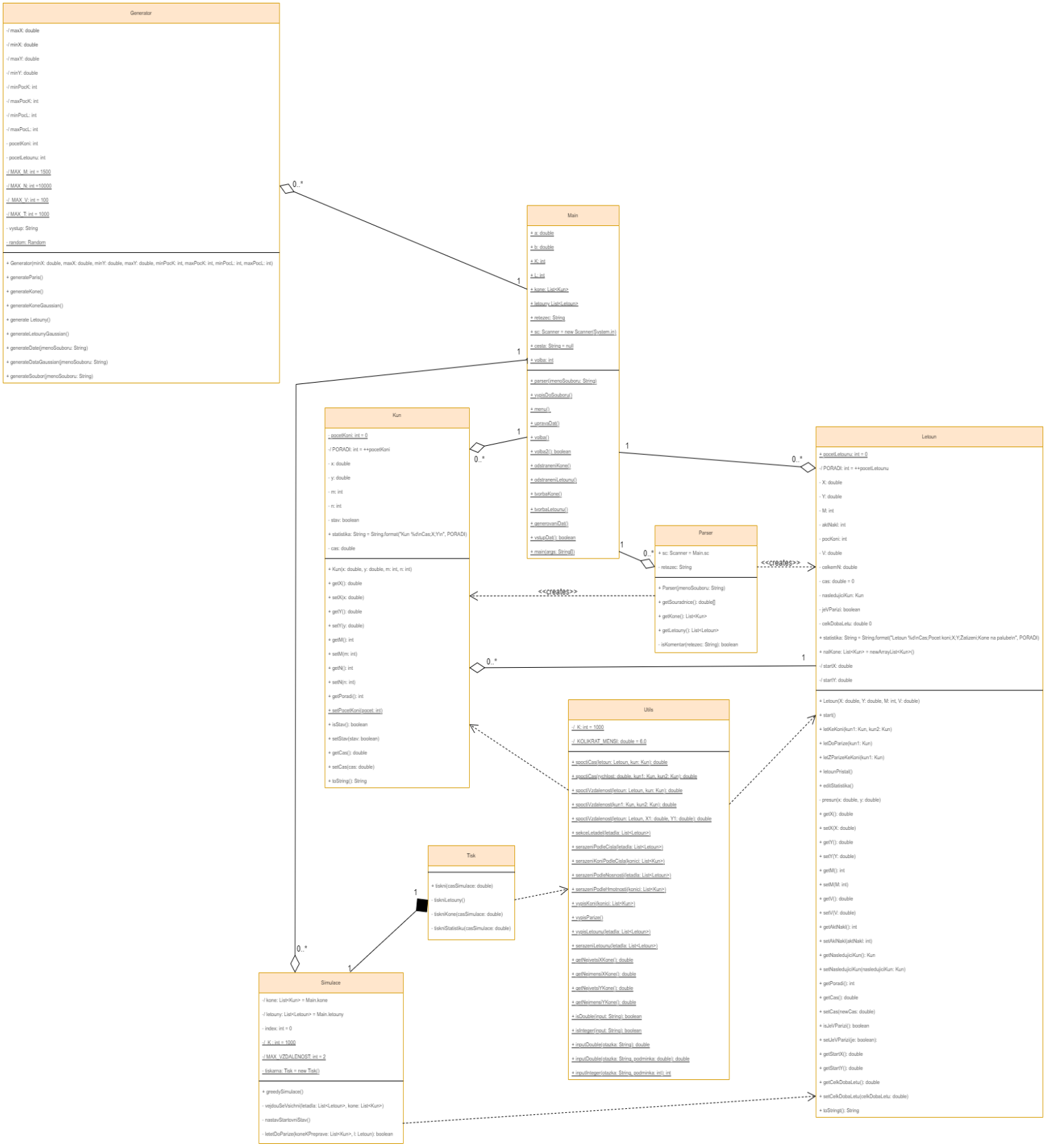
Bylo by vhodné implementovat grafovou strukturu. Implementace grafu by mohla být provedena tak, že koně boudou reprezentovat vrcholy a ohodnocené hrany boudou ohodnocené vzdálenostmi mezi koni.

Dalším problémem je zvolení vhodného algoritmu, který by pracoval efektivně a pokryl by všechny požadavky zadání. Při řešení připadá v úvahu využít princip dynamického programování, tedy předpočítat si výsledky a až poté začít se simulací. Další vhodným principem by mohl být backtracking, který se používá v podobných problémech, principem je, že vygeneruji nějaké řešení a postupně se ho snažím úpravami vylepšovat. Pokud úprava nevylepší výsledek vracím se zpátky na původní řešení. Poslední možný princip, který je nejlehčí na implementaci je greedy metoda, tedy budu brát vždy nejbližšího koně. Pro zjištění, kam má letoun letět je vhodné použít nějaký algoritmus na hledání nejkratší cesty, tedy: Dijkstrův, Floyd-Warshallův nebo Bellman-Fordův algoritmus.

Dále se také budeme zabývat otázkami typu, jak provést vygenerování zadané statistiky do souborů, vytvoření menu v konzoli, nebo jak vytvořit generátor vlastních dat. Generování dat můžeme provést rovnoměrně, normálově nebo logaritmicky normálovou distribucí.

3 Návrh programu

Níže uvedený UML diagram zobrazuje detailní popis všech tříd a metod, které byly v programu implementovány.



4 Uživatelská dokumentace

4.1 Spuštění a ovládání programu

Před spuštěním samotného programu musíme nejprve zkontrolovat, zda se nacházíme ve stejné složce, jako je soubor s názvem semestralkaPT.jar. Tento program spustíme v příkazové řádce příkazem: `java -jar semestralkaPT.jar`. Po spuštění je uživatel dotázán na cestu vstupních dat s jeho zvoleným souborem. Po úspěšném načtení dat se zobrazí hlavní menu (viz Obázek 1).

Obrázek 1: Hlavní menu

V hlavním menu má uživatel na výběr z 5 možností. Jednotlivou volbu zvolí pomocí zadání příslušné číslice + enter.

4.2 Start

Volba Start spouští simulaci přepravy koní. Po úspěšně proběhlé simulaci se vygenerují soubory se statistikou. Pro návrat do hlavního menu musí uživatel stisknout enter.

4.3 Generace dat

Volba Generace dat se uživatele zeptá na název exportovaného souboru a také na minimální a maximální počet koní, letounů, souřadnice X a Y. Následně si uživatel musí vybrat jednu ze dvou možností, jak chce generovat data. Buď pomocí rovnoměrného nebo normálového rozdělení.

4.4 Úprava dat

Volba Úprava dat umožňuje uživateli zobrazit pod menu (viz Obrázek 2). Umožňuje výpis přidání, odebrání koní a letadel. Volba pod číslem 7 opět zavede uživatele na hlavní menu.

Obrázek 2: Menu úpravy dat

4.5 Jiný soubor

Volba Jiný soubor slouží k načtení nových dat. V principu jsou vymazány veškeré letouny a koně a jsou nahrazeny novými.

4.6 Konec

Volba „Konec“ ukončí program.

5 Závěr

V naší semestrální práci jsme se zabývali vytvoření funkčního programu, který by měl splňovat jednotlivé body zadání.

Bohužel se nám nepodařilo splnit zadání v požadovaném rozsahu. Náš program není implementován grafem. Dále neobsahuje krokování, neboť nelze použít posluchače kláves a také nemáme dostatečné znalosti pro implementaci vlákna, Všechny ostatní body zadání jsou však splněny.

Jak je v kapitole 2 zmíněno, přemýšleli jsme nad vhodným zvolením příslušného algoritmu, který by byl efektivní, co se přepravy koní za použití všech letadel týče. Rozmýšleli jsme se mezi implementací Dijkstra algoritmu, Floyd-Warshallova algoritmu a Greedy algoritmu. Nakonec jsme se rozhodli právě pro Greedy algoritmus, protože implementace prvních dvou zmíněných algoritmů byly moc paměťově složité a také při velké velikosti dat program spadl.

Mezi úspěšné řešení jednotlivých úkolů bychom zařadili ošetření výjimečných stavů (velká hmotnost koně, žádná letadla) a také to, že všechna letadla létají zároveň.

Na druhou stranu víme, že některé naše použité metody jsou příliš složité a dlouhé. Možná by bylo vhodnější tyto metody v budoucnu naprogramovat efektivněji.

Naši spolupráci bychom označili za velmi dobrou. Hlavním cílem našeho projektu bylo nalézt vhodný efektivní algoritmus. Použít grafovou strukturu, což se nám bohužel nepodařilo, a vytvořit celkově funkční program. Samozřejmě jsme si také vědomi drobných nedostatků v programu, které by ale neměly nijak ovlivnit samotnou funkčnost kódu. Věříme, že jsme vytvořili funkční program bez kritických chyb.