1. Feladat – Matematikai kifejezések kiértékelése (50%)

Készítsen matematikai kifejezéseket reprezentáló osztályokat a következőképpen. Minden objektum *Expression* típusú, ezen belül vannak *Terminal* és *NonTerminal* altípusok. A *Terminal* típusú objektumok dupla lebegőpontos (*double*) értékeket tárolnak, a *NonTerminal* típusúak pedig műveleteket: összeadás, szorzás, kivonás, osztás, negálás, és az operandusokat. Figyelem, az első négy két operandusú, a negálás pedig egy operandusú.

Az *Expression* típusú objektumok rendelkezzenek legalább egy, következő függvénnyel:

- *double evaluate()* - Kiértékelés, értelemszerűen, Terminal típusúaknál csak az érték kerül visszaadásra, *NonTerminal* típusúaknál pedig kiértékeli a kifejezést, és annak az értékét adja vissza.

A *Terminal* és *NonTerminal* osztályoknak érdemes megfelelő konstruktorokat létrehozni

A *Terminal* és *NonTerminal* osztályokban az implementáció során dobjon kivételt (hibaüzenettel ami a konkrét hibának megfelel (nem *NullPointerException*)), ha logikailag vagy matematikailag értelmetlen műveletet akarunk végezni.

A matematikai kifejezés helyes fa adatszerkezetté alakítása nem feladat, így pl. a műveleti sorrenddel csak a fák kézzel való összeállítása során kell foglalkozni (tesztek megírásakor).

Írjon az osztályokat tesztelő kódot:

- (5p) Egyszerű *Terminal* objektumok kiértékelése
- (10p) Egyszerű kifejezés felépítése és kiértékelése, pl. 3.14+5
- (15p) Összetett kifejezés felépítése és kiértékelése, pl. -4.2*3+5.1
- (10p) Meglévő kifejezés (akár *Terminal* akár *NonTerminal*) beépítése más kifejezésbe, pl. A = 4*3+5 -> A/5.1
- (10p) Hibás műveletek tesztelése - minden lehetséges hibára 1-1, a kivételt elkapva és megjelenítve

2. Feladat - ATM (50%)

Ebben a feladatban az ITK-hoz közel eső ATM forgalmat kell szimulálni. Tegyük föl, hogy ezek az ATM-ek egy központi bankfiókkal vannak összekötve (általában egy bankfióknak van egy hatásköre, amelyen belül kezeli az automatákat), amelyet egyszerre csak egy ATM érhet el. Ezen kívül szimuláljunk 5 ATM-et, amelyeket a felhasználók rendelkezésére bocsátunk.

Az embereket, akik pénzt kívánnak fölvenni, szálak reprezentálják. Egy időben az ITK közelében véletlen időközönként, átlagosan másodpercenként (0.5...1.5 másodperc intervallumban) érkezik egy felhasználó. Megnézi a telefonján, hogy rendelkezésre áll-e a közelében szabad ATM (az 5 közül, amit definiáltunk), és ha igen, akkor távolról lefoglalja, és 5 másodperc alatt odamegy. Ha nem talál, akkor újra nézi 0.5 másodpercenként.

Ha odaért az ATM-hez, akkor 0.4 másodperc alatt leveszi a pénzt, ha szabad a csatorna a bankfiók felé (vagyis másik ATM nem használja). Ha nem sikerül, itt is 0.3 másodperc múlva újra próbálkozik egészen addig, amíg sikerül az elérés; ekkor innentől számolva 0.4 másodperc, hogy kivegye a pénzt. Ezután elmegy az ATM-től, azaz megszakad a kapcsolata a bankkal, és más is használhatja az ATM-et.

Az emberek átlagosan 12 500 forintot vesznek le a számlájukról e feladatban (5 000 - 20 000 közötti összegeket).

A program folyamatosan naplózza a tranzakciók tevékenységét, például így:

09:30:11, A(z) 59. személy elfoglalta a(z) 5. számú ATM-et.
09:30:14, A(z) 45. személy felvett 22000 forintot.
09:30:19, A(z) 41. személy elhagyta a(z) 2. számú ATM-et.

A program 10 másodpercenként kiírja, hogy mennyit vettek föl összesen az emberek, például így:

09:34:28, Az eddigi forgalom 180000 forint...

A megoldás során ne használj busy waitinget, használd helyette a Java platform/library-k lehetőségeit, amiket gyakorlatokon néztünk! Figyelj arra is, hogy a programod szálbiztos legyen!