Objektum orientált programozás
3. gyakorlat

Standard input, output kezelése; véletlenszám generálás és a Math osztály metódusai

1. Írja meg a 4 alapműveletet megvalósító **Kalkulátor** programot Java nyelven. Az elvégzendő műveletet és az operandusokat a felhasználó adja meg. A program a megadott művelettől függően végzi el a számítási feladatot. A művelet vizsgálatához használja a *switch* utasítást. Az osztás művelet végrehajtásakor ügyeljen az operandusok típusára, és ne engedje a 0-val való osztást. A felhasználói input megadását próbálja meg kétféleképpen:
   1. Külön olvassa be a két operandust (számok) és külön az operátort (char). Használja a *Scanner* osztály *nextXXX* metódusait.
   2. Egyetlen sztringként olvassa be az inputot space-el elválasztva a tagokat (pl.: 3 + 5). A beolvasott sztringet a space-ek mentén szét kell bontani tagokra a *split* metódussal, melynek paramétere egy sztring (a tagoló karaktersorozat). Az eredmény egy sztring tömb, melynek egyes elemeit a megfelelő típusra konvertálva kapjuk meg az operandusokat (a csomagoló osztályok *parseXXX* metódusait használva) és az operátort.

2. Deklaráljon, majd foglaljon helyet egy 10 elemű *int* tömb számára.
   - Töltse fel 1 és 50 közötti véletlenszámokkal a tömböt.
     1. Használja a *java.util.Random* osztályt! Első lépés egy ilyen típusú objektum létrehozása a véletlenszám generátor inicializálásához:
     ```
     java.util.Random rand = new java.util.Random();
     ```
     Második lépés az osztály *nextXXX* metódusának hívása. Ezek a metódusok 0 és az argumentumként megadott szám által határolt, felülről nyitott intervallumból állítják elő a megfelelő típusú véletlenszámot.
     ```
     int random = rand.nextInt(50)+1;
     ```
     2. Használja a *java.lang.Math* osztály *random()* metódusát! Ez a metódus a [0,1) felülről nyitott intervallumból egy lebegőpontos véletlenszámot ad vissza. A számtartományt a C-ből ismert *(felsőhatár-alsóhatár+1)+alsóhatár* képlettel adhatjuk meg.
     ```
     int random = (int )(Math.random() * 50) + 1;
     ```
   - Írja ki a tömbelemeket fordított sorrendben. Figyelem! Itt nem tudja használni a foreach ciklust.
   - Keresse meg a tömbben a legnagyobb páros számot. Figyelje meg mi a különbség, ha a for ciklust, illetve ha a foreach ciklust használja!

3. Deklaráljon, majd foglaljon helyet egy 10 elemű *double* tömb számára.
   - A tömbelemek értékét a standard inputról olvassa be. Használja a *Scanner* osztályt! Itt lehet használni a foreach ciklust?
   - Számítsa ki a tömbelemek mértani átlagát: a tömbelemek szorzata annyiadik gyök alatt, ahány tagú a szorzat. Gyökvonáshoz használja a *Math* osztály *pow()* hatványozó metódusát, amelynek első paramétere a hatványalap (double), második paramétere a hatványkitevő (double) és double értéket ad vissza.

Házi feladat:

1. Írja meg a másodfokú egyenlet valós gyökeit meghatározó Java programot. A másodfokú egyenlet konstansait a felhasználó adja meg. Beolvasáshoz használja a *Scanner* osztályt, a számításhoz pedig a *Math* osztály *sqrt()* metódusát, amely egy *double* típusú argumentumot vár és *double* értéket ad vissza. Az első feladathoz hasonlóan 1) kérje be egyenként a 3 számot, majd 2) egyetlen sztringként, ahol vesszővel választja el a számokat.

2. A *java.util* csomagban definiált *Arrays* osztály metódusaival kényelmesen kezelhetők a tömbök. Például:

- Tömb rendezése (nem ad vissza értéket): *java.util.Arrays.sort(myArray);*
- Bináris keresés tömbben (a megtalált elem indexét adja vissza): *java.util.Arrays.binarySearch(myArray, searchKey);*
- Két tömbben az elemek azonosságának a vizsgálata (boolean-el tér vissza): *java.util.Arrays.equals(array1, array2);*

Próbálja ki a metódusokat *int*, *double* és *char* tömbökön.