Egyetemisták

Néhány éve adták át ünnepélyes keretek között a pécsi székesegyház udvarában lévő középkori egyetem épületét. Ennek apropóján szimuláljunk egy gondolatkísérletet, és hasonlítsuk össze a mai és a középkori egyetemistákat.

Szimulációról lévén szó, az egyetemistákat csak egy egyedi azonosító kóddal azonosítják. Minden **egyetemistá**ra jellemző az, hogy *órán van()*, *tanul()*, illetve *kocsmázik()*. A tanulás során a *tanulási idő* a metódus paraméterében lévő időértékkel (egész perc) növekszik. Mivel a középkorban inkább kupát használtak ivásra, ma meg korsót, illetve poharat, ezért ezek helyett a *fogyasztott alkoholmennyiség*et vesszük figyelembe (centiliter). A kocsmázás során mindegyikük fogyasztott alkoholmennyisége a metódus paraméterében lévő double értékkel növekszik. Minden egyetemistának meghatározható a *tudás()*a (egy egész érték), de nem teljesen egyforma módon számolhatunk a mai és a középkori egyetemisták esetén, azonban mindegyikben közös, hogy a tudás egy része a tanulási idő és egy, az összes egyetemistára jellemző *állandó* szorzata, amelyből le kell vonni az alkohol hatását, vagyis a fogyasztott alkohol mennyiségének és egy *másik állandó*nak a szorzatát.

A középkorban élénk vita folyt az órákon. Vagyis ha egy **középkori egyetemista** órán van, akkor *vitakészség*e növekszik (esetünkben eggyel). Tudásának kiszámításakor az eddigi értékhez hozzáadódik még a vitázás során összeszedett tudás, vagyis a vitakészség és egy, az összes középkori egyetemistára jellemző *állandó* szorzata.

A **mai egyetemistá**k vita helyett sajnos inkább a kütyüikkel vannak elfoglalva. Még órán sem tudnak megválni tőlük. Nehéz lenne egyenként számolni, hogy hányszor használják, de létezik egy jól kimutatható statisztikai *állandó*, ami mindegyikükre egyformán érvényes. Ha egy mai egyetemista órán van, akkor *kütyühasználatának száma* evvel az értékkel növekszik. Sajnos a kütyühasználat ugyanannyival növeli a tudást, mint amennyit elvesz belőle.

A toString() definiálásakor minden egyetemista esetén adja vissza az illető azonosítóját, azt, hogy mai vagy középkori egyetemista és tudásának értékét (mértékegység: „egység”), a maiaknál azt is, hogy hány alkalommal használt valamilyen kütyüt.

A fentiek figyelembevételével írja meg a szimulációt végző programot, azaz:

A statikus adatok beállítása után olvassa be az *egyetemistak.txt* fájl tartalmát. Az M betűvel kezdődő azonosító mai egyetemistát, a K betűvel kezdődő középkorit jelent. Beolvasás után írassa ki a szimulációban résztvevő emberek adatait, majd kezdődjön a szimuláció. Ez a következőt jelenti:

Véletlen sokszor futtassa le azt, hogy egy véletlenül kiválasztott egyetemista órán van, majd utána véletlentől függően vagy leül tanulni, vagy kocsmába megy. Ha tanul, akkor véletlen

ideig tanuljon, ha kocsmázik, akkor véletlen mennyiségű alkoholt igyon (NE égesse be az adatokat).

A szimuláció után is írassa ki az adatokat, majd határozzon meg néhány statisztikai eredményt:

- Állapítsa meg, hogy a mai vagy a középkori egyetemistáknak nagyobb-e az átlagtudása (vagy esetleg egyforma).
- Kik azok, akiknek legnagyobb a tudásuk?
- Vannak-e olyanok, akik elitták az eszüket? Ha igen, hányan?

(Figyeljen rá, hogy az utolsó feladat miatt célszerű két külön is elérhető részre bontani a tudás kiszámítását. Sőt, mivel esetleg továbbfejlesztésként még a vitázásból eredő tudás is érdekelhet bennünket, azt is célszerű külön elérhetővé tenni.)

A középkori egyetem: http://www.kozepkoriegyetem.hu/

Megoldás-részletek:

Az ősosztályt most is célszerű absztraktként deklarálni, és az utolsó feladatrész miatt külön komponensekre bontani a tudás kiszámítását.

Az alaposztályok kódját getterek, setterek nélkül közöljük. Gettert minden mezőhöz célszerű írni, settert a statikusakhoz.

```
public abstract class Egyetemista {

    private String azonosito;

    private int tanulasiIdo;
    private double fogyasztottAlkoholMennyiseg;

    private static double tanulasSzorzo;
    private static double alkoholSzorzo;

    public Egyetemista(String azonosito) {
        this.azonosito = azonosito;
    }

    public abstract void oranVan();

    public void tanul(int perc){
        this.tanulasiIdo += perc;
    }

    public void kocsmazik(double alkoholMennyiseg){
        this.fogyasztottAlkoholMennyiseg += alkoholMennyiseg;
    }
```

```
    public int tanulasKomponens(){
        return (int) (this.tanulasiIdo * Egyetemista.tanulasSzorzo);
    }

    public int ivasKomponens(){
        return (int) (fogyasztottAlkoholMennyiseg * alkoholSzorzo);
    }

    public int tudasErtek(){
        return tanulasKomponens() - ivasKomponens();
    }

    public abstract String osztalyNev();

    @Override
    public String toString() {
        return String.format("%s %s, tudása %d egység",
                                    azonosito, osztalyNev(), tudasErtek());
    }
```

```
public class KozepkoriEgyetemista extends Egyetemista{

    private int vitaKeszseg;
    private static double vitaSzorzo;

    public KozepkoriEgyetemista(String azonosito) {
        super(azonosito);
    }

    @Override
    public void oranVan() {
        vitaKeszseg++;
    }

    @Override
    public String osztalyNev() {
        return "középkori egyetemista";
    }

    public int vitaKomponens(){
        return (int) (this.vitaKeszseg * KozepkoriEgyetemista.vitaSzorzo);
    }

    @Override
    public int tudasErtek() {
        return super.tudasErtek() + vitaKomponens();
    }
```

```java
public class MaiEgyetemista extends Egyetemista{

    private int kutyuhasznalatSzama;

    private static int kutyuIndex;

    public MaiEgyetemista(String azonosito) {
        super(azonosito);
    }

    @Override
    public void oranVan() {
        this.kutyuhasznalatSzama += MaiEgyetemista.kutyuIndex;
    }

    @Override
    public String osztalyNev() {
        return "mai egyetemista";
    }

    @Override
    public String toString() {
        return super.toString() +
                "\n\t" + kutyuhasznalatSzama +
                " alkalommal használt valamilyen kütyüt.";
    }
```

A vezérlésből csak néhány metódust emelünk ki. Magát a vezérlést, a beolvasást, a szimulációt, és annak megállapítását, hogy hányan itták el az eszüket.

```java
public class Main {

    private final String ADAT_ELERES = "/adatok/egyetemistak.txt";
    private final String CHAR_SET = "UTF-8";

    private final int SZIMULACIO_HATAR = 100;
    private double TANULAS_ESELY = 0.55;
    private int TANULASIPERC_HATAR = 120;
    private double ALKOHOL_HATAR = 8;

    private List<Egyetemista> egyetemistak = new ArrayList<>();

    public static void main(String[] args) {
        new Main().start();
    }
```

```
private void start() {
    statikusAdatok();
    adatBevitel();
    kiiratas("A felmérésben résztvevők: ");
    szimulacio();
    kiiratas("\nA szimuláció eredménye: ");
    statisztikak();
}

private void adatBevitel() {
    try(InputStream ins = this.getClass().getResourceAsStream(ADAT_ELERES);
        Scanner sc = new Scanner(ins, CHAR_SET)) {

        String azonosito;
        while (sc.hasNextLine()) {
            azonosito = sc.nextLine();
            try {
                if (!azonosito.isEmpty()) {
                    switch (azonosito.charAt(0)) {
                        case 'M':
                            egyetemistak.add(new MaiEgyetemista(azonosito));
                            break;
                        case 'K':
                            egyetemistak.add(new KozepkoriEgyetemista(azonosito));
                            break;
                        default:
                            throw new Exception("Hibás a " + azonosito +
                                                " azonosítót tartalmazó sor");
                    }
                }
            } catch (Exception ex) {
                System.out.println(ex.getMessage());
            }
        }
    } catch (Exception e) {
        System.out.println("Hiba a fájl beolvasása során!");
    }
}
```

```java
private void szimulacio() {
    Random rand = new Random();
    int n = rand.nextInt(SZIMULACIO_HATAR);
    int veletlenEgyetemistaIndex;
    Egyetemista egyetemista;

    for (int i = 0; i < n; i++) {
        veletlenEgyetemistaIndex = rand.nextInt(egyetemistak.size());
        egyetemista = egyetemistak.get(veletlenEgyetemistaIndex);
        egyetemista.oranVan();
        if(rand.nextDouble() < TANULAS_ESELY){
            egyetemista.tanul(rand.nextInt(TANULASIPERC_HATAR));
        }else{
            egyetemista.kocsmazik(rand.nextDouble()* ALKOHOL_HATAR);
        }
    }
}

private void statisztikak() {
    osszehasonlitas();
    legek();
    reszegesek();
}

private void reszegesek() {
    int ossz = 0;
    for (Egyetemista egyetemista : egyetemistak) {
        if (egyetemista.tanulasKomponens() < egyetemista.ivasKomponens()) {
            ossz++;
        }
    }
    if (ossz > 0) {
        System.out.printf("\n összesen %d ember itta el az eszét\n", ossz);
    } else {
        System.out.println("Senki sem itta el az eszét.");
    }
}
```