Focisták

Minden **focistá**t jellemez a *neve* és az *igazolási száma*. Mindegyikük *edz()*, ekkor az *edzési ideje* megnövekszik az edzésre szánt aktuális idővel (a metódus paraméterében megadott érték – lehet egész, mondjuk a megkezdett órák száma). Mindegyik *játszik()* is. Ekkor *meccseinek száma* eggyel növekszik, illetve időnként még gólt is rúg. Amikor *goltRug()*, akkor a *rúgott gólok száma* eggyel növekszik. Mindegyikőjük kap egy egységes alapfizetést, de a fizetés további része már függ attól, hogy milyen keményen és eredményesen dolgozott. Ha az edzésideje meghaladott egy, az összes focistára egységes *edzési határ*t, akkor az alapfizetéshez még hozzáadódik az edzésidő valahányszorosa. Ez az *edzési óradíj* egységes az összes focista esetén. A játékukra is kapnak fizetést, ha legalább *gólhatár* mennyiségű gólt rúgtak (a határ egységes). Ekkor a fizetésük az egységes *játékdíj* meccsszámmal szorzott értékével növekszik.

Írja meg a Focista osztályt, tesztelje, a vezérlésben pedig indítsa be a fantáziáját. Például: adott végjelig olvassa be a focisták adatait (de szerencsésebb lenne fájlból), a beolvasás után azonnal eddzen véletlen ideig, véletlenszerűen döntse el, hogy játszik-e vagy sem, ill. rúg-e gólt vagy sem, és írassa ki a fizetését.

Persze, az igazi az lenne, ha már tömböket vagy listát is használna, de egyelőre ezek a fogalmak menjenek.

Megoldás-részletek:

Különösebb magyarázat nélkül közöljük a `Focista` osztály kódját, majd néhány ötletet adunk a vezérléshez, de annak kódolását már Önre bízzuk.

Az osztály setterek, getterek nélküli kódja olvasható a következő oldalon.

(Gettert mindegyik mezőhöz írhatunk, settert a statikus változókhoz mindenképpen kell írni, a név, igazolásszám megváltoztathatósága megállapodás kérdése, de szigorúan véve inkább nem. Az edzésidő, gólszám, meccsek számának értéke metóduson belül változik, vagyis ezekhez egyáltalán nem szabad settert írni.)

```java
public class Focista {

    private String name;
    private int igazolasSzam;

    private int edzesIdo;
    private int meccsekSzama;
    private int golokSzama;

    private static int alapFizetes;

    private static int edzesiHatar;
    private static int edzesiOradij;

    private static int golHatar;
    private static int jatekDij;

    public Focista(String nev, int igazolasSzam) {
        this.name = nev;
        this.igazolasSzam = igazolasSzam;
    }

    public void edz(int ido) {
        this.edzesIdo += ido;
    }

    public void jatszik() {
        this.meccsekSzama++;
    }

    public void goltRug() {
        this.golokSzama++;
    }

    public int fizetes() {
        int temp = Focista.alapFizetes;
        if (this.edzesIdo >= Focista.edzesiHatar) {
            temp += this.edzesIdo * Focista.edzesiOradij;
        }
        if (this.golokSzama >= Focista.golHatar) {
            temp += this.meccsekSzama * Focista.jatekDij;
        }
        return temp;
    }

    @Override
    public String toString() {
        return name + " ebben a szezonban \n\t\t"
                + this.getEdzesIdo()+ " órát edzett,\n\t\t"
                + this.getMeccsekSzama() + " meccsen játszott,\n\t\t"
                + this.getGolokSzama() + " gólt rúgott,\n\t így "
                + this.fizetes() + " Forintot keresett.";
    }
```

A vezérlésből csak néhány részletet emelünk ki. Tegyük fel, hogy a focisták adataiból a focistak nevű listát hoztuk létre:

```
private List<Focista> focistak=new ArrayList<>();
```

A véletlen edzések például:

```
private final int EDZES_IDO_MAX = 10;

/**
 * Az edzések szimulálása
 */
private void edzes() {
    for (Focista focista : focistak) {
        int ido = (int) (Math.random() * EDZES_IDO_MAX);
        focista.edz(ido);
    }
}
```

Az `edzesIdoMax` változó értékét az osztály elején adjuk meg, és ide gyűjtjük a többi konstans értékadást is, hiszen itt szükség esetén könnyen lehet módosítani őket.

A focisták játékát pl. így lehetne szimulálni: Legyen például `meccsSzam` számú meccs, és minden egyes alkalommal döntsük el az egyes focistákról, hogy játszanak-e vagy sem. Mondjuk, legyen 70% annak esélye, hogy játszik valamelyik. Aki játszik, ő pedig 50% eséllyel rúgjon gólt.

```
private int meccsSzam = 20;
private double jatekEsely = 0.7;
private double golEsely = 0.5;

/**
 * A meccsek szimulálása
 */
private void meccsek(){
    for (int i = 0; i < meccsSzam; i++) {
        for (Focista focista : focistak) {
            if(Math.random() < jatekEsely){
                focista.jatszik();
                if(Math.random() < golEsely){
                    focista.goltRug();
                }
            }
        }
    }
}
```