Csipkerózsika

A mai gyerekek már elektronikus meséken nőnek fel (szegények). Sőt, a mesékhez nem egyszer készül játék szoftver is. Akár az Ön feladata is lehet valamikor később, hogy egy, mondjuk a Csipkerózsika mesét feldolgozó játékszoftvert író csapat munkatársa legyen. Talán emlékszik rá, hogy a mesében vannak tündérek és boszorkány(ok) is.

Hogy érdekesebb legyen a játék, ezek a karakterek rendelkezhetnek valamilyen, a játékhoz szükséges tulajdonságokkal. Például lehet varázserejük, ami változhat a játék során. Amikor találkoznak egy emberrel, akkor ennek az erőnek megfelelően valamilyen hatást gyakorolhatnak rájuk – a tündérek nyilván jó hatást, a boszorkányok rosszat, stb. Egy ilyen játék persze akkor érdekes, ha a felhasználó grafikus felületen, interaktív módon beleavatkozhat a játék menetébe, de még nem rendelkezik az ehhez szükséges ismeretekkel. A karakterek alap-tulajdonságait viszont már meg tudja fogalmazni, és egy egyszerűbb kis tesztprogrammal ellenőrizni is tudja a működésüket. Ez lesz most a feladata.

Vannak tehát **tündér**ek és **boszorkány**ok. Mindegyiknek van neve, és mindegyik rendelkezik valamekkora mágikus erővel (egy int érték). Mindegyikük tud áldani vagy átkozni. A „varázslás” (azaz áldás vagy átok) során mindegyikük kifejt valamekkora hatást. Ez a hatás egy egész szám, amelyet az illető hölgy ereje alapján számítunk, mégpedig úgy, hogy tündérek esetén ezt az erőt megszorozzuk egy, a tündérekre egyaránt érvényes (pozitív) szorzóval, boszorkányok esetén pedig egy, a boszorkányokra egyaránt érvényes (negatív) szorzóval.

Mivel a jócselekedetek növelik az ember erejét, a rosszak pedig – legalábbis a mesében – csökkentik, ezért a tündérek ereje minden egyes varázsláskor 1-gyel növekszik, a boszorkányoké 1-gyel csökken, de csak addig, amíg nulla nem lesz.

Mindenki tudja visszaadni (toString()) azt, hogy tündér vagy boszorkány, továbbá a nevét és az erejének értékét. Ha a boszorkány ereje nullára csökken, akkor még az is kerüljön az adatok mellé, hogy „ megjavult”.

Tesztelje néhány adattal az elkészült osztályokat.

Írja meg a mese „keresztelő” részét, vagyis: olvassa be a meghívottak névsorát a mellékelt *csipkerozsika.txt* fájlból. Az illető hölgy induló ereje legyen egy véletlen érték. Az pedig, hogy tündér-e vagy boszorkány, szintén a véletlenen múlik. Vegye figyelembe azt is, hogy a hölgyeknek kb. x %-a boszorkány.

A beolvasás után írassa ki a jelenlévő hölgyeket, majd egyenként mindenki „varázsoljon”. Végül számolja ki az átlagos varázserőt, majd adja meg a legnagyobb varázserejű tündér(ek) és a leggonoszabb boszorkány(ok) nevét.

Megoldás-részletek:

Az öröklődéshez fűzünk néhány megjegyzést, a megoldás többi részét különösebb magyarázat nélkül közöljük.

A feladatban tündérek és boszorkányok szerepelnek, de nyilvánvaló, hogy sok közös tulajdonságuk van. Ezeket célszerű egy közös ősosztályban megfogalmazni. Ugyanakkor ennek az osztálynak egyetlen példánya sem lesz, vagyis akár absztrakt osztály is lehet. Ténylegesen egyetlen megbeszélendő részlet van, mégpedig a mágikus erő és mágikus hatás kérdése. Ezek értéke ugyanis az utód-osztályokban változik. Ezért első ötletként célszerűnek tűnne, ha a `magikusEro` és `magikusHatas` nevű mezőket `protected` láthatósággal deklarálnánk. Csakhogy emellett főleg a lustaság és kényelem szól, ami nem igazán szempont a biztonságossággal szemben. Vagyis maradjunk abban, hogy továbbra is `private` láthatóságú minden mező, de mivel ezek az értékek az utód-osztályokban módosulnak, viszont senki más nem módosíthatja őket, ezért a hozzájuk tartozó settereket deklaráljuk majd `protected` módon.

A szóban forgó alaposztályok (getterek nélküli részlet, de persze, minden mezőhöz tartozik getter, az utód osztályok statikus mezőihez setter is és getter is):

```
public abstract class Holgy {
    private String nev;
    private int magikusEro;
    private int magikusHatas;

    public Holgy(String nev, int magikusEro) {
        this.nev = nev;
        this.magikusEro = magikusEro;
    }

    public abstract void varazsol();

    @Override
    public String toString() {
        return  this.nev + " "
                + this.getClass().getSimpleName().toLowerCase()
                + ", mágikus ereje: " + this.magikusEro + " egység";
    }

    protected void setMagikusEro(int magikusEro) {
        this.magikusEro = magikusEro;
    }

    protected void setMagikusHatas(int magikusHatas) {
        this.magikusHatas = magikusHatas;
    }
```

(Ha zavarja az ékezet nélküli tunder/boszorkany kiírás, akkor azt a bálozós feladatban leírtakhoz hasonlóan itt is ki lehet javítani.)

```java
public class Tunder extends Holgy {

    private static int tunderSzorzo;

    public Tunder(String nev, int magikusEro) {
        super(nev, magikusEro);
    }

    /** Kiszámolja a tündér mágikus hatását és növeli az erejét ...3 lines */
    @Override
    public void varazsol() {
        this.setMagikusEro(this.getMagikusEro()+1);
        this.setMagikusHatas(this.getMagikusEro()*Tunder.tunderSzorzo);
    }
```

```java
public class Boszorkany extends Holgy {

    private static int boszorkanySzorzo;

    public Boszorkany(String nev, int magikusEro) {
        super(nev, magikusEro);
    }

    /** Kiszámolja a boszi mágikus hatását és csökkenti az erejét ...3 lines */
    @Override
    public void varazsol() {
        if (this.getMagikusEro() > 0) {
            this.setMagikusEro(this.getMagikusEro() - 1);
        }
        this.setMagikusHatas(this.getMagikusEro() * Boszorkany.boszorkanySzorzo);
    }

    @Override
    public String toString() {
        String temp= super.toString();
        if(this.getMagikusEro() == 0){
            return temp+", megjavult";
        }
        return temp;
    }
```

A tesztet oldja meg önállóan.

A vezérlésben nincs semmi különös, magyarázat nélkül közöljük a kód nagy részét, csak a leggonoszabb boszorkányok meghatározását előzi majd meg egy rövid magyarázat.

```java
public class Main {

    private final String ADAT_ELERES = "/adatok/csipkerozsika.txt";
    private final String CHAR_SET = "UTF-8";
    private final int TUNDER_SZORZO = 5;
    private final int BOSZORKANY_SZORZO = -4;
    private final int MAX_ERO = 6;

    private final double BOSZORKANYSAG_ESELY = 0.4;

    private List<Holgy> holgyek = new ArrayList<>();

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

    private void start() {
        statikusAdatok();
        beolvasas();
        kiiratas("A meghívott hölgyek:\n");
        varazslas();
        kiiratasVarazslasUtan("\nA varázslás után:\n");

        atlagEro();
        leggonoszabbBoszik();
        legjobbTunderek();
    }

    private void statikusAdatok() {
        Tunder.setTunderSzorzo(TUNDER_SZORZO);
        Boszorkany.setBoszorkanySzorzo(BOSZORKANY_SZORZO);
    }

    private void beolvasas() {
        try (InputStream ins = this.getClass().getResourceAsStream(ADAT_ELERES);
                Scanner sc = new Scanner(ins, CHAR_SET)) {
            String nev;
            while (sc.hasNextLine()) {
                nev = sc.nextLine();
                if (!nev.isEmpty()) {
                    if (Math.random() < BOSZORKANYSAG_ESELY) {
                        holgyek.add(new Boszorkany(nev,
                                (int) (Math.random() * MAX_ERO)));
                    } else {
                        holgyek.add(new Tunder(nev,
                                (int) (Math.random() * MAX_ERO)));
                    }
                }
            }
        } catch (Exception e) {
            System.out.println("Hiba a fájl beolvasása során!");
        }
    }
```

```
private void kiiratas(String cim) {
    System.out.println(cim);
    for (Holgy holgy : holgyek) {
        System.out.println(holgy);
    }
}

private void varazslas() {
    for (Holgy holgy : holgyek) {
        holgy.varazsol();
    }
}

private void kiiratasVarazslasUtan(String cim) {
    System.out.println(cim);
    for (Holgy holgy : holgyek) {
        System.out.println(holgy +
                "\n\thatása: " + holgy.getMagikusHatas() + " egység");
    }
}

private void atlagEro() {
    if (holgyek.isEmpty()) {
        System.out.println("Nincsenek hölgyek.");
    } else {
        double ossz = 0;
        for (Holgy holgy : holgyek) {
            ossz += holgy.getMagikusEro();
        }
        System.out.printf("\nAz átlagos varázserő: %5.2f egység\n",
                                                ossz / holgyek.size());
    }
}
```

A leggonoszabb, vagyis legnagyobb mágikus erővel rendelkező boszorkányok meghatározása sima, egyszerű maximum-keresés. Megoldhatnánk úgy, hogy a max változó értékét nulláról indítjuk, hiszen most pozitív, illetve nem negatív számok maximumát keressük. De egy teljesen általános maximumkeresés esetén nem garantált, hogy helyes, ha a nulláról indulunk. Ezért most oldjuk meg teljesen általánosan, és kiinduló értékként tekintsük a legelső boszorkány erejét. Ilyet csináltunk már a bálkirálynő választáskor (Bálozók), de ott hallgatólagosan feltételeztük, hogy vannak lányok. Most viszont – mivel véletlenszerűen dől el, hogy a megadott hölgyek közül ki boszorkány, ki tündér –, előfordulhat, hogy egyáltalán nincs is köztük boszorkány. Vagyis a megoldás során figyeljük azt is, hogy egyáltalán van-e boszorkány, és csak akkor keressük a leggonoszabbat, ha van.

Egy lehetséges kódrészlet:

```
private void leggonoszabbBoszik() {
    int max = 0;
    boolean vanBoszi = false;
    for (Holgy holgy : holgyek) {
        if (holgy instanceof Boszorkany) {
            if (!vanBoszi || holgy.getMagikusEro() > max) {
                max = holgy.getMagikusEro();
                vanBoszi = true;
            }
        }
    }
    if (!vanBoszi) {
        System.out.println("Nem jött egy boszorkány sem.");
    } else {
        System.out.printf("\nA leggonoszabb boszorkány(ok) %d egység erővel:\n",
                                                                    max);
        for (Holgy holgy : holgyek) {
            if (holgy instanceof Boszorkany && holgy.getMagikusEro() == max) {
                System.out.println(holgy.getNev());
            }
        }
    }
}
```

A legerősebb tündérek meghatározása ugyanilyen.