Médiabirodalom 1.

Egy **újság**ot egyértelműen jellemez a *neve* és *megjelenési dátum*a (lehet sima String, de próbálkozhat igazi dátummal is, ha kedve tartja). Az újság *cikket közöl()*, vagyis a metódus paraméterében megadott cikket hozzáadja az újságban megjelent *cikkek listájá*hoz (úgy oldja meg, hogy minden cikk csak egyszer szerepelhet egy-egy újság listájában).

Egy **cikk** egyértelműen megadható a *szerző nevé*vel, a *cikk címé*vel, a *cikk méreté*vel (karakterszám) és egy, a cikkben lévő hazugság nagyságára utaló *százalékláb* értékkel.

Az újságok azt is meg tudják „mondani”, hogy mekkora a bennük lévő cikkek átlagos hazugságszázaléka.

A médiabirodalom nyomtatott és internetes újságot is megjelentet.

A **nyomtatott újság**ot a nevén és megjelenési dátumán kívül jellemzi még az újság *példány-száma* és a *mérete* (összesen hány karaktert tud megjelentetni). Mivel itt korlátozott a méret, ezért egy cikk közlése csak akkor lehetséges, ha annak mérete még belefér az újság méretébe.

Az **internetes újság** a néven és megjelenési dátumon kívül tartalmazza még az újság *link*jét (most csak egy String).

Olvassa be az adatokat – mégpedig úgy, hogy könnyen lehessen módosítani, ha adatfájlból vagy adatbázisból szeretnénk. (Adatfájlok: *ujsagok.txt*, *cikkek.txt*; az adatbázis tábláit az *ujsagok.sql* és a *cikkek.sql* fájl írja le.) Az *ujsagIras()* során véletlen sokszor egy-egy véletlenül választott újságban jelentessen meg egy-egy véletlenül választott cikket.

Írassa ki az újságokat a bennük megjelent cikkekkel együtt, illetve azt is állapítsa meg, hogy melyik újság hány napja jelent meg.

Megoldásrészletek:

Az alaposztályokat különösebb magyarázatok nélkül közöljük, illetve ahol nagyon szükséges, a kódba írunk magyarázó kommentet, de ne felejtse el, hogy egy tiszta kódban csak dokumentációs kommentek szerepelhetnek (ezeket most nem mutatjuk, és a getterek, setterek közül is csak azokat, amelyekben esetleg van egy kis eltérés a szokásoshoz képest).

```java
public class Cikk {
    private String szerzo;
    private String cim;
    private int karakterSzam;
    private int hazugsagSzazalek;
    private static int defaultHazugsagSzazalek;

    public Cikk(String szerzo, String cim, int karakterSzam,
                                           int hazugsagSzazalekLab) {
        this.szerzo = szerzo;
        this.cim = cim;
        this.karakterSzam = karakterSzam;
        this.hazugsagSzazalek = hazugsagSzazalekLab;

        // egy kis adatvédelem:
        if (hazugsagSzazalekLab < 0 || hazugsagSzazalekLab > 100) {
            this.hazugsagSzazalek = defaultHazugsagSzazalek;
        }
         // mert eleve hazugság mindkét eset :)
    }

    @Override
    public String toString() {
        return szerzo + ": " + cim;
    }

    public void setCim(String cim) {
        // Vegyük figyelembe, hogy a cím módosításakor
        // a karakterszám is változik.
        this.karakterSzam -= this.cim.length();
        this.cim = cim;
        this.karakterSzam += this.cim.length();
    }
```

A feladatkiírás külön felhívja rá a figyelmet, hogy egy újságban nem jelenhet meg kétszer ugyanaz a cikk, ezért most mindenképpen generáltatni kell az `equals()` és `hashCode()` metódusokat. (Éles alkalmazáskor ezeket egyébként is érdemes generálni, mert nem lehet előre tudni, hogy vajon szükség lesz-e rájuk valahol, vagy sem.)

Bár a feladatkiírás „megengedi”, hogy igazi dátum helyett string típussal dolgozzunk, de ideje lenne szót ejteni a dátumkezelésről is. Ez a 8-as Java előtt kicsit nehézkes volt, de mivel nagyon elterjedt, ezért most ezt a változatot mutatjuk meg, de önállóan megoldhatja a 8-as Java-ba beépített csomag segítségével is. Ehhez pl. itt talál segítséget:

http://javarevisited.blogspot.hu/2015/03/20-examples-of-date-and-time-api-from-Java8.html

```java
public class Ujsag {

    private String nev;
    private Date datum;

    private List<Cikk> cikkek = new ArrayList<>();

    private int osszHazugsag;
    private double atlagHazugsagSzazalek;
    private int olvasoSzam;

    public Ujsag(String nev, Date datum) {
        this.nev = nev;
        this.datum = datum;
    }

    public boolean cikketKozol(Cikk cikk) {
        if (!cikkek.contains(cikk)) {
            cikkek.add(cikk);
            osszHazugsag += cikk.getHazugsagSzazalekLab();
            atlagHazugsagSzazalek = osszHazugsag / cikkek.size();
            return true;
        }
        return false;
    }

    public String stringAlakuDatum(){
        return new SimpleDateFormat("yyyy.MM.dd").format(datum);
    }

    private static Date aktualisDatum;

    public long napokSzama(){
        return (aktualisDatum.getTime() - datum.getTime())/(24*60*60*1000);
    }

    public String tartalomJegyzek() {
        if(cikkek.isEmpty()){
            return "\nAz újságban nem jelent meg cikk :( ";
        }
        String temp = "\ncikkek: ";
        for (Cikk cikk : cikkek) {
            temp += "\n" + cikk;
        }
        temp += "\nAz újságban átlagosan " + getAtlagHazugsagSzazalek()
                + " % a hazugság.";
        return temp;
    }

    @Override
    public String toString() {
        return nev + " (" + stringAlakuDatum() + ") ";
    }
```

```
    public List<Cikk> getCikkek() {
        return new ArrayList<>(cikkek);
    }

    public double getAtlagHazugsagSzazalek() {
        if(cikkek.isEmpty()) return -1;
        return atlagHazugsagSzazalek;
    }
```

```
public class NyomtatottUjsag extends Ujsag{
    private int meret;
    private int peldanySzam;
    private int maradekMeret;

    public NyomtatottUjsag(String nev, Date datum,
                           int meret, int peldanySzam ) {
        super(nev, datum);
        this.meret = meret;
        this.peldanySzam = peldanySzam;
        this.maradekMeret = meret;
    }

    @Override
    public boolean cikketKozol(Cikk cikk) {
        if(cikk.getKarakterSzam() < this.maradekMeret){
            this.maradekMeret -= cikk.getKarakterSzam();
            return super.cikketKozol(cikk);
        }
        return false;
    }
```

```
public class NetesUjsag extends Ujsag{

    private String link;

    public NetesUjsag(String nev, Date datum, String link) {
        super(nev, datum);
        this.link = link;
    }

    @Override
    public String toString() {
        return super.toString() + "\n\tlinkje: " + link;
    }
```

A vezérlésből most is kiemeljük majd az adatbevitelt, és ezt alaposabban tárgyaljuk, a többi nem igényel különösebb magyarázatot:

```java
public class Main {

    private int CIKKSZAM_HATAR = 30;

    private List<Ujsag> ujsagok;
    private List<Cikk> cikkek;

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

    private void start() {
        beolvasas();
        ujsagIras();
        kiiratas();
        kiadasOtaElteltNapok();
    }

    private void beolvasas() {...17 lines }

    private void ujsagIras() {
        int hanyszor, veletlenCikkIndex, veletlenUjsagIndex;
        hanyszor = (int) (Math.random() * CIKKSZAM_HATAR);
        Ujsag ujsag;
        Cikk cikk;
        for (int i = 0; i < hanyszor; i++) {
            veletlenCikkIndex = (int) (Math.random() * cikkek.size());
            veletlenUjsagIndex = (int) (Math.random()*ujsagok.size());
            ujsag = ujsagok.get(veletlenUjsagIndex);
            cikk = cikkek.get(veletlenCikkIndex);
            ujsag.cikketKozol(cikk);
        }
    }

    private void kiiratas() {
        System.out.println("\nAzt újságok:");
        for (Ujsag ujsag : ujsagok) {
            System.out.println("\n"+ujsag + ujsag.tartalomJegyzek());
        }
    }

    private void kiadasOtaElteltNapok() {
        System.out.println("\n\nArchiválás:");
        for (Ujsag ujsag : ujsagok) {
            System.out.println(String.format("%s, eltelt napok száma: %d",
                                                ujsag, ujsag.napokSzama()));
        }
    }
```

Adatbevitel:

```
/**
 * Itt írjuk elő, hogy milyen metódusokra van szükségünk.
 * Egyúttal arra is kötelezzük az interfész megvalósítóját, hogy
 * a metódusok megírásakor figyeljen a kivételkezelésre is.
 */
public interface AdatBevitel {
    public List<Cikk> cikkListaBevitel() throws Exception;
    public List<Ujsag> ujsagListaBevitel() throws Exception;
}
```

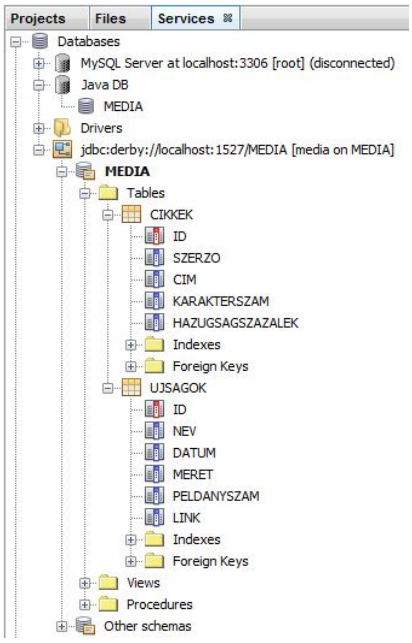
Az interfészt először úgy implementáljuk, hogy fájlból lehessen olvasni az adatokat. Ez megoldható úgy, hogy mindkét metódusban külön megírjuk a fájl megnyitását és a soronként való olvasást. Most azonban egy másik megoldást veszünk, főleg azért, hogy azt is megbeszéljük, hogy egy tiszta kódban, hacsak lehet, nincs kódismétlés. Ezért most a közös részeket csak egyszer írjuk meg, és csupán a sorok feldolgozásában teszünk különbséget. Esetünkben ez nem tűnik egyszerűbb megoldásnak, mint a két független beolvasó metódus, de talán nem árt, ha lát egy példát a kódismétlés elkerülésére.

```
public class FajlBevitel implements AdatBevitel{

    private static String DATUM_FORMATUM = "yyyy.MM.dd";
    private final String CHAR_SET = "UTF-8";

    private List<Cikk> cikkek;
    private List<Ujsag> ujsagok;

    private String cikkFajlEleres;
    private String ujsagFajlEleres;

    enum MitOlvas {CIKK, UJSAG}

    public FajlBevitel(String cikkFajlEleres, String ujsagFajlEleres) {
        this.cikkFajlEleres = cikkFajlEleres;
        this.ujsagFajlEleres = ujsagFajlEleres;
    }

    @Override
    public List<Cikk> cikkListaBevitel() throws Exception {
        cikkek = new ArrayList<>();
        fajlbol(CIKK, cikkFajlEleres);
        return cikkek;
    }

    @Override
    public List<Ujsag> ujsagListaBevitel() throws Exception {
        ujsagok = new ArrayList<>();
        fajlbol(UJSAG, ujsagFajlEleres);
        return ujsagok;
    }
```

```java
/**
 * Itt írjuk le, hogy mit és honnan olvasson.
 * Ez a fájlból való olvasás közös része.
 */
private void fajlbol(MitOlvas mit, String utvonal) throws Exception {

    try (InputStream ins = this.getClass().getResourceAsStream(utvonal);
            Scanner fajlScanner = new Scanner(ins, CHAR_SET)) {
        String sor;
        while (fajlScanner.hasNextLine()) {
            sor = fajlScanner.nextLine();
            if (!sor.isEmpty()) {
                feldolgoz(sor, mit);
            }
        }
    }
}

/** A beolvasandó adatok fajtájától függően dolgozza fel a sorokat ...7 lines */
private void feldolgoz(String sor, MitOlvas mit) throws Exception {

    String[] adatok = sor.split(";");
    switch (mit) {
        case CIKK: {
            // Fecsegő Ferdinánd;Hulla a bokorban;220;30
            cikkek.add(new Cikk(adatok[0], adatok[1],
                    Integer.parseInt(adatok[2]),
                    Integer.parseInt(adatok[3])));
        }
        break;

        case UJSAG: {
            // Kacsa hírek;2018.04.29;1000;30
            // Napi bulvár;2018.05.03;http:/napibulvar.hu
            String nev = adatok[0];
            Date datum = new SimpleDateFormat(DATUM_FORMATUM).parse(adatok[1]);
            try {
                int meret = Integer.parseInt(adatok[2]);
                int peldany = Integer.parseInt(adatok[3]);
                ujsagok.add(new NyomtatottUjsag(nev, datum, meret, peldany));
            } catch (NumberFormatException e) {
                String link = adatok[2];
                ujsagok.add(new NetesUjsag(nev, datum, link));
            }
        }
        break;

        default: {
            throw new Exception();
        }
    }
}
```

Esetleg még meg lehet írni a gettereket is.

Most az adatbázisból való olvasás következik. Az adatbázis szerkezete:




```
CREATE TABLE MEDIA.CIKKEK ( id int not null primary key,
                            szerzo varchar(40),
                            cim varchar(50),
                            karakterszam int,
                            hazugsagszazalek int);

INSERT INTO MEDIA.CIKKEK VALUES(1,'Fecsegő Ferdinánd','Hulla a bokorban',220,30);
INSERT INTO MEDIA.CIKKEK VALUES(2,'Csacsogó Csilla','Csacsogás',1000,40);
INSERT INTO MEDIA.CIKKEK VALUES(3,'Igazmondó Pál','Őszinte politika',300,80);
INSERT INTO MEDIA.CIKKEK VALUES(4,'Rossztollú Rezső','Mellébeszélés',250,60);


CREATE TABLE MEDIA.UJSAGOK ( id int not null primary key,
                           nev varchar(40),
                           datum date,
                           meret int,
                           peldanyszam int,
                           link varchar(50));

INSERT INTO MEDIA.UJSAGOK VALUES(1,'Kacsa hírek','2018-04-19',1000,30,null);
INSERT INTO MEDIA.UJSAGOK VALUES(2,'Napi bulvár','2018-03-23',null,null,'http://hapibulvar.hu');
INSERT INTO MEDIA.UJSAGOK VALUES(3,'Sztár világ','2018-04-21',null,null,'http://sztarvilag.hu');
INSERT INTO MEDIA.UJSAGOK VALUES(4,'Ki kivel','2018-05-03',null,null,'http://kikivel.hu');
INSERT INTO MEDIA.UJSAGOK VALUES(5,'Unicum','2018-04-30',2000,50,null);
INSERT INTO MEDIA.UJSAGOK VALUES(6,'Adatbázis hírek','2018-04-11',null,null,'http://adatbazishirek.hu');
```

Az interfészt implementáló osztály:

```java
public class AdatBazisBevitel implements AdatBevitel {

    private Connection kapcsolat;

    public AdatBazisBevitel(Connection kapcsolat) {
        this.kapcsolat = kapcsolat;
    }

    /** Beolvassa a cikkek adatait ...6 lines */
    @Override
    public List<Cikk> cikkListaBevitel() throws Exception {

        List<Cikk> cikkek = new ArrayList<>();

        if (kapcsolat != null) {
            String szerzo, cim;
            int karakterszam, hazugsagSzazalek;

            // Ezt az SQL utasítást kell majd végrehajtani.
            String sqlCommand = "select * from CIKKEK";

            // Megkérjük a kapcsolatot, hogy hozzon létre egy Statement
            // típusú, utasitasObjektum nevű változót.
            // Az utasitasObjektum hajtatja végre a megadott SQL utasítást.
            // Eredményként egy ResultSet típusú eredményhalmazt kapunk.
            try (Statement utasitasObjektum = kapcsolat.createStatement();
                    ResultSet eredmenyHalmaz =
                            utasitasObjektum.executeQuery(sqlCommand)) {

                // Végigjárjuk az eredményhalmazt, és a kapott adatokból
                // létrehozzuk a cikkek listáját.
                while (eredmenyHalmaz.next()) {
                    szerzo = eredmenyHalmaz.getString("szerzo");
                    cim = eredmenyHalmaz.getString("cim");
                    karakterszam = eredmenyHalmaz.getInt("karakterszam");
                    hazugsagSzazalek = eredmenyHalmaz.getInt("hazugsagszazalek");
                    cikkek.add(new Cikk(szerzo, cim, karakterszam,
                                        hazugsagSzazalek));
                }
            }
        }
        return cikkek;
    }
```

```
    /** Beolvassa az újságok listáját az adatbázisból ...6 lines */
    @Override
    public List<Ujsag> ujsagListaBevitel() throws Exception {

        List<Ujsag> ujsagok = new ArrayList<>();

        if (kapcsolat != null) {
            int meret, peldanySzam;
            String nev, link;
            Date datum;

            String sqlUtasitas = "select * from UJSAGOK";

            // Lehetne ezt is, ekkor névsorba rendezetten olvasná be az adatokat:
            // String sqlUtasitas = "select * from UJSAGOK ORDER BY nev";

            try ( Statement utasitasObjektum = kapcsolat.createStatement();
                    ResultSet eredmenyHalmaz =
                            utasitasObjektum.executeQuery(sqlUtasitas)) {

                while (eredmenyHalmaz.next()) {
                    nev = eredmenyHalmaz.getString("nev");
                    datum = eredmenyHalmaz.getDate("datum");
                    meret = eredmenyHalmaz.getInt("meret");
                    peldanySzam = eredmenyHalmaz.getInt("peldanyszam");
                    link = eredmenyHalmaz.getString("link");
                    if (link != null) {
                        ujsagok.add(new NetesUjsag(nev, datum, link));
                    } else {
                        ujsagok.add(new NyomtatottUjsag(nev, datum,
                                                        meret, peldanySzam));
                    }
                }
            }
        }
        return ujsagok;
    }
}
```

A vezérlés beolvasó metódusa:

a) Ha fájlból akarunk olvasni:

```
private String cikkFajlEleres = "/adatok/cikkek.txt";
private String ujsagFajlEleres = "/adatok/ujsagok.txt";

    private void beolvasas() {

    try{
        Date aktDate = new Date();
        Ujsag.setAktualisDatum(aktDate);

        AdatBevitel adatBevitel
                = new FajlBevitel(cikkFajlEleres, ujsagFajlEleres);

        cikkek = adatBevitel.cikkListaBevitel();
        ujsagok = adatBevitel.ujsagListaBevitel();
    } catch (Exception ex) {
        Logger.getLogger(Main.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```

és ha adatbázisból:

```
private void beolvasas() {

    try (Connection kapcsolat = kapcsolodas()){
        Date aktDate = new Date();
        Ujsag.setAktualisDatum(aktDate);

        AdatBevitel adatBevitel
                = new AdatBazisBevitel(kapcsolat);
        cikkek = adatBevitel.cikkListaBevitel();
        ujsagok = adatBevitel.ujsagListaBevitel();
    } catch (Exception ex) {
        Logger.getLogger(Main.class.getName()).log(Level.SEVERE, null, ex);
    }
}

private Connection kapcsolodas() throws ClassNotFoundException,
                                        SQLException {
    // az adatbázis driver meghatározása
    Class.forName("org.apache.derby.jdbc.EmbeddedDriver");
    // az adatbázis definiálása
    String url = "jdbc:derby://localhost:1527/MEDIA";
    // kapcsolodas az adatbázishoz
    return DriverManager.getConnection(url, "media", "media");
}
```

Ez utóbbi esetben ne felejtse el hozzáadni a projekthez a JavaDb driver-t. (Vagy készítsen Maven projektet.)