Programozási nyelvek (BSc, 18) Java 9. gyakorlat

1. feladat

Készítsen `divisors()` néven függvényt, amely a paraméterként kapott pozitív egész szám
osztóit adja vissza egy lista adatszerkezetben.

2. feladat

a

Készítsen egy `Book` osztályt, amellyel egy általános könyvet fogunk reprezentálni. A `Book` -nak van szerzője és címe (`String`), valamint oldalszáma (`int`). Készítsen `PrintedBook` és `EBook` osztályokat, amelyek a `Book` osztályból származnak. A könyv szerzőjéhez és nevéhez csak `Book` férhet hozzá, az oldalszámhoz a leszármazott osztályoknak
is hozzá kell férniük.

A `Book` -nak legyen paraméter nélküli konstruktora, amely beállítja az adattagjait a következőre: szerző: John Steinbeck, cím: Of Mice and Men, oldalszám: 107. Legyen olyan konstruktora is, amely paraméterként kapja meg a szerző, cím és oldalszám adatokat. Amennyiben a szerző neve rövidebb mint `2` , illetve a cím rövidebb mint `4` karakter, akkor dobjon `IllegalArgumentException` -t.

A `Book` osztálynak legyen egy `getShortName()` metódusa, amely visszatér a könyv adatainak sztringben tárolt, rövidített változatával: a szerző nevének első `2` , a cím első `4` karakterével illetve az oldalszámmal.

Írjon főprogramot, amelyben paraméterekkel illetve paraméter nélkül konstruál egy-egy `Book` -ot, majd írja ki a képernyőre a könyvek rövidített adatait (`getShortName()`).

b

Egy nyomtatott könyv lehet puhafedelű vagy keményfedelű. Ezen értékek tárolására készítsen

egy felsorolt típust `Softcover`, `Hardcover` értékekkel. A `PrintedBook` osztály egy adattagban tárolja a fedél típusát.

Egy `PrintedBook`-t lehessen paraméter nélkül illetve paraméterekkel is konstruálni. Paraméter nélkül hívódjon meg az ősosztály paraméter nélküli konstruktora, a könyv fedelének
típusa legyen `Hardcover`. Egy könyv kinyomtatása `6` oldallal növeli az oldalszámot (azaz az oldalszámot tároló adattaghoz adjon hozzá `6`-t). A paraméteres konstruktor tárolja
a nyomtatott könyv adatait (szerző, cím, oldalszám, fedéltípus).

A `PrintedBook` és `EBook` gyermekosztályoknak legyen `getPrice()` metódusa, amely a
könyv árát számolja ki. Egy puhafedeles nyomtatott könyv ára az oldalszám `2`-szerese, egy keményfedeles nyomtatott könyv ára az oldalszám `3`-szorosa.

Az `EBook` gyermekosztály tárolja egy `int` adattagban a PDF fájl méretét. Az `EBook` osztályt ne lehessen paraméterek nélkül konstruálni; a paraméteres konstruktor tárolja az elektronikus könyv adatait (szerző, cím, oldalszám, fájlméret). Egy `EBook` ára az oldalszám és a fájlméret összege.

Példányosítson `PrintedBook` és `EBook` objektumokat, írassa ki a képernyőre a könyvek rövidített adatait és árait.

c

Írjon `toString()` metódust a `Book` osztályhoz, amely visszatér a könyv szerzőjével, címével és oldalszámával sztringben tárolva. Ezt a `toString()`-et nyilván megörökli `PrintedBook` és `EBook` is. Az `EBook`-nak ez a `toString()` elég jó, `PrintedBook` viszont definiálja ezt felül (override): az ősosztály által visszaadott sztring reprezentáció végén tüntesse fel a fedél típusát is.

Könyveket gyakran idéznek cikkekből, ilyenkor gondosan el kell készíteni egy referenciagyűjteményt, amelyben könyvtípustól függően feltüntetik a szerző nevét, a könyv címét, az idézett oldalszámokat etc.

Írjon a `Book` osztályba `createReference()` metódust, amely paraméterként cikknevet (`String`),
kezdő- és végoldalszámot (`int`) fogad, valamint visszatér egy sztringgel, ami a könyvre mutató szöveges hivatkozást tartalmaz. A hivatkozás formája:

“getShortName() [kezdőoldalszám-végoldalszám] referenced in article: cikknév”

Nyomtatott valamint digitális könyveket ugyanezen paraméterekkel, de más módon kell idézni.
Definiálja felül a leszármazott osztályokban a `createReference()` metódust.

Nyomtatott könyv hivatkozási formája:

“ősosztály toString()-je [kezdőoldalszám-végoldalszám] referenced in article: cikknév”

Digitális könyv hivatkozási formája:

“ősosztály toString()-je (PDF size: méret) [kezdőoldalszám-végoldalszám] referenced in article: cikknév”

Digitális anyagok hivatkozásánál szokás feltüntetni a fájl elérési dátumát. Terhelje túl (overload)
az `EBook` osztály `createReference()` metódusát egy olyan metódussal, amely paraméterként
cikknevet és dátumot (`String`) fogad, majd a következő hivatkozási szöveggel tér vissza:

“ősosztály toString()-je (PDF size: méret) referenced in article: cikknév, accessing PDF date: dátum”

d

Írjon `isSameAuthor()` néven függvényt, amely paraméterként két `Book` referenciát vár, és eldönti, hogy a két könyv szerzője ugyanaz-e. Hívja meg a függvényt `Book` valamint valamely leszármazott osztályok példányaival.

1. gyakorló feladat

Készítse el a `ColouredPoint` osztályt a 6. gyakorlat 1. feladatában lévő `Point` leszármazottjaként. Az osztály rendelkezik egy beágyazott felsoroló osztállyal (`ColouredPoint.Colour`), amely értékei az alábbiak lehetnek: `RED, GREEN, BLUE` .

Az új osztály tartalmazzon privát attribútumként egy színt, és biztosítson lehetőséget a szín beállítására és lekérdezésére nyilvános metódusokon keresztül.

Készítse el a `ColouredCircle` osztályt a 6. gyakorlat 1. feladatában lévő `Circle` leszármazottjaként. Az osztály színét a középpontjában lévő `ColouredPoint` határozza meg, a színhez tartozzon egy lekérdező művelet.
Tesztelje fehérdoboz-teszteléssel a két új osztályt!

2. gyakorló feladat

Egy kávéház működését fogjuk szimulálni. A pultos (`Bartender`) szolgálja ki a vendégeket
(`Guest`). Vendégekből két típust különböztetünk meg, mindketten a `Guest` osztályból származnak:
`Adult` és `Minor` . A pultos különböző italokat (`Beverage`) adhat el a vendégeknek.
Először valósítsa meg a `Beverage` osztályt, ami az alábbi mezőkkel és metódusokkal rendelkezik:

- `name` , egy nemüres sztring
- `legalAge` , pozitív egész

Írjon egy konstruktort, amely minden tagváltozó értékét megkapja és beállítja azokat. A konstruktor
dobjon `IllegalArgumentException` -t, ha valamelyik argumentum nem megfelelő.
A tagváltozókhoz írjon gettereket.

A `Guest` osztálynak két protected láthatóságú adattagja van:

- szöveg típusú név (`name`)
- `int` típusú kor (`age`)

Mindkét adattag legyen elérhető gettereken keresztül.

A pultos rendelkezzen egy nyilvános metódussal:

- `order(Beverage, Guest)` , amely boolean értékű változóval tér vissza. Akkor térjen vissza hamissal,
  ha az ital `legalAge` attribútuma 18 és a `Guest` nem `Adult`

3. gyakorló feladat

Írjon egy `Stream` osztályt, amely karakterláncokat fog előállítani.
Hozzon létre egy `Logger` osztályt is, amely a kapott szövegeket logolja.

A `Logger` osztálynak egy visszatérési érték nélküli metódusa van:

- `log()`, amely egy sztringet vár paraméterül. A metódus törzse legyen üres.

A `Stream` osztály konstruktora argumentumokat vár, amelyeket privát adattagként tárol el:

- A karakterlánc maximális hossza (`maxStringLength`)
- Az előállítandó sztringek száma (`stringNumber`)
- Egy `Logger` amelynek küldi a sztringeket
- Ha valamelyik argumentum nem megfelelő, a konstruktor dobjon `IllegalArgumentException` -t

Legyen egy publikus `startStreaming()` metódusa, amely `stringNumber` alkalommal ħívja meg a `Logger` `log()` metódusát az előállított véletlenszerű sztringgel.

A `ConsoleLogger` osztály származzon a `Logger` -ből és definiálja felül a `log()` metódust - a kapott szöveget írja ki a standard outputra.

A `ConsoleCipherLogger` is a `Logger` -ből származzon, a `log()` metódusa először kódolja a
kapott sztringet a Caesar-kódolással, majd írja ki a standard outputra.

A `FileLogger` terjessze ki a `Logger` osztályt és a konstrukra paraméterként várjon egy fájlnevet. A `log()` metódus a kapott sztringeket írja ebbe a fájlba soronként.

A `Main` osztály tesztelje le a fenti osztályok működését.

4. gyakorló feladat

Készítsen egész számok listáját reprezentáló adatszerkezetet `IntList` néven. Egy (a) részfeladat
keretében készítse el az adatszerkezet hagyományos, tömbökön alapuló megoldását.

a

Az `IntList` osztálynak adattagokban tárolja az `IntList` aktuális és maximális méretét. A maximális méretet a konstruktor állítja be. Az osztálynak legyen egy `add()` metódusa, amellyel egy `int` típusú adatot tehetünk be az `IntList` -be. Írjon `concat()` függvényt, amely egy másik `IntList` referenciát vár paraméterként, és ha az aktuális `IntList` elég nagy,

akkor a végéhez fűzi a paraméterként kapott `IntList` -ben található egészeket. Ha az `IntList`
nem elég nagy elemek hozzáadásakor, akkor a metódusok dobjanak `IllegalStateException` -t.
Írjon `toString()` metódust, amely vesszővel elválasztva felsorolja az `IntList` elemeit. Amennyiben az `IntList` üres, akkor a `toString()` az “empty” sztringet adja vissza.
Írjon `removeItemsGreaterThan()` metódust, amely paraméterként egy egész számot (`limit`)
fogad, és az `IntList` csak azon elemeit hagyja meg, amelyek nem nagyobbak `limit` -nél.

b

A (b) részfeladat ugyanezt az adatszerkezetet valósítsa meg `ArrayList` vagy `LinkedList`
használatával (milyen előnyei vannak a (b) megoldásnak az (a) megoldással szemben?)
Az `IntList` -nek legyen olyan konstruktora, amely egész számok tömbjével inicializálja az újonnan létrehozott `IntList` -et.

c

A © részfeladatban származtasson `NamedIntList` néven osztályt a (b) feladatban megírt `IntList` osztályból. Egy egészek listájának mostantól legyen neve is, a `NamedIntList` osztály
tárolja egy sztring adattagban. Írjon konstruktorokat, amelyek egy nevet illetve egy nevet és
egy egészeket tartalmazó tömböt fogadnak, majd ezen adatokkal inicializálja a létrejött `NamedIntList` objektumot. Definiálja felül az ősosztálytól örökölt `toString()` metódust
úgy, hogy a sztringben tüntesse fel a `NamedIntList` nevét is.