Programozási nyelvek (BSc, 18) Java 6. gyakorlat

útmutatás

Terjedés követése: fordítási hiba javítva

```
import java.io.IOException;
class TestTime {
    public Time readTime( String fname ) throws IOException {
        ... new java.io.FileReader(fname) ...
    }

    public static void main( String[] args ) throws IOException {
        TestTime tt = new TestTime();
        Time wakeUp = tt.readTime("wakeup.txt");
        wakeUp.aMinutePassed();
    }
}
```

Kivételkezelés

```
import java.io.IOException;
class TestTime {
    public Time readTime( String fname ) throws IOException {
        ... new java.io.FileReader(fname) ...
    }
    public static void main( String[] args ){
        TestTime tt = new TestTime();
        try {
            Time wakeUp = tt.readTime("wakeup.txt");
            wakeUp.aMinutePassed();
        } catch( IOException e ){
            System.err.println("Could not read wake-up time.");
        }
    }
}
```

Kivételkezelés

A program tovább futhat a probléma ellenére

```
public class Receptionist {
    ...
    public Time[] readWakeupTimes( String[] fnames ){
        Time[] times = new Time[fnames.length];
        for( int i = 0; i < fnames.length; ++i ){
            try {
                times[i] = readTime(fnames[i]);
            } catch( java.io.IOException e ){
                times[i] = null;   // no-op
                System.err.println("Could not read " + fnames[i]);
            }
        }
        return times; // maybe sort times before returning?
    }
}
```

https://stackoverflow.com/questions/11589302/why-is-throws-exception-necessary-when-calling-a-function
https://stackoverflow.com/questions/2683958/why-is-nullpointerexception-not-declared-as-a-checked-exception
https://stackoverflow.com/questions/297303/printwriter-and-printstream-never-throw-ioexceptions

1. feladat

a

Módosítsa a 3. gyakorlat 5. feladatát úgy, hogy egy `Point` objektumot paraméterek nélkül is lehessen konstruálni! Ekkor az `x` és `y` adattagja 0-ra inicializálódjon.
Írjon a `Point` osztály `x` és `y` adattagjaihoz setter metódusokat. Oldja meg a `Circle` konstruktorában azt, hogy az osztály a paraméterként kapott `Point` -ról másolatot tároljon (ne csak referenciát).

Egy körnek mostantól legyen címkéje (neve) is! Bővítse a `Circle` osztályt egy címke (`String`) tárolására alkalmas adattaggal. Konstruáláskor egy `String` paraméterrel meg lehessen adni a kör címkéjét; a címke kerüljön be a `toString()` -be is. Az osztálynak legyen egy `defaultLabel` , “unnamed” értékű statikus `final` adattagja.
Ha a konstruktor címkének `null` referenciát kap paraméterként, akkor a létrejövő kör címkéje `defaultLabel` legyen.

A `Circle` osztálynak legyen olyan konstruktora, amely paraméterként nem `Point` referenciát vár, hanem `x` és `y` `double` típusú koordinátákat, majd hívja meg az előbb megírt `Circle` konstruktort.

Írjon a `Circle` osztályhoz statikus `readFromFile()` metódust, amely betölti a paraméterben kapott fájlból egy kör adatait (`x` és `y` koordináta, sugár és címke újsorral elválasztva), megkonstruál ezen adatokkal egy `Circle` objektumot, majd visszatér az objektum referenciájával. A függvény a fellépő kivételeket engedje tovább a hívóhoz.

Írjon a `Circle` osztályhoz `saveToFile()` metódust, amely az aktuális `Circle` objektum
adatait a paraméterként megadott fájlba menti. Amennyiben a fájlba írás kivételes eseménybe ütközik, a függvény engedje tovább a kivételes eseményt a hívóhoz. Gondoskodjon arról, hogy ha a kiírás menet közben ütközik kivételes eseménybe, akkor a már kiírt adatok ne vesszenek el (azaz a `PrintWriter` -t mindenképpen be kell zárni).

b

Készítse el az (a) megoldás egy olyan változatát, amelyben a `readFromFile()` metódus megpróbálja kezelni a fellépő kivételes eseményeket. Ha valamilyen kivételes esemény miatt nem sikerül a beolvasás, akkor jöjjön létre kör objektum csupa 0-szerű adatokkal.

útmutatás

JavaDoc formátum:

```
/**
* ...
* ...
*/
```

Hogyan kell kinézniük a Java dokumentációs megjegyzéseknek, ezen belül a blokkcímkéknek, szövegközi címkéknek (block tags, inline tags)?

@param,@return,@throws etc. -blokkcímke

{@code},{@link} -szövegközi címke

numbers/Rational.java:

```
package numbers;


public class Rational {
    /**
    *  Set {@code this} to {@code this} * {@code that}.
    *  @param that Non-null reference to a rational number,
    *              it will not be changed in the method.
    *  @throws NullPointerException When {@code that} is null.
    */
    public void multiplyWith( Rational that ){
        this.numerator *= that.numerator;
        this.denominator *= that.denominator;
    }
}
```

HTML generálás: `javadoc Rational.java`

2. feladat

Módosítsa az 5. gyakorlat 3. feladatának a forráskódját a következőképpen:

Rendezze osztályszintű metódusba a korábban megírt logikát, a metódus viszont a parancssori argumentumok tömbjén kívül kapjon egy double típusú értéket is, amire vizsgálja meg, hogy a két argumentumon elvégzett művelet után kapott érték lefele kerekítve egyenlő-e vele (`boolean` -nal térjen vissza). A metódus hiba esetén ne kezelje le a kivételeket, ez a `main()` -ből való meghíváskor történjen.

Írjon dokumentációs megjegyzést a fentebb leírt metódushoz, amiben leírja röviden a funkcionalitását. Tartalmazza legalább az alábbi címkéket:

- `@param`
- `@returns`
- `@throws`

Készítsen az osztályhoz is dokumentációs megjegyzést. Tartalmazza az `@author` , `@version` , `@since` tageket.

A `javadoc` program segítségével generáljon HTML dokumentációt a Java programhoz.

útmutatás

```
public enum Day {
    SUNDAY, MONDAY, TUESDAY, WEDNESDAY,
    THURSDAY, FRIDAY, SATURDAY
}
//...
Day d1 = Daay.MONDAY ;
Day d2 = Day.valueOf("MONDAY");
```

`values()` statikus metódus, mely visszaadja az összes enumértéket
`ordinal()` az adott `enum` érték az osztálydefinicióban való helyzetét adja meg

https://docs.oracle.com/javase/tutorial/java/javaOO/enum.html

3. feladat

Készítse el a `WildAnimal.java` fájlba a `WildAnimal` felsorolási típust (`enum` -ot), amelyben legyen négy felsorolási tag: majom, elefánt, zsiráf és mosómedve.
Az állatok konstruktorában első paraméternek megkapják azt, hogy melyik gyümölcsöt szeretik enni, második paraméterként pedig azt, hogy mennyi lenne ideális esetben egy napi adagjuk az adott gyümölcsből.

Készítse el a `listAllAnimals()` metódust, amely egy ilyen formátumú szöveggel tér vissza:

"*A vadállat sorszáma*: *a vadállat neve* szeretne enni *a vadállat gyümölcse* egy héten."

Például, ha az elefánt megadott napi mennyisége 30 málna volt:
"2: Elefánt szeretne enni 210 málnát egy héten."

Az `enum` elemeinek bejárásához használja a `values()` , illetve a sorszám lekérdezéséhez az `ordinal()` metódust.

Készítsen saját `toString()` metódust, amely az adott `enum` elem által meghívott állatról írja ki az információkat.

Próbálja ki az elkészített felsorolási típust és a hozzátartozó metódusokat egy `Main` osztályban.

1. gyakorló feladat

Készítsünk egy egyszerű `Color` felsorolt típust, mely a következő értékeket tárolhatja: `RED`, `GREEN`, `BLUE`.

Írjunk egy `Auto` osztályt, mely a következő információkat tárolja:

- rendszám (`String`)
- szín (`Color`)
- maximális sebesség (`int`)

Írjunk az osztályhoz egy konstruktort, mely ezt a három értéket várja paraméterként. Az osztályban legyen számláló, mely tárolja, hogy hány objektumot hoztunk eddig létre.

Írjunk egy paraméter nélküli konstruktort is, mely `AAA-000`, `BLUE` és `120` értékekkel hoz létre objektumot.

Írjunk egy osztályszintű összehasonlító metódust, mely két autó objektumot vár, és igazzal tért vissza, ha az első gyorsabb mint a második.

Helyezzük a `Color` osztályt az `auto.utils` csomagba, az `Auto`-t pedig az `auto` csomagba!

Hozzunk létre egy `input.txt` fájlt, melyben autók adatai vannak soronként megadva, vesszővel elválasztva. Pl: ABC-123,RED,100

Írjunk egy `Main` osztályt (a csomogokon kívül), amely a tesztprogramunkat fogja tartalmazni! A `Main` osztály `main()` metódusában olvassuk be az input fájl tartalmát, a létrehozott objektumok referenciáit pedig tároljuk el egy tömbben.

2. gyakorló feladat

Készítsen az 1. feladat forráskódjához JavaDoc dokumentációs megjegyzéseket. A kommentek tartalmazzanak információkat a metódusok paramétereiről és visszatérési értékeiről, a dobott kivételekről. Generáljon HTML fájlt a `JavaDoc` programmal.

3. gyakorló feladat

Készítse el az előadáson már előforduló `Time` osztályt teszőlegesen választható időreprezentációval. Kezelje le az előfoduló kivételeket és legyen egy metódusa, ami `String` -ként visszaadja az időt.

Készítsen egy `Pizza` osztály, amelynek legyen az összes mezője `private` és `final` ; egy mezője az átmérő (`double`), egy a feltétek listája (`String[]`), egy az elkészítési idő (`Time`) és egy a szállítási idő (`Time`). Írjon hozzá konstruktort, amely a feltétek szerint kiszámolja, hogy mennyi az elkészítési idő, tehát az ne legyen paraméterül átadva a konstruktornak. Az elkészítési idő a következő képlet szerint történjen: az összes feltétre a következők összege: a feltét nevének hossza szorozva a pizza átmérőjével (centiméterenként és feltétbetűnként egy másodperc).

Pl.: 32 cm-es pizza “cheese” feltéttel: 32*6 = 192, amit kerekítsünk 4 percre.
A konstruktor dobjon kivételt, ha valamely paraméter nem értelmezhető. Dobjon `TypeNotPresentException` -t, ha a feltét nincs benne a következő listában: `beef, cheese, corn, fish, ham, mushroom, salami, tomato` . Jelenítse meg a `throws`
kulcsszóval, hogy a `Pizza` osztály példányosításakor le kell kezelni a `TypeNotPresentException` -t.

Írjon `Main` programot, amelynek egy statikus metódusa beolvas a paraméterként átadott fájnévű fájlból pizzarendeléseket, és létrehozza egy listában az összes lehetséges pizzát, majd kiszámítja, hogy mennyi ideig kell aznap sütni. Kezelje le a kivételeket, amelyek felléphetnek a pizzák létrehozása során.

Végül mutassa be ennek a `Main` osztálynak statikus metódusának használatát a `Main` osztályon belül.

4. gyakorló feladat

Készítsen egy `TelevisionShop` felsorolási típust. A felsorolási tagok legyenek `SAMSUNG, LG, SKYWORTH, SONY, SHARP` . A konstruktorukban az első tag legyen, hogy hány db készülék van az adott márkából raktáron, a második és a harmadik az elérhető átmérők minimuma és maximuma legyen. Készítsen hozzá olyan metódukokat, amelyekkel
ki tudja íratni az összes lehetséges kapható méret minimumát és maximumát típustól

függetlenül (statikus) és olyat, amely adott márkára kiírja, hogy mekkora méretű tévéket lehet kapni. Készítsen statikus metódust, amellyel kiírja a rendelkezésre álló készletről minden tudhatót!

Használja a `final` kulcsszót, ahol lehet!