5. gyakorlat anyaga

Feladatok

1. 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.

2. feladat

Készítsünk egy, a nemek ábrázolásához használt `Gender` nevű felsorolási típust! Ebben szerepeljen két érték, amelyek rendre `Gender.MALE` (férfi) és `Gender.FEMALE` (nő).
Készítsünk `Person` névvel egy olyan osztályt, amelyben nyilvántartjuk a személyi adatokat! A rögzíteni kívánt adatok: a személy vezeték és keresztneve (mindkettő `String`), foglalkozása (`String`), neme (`Gender`) és születési éve (`int`).

Legyen a `Person` osztálynak egy olyan konstruktora, mely ezeket az adatokat paraméterként kapja.

Egészítsük ki a `Person` osztályt egy `toString()` metódussal, amely `String` típusú értékké alakítja az adott objektum belső állapotát! Készítsünk egy `equals()` nevű metódust a `Person` osztályhoz, amely eldönti a paraméterként megadott másik `Person` objektumról, hogy megegyezik-e az aktuális példánnyal. Vigyázzunk arra, hogy mivel referenciát adunk át paraméterként, az lehet (többnyire véletlenül) `null` érték is! Ilyenkor értelemszerűen az eredménye hamis lesz.

Tegyük az eddigi osztályokat a `person` csomagba és készítsünk hozzá egy főprogramot, amelyben létrehozunk két `Person` objektumot, megvizsgáljuk, hogy ugyanarról a két személyről van-e szó és az eredményt kiírjuk a szabványos kimenetre! A főprogram kerüljön a `main` csomagba!

3. 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 metódusokat, 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!

Gyakorló feladatok

1. gyakorló feladat

1. Készítsd el a `City` felsorolási típust.
   - a felsorolási típus elemei legyenek a városok nevei (csupa nagybetűvel)
   - a típusnak legyen `zipCode` mezője (irányítószám), egy `int`
   - lehessen az elemeket egy `int`-tel paraméterezni, amit vegyen át a mezőbe
   - lehessen egy másik várossal paraméterezni az elemeket, ekkor az aktuális elem irányítószáma a paraméterből jöjjön
     - számít-e a sorrend?
   - készíts saját `toString` metódust hozzá
2. Írd ki a `City` elemeit sorban.
3. Készítsd el a `WeekDay` felsorolási típust.
   - Az elemek legyenek a napok nevei rövidítve.
   - Készítsd el a `nextDay` metódust, ami kiadja a következő napot.
     - Legyen ebből túlterhelt változat, ami kapja meg, hány napot lépjünk előre.
     - Negatív szám esetén is működjön, értelemszerűen visszafelé lépésekkel.
   - Az elemek paraméterei legyenek a napok teljes nevei különböző nyelveken. Tetszőleges sok paramétert kaphasson az elem.
     - Egy `static` mező írja le a támogatott nyelvek szöveges kódjait egy tömbben.
   - Legyen egy `get(String lang)` metódus, amely adja vissza a nap nevét a megadott nyelven, illetve a `"?"` szöveget, ha a nyelv nem ismert, vagy a paraméterezésben nincsen jelen.
   - Lehessen úgy is megadni a neveket, hogy azok tetszőleges sorrendben legyenek leírhatók az adott naphoz, és ne feltétlenül egy előre meghatározott sorrendben kelljen őket megadni. Akár az is lehetséges így, hogy két különböző napnak nincs is közös nyelven neve.
4. Készíts három osztályt: a `Post` osztály kezdetben két `PostOffice` példányt kap meg, és `Letter`-öket (leveleket) fog elküldeni.
   - A `Post` osztály `send` metódusa egy `Letter` példányt kap meg, és kézbesíti: a páros irányítószámúakat az első, a páratlan irányítószámúakat a második postafiókba (`receive` metódus).
   - A `Letter` vagy a címzett irányítószámát kapja meg, vagy a várost, ami alapján az irányítószám egyértelműen kiderül.
   - A levélen legyen rajta a nap is: a postára feladáskor paraméterként kelljen megadni, hogy melyik napon történt a feladás. Amikor a levél a következő postaállomásra utazik, lépjen egy napot.

Extra feladat

Ez a feladat nem képezi a kötelező tananyag részét, érdeklődő hallgatóknak ajánlott. A szükséges ismereteket a gyakorlaton nem vesszük, nekikezdés előtt érdemes lehet a Java tutorialt elolvasni.

Annotációk

A múlt héten találkoztunk a JUnit által deklarált `@Test`, `@ParameterizedTest` stb. annotációkkal. Ezekhez hasonló, egyszerű annotációt hozhatsz létre a következő feladatban.

1. Készíts `@Author` annotációt típusokhoz, amely a típus készítőjének nevét írja le.
   a. Készíts metódust, amely megkapja a típus nevét szövegesen, és kiírja a készítő nevét (ha az annotáció szerepel a típuson).
   b. Az annotáció legyen alkalmazható metódusokra is.
      a. Készíts metódust, amely megkapja a típus nevét szövegesen, és kiírja a típus azon metódusainak nevét, amelyeknek más a készítője, mint a típusé.
   c. Az annotáció legyen többszörözhető (`@Repeatable`).

    a. A fenti metódusok működjenek akkor is, ha egy szerző szerepel, és akkor is, ha több.
2. Készíts `@Date` annotációt, amelyet metódusok kaphatnak meg.
    a. Készíts metódust, amely paraméterként kapott nevű osztály metódusai közül megkeresi azokat, amelyek rendelkeznek az annotációval, és paraméter nélküliek. Ezek közül hívd meg azokat, amelyek egy (paraméterben kapott) dátumnál nem régebbiek.
3. Készíts `@ParameterFor` annotációt, amely metódusra alkalmazható. A metódus valamilyen értékek tömbjét adja vissza. Az annotáció kap egy szöveget, ami ugyanabban az osztályban egy metódus neve; ez a metódus olyan típusú paramétert vár, mint amilyen értékek a másik metódus visszatérési értékének tömbjében szerepelnek. Az annotáció legyen többszörözhető.
    a. Készíts metódust, amely paraméterként kapott nevű osztály metódusai közül megkeresi azokat, amelyek rendelkeznek az annotációval. Az annotációban szereplő nevű metódust hívd meg sorban azokkal a paraméterekkel, amelyeket az annotált metódus visszatérési értékének tömbjében szerepelnek.