

Java PR1

Feladatok

Alapok

1. Készíts egy programot, mely kiírja a konzolra hogy „Hello world”!
2. Módosítsd az előbb elkészített programot, hogy a futtatáskor paraméterként átadott felhasználónévvel üdvözlje a felhasználót!

Vezérlési szerkezetek

3. Készíts egy programot, amely a paraméterként átadott két szám közül a nagyobbát írja ki! Ha nincs elég paraméter akkor jelezze a hibát! (A paraméterek típusával még nem kell foglalkoznod.)
4. Készíts egy programot, amely paraméterként egy betűt és egy számot kap, és a betűt annyiszor írja ki egymás után, amennyi a második szám!
5. Készíts egy programot, amely csillagokból kirajzol egy téglalapot. A téglalap méreteit paraméterként kapja meg!

Pl: Teglalap 3 5 esetén:

Algoritmizálási alapok

6. Készíts egy programot, amely egy egész számokból álló tömb elemeit adja össze és átlagolja!
7. Egészítsd ki az előző programot, hogy megkeresse az első negatív számot! Ha nincs benne negatív, akkor írja ki, hogy „Nincs a számok között egyetlen negatív sem.”!
8. Készíts programot, amely szavak tömbjéből írja ki a legrövidebb és a leghosszabb szót!
9. Készíts programot, amely szavak tömbjéből kiválogatja az összes „A” betűvel kezdődőt, és alfabetikus sorrendben kiírja őket egymás után a konzolra!

Osztályok

10. Készíts egy `Car` osztályt egyszerű szövegszerkesztővel!
Attribútumai:
`registrationNumber: String`
`positionX: int`
`positionY: int`
`speed: int`
Fordítsd le konzol ablakból, majd fordítsd vissza!
11. Importáld IDE-be a `Car` osztályt és egészítsd ki konstruktorral, amelyben minden attribútumot megkap paraméterként!

12. Készíts olyan konstruktort, amely csak a rendszámot kapja meg! Minden más attribútum legyen 0!
13. Módosítsd az előbb megkapott konstruktort, hogy az autó kezdő pozíciója (100, 100) legyen!
14. Egészítsd ki a `Car` osztályt getterekkel/setterekkel és az alábbi metódusokkal!
`accelerate()`: egy egységgel növeli az autó sebességét
`move(deltaX: int, deltaY: int)`: megváltoztatja az autó pozícióját a paraméterként kapott értékekkel
15. Egészítsd ki az osztályt úgy, hogy ha az `accelerate` metódus kap paramétert, akkor annnyival változtatja a sebességet.

Öröklődés

16. Készíts egy `Bicycle` osztályt!
Attribútumai:
`positionX: int`
`positionY: int`
`speed: int`
Metódusai:
`accelerate()`, `accelerate(int)`, `move()`

Módosítsd a `Car` osztályt: legyen egy `fuelLevel` és egy `maxFuelLevel` egész attribútuma. A `maxFuelLevel` mindig 50. Legyen két `fill` metódusa. Az egyik mindig tele tankol, a másik pedig a paraméterként átadott mennyiséget.

A járműveket egységesen szeretnénk kezelni, ezért készíts egy közös őst `Vehicle` néven. Milyen attribútumokat és metódusokat tennél bele? Valósítsd meg!
Készíts egy `VehicleCounter` osztályt, amely egy tömböt tartalmaz a különböző járművekről!
Metódusai:
`countMovingVehicles()`: az éppen mozgásban lévő járművek számát adja vissza.
`listVehicles()`: a konzolra kilistázza a járművek tulajdonságait olvasható formában (`toString`).
`countBycyclesInArea(int x, int y, int width, int height)`: a paraméterként átadott területen található biciklik számát adja vissza. A terület mindig téglalap alakú és a bal felső sarkának pozíciója (x, y).
`findByRegNumber(String regNumber)`: megkeresi az adott rendszámú autót és minden tulajdonságát kiírja a konzolra.

Kapcsolatok

17. `Person`, `Employee`, `Boss` és `BigBoss` osztály
Ezek egy munkahelyi hierarchiát reprezentálnak, a fenti sorrendben egymásból öröklődő osztályok. Jelen esetben a `Person` osztályt nem is példányosítjuk, ez az alatta levő osztályok egyfajta absztrakciójának tekinthető. A különböző alkalmazottak fizetését eltérő módon számítjuk. Míg az `Employee` alapfizetéssel rendelkezik, a `Boss` esetében az alapfizetéshez

hozzáadódik a vezetői pótlék (beosztottak száma * LEADERSHIP_FACTOR * alapfizetés), míg a BigBoss esetében ehhez hozzáadódik egy vezetői prémium is (bonus).

Person osztály String name és String address attribútumokkal

Publikus metódusok:

```
public Person(String name, String address)
public void migrate(String newAddress)
```

Employee osztály double salary attribútummal

Publikus metódusok:

```
public Employee(String name, String address, double salary)
public double getSalary()
public void raiseSalary(int percent)
```

Boss osztály LEADERSHIP_FACTOR = 0.1 és int numberOfEmployees attribútumokkal

Publikus metódusok:

```
public Boss(String name, String address, double salary, int
numberOfEmployees)
public double getSalary()
public int getNumberOfEmployees()
```

BigBoss osztály double bonus attribútummal

Publikus metódusok:

```
public BigBoss(String name, String address, double salary, int
numberOfEmployees, double bonus)
public double getBonus()
public double getSalary()
```

18. Item, Basket, ShoppingBasket osztály

Az öröklődés mellett/helyett kompozíció is alkalmazható, ahol az alkotó osztályok egymás szolgáltatásait használják ki új funkciók megvalósítására. Itt az alap Basket osztály Item objektumokat tárol, és a ShoppingBasket osztály attribútumként tárol egy Basket objektumot, magasabb szintű és részben másféle funkciók kiszolgálására.

Item osztály String barcode, double nettoPrice, int vatPercent attribútumokkal

Publikus metódusok:

```
public double getTaxAmount(): a nettoPrice és a vatPercent alapján
kiszámolja az adó összegét
public double getNettoPrice()
public String getBarcode()
public String toString()
```

Basket osztály List<Item> items attribútummal

Publikus metódusok:

```
public void addItem(Item item)
public void removeItem(String barcode)
```

```

public void clearBasket(): a Basket ürítése
public List<Item> getItems(): az Item lista másolatát adja vissza!

ShoppingBasket osztály Basket basket attribútummal
publikus metódusok:
public void addItem(Item item)
public void removeItem(String barcode)
public double sumNettoPrice(): az összes tételre
public double sumTaxValue(): az összes tételre
public double sumBruttoPrice()
public void checkout(): befejezzük a vásárlást, a kosár ürítése
public void removeMostExpensiveItem(): kikeresi és eltávolítja a kosárból a
legdrágább tételt

```

Enum

19. Készíts egy enum-ot Season néven, amelybe az évszakok neveit teszed! Készíts egy Zoo osztályt, amelynek a getOpeningHours(Season) metódusa kiírja a nyitvatartási időt a paraméterként átadott Season alapján. Tavasszal és ősszel 9:00-18:00-ig, télen 10:00-17:00-ig, nyáron 08:00-20:00-ig.
20. Módosítsd az előző feladatban elkészített Season enumot, hogy minden évszaknál le lehessen kérdezni, hogy melyik a következő (next) és az előző (previous), valamint az openingHours() metódusa írja ki, hogy az állatkert az adott évszakban mikor van nyitva! A Zoo osztály getOpeningHours(Season) metódusa csak delegálja a feladatot az enumnak.

Interfészek használata

Készíts interfészt Currency néven. Az implementáló osztályoknak mindig van rövidítése, pénzjele, és váltása bármilyen másik Currency felé. Ezt úgy valósítsd meg, hogy minden pénznem tud váltani alapegységre és alapegységről, valamint ezek használatával a paraméterként kapott pénznemre is. Készíts hozzá 3 megvalósítást: UsDollar, HunForint, Euro.

Metódusok:

```

getShortName(): USD, HUF, CHF, EUR
getSign(): $, Ft, €
toBase(double value): alapegységre vált
fromBase(double value): alapegységről vált
toOtherCurrency(double value, Currency currency): a megadott összeget
az adott valutára váltja

```

Most egészítsd ki új SwissFrank valutával: CHF, F
Nagy munka volt?

Generikusok

21. Valósíts meg egy keresőszolgáltatást, ahol minden `Item` típusú elemet tárolni lehet. (Az `Item` legyen egy interfész.) Ez lehet például `Book`, de legyen lehetőség más típusú elemek implementálására is. A könyvtárban különböző szempontok szerint lehet keresni az egyes konkrét elemek között, például szerző vagy cím alapján. A kereséshez egy generikus `SearchService<T extends Item>` osztályt kell létrehozni, melynek a `public T findFirst(List<T> items, SearchCriteria criteria)` metódusa implementálja a keresést. A `SearchCriteria` egy olyan interfész, melynek van egy `boolean pass(T target)` metódusa, mely azt adja vissza, hogy a paraméterként megadott objektum megfelel-e a feltételnek.

Készítsd el ennek két implementációját:

`BookAuthorSearchCriteria`: konstruktorban kap egy szerzőt, és az alapján keres

`BookTitleSearchCriteria`: egy címet kap, és ez alapján keres

Kollekciók

22. Készíts lottóhúzó programot! Lehesen paraméterezni, hogy hány számból hányat kell kihúzni. A nyertes számok kiírása növekvő sorrendben történjen!
23. Készíts feladatkezelő programot! A feladatoknak legyen prioritása, címe és leírása. A feladatok csak egymás után hajthatók végre, és legelőször a legfontosabbat kell elvégezni. Lehesen új feladatot felvenni, még nem elvégzettet törölni. Vissza lehesen nézni az elvégzett feladatokat a végrehajtásuk sorrendjében.
24. Készíts egy `Book` osztályt `int regNumber, String title, String author, double price` attribútumokkal! Az `OrderedLibrary` osztálynak egyetlen attribútuma van: `List<Book> libraryBooks`, ezt konstruktorból tudjuk feltölteni. A könyvek listáját le lehesen kérdezni bármelyik attribútum szerint rendezetten. Az alapértelmezett rendezés a `regNumber` szerint történik.
- Tipp: Ha az ékezetesek szerint helyesen szeretnél rendezni, akkor szükséged lesz a `Collator` osztályra. Nézz utána, hogyan tudod használni!