Objektum orientált programozás
12. gyakorlat
Gyakorlás féléves programozás beszámolóra

1. feladat: alapfeladat, szintidő: 30 perc

Definiáljon „homero” csomagban egy „Homerseklet” osztályt, melynek adattagjai: a hőmérséklet (valós) és a mértékegység (enum: CELSIUS, KELVIN).

- Definiáljon konstruktort, amellyel egy hőmérséklet adatai beállíthatók (2 paraméteres)
- Definiáljon konstruktort, amellyel inicializál egy hőmérsékletet (1 paraméteres) és a mértékegység automatikusan CELSIUS
- Definiálja át a toString metódust, amelyben visszaadja a hőmérséklet összes adatát.
- Definiáljon getter, setter metódusokat az adattagok értékének beállítására és lekérdezésére.
- Definiáljon osztályszintű „konvHomerseklet” metódust, amely paraméterben megkap egy hőmérsékelt értéket és egy mértékegységet és visszaadja az adott hőmérséklet értéket a paraméterben megadott mértékegységben (0 Celsius=273.15 Kelvin; Celsius to Kelvin x+273.15, Kelvin to Celsius x-273.15)

Definiáljon ugyanebben a „homero” csomagban egy futtatható „HoProba” osztályt majd oldja meg az alábbi feladatokat:
- Hozzon létre négy hőmérséklet objektumot. Az első kettő adatait ellenőrzött módon olvassa be: az 1. adatai 15 és Celsius, a 2. adatai 300 és Kelvin, az utolsó kettő értéke pedig 0-100 között véletlenszerűen generált Celsius. Tárolja el ezeket az objektumokat egy tömbben.
- Írjon metódust, amely paraméterként megkapja ezt a tömböt és kiírja a tömb objektumainak adatait (foreach ciklussal).
- Konvertálja az 1. Celsius mértékegységű objektum értékét Kelvin-re majd írja ki az értékét.
- Konvertálja a 2. Kelvin mértékegységű objektum értékét Celsius-ra majd írja ki az értékét.
- Írjon metódust amely paraméterként megkapja ezt a tömböt és visszaadja az átlaghőmérsékletet Kelvinben. Ügyeljen arra, hogy ha a hőmérséklet Celsius volt akkor azt előbb Kelvinre kell konvertálni! Írassa is ki ezt az átlaghőmérsékletet.

Elvárt output:
A tömb elemeinek listázása:

```
Homerseklet [ertek=15.0, mertekegyseg=Celsius]
Homerseklet [ertek=300.0, mertekegyseg=Kelvin]
Homerseklet [ertek=58.2614336710285, mertekegyseg=Celsius]
Homerseklet [ertek=86.99428465068647, mertekegyseg=Celsius]
```

Konverzió:

```
15 Celsius : 288.15 Kelvin
300 Kelvin : 26.850000000000023 Celsius
```

Számítás:

```
Az átlaghőmérséklet: 319.9264295804287 Kelvin
```

2. feladat: emelt szintű feladatkiegészítés, szintidő: 15-20 perc. Az első feladatban megírt futtatható osztály kiegészítése az alábbi részfeladatokkal:

a) Oldja meg az első feladatot úgy, hogy egy szövegállományból olvassa be a megadott adatokat. Váltsa át azonos mértékegységre az adatokat, rendezze a tömböt és írja ki egy szöveges állományba.

b) Keressen meg a tömbben egy adott elemet az Arrays osztály binarySearch metódusával.

Első lépés: a tömb elemeit azonos mértékegységre átváltani (pl. CELSIUS-ra)
Második lépés: a tömb rendezése a hőmérséklet értékek alapján
Harmadik lépés: a kereséshez Comparator írása a futtatható osztályban névtelen osztályként

```
Comparator<Temperature> c = new Comparator<Temperature>() {
	public int compare(Temperature t1, Temperature t2) {
		return t1.getValue() - t2.getValue();
	}
};
```

Megjegyzés: ha sztring adattagról van szó, akkor :

```
	return o1.getName().compareTo(o2.getName());
```

Negyedik lépés: keresés (tömb, keresett elem, Comparator) ; a keresés a megtalált elem indexét adja vissza. Ha a keresett elem nincs a tömbben, -1-et ad.

```
int index = Arrays.binarySearch(tempArray, new Temperature(100), c);
```

Figyelem! Bináris keresés csak rendezett tömbbön értelmezett művelet. A rendezés iránya és a Comparator compare metódusában az összehasonlítás iránya legyen összhangban!

c) Az előző feladatot oldja meg dinamikus tömbbel.

Tömb átalakítása listává:

```
ArrayList<Temperature> tempList =
                        new ArrayList<Temperature>(Arrays.asList(tempArray));
```

Lista rendezése:

```
tempList.sort(...);
```

Bináris keresés rendezett listában:
Először létrehozunk egy Comparator-t névtelen osztályként (ugyanaz, mint a tömbös megoldásnál).

```
Comparator<Temperature> c = new Comparator<Temperature>() {
	public int compare(Temperature t1, Temperature t2) {
		return t1.getValue() - t2.getValue();
	}
};
```

Majd bináris keresés rendezett listában (lista, keresett elem, Comparator). A megtalált elem indexével tér vissza:

```
Collections.binarySearch(tempList, new Temperature(100), c);
```

3. feladat: alapfeladat, szintidő: 40 perc

Definiáljon saját csomagban bankkártya osztályt **Card** néven.
Adattagjai (csak az osztályon belül érhetők el): tulajdonos neve (String), érvényesség (LocalDate), kibocsátó bank (enum: OTP, K&H, ERSTE, CIB), egyenleg (int).
Konstruktora: a négy paraméterként kapott értékkel inicializálja az adattagokat.
Metódusai:
- Getter metódusok az adattagok lekérdezéséhez.
- toString: egy sztringbe összefűzve adja vissza a kártya adatait.
- Az egyenleget a paraméterben megadott értékre beállító metódus.

- Pénzfelvét: a paraméterként kapott összeggel csökkenti az egyenleget és igazat ad vissza (sikeres pénzfelvét). Ha nem tudja, mert nincs elég fedezet, akkor hamisat ad vissza. Akkor is hamisat ad vissza, ha érvénytelen a kártya (a kártya érvényessége az aktuális dátumnál régebbi). Az összehasonlításhoz használja a LocalDate osztály isBefore() metódusát.

Definiáljon ugyanebben a csomagban hitelkártya osztályt **CreditCard** néven a bankártya osztály kiterjesztéseként.
Adattagjai (csak az osztályon belül érhető el): hitelkeret (int)
Konstruktorai:
1. paraméterei: a kártya tulajdonosa, érvényessége, a kibocsátó bank és az egyenleg. A hitelkeret 100000 Ft.
2. paraméterei: a kártya tulajdonosa, érvényessége, a kibocsátó bank, az egyenleg és a hitelkeret.
Metódusai:
- A pénzfelvétel annyiban módosul, hogy ha az egyenleg nem fedezi a paraméterként kapott összeget, akkor a hitelkeretet lehet csökkenteni és sikeres a pénzfelvét. Ha a hitelkeret sem elegendő, akkor sikertelen a pénzfelvét.
- toString: egy sztringben összefűzve adja vissza a kártya adatait
- Hitelkeret lekérdezése, hitelkeret beállítása a paraméterben megadott értékre

Definiáljon **CardTest** futtatható osztályt új csomagban.
1. Külön metódusban ellenőrzött módon olvasson be egy számot (1 és 10 közötti érték). Most legyen 4.
2. Hozzon létre a beolvasott értéknek megfelelő méretű bankkártya tömböt. Töltse fel az alábbiadatokkal:

```
Owner=Kiss Tamás, Valid=2020-04-30, Bank=OTP, Balance=150000, Credit=100000
Owner=Nagy Levente, Valid=2022-05-31, Bank=ERSTE, Balance=100000
Owner=Szabó László, Valid=2019-03-31, Bank=OTP, Balance=200000, Credit=100000
Owner=Kovács Edit, Valid=2021-01-31, Bank=CIB, Balance=250000
```

3. Minden kártyával végezzen egy pénzfelvételi műveletet és írja ki, hogy sikerült-e a művelet,valamint az aktuális egyenleget és a hitelkeretet (ha hitelkártyáról van szó).

```
1. Pénzfelvét: 280000 Sikertelen Új egyenleg: 150000 Hitelkeret: 100000
2. Pénzfelvét: 80000 Sikeres Új egyenleg: 20000
3. Pénzfelvét: 50000 Sikertelen Új egyenleg: 200000 Hitelkeret: 100000
4. Pénzfelvét: 100000 Sikeres Új egyenleg: 150000
```

4. Írjon külön metódust, amelyik megszámolja hány CIB bankos kártya van a tömbben. (1)
5. Rendezze a tömböt a kártyák érvényességi dátuma szerint és listázza ki a rendezett tömb adatait. A tömb elemeinek kiírására írjon külön metódust.

4. feladat: emelt szintű feladatkiegészítés, szintidő: 20 perc. A 3. feladatot egészítse ki egy Chargeable (díj vethető ki rá) interfésszel, amelynek egy metódusa van, a díj levonását végző metódus. Ennek paramétere a kártyahasználat díja (egész szám), amit megpróbál levonni a számláról. Ha van fedezet, sikerül a díj levonás akkor igazzal tér vissza, ha nem akkor hamissal.

a) A Card osztály valósítsa meg az interfészt és legyen egy új adattagja, a kártya díja (egész szám). A kártya díját a konstruktorban állítsa be, ami bankonként legyen eltérő. A CardTest osztályban próbálja meg a kártyahasználati díjat levonni a számlákról.

b) A CreditCard osztály definiálja felül a kártyadíj felszámítását végző metódust. Hitelkártya esetén legyen plusz díj (itt is lehet banktól függő érték). A CardTest osztályban próbálja meg a kártyahasználati díjat levonni a számlákról.

c) Írjon saját kivételosztályt arra az esetre, amikor nincs elég fedezet a számlán a tranzakció elvégzéséhez. A pénzfelvét és a kártyadíj levonását végző metódusok dobják meg ezt a kivételt, ha szükséges.