10. gyakorlat anyaga

1. feladat

Készítsünk a `BookSuite` alapján könyveket.

Az osztály mindkét konstruktora az `initBook` metódust hívja meg. A paraméter nélküli konstruktor a `default...` mezők szerint tölti fel az adattagokat, a másik előbb megvizsgálja a kapott adatokat a `checkInitData` metódus segítségével. A vizsgálat `IllegalArgumentException` kivételt vált ki, ha az `author` szöveg rövidebb két betűnél, a `title` rövidebb négy betűnél, vagy ha `pageCount` nem pozitív szám.

A könyv ára az oldalszámmal megegyező összeg.

Teszteléshez használjuk az alábbi három könyvet.

```
John Steinbeck, Of Mice and Men, 107
Kozsik Tamás, Java programozás, 234
Alan Alexander Milne, Winnie-the-Pooh, 145
```

A könyvek elvárt szöveges alakja az alábbi. A `decode()` metódus ilyen szövegekből alakít ki könyveket, tesztelendő, hogy a visszakapott könyvek egyes adatai megegyeznek-e az eredeti példányéival.

- Tipp: szövegek elejéről és végéről a felesleges szóközök a `strip()` metódussal távolíthatók el.

```
John Steinbeck: Of Mice and Men; 107
Kozsik Tamás: Java programozás; 234
Alan Alexander Milne: Winnie-the-Pooh; 145
```

A `getShortInfo()` rövidített alakban írja le a könyvet. A szerzők neveit iniciálékkal rövidíti, ezt a `getAuthorWithInitials()` valósítsa meg egy `StringBuilder` használatával.

```
J. Steinbeck: Of M; 107
K. Tamás: Java; 234
A. A. Milne: Winn; 145
```

Tesztelendő továbbá, hogy olyan könyveket, amelyek nem mennek át a `checkInitData` vizsgálatain, nem tudunk létrehozni, és `InvalidBookException` kivételt váltanak ki. Tesztelendő, hogy ezek a kivételek tartalmazzák a könyv leendő szerzőjét és címét.

Gyermekosztály: `EBook`

Az `EBook` osztály konstruktora egy `Book` példány mellett a három új mezőnek megfelelő adatokat kap meg, ez utóbbiakat eltárolja.

A könyv ára az oldalszámnál `pdfSize` értékkel több.

Ha a fenti könyvek `pdfSize` értéke rendre 5, 10000 és 8, akkor a szöveges reprezentációik az alábbiak legyenek. Itt a vásárlók szemét megtévesztő évtizedes gyakorlat szerint a valós árnál `0.01` egységgel kevesebb van feltüntetve, ha az ár `1000` egység vagy kevesebb.

```
EBook(John Steinbeck: Of Mice and Men; 107, $111.99)
EBook(Kozsik Tamás: Java programozás; 234, 10233 Ft)
EBook(Alan Alexander Milne: Winnie-the-Pooh; 145, £152.99)
```

Ezeket dekódolni nehéz lenne, a `decode()` metódus váltson ki `UnsupportedOperationException` kivételt.

Gyermekosztály: `PrintedBook`

Egy `PrintedBook` a `Book` szokásos adatai mellett kaphasson meg egy `CoverType` értéket, amit tároljon el az adattagba. Egy `PrintedBook` a szokásos könyveknél hat oldallal többel rendelkezik.

Egy `PrintedBook` példányt egy `Book` példány adatai segítségével is elő lehessen állítani. Ez legyen mindig keménykötésű. Ez a konstruktor hívjon át az előzőre.

A könyv ára az oldalszám többszöröse legyen: puhafedelű könyvek esetén kétszerese, keménykötésűek esetén háromszorosa.

Amennyiben a könyveket a második konstruktorral állítjuk elő, a szöveges reprezentációjuk az alábbi alakú legyen.

```
John Steinbeck: Of Mice and Men; 113 (HARDCOVER)
Kozsik Tamás: Java programozás; 240 (HARDCOVER)
Alan Alexander Milne: Winnie-the-Pooh; 151 (HARDCOVER)
```

Ezt könnyű dekódolni, a `decode()` metódus itt legyen megvalósítva.

- Tipp: mivel a zárójelek a `String` osztály `split()` metódusában speciális jelentésűek, így használandó: `txt.split("[(]")`

Könyvgyűjtemény

A `BookCollection` tároljon el sok könyvet egy listában, és ezeket lehessen elmenteni és betölteni az alábbi formátumban.

```
Book--John Steinbeck: Of Mice and Men; 107
Book--abcd: ABCD; 123
EBook--EBook(abcd: ABCD; 123, 356.99$)
PrintedBook--abcd: ABCD; 129 (HARDCOVER)
```

Ha `EBook` vagy érvénytelen adatú könyv érkezne, a helyére egy `DamagedBook` példány kerüljön.

A mentést és töltést próbáljuk ki üres listával, egyetlen (alapértelmezett adatokkal elkészített) könyves listával, és egy mindenfélét (`EBook` és `PrintedBook` könyvet egyaránt tartalmazó) listával.

2. feladat

A `MultiDimensionalPointSuite` alapján készítsünk többdimenziós pont típust. Ennek a konstruktora a kezdeti koordinátákat veszi át. A `get` és `set` metódusokkal egyes koordinátákat lehet beállítani és lekérdezni.

Az osztály két interfészt valósít meg.

- `Scalable`: minden koordinátát a megkapott mértékben növel meg
- `Undoable`: itt összetettebb a teendő
  - Minden olyan metódus működését vissza akarjuk tudni vonni, ami megváltoztatja a pont állapotát.
  - Ezt elősegítendő, a `lastCoordinates` adattag mindig a pont legutóbbi állapotát tartalmazza.
    - A pont elkészítése után közvetlenül a kezdeti koordinátákkal legyen feltöltve.
  - Az interfész metódusa az utolsó állapotot állítja vissza.
    - Ha közvetlenül ezután újra meghívják a műveletet, az éppen lecserélt állapot álljon vissza.

Az alábbi három teszteset mindegyikében a következő lépéseket kell megtenni.

1. Pont elkészítése tetszőleges adatokkal.
2. A megteendő lépés elvégzése, lásd lentebb.
3. Megvizsgálandó mindegyik koordináta elvárt értéke.
4. Visszalépés meghívása.

5. Megvizsgálandó, hogy mindegyik koordináta értéke visszaállt az elkészítés utáni állapotra.
6. Visszalépés meghívása ismét.
7. Megvizsgálandó mindegyik koordináta elvárt értéke.

A következők tesztelendők.

- `testInitUndo`: a pontot csak elkészítjük
- `testSet`: a pont elkészítése után beállítjuk egy koordinátáját
- `testScale`: a pont elkészítése után felskálázzuk

Két pont akkor minősüljön tartalmilag egyenlőnek, ha a koordinátáik megegyeznek. A korábbi állapotuk nem számít.

A `testEquality` metódus tesztelje, hogy két olyan lista egyenlő, amelyben egymással egyenlő pontok szerepelnek.

- Mindkét listához használható a `List.of` művelet.
- A pontok különbözőek legyenek a két listában.

Legyen természetes rendezés is adott a pontokon az alábbiak szerint. Ha valamelyiknek nagyobb a dimenziószáma, az minősül nagyobbnak. Különben az első eltérő koordinátaérték dönt, ha pedig mindegyik koordináta megegyezik, akkor egyenlőek.

A `testOrdering` metódusban kerüljön sok különféle pont egy listába. Erre aztán hívjuk meg a Collections.shuffle, majd a Collections.sort műveletet. Ezután teszteljük, hogy a kijött lista elemei az elvárt, növekvő sorrendbe kerültek.

- Mivel a lista tartalma módosul, az elkészítéséhez *nem* használható a `List.of` művelet.
  - Az elvárt érték elkészítéséhez viszont igen.
- Itt nem szigorúan egységtesztelést végzünk, mert az egyes lefutások során különbözően keverjük össze a lista elemeit.

3. feladat

A `GoAroundStructureTest` szerint készítsünk olyan adatszerkezetet, ami egypár elemet tárol. A bejárója (`GoAroundIteratorStructureTest`) működjön úgy, hogy a tárolt elemeket `roundCount` alkalommal adja elő.

Teszteljük, hogy ha az eltárolt elemek `1, 2, 3, 4` és `roundCount` értéke `2`, akkor a bejáró kimenete `1, 2, 3, 4, 1, 2, 3, 4`. Mindegyik lépés megtétele előtt próbáljuk ki, hogy `hasNext()` értéke `true`, de az utolsó lépés megtétele után már `false`.