Haladó Java beadandó feladat 2022-23/2

1. Feladat — 10 pont

`Either<L, R>`

Készíts el a fenti néven összeg típust, amely két tetszőleges típus közül pontosan az egyik fajtából képes objektumot tárolni. Publikus konstruktor helyett osztályszintű metódusokkal lehet előállítani.

Koncepcionálisan az `R` típusú (Right) érteket tekintjük *jó* értéknek, az `L` (Left) típusút pedig *hibás* értéknek. Ennek megfelelően egyes műveleteket csak az `R` típusú értéken hajtunk végre. Ennél a típusnál közvetlenül nem módosíthatók az objektumok.

Osztályszintű műveletei:

- `left`: egy `L` típusú értékből előállít egy `Either<L, R>` objektumot.
- `right`: hasonlóan a `left`-hez, csak `R` típusú értékből.
- `iterate`: egy olyan `Either` példányt vár, aminek mindkét értéke egyforma típusú, továbbá egy `n` egész számot és egy `R -> R` névtelen függvényt is, és egy `R` objektummal tér vissza. Ha bal érték van tárolva, azt adja vissza. Különben a tárolt jobb értékre `n`-szer alkalmazza a leképezést, és ezt az értéket adja vissza.

Példányszintű műveletei:

- `swap`: olyan `Either`-t ad ki, amelyben fel van cserélve a `left` és a `right`.
- `isLeft`, `isRight`: jobb vagy bal értéket tárol az objektum?
- `getRight`: visszaadja a jobb értéket, kivételt (`NoSuchElementException`) vált ki, ha nem jobb értéket tarolt.
- `getLeft`: visszaadja a bal értéket, kivételt (`NoSuchElementException`) vált ki, ha nem bal értéket tarolt.
- `orElseGet`: egy lustán kiértékelhető, `R` értéket visszaadó névtelen függvényt (`other`) kap, `R` értékkel tér vissza. Ha jobb értéket tárolunk, adjuk azt vissza, egyébként az `other` által kiadott értéket.
- `map`: egy `R -> T` névtelen függvényt vár, és egy `Either<L, T>` objektummal tér vissza. Ha jobb érték volt tárolva, alkalmazzuk a leképezést és adjuk vissza egy új `Either`-ben az eredményt, egyébként változatlanul adjuk vissza a bal értéket egy új `Either` objektumba csomagolva.
- `bind`: egy `R -> Either<L, T>` névtelen függvényt kap és egy `Either<L, T>` objektummal tér vissza. Jobb érték esetén alkalmazzuk a műveletet és annak eredményével térjünk vissza, egyébként a bal érték újracsomagoltjával.

Írj `EitherTest` egységtesztelőt az osztályhoz, ami leteszteli mindegyik

függvényét az osztálynak. Legalább az egyik teszteset legyen paraméterezett.

2. Feladat — 10 pont

Készíts egy `ClassDumper` osztályt, melynek legyen egy `dump` függvénye. Ez a függvény paraméterül kap egy `Class`-t, és visszatér az osztály forráskódját jól közelítő szöveggel.

A szöveg előállításánál az adattagokat, a metódusokat, a módosítókat figyelembe kell venni, a konstruktorokat, metódustörzseket, kezdeti értékadásokat nem.

Teljes pont akkor jár a feladatra, ha a `dump` metódus törzse úgy néz ki, hogy az eleje néhány `Stream`/`Optional` értelmes felhasználásával kialakítja a kimenet megfelelő részeit, és az utolsó lépés egy lényegében triviális konkatenáció.

Írj `ClassDumperTest` egységtesztelőt az osztályhoz, ami legalább az alábbi három esetet próbálja ki. (Természetesen a kódnak értelmesen működnie kell mindenféle osztályra.)

Ha az eredeti osztály kódja az alábbi:

```
import java.io.Serializable;

public class TestClass1 extends Thread implements Runnable, Serializable {
    private static final int CONST = 1;
    volatile transient boolean flag;
    protected Object obj;
    public int n = 2;

    public static void printConst() {
        System.out.println(CONST);
    }
}
```

... akkor a `dump` függvény eredménye karakterre pontosan a következő kell, hogy legyen:

```
public class TestClass1 extends java.lang.Thread implements java.lang.Runnable, jav
a.io.Serializable {
    private static final int CONST;
    transient volatile boolean flag;
    protected java.lang.Object obj;
    public int n;

    public static void printConst(...) { /* method body */ }
}
```

Ha az interfészeket levesszük az osztályról, a kimenet:

```
public class TestClass2 extends java.lang.Thread {
    private static final int CONST;
    transient volatile boolean flag;
    protected java.lang.Object obj;
    public int n;
```

```
    public static void printConst(...) { /* method body */ }
}
```

Ha az ősosztályt és a nyilvános minősítőt is levesszük az osztályról:

```
 class TestClass3 extends java.lang.Object {
    private static final int CONST;
    transient volatile boolean flag;
    protected java.lang.Object obj;
    public int n;

    public static void printConst(...) { /* method body */ }
}
```