Domače naloge za 5. vaje

Razreda `Posta` in `Pismo`

V skladu s sledečimi navodili napišite razreda `Posta` in `Pismo` ter ju preizkusite v posebnem testnem razredu.

Razred `Posta` definirajte tako, da bo vsak njegov objekt predstavljal neko pošto s poštno številko (npr. 1000) in nazivom (npr. Ljubljana). Poleg atributov za predstavitev poštne številke in naziva pošte naj razred `Posta` vsebuje še konstruktor za hkratno nastavitev obeh atributov, metodi vrste "getter" za oba atributa in metodo `toString()`, ki vrne predstavitev pošte v obliki niza (npr. kot niz `"1000 Ljubljana"`).

Razred `Pismo` definirajte tako, da bo vsak njegov objekt predstavljal neko pismo s sledečimi podatki:

- izvorna pošta (npr. 1000 Ljubljana);
- ciljna pošta (npr. 2000 Maribor);
- podatek o tem, ali je pismo priporočeno ali navadno;
- celoštevilska razdalja (v kilometrih) med izvorno in ciljno pošto.

Poleg atributov za predstavitev posameznih podatkov (premislite o njihovih tipih!) naj razred `Pismo` vsebuje še konstruktor za hkratno nastavitev vseh štirih atributov in sledeče metode:

- Metode vrste "getter" za vse štiri atribute.
- Metodo `toString()`, ki vrne niz oblike

  vrsta_pisma `za` *izvorna_pošta* `->` *ciljna_pošta* `(`*razdalja* `km)`

  pri čemer je *vrsta_pisma* bodisi `priporočeno` ali pa `navadno`. Na primer:

  `priporočeno za 1000 Ljubljana -> 2000 Maribor (130 km)`
- Metodo `izviraOd(Posta posta)`, ki vrne `true` natanko v primeru, če je pošta, ki je podana kot parameter, izvorna pošta za dano pismo (torej za pismo `this`).
- Metodo `staIzvorInCiljIsta()`, ki vrne `true` natanko v primeru, če sta za dano pismo izvorna in ciljna pošta isti (npr. če je pismo poslano s pošte 1000 Ljubljana na pošto 1000 Ljubljana).
- Metodo `imaIstiCiljKot(Pismo pismo)`, če ima dano pismo (`this`) isto ciljno pošto kot pismo, ki je podano kot parameter metode.
- Statično (!) metodo `imataIstiCilj(Pismo p1, Pismo p2)`, ki vrne `true` natanko v primeru, če imata obe pismi isto ciljno pošto.
- Metodo `cena()`, ki vrne ceno danega pisma. Za navadno pismo se cena izračuna glede na razdaljo: za razdaljo od 0 do vključno $(r-1)$ km cena znaša k EUR, za razdaljo od r km do vključno $(2r-1)$ km je cena enaka $2k$ EUR, za razdaljo od $2r$ km do vključno $(3r-1)$ km je cena enaka $3k$ EUR itd. Ceno priporočenega pisma izračunamo tako, da ceni navadnega pisma prištejemo priporočnino p EUR, ki je neodvisna od razdalje. Števila k, r in p definirajte kot statične globalne konstante, seveda pa jim namesto k, r in p dodelite kakšna bolj smiselna (in daljša!) imena.

- Metodo `jeDrazjeOd(Pismo pismo)`, ki vrne `true` natanko v primeru, če je cena danega pisma (`this`) večja od cene pisma, podanega kot parameter metode.
- Statično (!) metodo `vrniDrazje(Pismo p1, Pismo p2)`, ki vrne tisto pismo izmed pisem `p1` in `p2`, ki ima večjo ceno. (Metoda vrača vrednost tipa `Pismo`.)
- Metodo `izdelajPovratno()`, ki ustvari (`new`) in vrne (`return`) novo pismo, ki ima enake atribute kot pismo `this`, le izvorna in ciljna pošta sta zamenjani. Na primer, če (v testnem razredu) spremenljivka `lj2mb` kaže na priporočeno pismo s pošte 1000 Ljubljana na 130 km oddaljeno pošto 2000 Maribor, potem naj stavki

  ```
  System.out.println(lj2mb.toString());
  Pismo mb2lj = lj2mb.izdelajPovratno();
  System.out.println(mb2lj.toString());
  System.out.println(mb2lj.izdelajPovratno().toString());
  ```

  izpišejo

  ```
  priporočeno za 1000 Ljubljana -> 2000 Maribor (130 km)
  priporočeno za 2000 Maribor -> 1000 Maribor (130 km)
  priporočeno za 1000 Maribor -> 2000 Maribor (130 km)
  ```

Opomba: Nize lahko sestavljate z operatorjema `+` in `+=` ali pa s pomočjo metode `String.format`, ki deluje na enak način kot metoda `System.out.printf`, le da niz vrne, namesto da bi ga izpisala. Na primer, stavki

```
double pi = Math.PI;
String s = String.format("pi = %.2f (približno)", pi);
System.out.println(s);
```

izpišejo

```
pi = 3.14 (približno)
```

Razred `Ulomek`

Napišite razred `Ulomek` tako, da bodo njegovi objekti predstavljali posamezne okrajšane ulomke. Okrajšani ulomek je sestavljen iz števca (poljubno celo število) in imenovalca (poljubno *pozitivno* celo število). Ulomek a/b okrajšamo tako, da števec in imenovalec delimo z $\gcd(|a|, |b|)$, nato pa upoštevamo še relaciji $a/-b = -a/b$ in $-a/-b = a/b$ (da imenovalec postane pozitivno število). Poleg atributov za predstavitev števca in imenovalca naj razred `Ulomek` vsebuje še sledeče:

- Konstruktor, ki sprejme števec in imenovalec. Lahko predpostavite, da je podani imenovalec različen od 0, ne smete pa predpostaviti, da je imenovalec pozitiven ali pa da je ulomek s/i, kjer s predstavlja podani števec, i pa podani imenovalec, že okrajšan.
- Metodo `toString()`, ki vrne niz oblike *števec*`/`*imenovalec*. Na primer, stavka

  ```
  Ulomek u = new Ulomek(6, -4);
  System.out.println(u.toString());
  ```

  izpišeta besedilo `-3/2`.

- Metodo `negacija()`, ki vrne nasprotno vrednost danega ulomka (torej ulomka `this`) kot nov (!) objekt razreda `Ulomek`. (Metoda ne sme spremeniti ulomka `this`!) Nasprotna vrednost ulomka a/b je ulomek $-a/b$.
- Metodo `obrat()`, ki vrne obratno vrednost ulomka `this` kot nov objekt razreda `Ulomek`. Obratna vrednost ulomka a/b je ulomek b/a.
- Metodo `vsota(Ulomek u)`, ki vrne vsoto ulomka `this` in ulomka `u` kot nov objekt tipa `Ulomek`. Na primer, stavki

  ```
  Ulomek u = new Ulomek(6, -4);
  Ulomek v = new Ulomek(2, 5);
  Ulomek w = u.vsota(v);
  System.out.println(w.toString());
  ```

  izpišejo besedilo `-11/10`.
- Metode `razlika`, `zmnozek` in `kolicnik`, ki so definirane na enak način kot metoda `vsota`. Premislite, kako bi pri računanju razlike in količnika učinkovito uporabili že napisane metode.
- Metodo `potenca(int eksponent)`, ki vrne potenco ulomka `this` na podani eksponent. Eksponent je lahko tudi negativen.
- Metodo `jeManjsiOd(Ulomek u)`, ki vrne `true` natanko v primeru, če je ulomek `this` manjši od ulomka `u`.

V testnem razredu izračunajte in izpišite vrednosti izrazov

$$p = \left(-\frac{2}{3}\right)^{-3} + \frac{6}{4} \cdot \left(\frac{1}{3}\right)^{2} + \frac{3}{1}$$

in

$$q = -\left(\frac{1}{2} - \frac{1}{4}\right) : \left(\frac{1}{2} + \frac{1}{4}\right)^{-2}$$

ter nato preverite, ali velja $a < b$. (Rešitve: $a = -5/24$, $b = -9/64$, $a < b$.)

Opomba: Nobena od metod v razredu `Ulomek` ne spremeni stanja objekta. Z drugimi besedami: ko objekt tipa `Ulomek` ustvarimo, ga ne moremo več spremeniti. Objekti tipa `Ulomek` (pa tudi objekti razredov `Posta`, `Pismo` in `Datum`, ki jih obravnavamo v drugih domačih nalogah) so torej podobno kot objekti tipa `String` ali `Integer` *nespremenljivi* (angl. immutable). Nespremenljivi objekti imajo vrsto prednosti (`http://www.javapractices.com/topic/TopicAction.do?Id=29`) in lahko programerja odrešijo marsikatere zahrbtnosti. Na primer, sledeči potencialno zahrbtni pojavi so možni zgolj pri spremenljivih objektih, saj nespremenljivi objekti po definiciji sploh ne morejo imeti metod vrste "setter":

```
Delavec d = new Delavec(12345, "Gorišek", "Jože", 150);
Delavec d1 = d;
d.nastaviStUr(200);
d.izpisi();     // Jože Gorišek (12345), 200 ur
d1.izpisi();    // Jože Gorišek (12345), 200 ur (past!)
```

Razred `Datum`

Napišite razred `Datum` tako, da bodo njegovi objekti predstavljali veljavne datume po gregorijanskem koledarju. Vsak datum je predstavljen s tremi celoštevilskimi atributi: dan (1–31), mesec (1–12) in leto (od 1583 naprej). (Gregorijanski koledar je stopil v veljavo na sredini leta 1582, leta 1583 pa je veljal že od 1. januarja naprej.) Poleg navedenih atributov naj razred `Datum` vsebuje še sledeče:

- Privatni (!) konstruktor, ki nastavi vrednosti vseh treh atributov. Konstruktor naj ne preverja veljavnosti podanih parametrov.
- Statično (!) metodo

  `public static Datum ustvari(int dan, int mesec, int leto)`.

  Če podani parametri predstavljajo veljaven datum, naj metoda ustvari in vrne nov objekt razreda `Datum`, sicer pa naj vrne `null`. Pri preverjanju veljavnosti datuma upoštevajte pravilo za prestopna leta, ki se glasi takole: leto je prestopno, če je deljivo s 400 ali pa če je deljivo s 4, vendar ni deljivo s 100. Leta 1700, 1800, 1900, 2100, 2200, 2300, 2500, ... tako niso prestopna, leta 1600, 2000, 2400, ... pa so.

  V konstruktorju bi sicer tudi lahko preverjali veljavnost podanih komponent datuma, vendar pa ne bi mogli vrniti vrednosti `null` v primeru neveljavnega datuma, saj konstruktor vedno ustvari nov objekt in vrne referenco nanj. Če pa objekte tipa `Datum` izdelujemo v posebni metodi, se v njej lahko svobodno odločimo, ali bomo objekt dejansko ustvarili ali ne. Če napišemo takšno metodo, moramo dostop do konstruktorja onemogočiti, saj bi sicer lahko uporabnik razreda z neposrednim klicem konstruktorja ustvaril tudi neveljaven datum. Opisana tehnika — privaten konstruktor v kombinaciji z javno dostopno statično metodo za nadzorovano ustvarjanje objektov — se v objektnem programiranju kar pogosto uporablja. Tovrstna statična metoda se imenuje *tovarna* (angl. factory method).
- Metodo `toString()`, ki vrne predstavitev datuma v obliki niza `DD.MM.LLLL`, npr. `15.07.2012`.
- Metodo `jeEnakKot(Datum datum)`, ki vrne `true` natanko v primeru, če objekt `this` predstavlja isti datum kot objekt `datum`.
- Metodo `jePred(Datum datum)`, ki vrne `true` natanko v primeru, če je datum, ki ga predstavlja objekt `this`, kronološko pred datumom, ki ga predstavlja objekt `datum`. Na primer, datum 30.09.2011 je pred datumom 27.10.2011, ta pa je pred datumom 20.01.2012.
- Metodo `naslednik()`, ki ustvari in vrne nov (!) objekt razreda `Datum`, ki predstavlja neposrednega naslednika datuma `this`. Na primer, naslednik datuma 28.02.2012 je datum 29.02.2012, njegov naslednik pa je datum 01.03.2012. Naslednik datuma 28.02.2013 je datum 01.03.2013.
- Metodo `predhodnik()`, ki ustvari in vrne nov objekt razreda `Datum`, ki predstavlja neposrednega predhodnika datuma `this`. Za vsak objekt `d` tipa `Datum` velja relacija `d.naslednik().predhodnik().jeEnakKot(d)`. Če datum nima veljavnega predhodnika (edini tak datum je 01.01.1583), naj metoda vrne `null`.
- Metodo `cez(int stDni)`, ki izračuna (in vrne kot nov objekt tipa `Datum`) datum, ki je za `stDni` oddaljen od datuma `this`. Parameter `stDni` je lahko tudi negativen; v

tem primeru metoda izračuna datum, ki je `-stDni` pred datumom `this`. Če je ciljni datum pade pred datum 01.01.1583, naj metoda vrne `null`.

- Metodo `razlika(Datum datum)`, ki izračuna razliko (v številu dni) med datumoma `this` in `datum`. Če je datum `this` pred datumom `datum`, je razlika seveda negativna.

Poleg navedenih metod lahko seveda dodate še poljubno mnogo pomožnih privatnih metod. Pri vsaki metodi premislite, ali si lahko morda pomagate s katero od že napisanih metod.

Razred `Datum` preizkusite v posebnem testnem razredu.

Dopolnitve razreda `Oseba`

Razred `Oseba`, ki smo ga napisali na vajah, dopolnite tako, kot zahtevajo naloge, ki jih bomo predstavili v nadaljevanju. Zadošča, če rešite 3 naloge od šestih. V nasprotju z nadgradnjami seminarskih nalog lahko vse dopolnitve realizirate kar v isti datoteki. Dopolnitve preizkusite v testnem razredu.

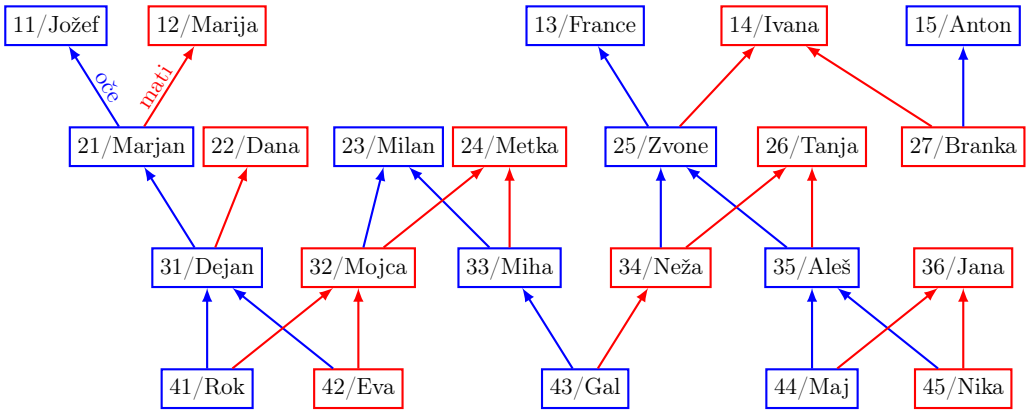
Naloge so sledeče:

- Atribut `letoRojstva` nadomestite z atributom `datumRojstva` tipa `Datum` (gl. domačo nalogo `Datum`), dodajte metodo `starostVDneh(Datum datum)`, ki vrne trenutno starost osebe `this` na datum `datum` in po potrebi posodobite še druge metode, povezane s starostjo osebe.
- Dodajte atribut `partner`, ki predstavlja moža oz. ženo osebe `this`, in dopolnite razred tako, da bo mogoče ta atribut nastavljati in pridobivati njegovo vrednost. Po vzoru ostalih metod `je`*`Nekaj`*`Od` napišite še metodi `jeTascaOd(Oseba os)` in `jeSvakOd(Oseba os)`. Tašča je moževa oz. ženina mati, svak pa je bodisi mož oz. ženin brat ali pa sestrin mož.
- Napišite metodo `ocetovskaGeneracijskaRazlika(Oseba os)`, ki deluje po sledečih pravilih:
  - Če sta osebi `this` in `os` identični, je generacijska razlika enaka 0.
  - Če je oseba `this` očetovski prednik osebe `os`, je generacijska razlika število očetov med obema osebama. (Če je oseba `this` oče osebe `os`, je generacijska razlika enaka 1, če je `this` stari oče osebe `os`, je razlika enaka 2 itd.)
  - Če je oseba `os` očetovski prednik osebe `this`, je generacijska razlika negativno število očetov med obema osebama.
  - Če ni izpolnjen nobeden od zgornjih pogojev, naj metoda namesto vračila vrednosti (torej namesto stavka `return`) *vrže izjemo*:

    ```
    throw new IllegalArgumentException(
            "Osebi nista sorodnika po očetovski liniji.");
    ```

    V testnem razredu boste lahko izjemo *ujeli*:

    ```
    try {
        // eden ali več klicev metode ocetovskaGeneracijskaRazlika
    } catch (IllegalArgumentException ex) {
        System.out.println("Izjema: " + ex.getMessage());
    }
    ```



Slika 1: *Starševski odnosi med osebami v testnem razredu.*

- (⋆) Napišite metodo `jePrednikOd(Oseba os)`, ki vrne `true` natanko v primeru, če je oseba `this` prednik osebe `os`. Oseba A je prednik osebe B, če je izpolnjen eden od sledečih pogojev:
  - $A = B$ (denimo, da je vsaka oseba sam svoj prednik).
  - A je prednik B-jevega očeta.
  - A je prednik B-jeve matere.
- (⋆) Napišite metodo `nastejPrednike()`, ki izpiše vse prednike osebe `this`. Za vsakega prednika naj metoda izpiše tudi starševsko verigo, ki vodi od osebe `this` do trenutnega prednika. Za testni primer, ki smo ga obravnali na vajah (slika 1) bi klic `os43.nastejPrednike()` izpisal sledeče:

  ```
  oseba: Gal Smole [M] (2009)
  oseba.oce: Miha Smole [M] (1978)
  oseba.oce.oce: Milan Smole [M] (1953)
  oseba.oce.mati: Metka Smole [Z] (1953)
  oseba.mati: Neža Smole [Z] (1980)
  oseba.mati.oce: Zvone Kotnik [M] (1956)
  oseba.mati.oce.oce: France Kotnik [M] (1932)
  oseba.mati.oce.mati: Ivana Kotnik [Z] (1931)
  oseba.mati.mati: Tanja Kotnik [Z] (1954)
  ```

  Vrstni red izpisa ni pomemben.

  Namig: Napišite in uporabite privatno pomožno metodo `nastejPrednike(String veriga)`.
- (⋆) Napišite metodo `jeSorodnikOd(Oseba os)`, ki vrne `true` natanko v primeru, če sta osebi `this` in `os` sorodnika. Osebi A in B sta sorodnika, če obstaja oseba C, ki je prednik tako osebe A kot osebe B. (Tudi sorodstvo je možno definirati na eleganten rekurzivni način. Odkrijte in realizirajte ga sami!)

  V primeru z vaj (slika 1) sta osebi `os41` in `os43` (Rok in Gal) sorodnika. Sorodnika sta tudi osebi `os24` in `os42` (Metka in Eva), osebi `os33` in `os34` (Miha in Neža) pa nista.