Objektno orijentirano programiranje: međuispit (a.g. 2019./2020.)

Zadatak 1 (2+2+1 bod).

Razmatramo modeliranje programa koji radi s više vrsta osoba. Svaka osoba ima ime. Student je osoba koja studira na fakultetu. Zaposlenik je osoba koja radi u tvrtki. Direktor je zaposlenik koji dobiva i bonus. Nije moguće izravno stvoriti primjerak razreda `Osoba`.

a) Napišite programski kod svih navedenih razreda. Jednom stvorene primjerke tih razreda ne smije biti moguće više mijenjati. Nacrtajte i prikladni dijagram razreda.

b) Dan je demonstracijski program u nastavku. U trenutku kada izvođenje programa dođe do naredbe koja ispisuje X, prikažite kakvo je stanje u memoriji (što se nalazi na stogu, što na gomili)

```
Osoba[] osobe = new Osoba[] { new Student("Janko", "FER"), new Zaposlenik("Kristina", "ACME"),
                new Direktor("Ana", "ACME", 10000.0)};

int[] duljine = new int[osobe.length];
for(int i = 0; i < osobe.length; i++) {
        duljine[i] = osobe[i].getIme().length();
}

for(Osoba o : osobe) {
        System.out.println(o.getIme());
        if(o instanceof Zaposlenik) {
                System.out.println(" Zaposlenje u tvrtki "+((Zaposlenik)o).getTvrtka());
        }
}

Osoba janko = new Student("Janko", "FER");
for(Osoba o : osobe) {
        System.out.println(o==janko);
}

System.out.println("X");

System.out.println(osobe[0].getIme()+" "+duljine[0]+" "+janko.getIme());
```

c) Što će ispisati demonstracijski program?

Zadatak 2 (1+4 boda).

Definirano je parametrizirano sučelje `Queue` s parametrom `E` koje opisuje kolekciju koja se ponaša kao red (FIFO), te propisuje sljedeće metode:

```
boolean isEmpty();
int size();
E take(); // Može baciti java.util.NoSuchElementException
Queue add(E elem);
```

a) Napišite programski kod tog sučelja. Razredi koji implementiraju ovo sučelje moraju se moći prevesti i raditi korektno, čak i ako ne ponude svoju implementaciju metode `isEmpty`, i mora se moći stvarati primjerke tih razreda.

b) Napišite programski kod razreda `LinkedListQueue` koji implementira ovo sučelje. Razred interno elemente pamti u jednostruko povezanoj ulančanoj listi te pamti reference na prvi i zadnji čvor (sami to morate implementirati!); vrijednosti skida s početka liste, a nove dodaje na kraj liste.

Primjer uporabe:

```
Queue<String> imena = new LinkedListQueue<>();
imena.add("Ivana").add("Janko").add("Ana").add("Branko");
while(!imena.isEmpty()) System.out.println(imena.take());
```

Pokretanjem ovog isječka ispisat će se:

```
Ivana
Janko
Ana
Branko
```

Zadatak 3 (2+2 boda).

Kvadratna jednadžba oblika $ax^2+bx+c=0$ modelirana je razredom `QuadraticEquation`. Razmotrite demonstracijski primjer dan u nastavku koji je smješten u razred `Demo` (prikazane su samo metode tog razreda). Ako je $a=0$, konstruktor treba baciti iznimku `IllegalArgumentException`.

```
public static void main(String[] args) {
	QuadraticEquation qe = new QuadraticEquation(2, 4, -30);

	// PAZI! VAŽNO!!! Kada bismo bilo koji od dva retka ispod ovog
	// komentara odkomentirali, kod se ne bi preveo!
	//qe.getUniqueSolution();
	//qe.getTwoSolutions();

	try {
		double x = qe.getUniqueSolution();
		System.out.println("x = " + x);
	} catch(QuadraticException e) {
		info(e);
	}

	try {
		Solutions sol = qe.getTwoSolutions();
		System.out.println("Manje rješenje je:" + sol.getSmaller());
		System.out.println("Veće rješenje je:" + sol.getBigger());
	} catch(QuadraticException e) {
		info(e);
	}

	try {
		qe.getUniqueSolution();
	} catch(NoUniqueSolutionException ex) {
		System.out.println("Ovo očekujemo!");
	} catch(NoRealSolutionsException ex) {
		System.out.println("X!");
	}

	try {
		qe.getTwoSolutions();
	} catch(NoMultipleSolutionsException ex) {
		System.out.println("Y!");
	} catch(NoRealSolutionsException ex) {
		System.out.println("Z!");
	}
}

private static void info(QuadraticException e) {
	System.out.println("Iznimka: " + e.getClass().getName());
	QuadraticEquation qe = e.getQuadraticEquation();
	System.out.printf("  a=%f, b=%f, c=%f%n",
				qe.getA(), qe.getB(), qe.getC());
}
```

Pokretanjem programa ispis će biti sljedeći.

```
Iznimka: iznimke.NoUniqueSolutionException
  a=2.000000, b=4.000000, c=-30.000000
Manje rješenje je:-5.0
Veće rješenje je:3.0
Ovo očekujemo!
```

Pažljivo proučite primjer i dani ispis. Napišite programski kod svih razreda potrebnih da bi se program mogao uspješno prevesti i pokrenuti, osim razreda `Demo` i `Solutions`. Pretpostavite da razred `Solutions` ima konstruktor kroz koji prima manji pa veći broj (tim redoslijedom).

Zadatak 4 (3 boda).

Razmatramo programski sustav za upise studenata u akademsku godinu diplomskog studija. Programski sustav studentima prikazuje popise kolegija. Svaki kolegij (razred `Kolegij`) ima pridruženu vrstu kolegija koja je modelirana enumom `VrstaKolegija`. Razmotrite sljedeći isječak koda.

```
Kolegij[] kolegiji = new Kolegij[] {
                new Kolegij("Strojno učenje", VrstaKolegija.TEORIJSKI),
                new Kolegij("Duboko učenje", VrstaKolegija.SPECIJALIZACIJA),
                new Kolegij("Bioinformatika", VrstaKolegija.IZBORNI)
};

for(Kolegij k : kolegiji) {
        VrstaKolegija vk = k.getVrstaKolegija();
        System.out.printf("Kolegij %s: tip=%d, opis: %s%n", k.getIme(), vk.getTip(), vk.getNaziv());
}
```

Pokretanjem, očekujemo sljedeći ispis:

```
Kolegij Strojno učenje: tip=1, opis: Teorijski predmet profila
Kolegij Duboko učenje: tip=2, opis: Predmet specijalizacije profila
Kolegij Bioinformatika: tip=3, opis: Izborni predmet profila
```

Pretpostavite da razred `Kolegij` već postoji. Napišite samo programski kod od `VrstaKolegija`.

Zadatak 5 (4 boda).

Razred `Potencije`, za pozitivnu bazu `B` i pozitivan broj `N`, predstavlja kolekciju od prvih `N` potencija te baze (demonstracijski primjer u nastavku stvara kolekciju koja predstavlja prvih 5 potencija od broja 2). U slučaju da `B` ili `N` nisu pozitivni, konstruktor treba baciti `IllegalArgumentException`. Napišite korektnu implementaciju razreda `Potencije` uz koju će se demonstracijski kod dan u nastavku uspješno pokrenuti:

```
Potencije od2 = new Potencije(2, 5);

System.out.println("Ispisujem ih:");
for(Integer p : od2) {
        System.out.println(p);
        if(p > 4) break;
}

System.out.println("Još jednom ih ispisujem:");
for(Integer p : od2) {
        System.out.println(p);
}
```

te čijim će se izvođenjem ispisati:

```
Ispisujem ih:
1
2
4
8
Još jednom ih ispisujem:
1
```

```
2
4
8
16
```

Nigdje niti u kojem trenutku ne smijete pamtiti više elemenata odjednom u memoriji (uporaba bilo kakvih polja ili javinih kolekcija nije dopuštena).

Zadatak 6 (4 boda).

Korisnik preko tipkovnice unosi niz imena (koristite razred `Scanner`), sve dok se ne unese “kraj”. Program najprije treba ispisati sva imena čija je duljina veća ili jednaka prosječnoj duljini, sortirano leksikografski (tj. “abecedno”); ako je neko ime bilo uneseno više puta, treba ga ispisati toliko puta. Program potom treba ispisati sva unesena imena, redoslijedom kojim su bila unesena, ali bez ispisivanja duplikata. Napišite traženi program i zadatak riješite uporabom standardnih javinih kolekcija (koje su barem `java.util.Collection`). Primjer pokretanja (nakošeno je korisnikov unos):

```
/home/javko> java -cp bin imena.Glavni
kraj
Niste dali niti jedno ime.

/home/javko> java -cp bin imena.Glavni
Ivana Janko Ana Branko Lahorija Ivo Ivana Jasen Kristina
Branko Anamarija Pero Kristina Žarko Ana Danko kraj
Ispis 1 (prosječna duljina: 6):
Anamarija
Branko
Branko
Kristina
Kristina
Lahorija
Ispis 2:
Ivana
Janko
Ana
Branko
Lahorija
Ivo
Jasen
Kristina
Anamarija
Pero
Žarko
Danko
```