

Objektno orijentirano programiranje

Završni ispit

27. siječnja 2014.

Ispit nosi ukupno 30 bodova i piše se 120 minuta.

1. zadatak (10 bodova)

a) Uporabom neke od Javinih standardnih implementacija liste napišite parametriziranu implementaciju razreda `RedStog` (parametar `T` određuje tip objekata koje razred sprema) koji predstavlja red neograničenog kapaciteta te ima sljedeće metode (opisano pseudokôdom):

- `imaElemenata()` : vraća `true` (ako ima elemenata) ili `false` (ako nema)
- `skiniIDohvatiSPocetka()` : skida i vraća prvi objekt
- `skiniIDohvatiSKraja()` : skida i vraća zadnji objekt
- `void dodajNaPocetak(objekt);`
- `void dodajNaKraj(objekt);`

Rješenje mora biti takvo da se operacije dodavanja i skidanja obavljaju garantirano u $O(1)$.

b) Definirajte razred `Student` koji ima privatne članske varijable `ime`, `prezime` te `JMBAG` i odgovarajuće *gettere*. Pretpostavite da u razredu `Ucitavac` postoji statička metoda `ucitaj()` koja iz predefiniране datoteke učitava studente i vraća ih kao `java.util.List<Student>`. U razred `Student` je potrebno još dodati sve što je potrebno da bi sljedeći isječak koda bio uspješno izveden:

```
List<Student> lista = Ucitavac.ucitaj();
Set<Student> studenti = new TreeSet<>(lista);
```

c) Prikazana je metoda `main` u kojoj nedostaju dijelovi koda:

```
public static void main(String[] args) {
    List<Student> lista = Ucitavac.ucitaj();
    List<Integer> duljine = lista.stream()
        .filter( ... )
        .map( ... )
        .collect(Collectors.toList());
    int[] statistika = new int[] {-1, -1, -1};
    duljine.forEach( ... );
    System.out.printf("Suma duljina: %d, min duljina: %d, max duljina: %d%f",
        statistika[0], statistika[1], statistika[2]
    );
}
```

Napišite lambda izraze koji se traže kao argumenti metoda `filter`, `map` i `forEach` kako bi se iz liste studenata dobila kolekcija samo onih studenata kod kojih je suma duljina imena i prezimena veća od 10 znakova, nakon čega se ta kolekcija pretvara u listu duljina prezimena studenata koji su zadovoljili prethodni uvjet. Program potom skuplja jednostavnu statistiku i ispisuje ju. Metoda `filter()` prima referencu na objekt tipa `Predicate`, `map()` na `Function`, a `forEach()` na `Consumer`.

```
public interface Predicate<T> {
    boolean test(T t);
}
public interface Function<T, R> {
    R apply(T t);
}
public interface Consumer<T> {
    void accept(T t);
}
```

2. zadatak (10 bodova)

Lijepo vrijeme izvuklo je na trg građane Javograda. Pažnju su plijenile gluha soba, bestežinska soba te kiosk s hotdogovima na kojem je robot pripremao hotdogove i posluživao kupce koji su stajali u redu. Prilikom posluživanja, kupac bi dobio i račun (s jedinstvenim serijskim brojem). Robot ne bi nastavljao s pripremom novog hotdoga dok prethodnog nije predao kupcu. U gluhoj i bestežinskoj sobi građani bi se zadržavali neko vrijeme, ali istovremeno je u svakoj od njih mogla biti samo jedna osoba.

Mali Javko je pokušao modelirati gornju situaciju, međutim nije naučio sinkronizacijske mehanizme u Javi, pa mu je potrebna pomoć. Izmijenite razrede `Robot`, `Gradjanin` i `HotDogKiosk` dane na u okviru ovog zadatka tako da se zadovolje uvjeti iz gornje priče.

Napomena: Nije potrebno prepisivati razrede i metode koji se ne mijenjaju.

```
package hr.fer.oop.zavrsni.dretve;
import java.util.Random;
public class Main {
    public static void main(String[] args) throws InterruptedException {
        Random r = new Random();
        HotDogKiosk kiosk = new HotDogKiosk();
        new Thread(new Robot(kiosk)).start();

        int id = 0;
        while(true){
            Thread thread = new Thread(new Gradjanin(++id, kiosk));
            thread.start();
            Thread.sleep(r.nextInt(1000));
        }
    }
}
```

```
package hr.fer.oop.zavrsni.dretve;
public class Robot implements Runnable {
    private HotDogKiosk kiosk;
    public Robot(HotDogKiosk kiosk){
        this.kiosk = kiosk;
    }
    @Override
    public void run() {
        while(true){
            kiosk.pripremi();
            try {
                Thread.sleep(500);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```

```

package hr.fer.oop.zavrsni.dretve;
public class HotDogKiosk {
    private int brojRacuna = 0;
    public void pripremi(){
        System.out.println("Robot pripremio hotdog");
    }
    public int prodajKupcu(){
        System.out.format("Hot dog prodan. \n");
        return noviRacun();
    }
    private int noviRacun(){
        brojRacuna++;
        return brojRacuna;
    }
}

```

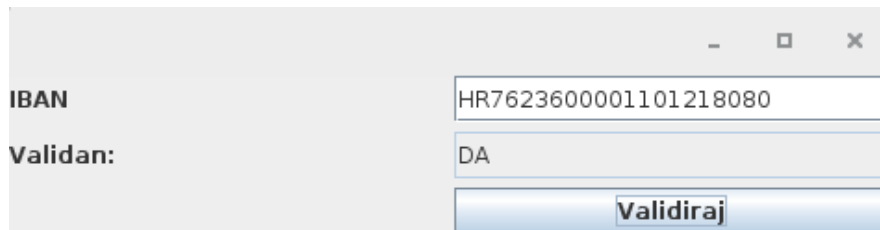
```

package hr.fer.oop.zavrsni.dretve;
import java.util.Random;
public class Gradjanin implements Runnable {
    private int id;
    private HotDogKiosk kiosk;
    public Gradjanin(int id, HotDogKiosk kiosk){
        this.id = id; this.kiosk = kiosk;
    }
    @Override
    public void run() {
        Random random = new Random();
        switch(random.nextInt(5)){
            case 0: gluhaSoba(); break;
            case 1: bestezinskaSoba(); break;
            default: kupiHotDog(); break;
        }
    }
    private void gluhaSoba() {
        System.out.format("%d ušao u gluhu sobu\n", id);
        try {
            Thread.sleep(3000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        System.out.format("%d izašao iz gluhe sobu\n", id);
    }
    private void bestezinskaSoba() {
        System.out.format("%d lebdi\n", id);
        try {
            Thread.sleep(3000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        System.out.format("%d prestao lebdjeti\n", id);
    }
    private String kupiHotDog() {
        System.out.format("Građanin %d stao u red za hotdog\n", id);
        int brojRacuna = kiosk.prodajKupcu();
        System.out.format("%d dobio hotdog i račun br %d\n",
                                                                    id, brojRacuna );
        return null;
    }
}

```

3. zadatak (10 bodova)

Potrebno je napisati cjelokupni Java kod (uključujući metodu `main`) jednostavne aplikacije s grafičkim korisničkim sučeljem za validaciju oznake bankovnog računa u IBAN formatu. Očekivani izgled aplikacije se nalazi na slici ispod. Za validaciju unesene oznake je potrebno koristiti klasu `IBANCheckDigit` iz paketa `org.apache.commons.validator.routines.checkdigit`. Ova klasa ima podrazumijevani konstruktor i metodu `boolean isValid(String iban)` koja vraća vrijednost `true` ako je predani `iban` validan, a inače vraća `false`.



The screenshot shows a Java Swing window with a light gray background. On the left side, there are two labels: "IBAN" and "Validan:". To the right of the "IBAN" label is a text input field containing the string "HR7623600001101218080". To the right of the "Validan:" label is a text input field containing the string "DA". Below these two input fields is a blue button with the text "Validiraj" in white. The window has standard macOS-style window controls (minimize, maximize, close) in the top right corner.