

## Objektno orijentirano programiranje Završni ispit

15.6.2021.

*Ispit nosi ukupno 25 bodova i piše se 150 minuta.*

*U zadacima nije potrebno pisati dio u kojem se uključuju klase ili paketi klasa (import)*

***Rješenja je potrebno pisati na stranice ispita. Ako na nekoj stranici ne budete imali dovoljno mjesta, ostatak možete napisati na zadnju (praznu) stranicu ispita.***

### IZJAVA

Tijekom ove provjere znanja neću od drugoga primiti niti drugome pružiti pomoć te se neću koristiti nedopuštenim sredstvima.

Ove su radnje povreda Kodeksa ponašanja te mogu uzrokovati trajno isključenje s Fakulteta.

Zdravstveno stanje dozvoljava mi pisanje ovog ispita.

**Vlastoručni potpis studenta:** \_\_\_\_\_

## Zajednički tekst za 1. i 2. zadatak

Neka trgovina pohranjuje račune svojih kupaca tako da svaki račun zapisuje kao datoteku naziva *bill-[identifikator].csv*. Računi su onda pohranjeni u mape oblika *YYYY-MM-DD* prema datumu provedbe.

Svaki račun (*Bill*) ima svoje atribute *id:String* i *articles:List<Article>*, gettere za svaki od njih, konstruktor koji prima atribut *id* te metodu *addArticle(Article)* koja dodaje artikl u listu artikala *articles*. Klasa *Article* sadrži atribute *name:String*, *basePrice:double*, *discount:double* i *int:quantity*, gettere za svaki od njih te konstruktor koji prima sve atribute.

Opis atributa: *id* – identifikator računa, *articles* – lista artikala u računu, *basePrice* – osnovna cijena, *discount* – trenutni popust izražen u postotku (npr. 0.2), *quantity* – količina kupljenih proizvoda

Račun je pohranjen kao datoteka, u čijem se imenu nalazi identifikator i uvijek je brojčana vrijednost, a svaki redak datoteke predstavlja jedan kupljeni artikl. Atributi artikla su pohranjeni kao zarezom odvojene vrijednosti (*comma separated value*). Primjer zapisa artikla je „*jabuka,4.0,0,5*“.

### 1. Zadatak (6 bodova)

Potrebno je dovršiti klasu *MyFileVisitor* i glavni program tako da glavni program u varijablu *bills* spremi sve račune koji su pohranjeni u datotekama unutar nekog direktorija i njegovih poddirektorija.

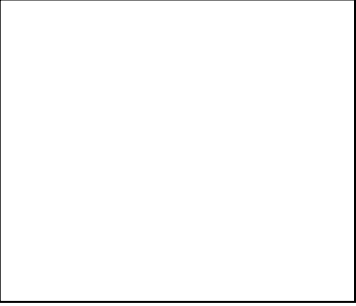
```
public static void main(String[] args) throws IOException {
    Path path = Paths.get(".");
    List<Bill> bills = new ArrayList<>();

    //TO DO: dovršiti

}
```

Koristeći *FileVisitor* dovršiti program koji ispravno učitava račune i pripadne artikle po računima iz predane putanje. U sljedećem programskom isječku potrebno je napisati metode *parseArticle* i *loadBill* te pomoću njih napraviti traženu funkcionalnost razreda *MyFileVisitor*. Metoda *parseArticle* parsira liniju datoteke prema kojoj stvara instancu razreda *Article* te istu vraća kao rezultat metode. *loadBill* provjerava nad primljenom putanjom radi li se o *.csv* datoteci te u slučaju da je tako, stvara račun (*Bill*) te dodaje sve artikle iz datoteke u račun. U slučaju da se kod provjere ekstenzije datoteke ne radi o *.csv* datoteci, potrebno je vratiti *null* vrijednost. *id* svakog računa nalazi se unutar naziva datoteke, dok se pojedinosti svakog artikla nalaze u datoteci.

NAPOMENA: Pretpostavite da su imena datoteka i njihov sadržaj uvijek ispravni.



```
public class MyFileVisitor extends SimpleFileVisitor<Path> {  
    private List<Bill> bills = new ArrayList<>();  
  
    // TODO dovršiti
```

```
private Bill loadBill(Path path) throws IOException {  
    // TODO dovršiti
```

```
}
```

```
private Article parseArticle(String line) {  
    // TODO dovršiti
```

```
}
```

```
public List<Bill> getBills() {  
    return bills;
```

```
}
```

```
}
```

## 2. Zadatak (6 bodova)

Koristeći koleksijske tokove, potrebno je nadopuniti klasu *BillStats* kako bi metode *getArticlesSortedByQuantityAndDiscount*,

*getTotalBillDiscount*, *getBillSums* radile ispravno. Metoda *getArticlesSortedByQuantityAndDiscount* prima jedan račun

i vraća listu svih artikala sortiranih najprije prema količini padajuć, a onda prema popustu izraženom u postotku (*discount*) uzlazno. Metoda *getTotalBillDiscount* prima račun i izračunava ukupni popust za sve artikle toga računa. Metoda *getBillSums* prima listu računa i vraća listu suma konačnih cijena svih artikala unutar računa. (Napomena: konačna cijena računa se kao bazna cijena – trenutni popust)

Napomena: smijete koristiti lambda izraze, anonimne klase, reference na metode, ugrađene komparatore, ... , međutim nije dozvoljeno riješiti zadatak iterativno bez koleksijskih tokova. Točan naziv neke od metoda iz koleksijskih tokova ili ugrađenih *default* metoda Javinih sučelja nije bitan sve dok se po smislu i argumentima jednoznačno može odrediti o kojoj postojećoj metodi se radi.

```
public class BillStats {
    public static List<Article> getArticlesSortedByQuantityAndDiscount(Bill bill){
        return bill.getArticles().stream()    // TODO dovršiti

    }

    public static Double getTotalBillDiscount(Bill bill){
        return bill.getArticles().stream()    // TODO dovršiti

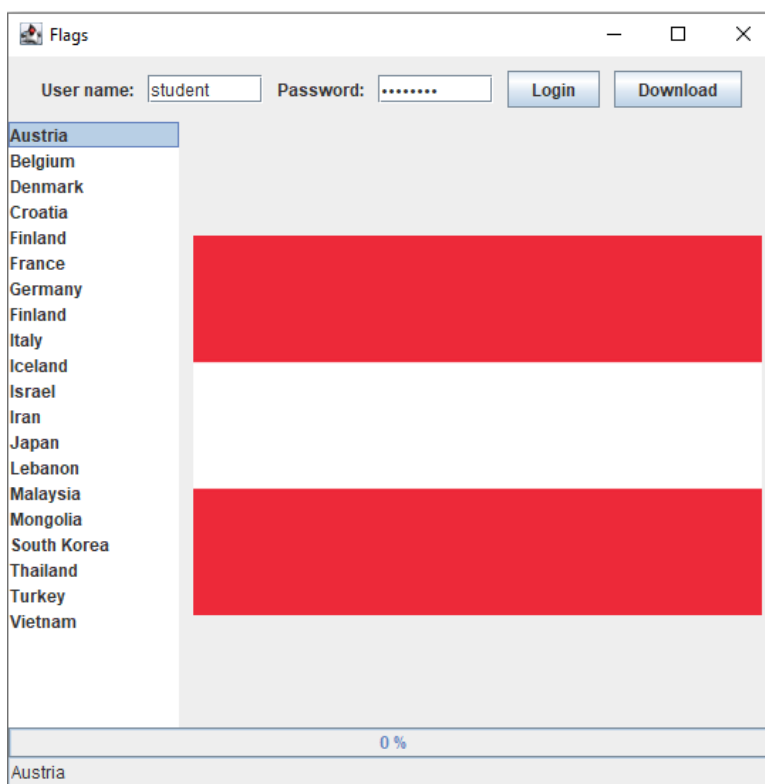
    }

    public static List<Double> getBillSums(List<Bill> bills){
        return bills.stream()    // TODO dovršiti

    }
}
```

### 3. Zadatak (6 bodova)

Dopunite razred *ImagesDownloader* koji predstavlja Swing aplikaciju pomoću koje se mogu prikazivati slike zastave određene države. Izgled aplikacije prikazan je na slici.



Slika 1

U gornjem dijelu aplikacije nalaze se polja za korisničko ime i lozinku te gumbi *Login* i *Download*. Na lijevoj strani se nalazi objekt tipa *JList* u kojem se nalaze države za koje su preuzete slike zastava. Najveći dio aplikacije zauzima objekt tipa *JLabel* u kojem će se prikazivati slika zastave. Na dnu prozora još se nalazi i *JProgressBar* za prikaz napretka prilikom preuzimanja zastava te polje za ispisivanje statusa.

U konstruktoru razreda *ImagesDownloader* kreirajte grafičko sučelje prikazano na slikama. Također je potrebno definirati ponašanje za gumb *Login*, kao i ponašanje aplikacije prilikom odabira države.

Na početku, kada se aplikacija pokrene, lista država i slika zastave su prazni, gumb *Download* je onemogućen, progress bar u dnu prozora je postavljen na 0, dok je tekst u statusnoj traci prazan. Pritiskom na gumb *Login* provjerava se ispravan upis korisničkog imena i lozinke (metoda *checkUser()*), i ako su unesene vrijednosti ispravne, potrebno je omogućiti gumb *Download*.

Gumbom *Download* pokreće se učitavanje zastava i punjenje država u *JList*. Funkcionalnost gumba *Download* dio je sljedećeg zadatka, i ovdje je **nije potrebno** implementirati!

Kada se u *JList*-i na lijevoj strani ekrana aplikacije promijeni odabir države, potrebno je osvježiti prikaz slike te u statusnoj traci upisati naziv odabrane države. Promjena odabira detektira se objektom koji implementira *ListSelectionListener*, i pri tome se poziva metoda *public void valueChanged(ListSelectionEvent e)*. U ovom slučaju sučelje *ListSelectionListener* je implementirano unutar razreda *ImageDownloader*.

**Uputa:** objekt tipa *JList* sadrži model (u zadatku tipa *DefaultListModel*), pomoću kojeg se može manipulirati elementima koje lista prikazuje. Pogledajte popis metoda na kraju.

```

public class ImagesDownloader extends JFrame implements
    ListSelectionListener
{
    private final List<String> countries = Arrays.asList("Austria",
        "Belgium" ... // Lista s imenima država

    // Mapa u koju spremamo države i slike zastava
    private Map<String, ImageIcon> images = new HashMap<>();
    // Model za listu država
    private DefaultListModel<String> model = new DefaultListModel<>();

    private JList<String> imagesList = new JList<>(model); // Lista država
    private JLabel lbImage; // Labela za prikaz zastave

    private JButton btnDownload = new JButton("Download"); // Gumb Download
    private JProgressBar progressBar = new JProgressBar(0, 100); // Progress bar u dnu prozora
    private JTextField txStatus = new JTextField(" "); // Statusna traka aplikacije
    private final JLabel lblUser = new JLabel("User name:");
    private final JTextField userField = new JTextField();
    private final JLabel lblPassword = new JLabel("Password:");
    private final JPasswordField passwordField = new JPasswordField();
    private final JButton btnLogin = new JButton("Login"); // Gumb Login

    // Metoda koja provjerava jesu li korisničko ime i lozinka ispravno uneseni, NE TREBA PISATI
    private boolean checkUser() { ... }

    // Metoda koja prikazuje zastavu za zadanu državu, NE TREBA PISATI
    public void updateLabel (String name) { ... }

    public ImagesDownloader() {
        super("Flags");
        imagesList.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
        imagesList.addListSelectionListener(this); // ImagesDownloader sluša promjene u JList-i
        lbImage = new JLabel();
        lbImage.setHorizontalAlignment(SwingConstants.CENTER);

        // TODO Definiranje izgleda aplikacije

```

```
btnLogin.addActionListener(e -> {  
    // TODO Akcija za gumb Login
```

```
});
```

```
btnDownload.addActionListener(e -> {  
    downloadFlagImages();  
});  
pack();  
}
```

```
@Override  
public void valueChanged(ListSelectionEvent e) {  
    // TODO Akcija za promjenu odabira u listi država
```

```
    }  
}
```

#### 4. Zadatak (7 bodova)

Nastavite implementaciju razreda *ImagesDownloader*. Pritiskom na gumb *Download* poziva se metoda *downloadFlagImages()*. Navedena metoda treba pokrenuti *DownloadWorker* koji predstavlja unutarnji razred od razreda *ImagesDownloader* i nasljeđuje razred *SwingWorker*. *DownloadWorker* će u pozadinskoj dretvi puniti *JList imageList* imenima država, te mapu *images* sa imenima država i slikama zastava. Države se nalaze u listi *countries*.

Preuzimanje zastave za pojedinu državu obavlja se metodom *loadFlagForCountry(String country)*, koja će vratiti sliku zastave, ili *null* ako sliku nije moguće dohvatiti. Ako preuzimanje nije moguće, metoda *doInBackground()* razreda *DownloadWorker* prestaje s radom i vraća string "Problems with network" koji se ispisuje u poruci na statusnoj liniji. Ukoliko je sve u redu preuzima se slika po slika. Na statusnoj liniji se ispisuje naziv države čija slika zastave se trenutno preuzima, a također se ažurira i trenutna vrijednost napredovanja. Zatim se pokazivač napredovanja ponovno postavlja na vrijednost 0 te se odabire prva država u listi.

```
public void downloadFlagImages() {      // TODO Pokretanje preuzimanja slika zastava
```

```
}
```

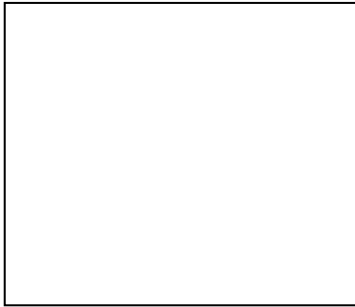


}

}

}

}



### Class SimpleFileVisitor<T>

Modifier and Type	Method	Description
FileVisitResult	postVisitDirectory(T dir, IOException exc)	Invoked for a directory after entries in the directory, and all of their descendants, have been visited.
FileVisitResult	preVisitDirectory(T dir, BasicFileAttributes attrs)	Invoked for a directory before entries in the directory are visited.
FileVisitResult	visitFile(T file, BasicFileAttributes attrs)	Invoked for a file in a directory.
FileVisitResult	visitFileFailed(T file, IOException exc)	Invoked for a file that could not be visited.

### Enum FileVisitResult

Enum Constants: CONTINUE, SKIP\_SIBLINGS, SKIP\_SUBTREE, TERMINATE

### Class Files

Modifier and Type	Method	Description
static Path	walkFileTree(Path start, FileVisitor<? super Path> visitor)	Walks a file tree.
static List<String>	readAllLines(Path path)	Read all lines from a file.

### Interface Path

Modifier and Type	Method	Description
Path	getFileName()	Returns the name of the file or directory denoted by this path as a Path object.
String	toString()	Returns the string representation of this path.

### Class JList<E>

Modifier and Type	Method	Description
E	getSelectedValue()	Returns the value for the smallest selected cell index; the selected value when only a single item is selected in the list.
ListModel<E>	getModel()	Returns the data model that holds the list of items displayed by the JList component.
void	setSelectedIndex(int index)	Selects a single cell.

### Class DefaultListModel<E>

Modifier and Type	Method	Description
int	getSize()	Returns the number of components in this list.
void	clear()	Removes all of the elements from this list.
void	add(int index, E element)	Inserts the specified element at the specified position in this list.
void	addElement(E element)	Adds the specified component to the end of this list.
void	addAll(Collection<? extends E> c)	Adds all elements from given collection into this list.
E	get(int index)	Returns the element at the specified position in this list.