Objektno orijentirano programiranje
Završni ispit
18.6.2019.

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)

1. zadatak (5 bodova)

Svaki automobil (`Car`) ima svoje atribute *`name:String`*, *`type:CarType`*, *`maxSpeed:int`*, *`power:int`*, *`consumption:double`* i *`price:double`* te *gettere* za svaki od njih. Tip auta je enumeracija *`CarType`* s vrijednostima *`Diesel`*, *`Petrol`*, *`Hybrid`*, *`Electric`*.

Koristeći kolekcijske tokove dopunite sljedeća 2 programska odsječka koji trebaju raditi sljedeće:

a) ispisati sve aute sortirane silazno po cijeni
b) ispisati prosječnu cijenu benzinaca ako takvi postoje u katalogu. Ako ne postoje, ne ispisuje se ništa

Napomena: smijete koristiti lambda izraze, anonimne klase, reference na metode, ugrađene komparatore, ... , međutim nije dozvoljeno riješiti zadatak iterativno bez kolekcijskih tokova. Točan naziv neke od metoda iz kolekcijskih 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.

```
List<Car> list =  CarCatalog.loadCars();
//print all cars sorted descending by price









//print average price of petrol cars (if such exist)
```

2. zadatak (5 bodova)

Dovršiti klasu *`MyFileVisitor`* i glavni program tako da glavni program ispiše broj pojedinih tipova (ekstenzija) datoteka unutar nekog direktorija i njegovih poddirektorija. Ne razlikovati velika i mala slova ekstenzija i obratiti pažnju da u poddirektorijima može biti datoteka bez ekstenzije (takve se ne uzimaju u obzir) i datoteka koje u svom imenu imaju više točaka pa je za njih ekstenzija tekst iza zadnje točke.

Npr. jedan od mogućih ispisa je

```
File types in: D:\GitRepositories\FER-OOP\Lectures\10_InputOutput\.
{JAVA=13, UCLS=2, PROJECT=1, XML=1, CLASS=13, PREFS=3, CLASSPATH=1}
```

```
public static void main(String[] args) throws IOException {
	Path path = Paths.get(".");
	System.out.println("File types in: " + path.toAbsolutePath().toString());
	//TO DO: dovršiti









}
```

Koristeći `FileVisitor` dovršiti program koji ispisuje broj pojavljivanja pojedinih ekstenzija datoteka.

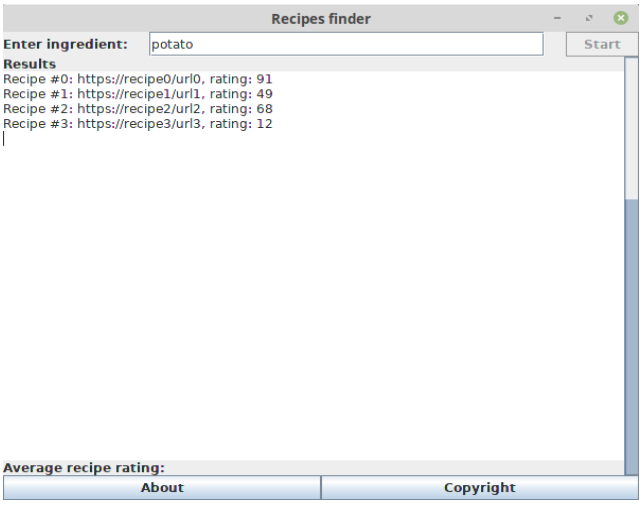
```
public class MyFileVisitor extends SimpleFileVisitor<Path> {

















}
```

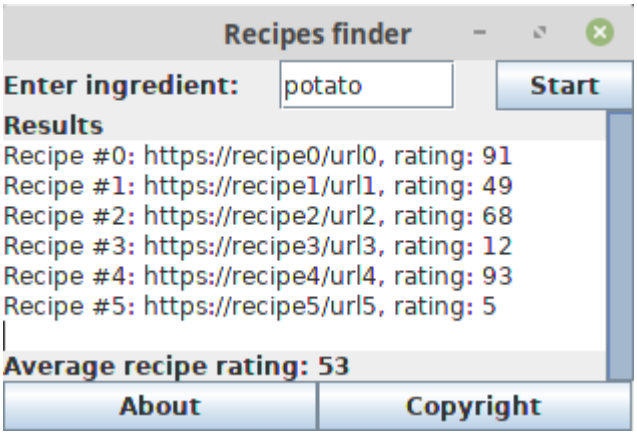
Zajednički tekst za 3., 4. i 5. zadatak

Potrebno je dovršiti sve potrebne klase za aplikaciju s grafičkim korisničkim sučeljem koja za uneseni sastojak traži sve recepte koji sadrže taj sastojak i za svaki pronađeni recept ispisuje naziv recepta (npr. Recipe #7), adresu gdje se recept može pronaći i ocjenu recepta (cijeli broj) u tekstualnom okviru i formatu kao što je prikazano na slici. Nakon što svi recepti budu ispisani ispisuje se prosječna ocjena pronađenih recepata.

Posao pronalaska recepta i ocjene recepta obavlja se unaprijed zadanim klasama i već je napisan u *SwingWorkeru* kojeg treba dopuniti u 4. zadatku. Naziv recepta, adresa recepta i njegova ocjena čine trojku podataka koja će se ispisivati na grafičkoj formi kao jedan string. SwingWorker je napisan tako da je neovisan o grafičkom sučelju na kojem se koristi i neće direktno moći pisati po grafičkom sučelju (jer ni ne zna za te kontrole). Format ispisa i mjesto ispisa određivat će sama grafička forma i proslijediti mu odgovarajuće objekte zadužene za te stvari (detaljnije u 4. i 5. zadatku). Promotrite npr. kako je u 5. zadatku definiran objekt *onDone* koji SwingWorker treba iskoristiti kad završi svoj posao.

Slike izvršavanja programa u različitim trenutcima izvođenja i s različitim veličinama prozora prikazane su na sljedećim slikama. Prikaz napretka odvija se vertikalnim *progress barom* s desne strane ekrana.





3. zadatak (5 bodova)

Potrebno je dovršiti programski kod kojim se kontrole razmještaju tako da izgled ekrana bude kao na navedenim slikama. U donjem programskom odsječku instancirane su sve komponente koje se vide na ekranu, a vaš je zadatak da po potrebi definirate dodatne panele i upravljače rasporedom kako bi se kontrole rasporedile kao na prikazanim slikama.

Napomena: *U 3. zadatku ne implementirate obradu događaja, pokretanje workera i slično, već samo izgled ekrana.* Udaljenost između teksta „*Enter ingredient*”, okvira za unos teksta i gumba *Start* je posljedica postavke hgap=20, vgap=20 kod odabranog upravljača rasporedom što možete ignorirati i riješiti bez razmaka.
Konstante za *BorderLayout* su *BorderLayout*.[NORTH/WEST/SOUTH/CENTER/EAST].

Ovdje upišite rješenje 3. zadatka

```
public class RecipesFinderWindow extends JFrame {
	public RecipesFinderWindow() {
		JTextField tfIngredient = new JTextField();
		JButton btnStart = new JButton("Start");
		JLabel lblIngredient = new JLabel("Enter ingredient:");
		JLabel lblResults = new JLabel("Results");
		JTextArea resultsArea = new JTextArea();
		JProgressBar progressBar = new JProgressBar();
		progressBar.setOrientation(SwingConstants.VERTICAL);
		JButton btnCopy = new JButton("Copyright");
		JButton btnAbout = new JButton("About");
		String avgRatingFormat = "Average recipe rating: %s";
		JLabel lblAvgRating = new JLabel(String.format(avgRatingFormat, ""));
		//TO DO: Add appropriate panels, layouts, ...











		//
		setSize(600, 500);
		setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
		setTitle("Recipes finder");
	}
	//main program ne treba pisati...
```

4. zadatak (5 bodova)
Dopuniti do kraja implementaciju *SwingWorkera* tako da u svom pozadinskom poslu za odabrani sastojak uzima popis svih recepata s nekim sastojkom, a zatim za svaki recept objavljuje i procesira tekst koji nastaje formatiranjem trojke (naziv recepta, adresa recepta, ocjene recepta) korištenjem objekta *formatter* (primljen u konstruktoru).

Za trojku je na raspolaganju klasa *Triple* koja predstavlja parametriziranu trojku (izvadak metoda je na kraju ispita).

SwingWorker zna ništa o tome iz koje grafičke forme će biti korišten niti koje kontrole se koriste za prikaz međurezultata i konačnog rješenja, pa kad bude gotov ili kad bude imao spreman međurezultat za procesiranje neće moći pisati direktno po grafičkoj formi, već će za to koristiti objekte iz konstruktora (objekti *consumer* prilikom procesiranja međurezultata i *onDone* kad bude gotov) – primijetiti kako je npr. *onDone* definiran u 5. zadatku tako da upiše rezultat na pravo mjesto.

```
public class RecipeWorker
   extends SwingWorker<_____________________________________________________________> {

       private String ingredient;

       private ____________________________________________________ consumer;

       private Consumer<Integer> onDone;
       private int avgRating;

       private Function<_____________________________________________________> formatter;
       public RecipeWorker(String ingredient,
                       [ne dopisivati, jer je istog tipa kao varijabla] consumer,
                       Consumer<Integer> onDone,
                       [ne dopisivati, jer je istog tipa kao varijabla] formatter )
      {
               this.ingredient = ingredient;
               this.consumer = consumer;
               this.onDone = onDone;
               this.formatter = formatter;
       }

       @Override
       protected Integer doInBackground() throws Exception {
               // Dohvaćanje mape recepata <naziv, adresa>
               Map<String, String> recipes = RecipeUtil.getRecipes(ingredient);
               int sum = 0;
               int counter = 0;
               for(Map.Entry<String, String> entry : recipes.entrySet()) {
                       int rating = RecipeUtil.rating(entry.getValue());
                       sum += rating;
                       ++counter;

                       //TO DO: dopisati kod za napredak (od 0 do 100) i međurezultat
```

```
            }
            avgRating = sum != 0 ? sum / recipes.size() : 0;
            return avgRating;
        }
    // TO DO:  dopisati preostale potrebne metode RecipeWorkera

}
```

5. zadatak (5 bodova)

Dovršiti kod za obradu klika na gumb *Start*. Prije pokretanja *workera* potrebno je dopisati kod kojim će se definirati način formatiranja trojke (naziv recepta, adresa recepta, ocjena) što je potrebno *SwingWorkeru* iz prethodnog zadatka da bi od trojke dobio tekst koji služi kao međurezultat, kod zadužen za nadopisivanje tako nastalog teksta u tekstualni okvir i kod za prikaz napretka.

```
ActionListener listener = e -> {
       String ingredient = tfIngredient.getText();
       if (ingredient.length() > 0) {
           btnStart.setEnabled(false);
           Consumer<Integer> onDone = i -> {
                     lblAvgRating.setText(String.format(avgRatingFormat, i));
                     btnStart.setEnabled(true);
           };
           //TO DO: definirati formatter









           //TO DO: Definirati kod za prikaz formatiranog teksta (objekt consumer)
           // koristi se resultsArea.append(neki tekst)









            RecipeWorker worker = new RecipeWorker(ingredient, consumer, onDone, formatter);
            //TO DO: definirati prikaz napretka
            // koristi se  progressBar.setValue(vrijednost)

















           worker.execute();
       }
};
btnStart.addActionListener(listener);
```

Class SwingWorker<T,V>

Modifier and Type	Method	Description
protected abstract T	doInBackground()	Computes a result, or throws an exception if unable to do so.
protected void	done()	Executed on the *Event Dispatch Thread* after the doInBackground method is finished.
void	execute()	Schedules this SwingWorker for execution on a *worker* thread.
T	get()	Waits if necessary for the computation to complete, and then retrieves its result.
protected void	process(List<V> chunks)	Receives data chunks from the publish method asynchronously on the *Event Dispatch Thread*.
protected void	publish(V... chunks)	Sends data chunks to the process(java.util.List<V>) method.
int	getProgress()	Returns the progress bound property.
protected void	setProgress(int progress)	Sets the progress bound property.

Class Triple<T, U, V> - vlastita klasa koju možete koristiti u zadatku

Modifier and Type	Method
public	Triple(T first, U second, V third)
public T	getFirst()
public void	setFirst(T value)
public U	getSecond()
public void	setSecond(U value)
public V	getThird()
public void	setThird(V value)

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

Klasa Files

Modifier and Type	Method	Description
static Path	walkFileTree(Path start, FileVisitor<? super Path> visitor)	Walks a file tree.

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.