Objektno orijentirano programiranje: završni ispit

Zadatci koji slijede napravljeni su oko zajedničkog modela podataka: znanstvenim radovima koji su objavljeni u časopisima. Autor znanstvenog rada modeliran je razredom `Author`. Znanstveni rad modeliran je razredom `ScientificPaper`. Važniji dijelovi oba razreda prikazani su u nastavku; pretpostavite da u razredima za sve članske varijable postoje i getteri te korektno napisane metode `hashCode`, `equals` i `toString`. Niti za jedan zadatak nije potrebno pisati Javadoc ili importe paketa. Na zadnjoj stranici nalazi se specifikacija korisnih razreda/sučelja/metoda koje možete koristiti u zadacima.

```
public class ScientificPaper {
        private String title;
        private Set<Author> authors;
        private Set<String> keywords;
        private String text;
        private String journal;
        private int year;
        private Path path;

        public ScientificPaper(
    String title,
    Set<Author> authors,
    Set<String> keywords,
    String text, String journal,
    int year, Path path) {
                super();
                this.title = title;
                this.authors = authors;
                this.keywords = keywords;
                this.text = text;
                this.journal = journal;
                this.year = year;
                this.path = path;
        }

        // plus getteri, equals,
   // hashcode...
}
```

```
public class Author {

        private String firstName;
        private String lastName;
        private String email;
        private String institution;

        public Author(String firstName,
    String lastName, String email,
    String institution) {
                super();
                this.firstName = firstName;
                this.lastName = lastName;
                this.email = email;
                this.institution =
            institution;
        }

        // plus getteri, equals,
   // hashcode...
}
```

Zadatak 1. (5 bodova)

Potrebno je napisati program koji pri pokretanju prima jedan argument u naredbenom retku: putanju do nekog direktorija. U tom direktoriju (odnosno u podstablu čiji je to korijen) nalazi se pohranjeno mnoštvo znanstvenih radova u obliku datoteka s ekstenzijom `*.xml` (datoteke ostalih ekstenzija treba zanemariti). Ako se u nekom poddirektoriju nalazi datoteka imena `IGNORE.ME`, tada taj direktorij te čitavno njegovo podstablo ne sadrži znanstvene radove (odnosno takvo čitavo podstablo treba zanemariti). Želimo napisati program koji će učitati znanstvene radove te ispisati koliko ih ima. Pretpostavite da za učitavanje rada iz datoteke čiju imate stazu možete iskoristiti statičku metodu razreda `LoaderUtil` (ne treba pisati kod ovog razreda!):

```
public static ScientificPaper load(Path file) throws IOException;
```

Osnovni kostur programa već je napisan u nastavku i ne smije ga se mijenjati. Vaš je zadatak nadopuniti tražene dijelove (upišite na crte što nedostaje), te na vlastitom papiru napišite programski kod razreda `PapersLoader` (pogledajte kako ga koristimo u metodi `main`) uz koji će napisani program generirati očekivani ispis.

```
public class Main {

	public static void main(String[] args) throws IOException {
		if(args.length != 1) {
			System.out.println("Očekivao sam stazu.");
			return;
		}

		// staza na temelju jedinog predanog argumenta:

		Path dir = ____________________________________________;

		// ako dir nije direktorij:

		if(____________________________________________) {
			System.out.println("Argument nije direktorij.");
			return;
		}

		PapersLoader loader = new PapersLoader();

		// obiđi čitavo podstablo oslanjajući se na postojeću funkcionalnost
		// razreda Files:

		Files.____________________________(dir, loader);

		List<ScientificPaper> papers = loader.getPapers();
		System.out.println("Broj znanstvenih radova: " + papers.size());
	}
}
```

Zadatak 2. (5 bodova)

Komparator c1 uspoređuje dva znanstvena rada po godini. Komparator c2 uspoređuje dva znanstvena rada po naslovu. Dopunite (na ovom papiru na predviđenom mjestu) definiciju komparatora c1 koristeći sintaksu stvaranja primjerka anonimnog razreda, te definiciju komparatora c2 koristeći sintaksu lambda izraza.

Comparator<ScientificPaper> c1 =

Comparator<ScientificPaper> c2 =

Pretpostavimo da imamo na raspolaganju listu znanstvenih radova:

```
List<ScientificPaper> papers = ...;
```

te objekte `c1` i `c2`. Koristeći funkcionalnost podatkovnih tokova koje nude kolekcije riješite sljedeće podzadatke (svaki podzadatak treba riješiti nadopunjavanjem konstrukcijom jedne ulančane naredbe; ne smijete pisati više naredbi).

Odredite najraniju godinu u kojoj postoji objavljen znanstveni rad:

```
Optional<Integer> godina = papers.
```

```
;
if(godina.isPresent()) {
	System.out.println("Najranija godina je: " + godina.get());
}
```

Stvorite listu svih radova koji u imenu imaju riječ “environmental”, pri čemu radovi u listi moraju biti poredani prema godini objavljivanja.

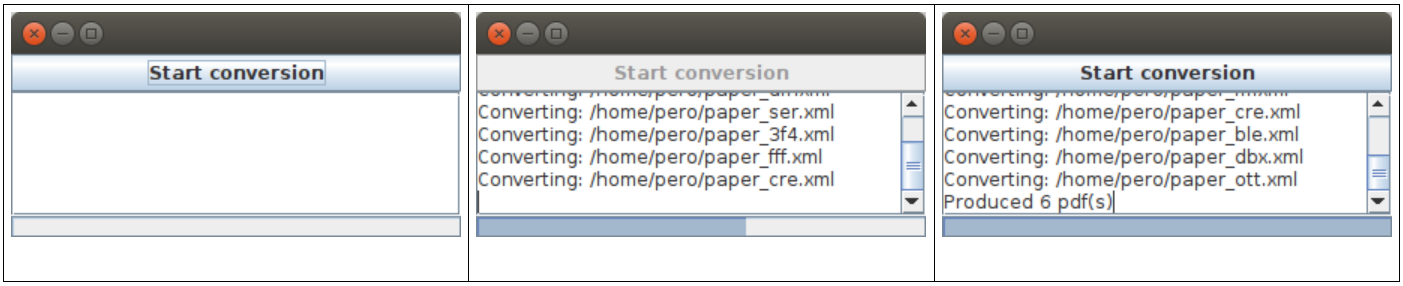
```
List<ScientificPaper> envPapers = papers.
```

Ispišite sve ključne riječi koje se pojavljuju u znanstvenim radovima. Istu ključnu riječ ne smijete ispisati više puta. Ključne riječi treba ispisati u abecednom poretku.

```
papers.
```

Zadatak 3. (5 bodova)

Radimo aplikaciju s grafičkim korisničkim sučeljem koja će omogućiti pretvorbu znanstvenih radova u PDF format: vidi slike u nastavku.



Kostur koda već je dan u nastavku. Metoda `initGUI` stvorit će korisničko sučelje i pripremiti listu znanstvenih radova (**nju ne trebate pisati**). Pritiskom na gumb start bit će pozvana metoda `startClicked()`. Vaša je zadaća napisati programski kod te metode, te odgovarajući `SwingWorker` (ugniježđeni razred `Worker`) koji će u pozadinskoj dretvi obavljati konverziju. Prije no što započne konverziju svakog rada, potrebno je ažurirati postotak obavljenog posla (koristeći za to predviđeni mehanizam koji nudi `SwingWorker`) koji GUI prikazuje; potom u tekstovnu komponentu ispisati da kreće konverzija određene datoteke. Za pretvorbu u PDF koristiti gotovu statičku metodu:

```
public static Path convert(ScientificPaper p);
```

razreda `ConvertUtil` (samo pozovite metodu – ne treba pisati kako je ona implementirana). Ako pretvorba uspije, metoda će vratiti stazu do generirane PDF datoteke. Ako pretvorba ne uspije, metoda će vratiti `null`. Zadaća SwingWorkera je stvoriti listu staza generiranih PDF datoteka, te kad je pretvorba gotova, omogućiti gumb za ponovno pokretanje te u tekstovnu komponentu dodati poruku koliko je PDF datoteka doista generirano (primijetite da taj broj može biti manji od broja radova koje se pokušalo konvertirati).

```
public class Main extends JFrame {

	private JButton start;
	private JTextArea log;
	private JProgressBar progressbar;
	private List<ScientificPaper> papers;

	public static void main(String[] args) {
		SwingUtilities.invokeLater(()->{
			new Main().setVisible(true);
		});
	}

	public Main() {
		setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
		setSize(300, 150);
		initGUI();
	}

	private void initGUI() {...}

	private void startClicked() { ... }

	class Worker ...
}
```

Ponovimo: napišite samo kod metoda `startClicked` te razreda `Worker` na vlastitim papirima. Ništa osim toga ne treba pisati. Pri tome u njima ne smijete koristiti `SwingUtilities.invoke*` metode.

Zadatak 4. (5 bodova)

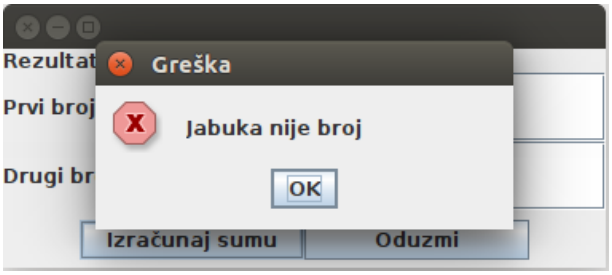
Radimo naredbeno-retčani program čiji je kostur dan u nastavku. Program učitava listu znanstvenih radova (tu metodu ne trebate pisati). Potom učitani popis dijeli u 4 podjednako velika dijela. Zatim pokreće četiri dretve – svaka odrađuje pretvorbu radova iz svoje četvrtine u PDF koristeći istu statičku metodu iz prošlog zadatka. Kada su sve pretvorbe gotove, program ispisuje poruku da su pretvorbe gotove.

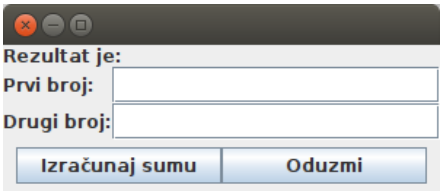
```
public class Main {

	public static void main(String[] args) throws IOException {
		List<ScientificPaper> papers = loaderPapers();

		List<List<ScientificPaper>> list = divide(papers, 4);

		// Stvori 4 dretve - svaka radi konverziju svojih radova

		// Pričekaj da pretvorbe završe

		System.out.println("Pretvorbe su gotove.");
	}
	private static List<List<ScientificPaper>> divide(List<ScientificPaper> papers, int sublists) {
		...
	}

	private static List<ScientificPaper> loaderPapers() {
		... // ne treba implementirati; pretpostavite da radi dobro!
	}
}
```

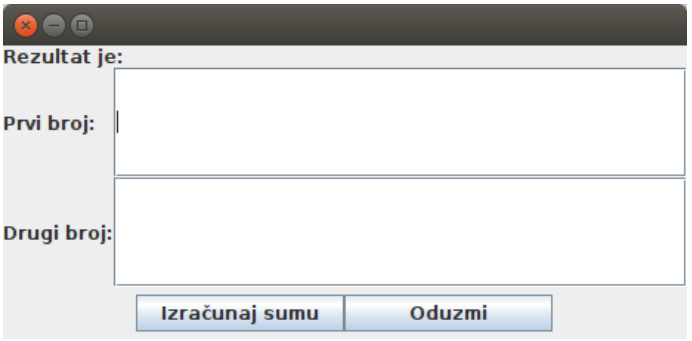
Na zasebnom listu papira napišite rješenje zadatka. Ne trebate implementirati metodu `loadPapers`.

Zadatak 5. (5 bodova)

Potrebno je napisati program koji će prikazati prozor (vidi slike u nastavku). Program omogućava unos dva broja te izračun njihovog zbroja ili razlike. Dio programa već je napisan pa najprije proučite zadani kod. Pritisak na gumbe treba pozvati metodu `calculateResult` koja je već napisana. Vaš je zadatak na zasebnom listu papira napisati metode `initGUI()` te `convertToInt`. Ova posljednja, ako se predani tekst ne da protumačiti kao broj, prikazuje prikladnu poruku u dijaloškoj kutiji. Slike u nastavku prikazuju kako se komponente trebaju razmiještati kada se mijenja veličina prozora. Ostvarite prikazani razmještaj oslanjajući se isključivo na upravljače razmještajem `FlowLayout`, `GridLayout` i `Borderlayout` (druge ne smijete koristiti).








```
public class Main extends JFrame {
	private JLabel result;
	private JTextField first;
	private JTextField second;

	public static void main(String[] args) {
		SwingUtilities.invokeLater(()->{
			new Main().setVisible(true);
		});
	}

	public Main() {
		setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
		setSize(300, 150);
		initGUI();
	}
	private void calculateResult(IntBinaryOperator operation) {
		OptionalInt a = convertToInt(first.getText());
		if(a.isEmpty()) return;
		OptionalInt b = convertToInt(second.getText());
		if(b.isEmpty()) return;

		int res = operation.applyAsInt(a.getAsInt(), b.getAsInt());

		result.setText("Rezultat je: " + res);
	}

	private void initGUI() {
		// Napišite na zasebnom papiru
	}

	private OptionalInt convertToInt(String text) {
		// Napišite na zasebnom papiru
	}

}
```

Specifikacija korisnih razreda/sučelja/metoda:

```
public interface FileVisitor<T> {
 FileVisitResult preVisitDirectory(T dir, BasicFileAttributes attrs) throws IOException;
 FileVisitResult visitFile(T file, BasicFileAttributes attrs) throws IOException;
 FileVisitResult visitFileFailed(T file, IOException exc) throws IOException;
 FileVisitResult postVisitDirectory(T dir, IOException exc) throws IOException;
}

public enum FileVisitResult {
 CONTINUE, TERMINATE, SKIP_SUBTREE, SKIP_SIBLINGS;
}

public interface Stream<T> extends BaseStream<T, Stream<T>> {
 <R, A> R collect(Collector<? super T, A, R> collector);
 Stream<T> distinct();
 Stream<T> filter(Predicate<? super T> predicate);
 void forEach(Consumer<? super T> action);
 <R> Stream<R> flatMap(Function<? super T, ? extends Stream<? extends R>> mapper);
 <R> Stream<R> map(Function<? super T, ? extends R> mapper);
 Optional<T> max(Comparator<? super T> comparator);
 Optional<T> min(Comparator<? super T> comparator);
 Stream<T> sorted();
 Stream<T> sorted(Comparator<? super T> comparator);
}

public final class Optional<T> {
 public static<T> Optional<T> empty();
 public static <T> Optional<T> of(T value);
 public T get();
 public boolean isPresent();
 public boolean isEmpty();
}

public abstract class SwingWorker<T, V> implements RunnableFuture<T> {
 protected abstract T doInBackground() throws Exception;
 protected final void publish(V... chunks);
 protected void process(List<V> chunks);
 protected void done();
 protected final void setProgress(int progress);
 public final void execute();
 public final T get() throws InterruptedException, ExecutionException;
 public final void addPropertyChangeListener(PropertyChangeListener listener);
}

public interface PropertyChangeListener extends java.util.EventListener {
 void propertyChange(PropertyChangeEvent evt);
}

public class PropertyChangeEvent extends EventObject {
 public String getPropertyName();
 public Object getNewValue();
 public Object getOldValue();
}

class JOptionPane ... {
 public static void showMessageDialog(Component parentComponent,
        Object message, String title, int messageType);
}

public interface IntBinaryOperator {
    int applyAsInt(int left, int right);
}
```