Šolski center Novo mesto

Srednja elektro šola in tehniška gimnazija

Šegova ulica 112

8000 Novo mesto

APLIKACIJE IN INFORMACIJSKI SISTEMI –

ORODJE ZA RAČUNALNIŠKO PODPRTO PREVAJANJE

(Maturitetna seminarska naloga)

Predmet: Računalništvo

Avtor: Anže Pintar

Mentor: dr. Albert Zorko

Novo mesto, april 2023

POVZETEK

V seminarski nalogi predstavim celoten postopek izdelave orodja za računalniško podpro prevajanje. Na začetku opišem cilj seminarske naloge in tehnologije, ki sem jih uporabil pri izdelavi orodja.

Nato predstavim načrtovanje, izdelavo in testiranje programa ter predstavim delovanje programa z vidika uporabnika in iz vidika programske kode.

Priloga vsebuje tudi povezavo do izvorne kode in datotečno strukturo aplikacije.

Ključne besede: Orodje za računalniško podprto prevajanje, Java, JavaFX

KAZALA

KAZALO VSEBINE

KAZALO SLIK

1 UVOD

Številni posamezniki se dnevno soočajo z željo po prevajanju določenih dokumentov. Obstajajo profesionalna orodja, kot so računalniško podprta prevajalska orodja (angl. CAT tools), vendar so ta orodja za povprečne uporabnike, ki le občasno prevajajo, pogosto predraga ali prezahtevna. Med takim orodji najdemo programe, kot so Trados Studio, MemoQ in OmegaT, po katerih sem se tudi zgledoval pri izdelavi mojega orodja.

Zaradi vsega tega sem se odločil ustvariti program, ki je enostaven za uporabo in ima nizko učno krivuljo, kar pomeni, da se ga lahko uporabnik hitro nauči.

Za izdelavo orodja sem si izbral programski jezik Java, ker je to jezik, ki ga najbolje poznam.. Java privzeto podpira starejšo grafično knjižnico Java Swing, zato sem se odločil, da bom grafični del programa zasnoval z uporabo sodobnejše knjižice JavaFX, znane tudi kot OpenJFX. (1) Med izdelavo programa sem uporabil tudi druge knjižice, ki jih bom predstavil v nadaljnjih poglavjih.

1.1 UPORABLJENE TEHNOLOGIJE

Pri izdelavi Orodja za računalniško podprto prevajanje sem poleg programskega jezika Jave uporabil tudi druge tehnologije, ki jih bom predstavil nadaljevanju

1.1.1 JavaFX

JavaFX je zmogljiva knjižnica za izdelavo grafičnih uporabniških vmesnikov v programskem jeziku Java. Razvilo jo je podjetje Sun Microsystems, kasneje pa je postala del Jave pod okriljem Oracle. mogoča razvoj sodobnih in interaktivnih aplikacij za različne platforme, kot so Windows, macOS, Linux in je celo mobilne naprave. JavaFX temelji na hierarhiji grafičnih vozlišč (angl. Scenegraph), ki predstavljajo grafične elemente, kot so gumbi, okenca, drsniki, slike in besedila. To omogoča enostavno organizacijo in oblikovanje elementov ter upravljanje z dogodki, kot so pritiski na gumbe ali vnos podatkov. Prav tako pa tudi podpira večpredstavnostne vsebine, kot so video in zvok, kar omogoča ustvarjanje različnih uporabniških izkušenj. (2)

Za lažje oblikovanje uporabniških vmesnikov JavaFX uporablja FXML format, ki je različica XML jezika, namenjena opisu strukture elementov uporabniškega vmesnika. FXML omogoča ločitev programske logike aplikacije od njenega grafičnega videza, kar izboljša preglednost in modularnost programske kode ter olajša sodelovanje med

razvijalci in oblikovalci. JavaFX tudi omogoča uporabo CSS za oblikovanje uporabniških vmesnikov, s čimer olajša prilagajanje videza aplikacij in olajša uporabo oblikovalskih smernic in doseganje enovitosti grafičnega videza med platformami. (3)

1.1.2 TMX in XLIFF format

TMX in XLIFF sta dva pomembna formata za upravljanje prevodov in lokalizacijo v industriji prevajanja. TMX (Translation Memory eXchange) je standardni format, ki omogoča izmenjavo prevodnih spominov med različnimi prevajalskimi orodji. Razvit je bil s strani LISA (Localization Industry Standards Association) v 90-ih letih (4).

XLIFF (XML Localization Interchange File Format) pa je drugi format, ki je namenjen izmenjavi prevodov in lokalizacijskih podatkov med različnimi orodji za lokalizacijo. Razvit je bil s strani OASIS (Organization for the Advancement of Structured Information Standards) v začetku 2000-ih (5).

TMX in XLIFF se uporabljata zato, ker omogočata združljivost med različnimi prevajalskimi orodji in procesi. To pomaga pri izboljšanju učinkovitosti in kakovosti prevodov ter zmanjšuje stroške lokalizacije (6).

TMX format temelji na XML in vsebuje prevodne enote (TU), ki so sestavljene iz izvornega besedila, ciljnega besedila in metapodatkov, kot so jezik, datum in referenčni viri (4). TMX datoteke se lahko uvozijo in izvozijo v večino prevajalskih orodij, kar omogoča enostavno izmenjavo prevodnih spominov med različnimi prevajalskimi ekipami.

XLIFF format prav tako temelji na XML in se osredotoča na lokalizacijske podatke, kot so izvorna in ciljna besedila, metapodatki, opombe in označevalniki za zaznavanje prelomov besedila (5). XLIFF omogoča izmenjavo prevodov in lokalizacijskih podatkov, ne da bi bilo treba skrbeti za združljivost med različnimi orodji in platformami.

1.1.3 ControlsFX

v ControlsFX je odprtokodna knjižnica, ki ponuja dodatne, zmogljive in prilagodljive UI komponent za JavaFX, namenjene izboljšanju uporabniške izkušnje. ControlsFX ponuja široko paleto komponent, kot so obvestila, nabor znakov, obrazci, območja z gumbi, številčne drsnike, zavihke z zapiranjem, poševne črte in mnoge druge. Poleg tega vključuje

podporo za validacijo podatkov v obrazcih, grafične orodje za gradnjo in razširjene funkcije dialogov ControlsFX je bil zasnovan tako, da je enostaven za uporabo, z jasnim in enotnim API-jem, kar omogoča razvijalcem, da enostavno dodajo napredne funkcije v svoje JavaFX aplikacije. (7)

1.1.4 JUnit

JUnit je priljubljena in razširjena knjižnica za testiranje enot (angl. unit testing) Java aplikacij, ki omogoča razvijalcem, da preverijo pravilno delovanje svoje kode in zagotovijo njeno kakovost. JUnit je bil razvit z namenom ponuditi enostavno, hitro in učinkovito orodje za pisanje in izvajanje testov v Java okolju. (8)

JUnit temelji na konceptu testnih primerov (angl. test cases), ki so razredi, ki vsebujejo testne metode za preverjanje določenih delov kode. Testne metode običajno vključujejo niz pričakovanih rezultatov in dejanskih rezultatov, pridobljenih iz testirane kode, ter uporabo trditev (angl. assertions), s katerimi se primerja pričakovane in dejanske rezultate. (9)

JUnit ponuja številne funkcije, kot so testni pripomočki (angl. test fixtures), testni zbirniki (angl. test suites), testni golniki (angl. test runners) in testni poročevalci (angl. test reporters). (9) Testni pripomočki omogočajo enostavno inicializacijo in čiščenje stanja, ki ga potrebujejo testne metode. Testni zbirniki združujejo več testnih primerov in omogočajo izvajanje testov za več razredov hkrati. Testni gonilniki so odgovorni za izvajanje testov in prikaz rezultatov, medtem ko testni poročevalci ustvarjajo poročila o testiranju, ki jih je mogoče uporabiti za analizo rezultatov. (9)

1.1.5 ODF Toolkit

Odf Toolkit je knjižnica za programski jezik Java, ki omogoča enostavno branje, zapisovanje in manipulacijo dokumentov, ki temeljijo na Open Document Format (ODF) standardu, kot so ODT (OpenDocument Text), ODS (OpenDocument Spreadsheet) in ODP (OpenDocument Presentation) (10) (11)

Odf je odprtokodni standard za shranjevanje in izmenjavo pisarniških dokumentov, ki ga je razvila organizacija OASIS in ga podpirajo številni pisarniški paketi, kot so LibreOffice, Microsoft Office in OpenOffice. (11)

Odf Toolkit omogoča razvijalcem delo z Odf dokumenti v programskem jeziku Java, brez uporabe zunanjih rešitev. To omogoča avtomatizacijo generanja poročil, analize podatkov, pretvorba dokumentov in drugi procesi, ki zahtevajo interakcijo z Odf dokumenti. (11)

Knjižnica vključuje številne komponente za delo z Odf dokumenti, kot so Odf Validator, ODFDOM in XSLT Runner. Odf Validator je knjižica z preverjanje pravilne strukture Odf datotek, medtem ko ODFDOM omogoča nizkonivojski dostop do Odf strukture, kar omogoča večjo prilagodljivost in več funkcionalnosti delu z dokumenti. XSLT Runner pa omogoča izvajanje XSLT transformacij na Odf dokumentih, kar lahko uporabimo za pretvorbo, analizo ali obdelavo dokumentov. (10)

Knjižica omogoča tudi ustvarjanje, spreminjanje, branje in zapisovanje vsebine v Odf dokumente, vključno z besedilom, tabelami, grafikoni, slikami in formulami. Poleg tega knjižnica omogoča tudi delo z naprednimi funkcijami, kot so stili, oblikovanje, vrtilne tabele, komentarji in zaščita dokumentov. (12)

1.1.6 Apache POI

Apache POI je odprtokodna knjižnica za programski jezik Java, ki omogoča branje in zapisovanje Microsoft Office dokumentov, kot so Excel (XLS in XLSX), Word (DOC in DOCX) in PowerPoint (PPT in PPTX) (13)

Glavna značilnost Apache POI je, da omogoča enostavno delo z Microsoft Office dokumenti v Javi brez uporabe Microsoft Office ali drugih zunanjih komponent Uporablja se za avtomatizacijo generiranja poročil, analizo podatkov in pretvorbo dokumentov. Poleg preprostega urejanja vsebine dokumentov, knjižica omogoča tudi delo s grafikoni, slikami, tabelami in naprednimi funkcijami, kot so stili, oblikovanje, vrtilne tabele, komentarji in zaščito dokumentov. (14)

1.1.7 Jakarta

Jakarta EE je zbirka specifikacij, ki določajo standardne funkcionalnosti in storitve za razvoj, izvajanje in upravljanje velikih, večkomponentnih in razširljivih aplikacij v programskem jeziku Java. Knjižnica poenostavi razvoj in zagotavlja prenosljivost med različnimi implementacijami, ki temeljijo na teh specifikacijah. Vključuje širok nabor funkcionalnosti in storitev, kot so sistemi za upravljanje podatkovnih baz, upravljanje transakcij, varnost, vmesniki za elektronsko pošto, obdelavo sporočil in delo z različnimi

datotekami. Ena od ključnih značilnosti Jakarta EE je podpora za delo z XML datotekami. (15)

Jakarta EE omogoča delo z XML datotekami prek različnih tehnologij, kot so JAXB (Java Architecture for XML Binding) in JAXP (Java API for XML Processing). JAXB omogoča pretvorbo med XML dokumenti in Java objekti, kar poenostavlja branje in pisanje XML podatkov. Z JAXB lahko razvijalci enostavno ustvarijo Java razrede, ki ustrezajo določeni XML shemi, in učinkovito prenašajo podatke med XML in Java objekti. JAXP pa omogoča obdelavo XML dokumentov s pomočjo različnih pristopov, kot so DOM (Document Object Model), SAX (Simple API for XML) in StAX (Streaming API for XML). Ti pristopi omogočajo branje, pisanje, iskanje in manipulacijo XML dokumentov na različne načine, odvisno od potreb aplikacije. (16)

1.1.8 Log4j

Log4j je zmogljiva in prilagodljiva knjižnica za beleženje dogodkov (angl. logging) v Java aplikacijah, ki jo je razvila Apache Software Foundation. Knjižnica omogoča razvijalcem, da zbirajo in upravljajo informacije o dogodkih, ki se pojavljajo med izvajanjem aplikacije. (17) Log4j je široko uporabljen v Java ekosistemu in se pogosto uporablja kot pomožna knjižnica za beleženje dogodkov v drugih knjižnicah. Ponuja več ravni beleženja, kot so TRACE, DEBUG, INFO, WARN, ERROR in FATAL, ki omogočajo razvijalcem, da prilagodijo količino in vrsto informacij, ki jih želijo zabeležiti. Ta značilnost je koristna pri diagnosticiranju in odpravljanju napak, saj lahko razvijalci enostavno prilagodijo raven beleženja glede na potrebe. (18)

Log4j je zelo konfigurabilen in omogoča razvijalcem, da prilagodijo beleženje dogodkov glede na svoje potrebe. Konfiguracija se lahko izvede s pomočjo konfiguracijskih datotek, kot so XML, JSON, YAML ali lastnostne datoteke, ali s programsko konfiguracijo v Java kodi. (17)

1.1.9 Maven

Maven je orodje za upravljanje vtičnikov in odvisnosti projektov, ki so napisani v programskem jeziku Java. Razvila ga je Apache Software Foundation. Maven pomaga avtomatizirati in poenostaviti procese kompilacije, dokumentiranja, pregleda in upravljanja odvisnosti v projektih. Ena izmed ključnih funkcionalnosti Maven je njegova možnost upravljanja z odvisnostmi. Maven poskrbi za prenos potrebnih knjižnic in njihovih različic, na podlagi tega, kaj je določeno v konfiguracijski datoteki projekta. To

omogoča enostavnejše vzdrževanje projekta, saj lahko razvijalci hitro in učinkovito posodobijo različice knjižnic ter zagotovijo kompatibilnost z uporabo ustreznih verzij v celotnem projektu. (19)

Osrednja konfiguracijska datoteka v Maven projektu je pom.xml (Project Object Model). Pom.xml določa projektno metapodatke, strukturo, odvisnosti, kompilacijske cilje in vtičnike projekta. Z uporabo pom.xml datoteke Maven omogoča enotno strukturo projekta in doslednost pri upravljanju odvisnosti, kar olajša sodelovanje med razvijalci ter zmanjšuje napake, povezane z neskladnostmi v konfiguracijah. Vključuje tudi sistem vtičnikov, ki omogoča dodajanje funkcionalnosti, kot so preverjanje kode, generiranje dokumentacije, objavljanje artefaktov v oddaljene repozitorije. (20)

1.1.10 Git

Git je pogosto uporabljeno orodje za nadzor različic, ki omogoča učinkovito sledenje spremembam v kodi in sodelovanje med uporabniki. (21) Deluje na osnovi distribuiranega načina nadzora različic, kar pomeni, da vsak uporabnik dela s svojo lokalno kopijo repozitorija. To omogoča večjo zasebnost, hitrost in enostavnejše vzporedno delo na več vej razvoja. Git omogoča hitro združevanje sprememb z različnih vej in s tem preprečuje konflikte, kar je ključnega pomena pri skupnem delu. (21)

Git kot orodje za nadzor različic shranjuje zgodovino sprememb vseh datotek, sledenje avtorstvu, kreiranje vej za ločeno delo na določenih funkcionalnostih in združevanje teh sprememb z glavno vejo, ko so funkcionalnosti dokončane. Prav tako omogoča uporabo oznak za označevanje pomembnih trenutkov v zgodovini repozitorija, kar olajša razumevanje in sledenje napredku. (22)

1.1.11 Scene Builder

Scene Builder je grafično orodje za urejanje fxml datotek, ki ga je razvilo podjetje Oracle, zdaj pa ga vzdržuje skupnost Gluon. To orodje omogoča enostavno grafično urejanje uporabniških vmesnikov aplikacije, ki uporabljajo JavaFX. S pomočjo tega orodja lahko razvijalci z uporabo tehnike »povleci in spusti« izdelajo kompleksne in privlačne uporabniške vmesnike, ne da bi pri tem morali ročno urejati fxml datoteke. (23)

Orodje omogoča široko paleto funkcionalnosti, vključno z grafičnim urejanjem fxml datotek, pregledovanjem hierarhije prizorišča, urejanjem CSS slogov, vključevanjem

lastnih JavaFX komponent ter povezovanja kontrolorjev s krmilniki dogodkov (angl. event handlers) in lastnostmi.

Grafični vmesnik programa Scene Builder omogoča tudi enostavno preklapljanje med različnimi predlogami za različne zaslonske resolucije, kar olajša razvoj aplikacij za naprave in platforme. S tem programom lahko razvijalci hitreje izdelujejo prototipe in končne uporabniške vmesnike, kar skrajša čas, potreben za razvoj, in povečuje produktivnost. (24)

1.1.12 XJC

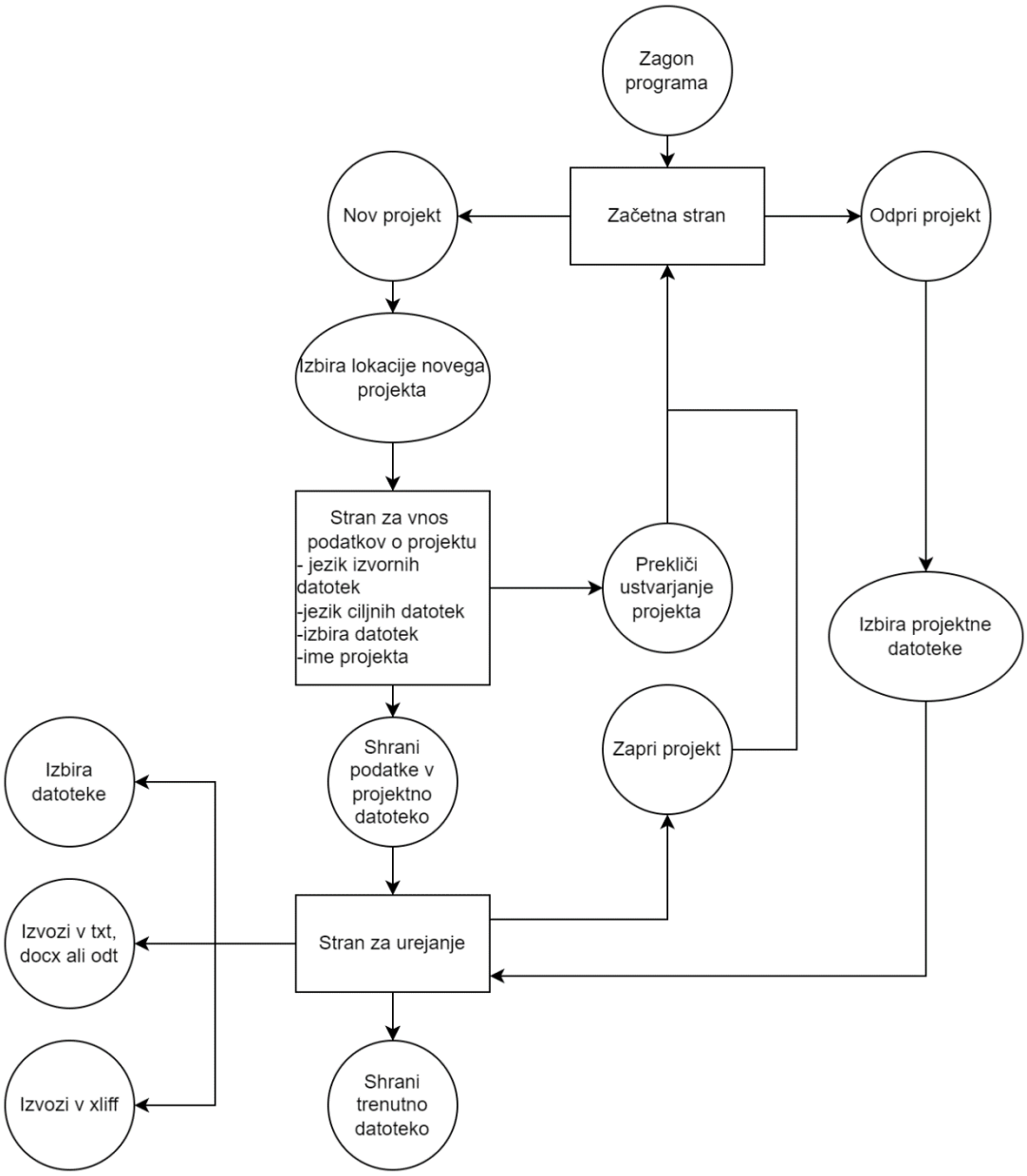
XJC (XML Java Compiler) je orodje, ki je del Java Architecture for XML Binding (JAXB) ogrodja. JAXB omogoča pretvorbo XML podatkov v Java objekte in nazaj, kar olajša branje, pisanje in manipulacijo XML dokumentov znotraj Java aplikacij. XJC omogoča generiranje Java razredov iz XML shem (XSD), kar pomaga pri enostavni uporabi XML podatkov v Java aplikacijah. (25)

XJC prebere XML shemo in nato generira ustrezne Java razrede, ki predstavljajo elemente, atribute, tipove in druge konstrukte, določene v shemi. Razvijalci lahko nato uporabljajo te generirane razrede za enostavno delo z XML podatki v svojih Java aplikacijah, brez potrebe po ročnem pisanju kode za razčlenjevanje ali ustvarjanje XML dokumentov. (26)

2 NAČRTOVANJE IN IZDELAVA PROGRAMA

2.1 NAČRTOVANJE

Na začetku načrtovanja programa sem si naredil okvirno skico vseh funkcij programa z uporabniškega vidika. Ugotovil sem, da je najbolje, če uporabim tri različna okna, preko katerih bo uporabnik uporabljal program. Ta okna sem poimenoval začetna stran, stran za vnos podatkov o projektu in stran za urejanje. Poleg tega sem si zamislil tudi katere funkcije se nahajajo na kateri strani. Vse to je prikazano na spodnji sliki (Slika 1).



Slika 1 Začetni načrt aplikacije

2.2 PRETVORBA ZAMISLI V OSNUTEK PROGRAMA

Nato sem se lotil preučevanja literature. Ugotovil sem, da je za je pri projektih, ki so napisani v javi zelo dobro uporabljati program, ki ureja knjižnice in odvisnosti, zaradi česa sem se odločil, da bom uporabil Maven.

Nato sem se lotil izdelave grafične podobe aplikacije, za katero sem uporabil program Scene Builder, ki je zelo prijazen za uporabnike, ki se prvič srečajo z JavaFX, saj za izdelavo grafičnega vmesnika ni potrebno poznavanje razredov in metod gradnikov, ker celotna izdelava poteka po sistemu povleci in spusti.

2.3 DODAJANJE FUNKCIONALNOSTI

Dodajanje funkcionalnosti sem se lotil po iz vidika uporabnika, kar pomeni, da sem funkcionalnosti implementiral po vrsti, od tiste, ki se potrebuje takoj na začetku programa, do tiste, ki ni pomembna za celotno izvajanje programa in ni bila nujno potrebna. Tako sem se najprej lotil implementacije začetnega zaslona in nato zaslona za urejanje datotek, šele nato pa sem se lotil implementacije ustvarjanja projekta in izvažanja v xliff datoteko.

2.4 KOMPILACIJA IN ZAPAKIRANJE V EXE

Za pakiranje mojega programa sem uporabil Maven Shader Plugin in Launch4j. Najprej sem v `pom.xml` datoteko dodal vtičnik Maven Shader Plugin in konfiguriral njegov cilj, ki zapakira vse razrede in odvisnosti v en samo JAR datoteko. Zaradi tega vtičnika sem moral tudi ustvariti nov razred `OzrppLauncher`, ker le ta potrebuje zaganjalni razred, ki nima dodatnih metod.

Nato sem v `pom.xml` dodal Launch4j in ustvaril konfiguracijska datoteko, kjer sem določil vhodno JAR datoteko in konfiguriral kompilacijo v exe datoteko tako, da mi sedaj ob vsaki uspešni kompilaciji ustvari exe datoteko.

Konfiguracija za vtičnik Maven Shader Plugin

```
<plugin>
  <artifactId>maven-shade-plugin</artifactId>
  <executions>
    <execution>
      <configuration>
        <transformers>
          <transformer
            implementation=
"org.apache.maven.plugins.shade.resource.ManifestResourceTransformer">
            <mainClass>com.anzepintar.ozrpp.OzrppLauncher</mainClass>
```

```
          </transformer>
        </transformers>
      </configuration>
      <goals>
        <goal>shade</goal>
      </goals>
      <phase>package</phase>
    </execution>
  </executions>
  <groupId>org.apache.maven.plugins</groupId>
  <version>3.0.0</version>
</plugin>
```

Konfiguracija za Lauch4j vtičnik

```
<plugin>
  <groupId>com.akathist.maven.plugins.launch4j</groupId>
  <artifactId>launch4j-maven-plugin</artifactId>
  <version>2.3.3</version>
  <executions>
    <execution>
      <id>l4j-clui</id>
      <phase>package</phase>
      <goals>
        <goal>launch4j</goal>
      </goals>
      <configuration>
        <headerType>gui</headerType>
        <jar>target/ozrpp-1.0.0.jar</jar>
        <outfile>target/Ozrpp.exe</outfile>
        <downloadUrl>http://java.com/download</downloadUrl>
        <classPath>
          <mainClass>com.anzepintar.ozrpp.OzrppLauncher</mainClass>
        </classPath>
        <icon>src/main/resources/img/icon.ico</icon>
        <jre>
          <path>%JAVA_HOME%</path>
        </jre>
        <versionInfo>
          <fileVersion>1.0.0.0</fileVersion>
          <txtFileVersion>1.0.0</txtFileVersion>
          <productVersion>1.0.0.0</productVersion>
          <txtProductVersion>1.0.0</txtProductVersion>
          <productName>Ozrpp</productName>
          <internalName>Ozrpp</internalName>
          <originalFilename>Ozrpp.exe</originalFilename>
        </versionInfo>
      </configuration>
    </execution>
  </executions>
</plugin>
```

2.5 TESTIRANJE PROGRAMA

Metode, ki sem jih vključil v program sem sproti testiral z testi za preizkušanje t.j. JUnit testi. Ustvaril sem teste, ki testirajo delovanje določenih funkcij programa kot so uvažanje datotek, generiranje tmx in xliff datotek ter shranjevanje tmx datotek v formate vhodnih datotek. Primer testa za testiranje razreda za uvažanje datotek lahko vidite spodaj. Za namen tega programa sem ustvaril vhodno txt datoteko, ki jo test posreduje metodi `parseTxtFile()` ter nato primerja vsebino seznama s pričakovanimi rezultati

FileImporterTest.java

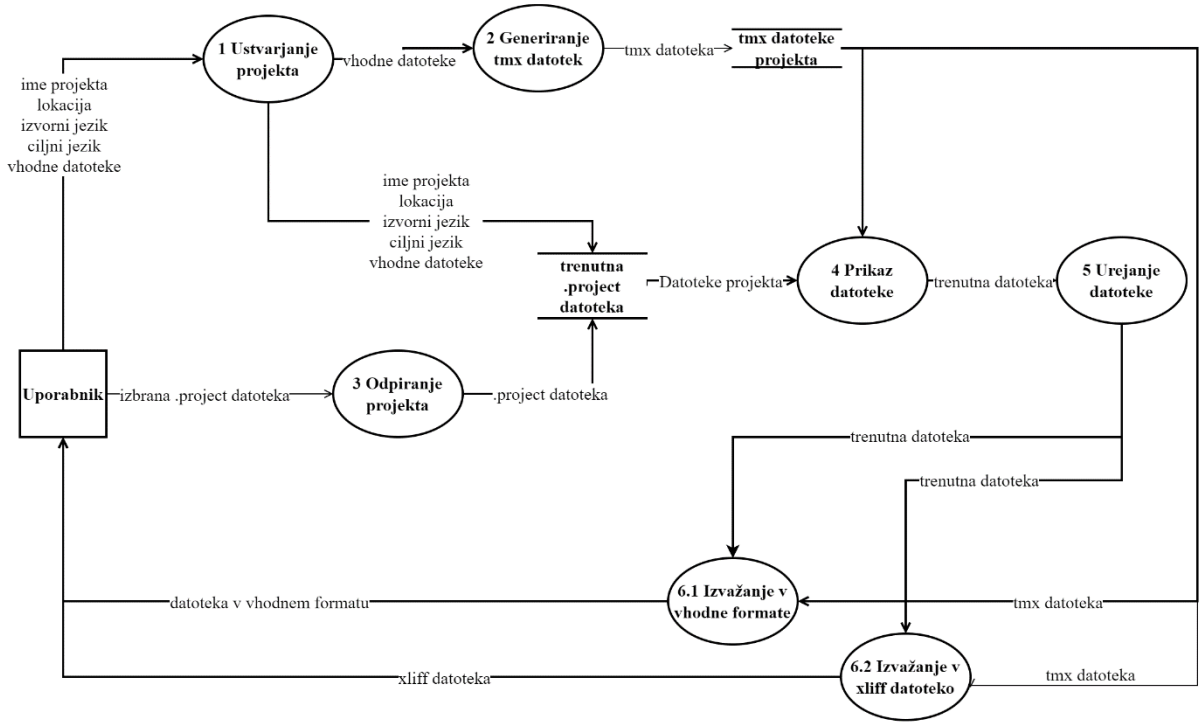
```
public class FileImporterTest {
  private FileImporter fileImporter;
  @BeforeEach
  public void setUp() {
    fileImporter = new FileImporter();
  }
  @Test
  public void testParseTxtFile() throws IOException {
    URL testFileUrl =
    getClass().getClassLoader().getResource("fileimport/sloveniatext.txt");
    if (testFileUrl == null) {
      throw new FileNotFoundException("Test file not found");
    }
    List<Tu> tuList = fileImporter.parseTxtFile(
      new File(testFileUrl.getFile())
      );
    assertEquals(
        "SloveniaTXT, officially the Republic of Slovenia, is a country in
      Central Europe.",
        tuList.get(0).getTuv().get(0).getSeg());
    assertEquals(
        "It is bordered by Italy to the west, Austria to the north, Hungary
       to the northeast, Croatia to the southeast, and the Adriatic Sea to
       the southwest.",
        tuList.get(1).getTuv().get(0).getSeg());
    assertEquals(
        "Slovenia is mostly mountainous and forested, covers 20,271 square
      kilometres (7,827 sq mi), and has a population of 2.1 million
      (2,108,708 people).",
        tuList.get(2).getTuv().get(0).getSeg());
  }
}
```

Po končani izdelavi programa pa sem le tega testiral z večjim številom datotek, za katera sem besedilo pridobil iz strani Wikipedie o Sloveniji (27), pri čemer sem ugotovil nekaj manjših napak, kot so napačno zapisovanje jezika v tmx datoteko, ki sem jih tudi popravil. Dodatno sem testiral tudi generirane tmx in xliff datoteke, za katere sem ugotovil, da so skladne s standardom

3 UPORABNIŠKI VMESNIK IN IZVORNA KODA

Program se sestavljen iz treh različnih scen (angl. Scene). Scena v JavaFX pomeni vsebino nekega odra (angl. Stage). Oder je najvišji okvir prikazane aplikacije in se v moji aplikaciji ne spreminja, saj ne potrebujem funkcionalnosti večih oken, razen v primeru pojavnega dialoga za izbiro datotek oziroma lokacije, za kar pa ustvarim nov oder.

Opis delovanja mojega orodja prikazuje tudi spodnji diagram toka podatkov po sledeh katerega bom ob slikah uporabniškega pogleda in izsekov iz programske kode orodja.



Slika 2 Diagram toka podatkov pri uporabi orodja

3.1 PREDPRIPRAVA PROGRAMA

Orodje za računalniško podprto prevajanje pred prvim prikazom okna zažene dva razreda. Prvi razred, ki se zažene je `OzrppLauncher`, ki predstavlja zaganjalni razred, ki je potreben zaradi pravilnega oblikovanja jar in kasneje exe datoteke programa. Kot lahko vidimo na spodnjem odseku programa je koda tega programa precej preprosta, saj zgolj zažene naslednji razred `Ozrpp`.

OzrppLauncher.java

```
public class OzrppLauncher {
  public static void main(String[] args){
    Ozrpp.main(args);
  }
}
```

Naslednji razred, ki se zažene je `Ozrpp`. Ta razred razširja JavaFX razred `Application` in v metodi `start()` ustvari glavni Stage aplikacije ter ustvari sceno iz `launcherScene.fxml`, katero na koncu tudi nastavi kot novo vsebino okna. Poleg tega nastavi tudi ime in ikono okna.

Razred `Ozrpp` vsebuje tudi dve statični spremenljivki. To sta scene in `projectProperties`. Prva ima shranjuje trenutno sceno programa, druga pa inicializira nov razred `ProjectProperties`, v katerega se bodo kasneje shranile lastnosti projekta, kot so ime, lokacija, vhodne datoteke ter izhodiščni in ciljni jezik.

Razred vsebuje tudi štiri pomožne metode in sicer `getStageM()` in `getStageA()`, ki se v drugih razredih uporabljata za pridobivanje trenutne scene iz dogodka povezanega z klikom oziroma akcijo. Drugi dve metodi `setRoot()` in `loadFXML()` pa sta pomožni metodi za nalaganje in prikaz fxml datotek.

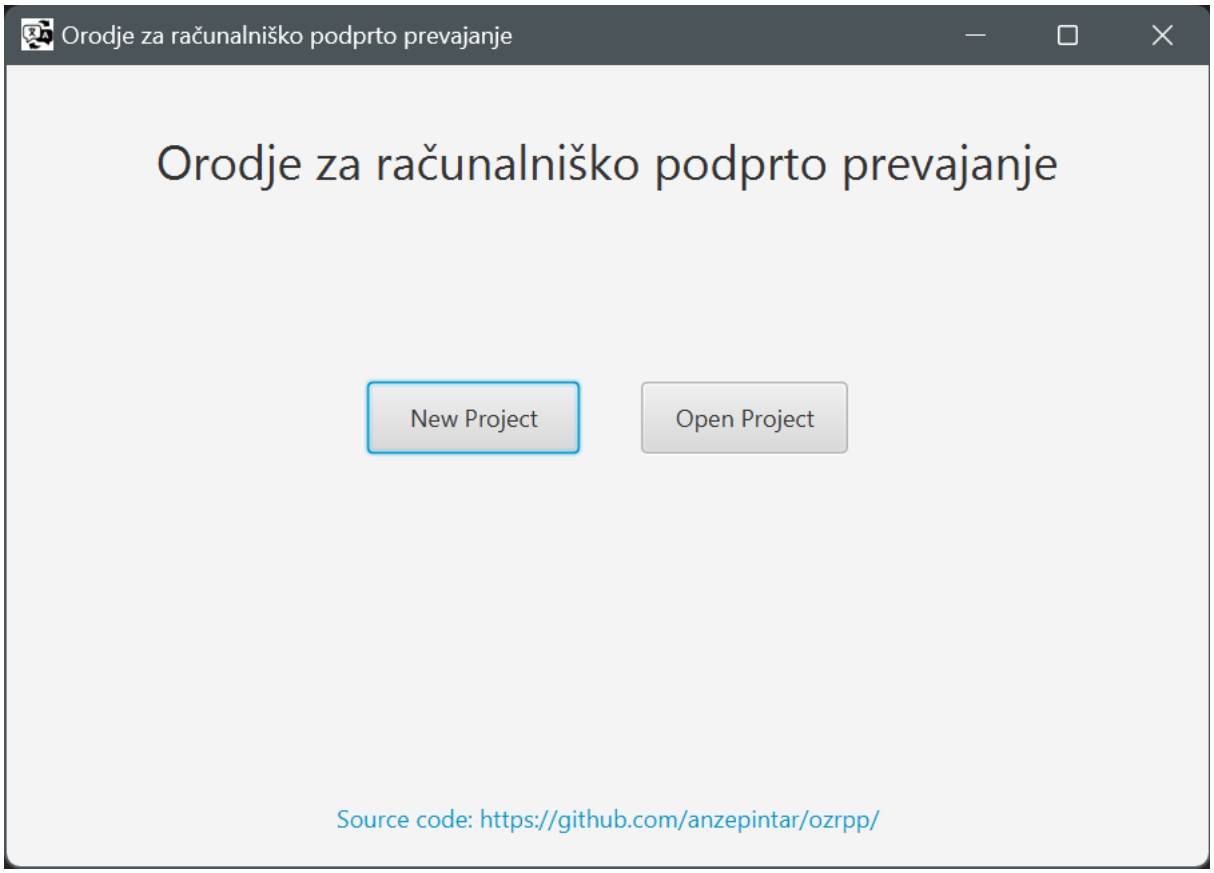
Ozpp.java

```
public class Ozrpp extends Application {

  public static ProjectProperites projectProperites = new ProjectProperites();
  public static Scene scene;
  public static void setRoot(String fxml) throws IOException {
    scene.setRoot(loadFxml(fxml));
  }

  public static Parent loadFxml(String fxml) throws IOException {
    FXMLLoader fxmlLoader = new FXMLLoader(Ozrpp.class.getResource(fxml));
    return fxmlLoader.load();
  }

  public static Stage getStageM(MouseEvent event) {
    return (Stage) ((Node) event.getSource()).getScene().getWindow();
  }


  public static Stage getStageA(ActionEvent event) {
    return (Stage) ((Node) event.getSource()).getScene().getWindow();
  }

  public static void main(String[] args) {
    launch();
  }
```

```
  @Override
  public void start(Stage stage) throws Exception {
    scene = new Scene(loadFxml("/ui/launcherScene.fxml"));
    stage.setScene(scene);
    stage.setTitle("Orodje za računalniško podprto prevajanje");
    stage.getIcons().add(new Image(
      Ozrpp.class.getResource("/img/icon.png").openStream()
      ));
    stage.show();
  }
}
```

3.2 ZAČETNO OKNO

Ko se `Ozrpp` zaključi in za prikaz nastavi `launcherScene.fxml`, se uporabniku odpre naslednje, dokaj preprosto začetno okno, ki ga lahko vidimo na spodnji sliki (Slika 3). Okno ima štiri elemente in sicer naslovno besedilo, gumb New Project (slov. Nov projekt), gumb Open Project (slov. Odpri projekt) in hiperpovezavo do Git repozitorija z izvorno kodo programa.



Slika 3 Začetno okno programa

Osnovna fxml struktura tega okna je shranjena v `launcherScene.fxml`. Struktura te datoteke z odstranjenim oblikovanjem je prikazana spodaj. Kot lahko vidimo je za celotno datoteko nastavljen krmilnik (angl. Controller) `LauncherControler`, v katerem se nahajajo vse metode in vse lastnosti, ki so definirane v fxml datoteki. Prikaz je zgrajen iz glavnega `VBox` objekta, ki omogoča dodajanje elementov enega pod drugega. V tem elementu se nahajajo trije elementi in sicer Label, ki predstavlja naslovno besedilo, `HBox`, ki omogoča dodajanje elementov enega zraven drugega, kar sta v tem primeru gumba `Button`, ter `Hyperlink`, ki predstavlja povezavo do repozitorija z izvorno kodo.

launcherScene.fxml

```
<VBox fx:controller="com.anzepintar.ozrpp.controllers.LauncherController">
  <children>
      <Label text="Orodje za računalniško podprto prevajanje">
         <font>
            <Font size="24.0" />
         </font>
      </Label>
    <HBox>
      <children>
        <Button onMouseClicked="#newProjectDialog" text="New Project">
        </Button>
        <Button onMouseClicked="#openProjectDialog" text="Open Project">
        </Button>
      </children>
    </HBox>
      <Hyperlink onAction="#openRepo" text="Source code:... />
  </children>
</VBox>
```

Z začetnim oknom upravlja kontroler `LauncherController`, ki ima tri metode. Prva metoda je `newProjectDialog()`, ki odpre okno za izbiro lokacije novega projekta in nato kot sceno nastavi `launcherScene.fxml`, na kateri lahko uporabnik vnese podatke o projektu.

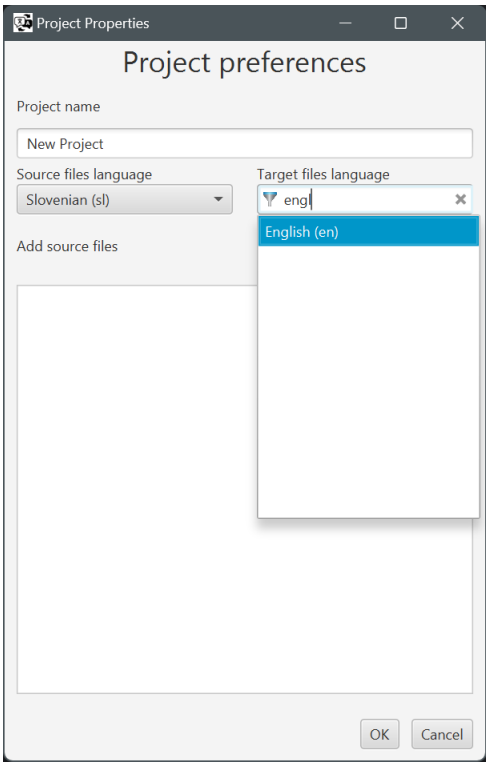
Druga metoda je `openProjectDialog()`, ki odpre okno za izbiro .project datoteke, iz katere se s pomočjo razreda `ProjectPropertiesManager` preberejo lastnosti projekta ter nato prikaže `editorScene.fxml`. Zadnja metoda pa s pomočjo knjižnice `java.awt.Desktop` odpre spletno stran z izvorno kodo.

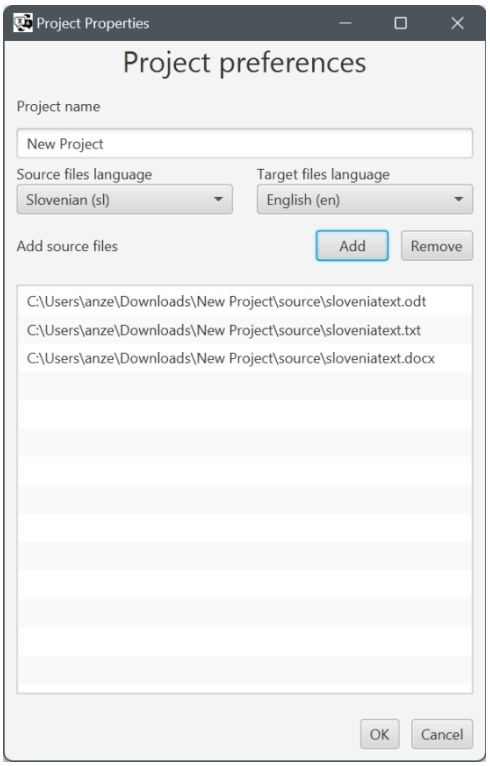
LauncherController.java

```
public class LauncherController {

  @FXML
  private void newProjectDialog(MouseEvent event) throws IOException {
    /koda ki se izvede ob pritisku na gumb*
     */
  }
  @FXML
  private void openProjectDialog(MouseEvent event) throws IOException,
JAXBException {
    /koda ki se izvede ob pritisku na gumb*
     */
  }
  @FXML
  private void openRepo(ActionEvent event) throws URISyntaxException,
IOException {
    /koda ki se izvede ob pritisku na gumb*
     */
  }
}
```

3.3 OKNO ZA USTVARJANJE PROJEKTA

Če uporabnik na začetnem oknu pritisne na gumb New Project in izbere primerno lokacijo novega projekta, se mu odpre okno za ustvarjanje projekta. To okno vsebuje naslovno besedilo, polje za vnos imena projekta, dva izbirna menija z iskanjem za vnos originalnega in željenega jezika, dva gumba za dodajane in odstranjevanje originalnih datotek, dva gumba za ustvarjanje in preklic ustvarjanja projekta ter seznam na katerem se prikažejo izbrane datoteke, kar lahko vidimo na spodnjih slikah (Slika 4, Slika 5).





Slika 4 Okno za ustvarjanje projekta - po koncu izbiranja datotek (levo)

Slika 5 Okno za ustvarjanje projekta - izbira jezika (desno)

To okno je zgrajeno iz fxml datoteke `projectPropertiesScene.fxml`, ki ima definirane vse prej omenjene elemente, izmed katerih bom v nadaljevanju izpostavil pomembnejše.

Prvi zapis ustvari izbirni meni, po katerem lahko iščemo. V tem primeru vsebino le tega nastavi metoda `getLanguageCodesAndNames()` v razredu `ProjectPropertiesController`, ki iz csv datoteke prebere imena in kratice jezikov

ter jih nato shrani v objekt `Map<String, String>`, tako da uporabnik vidi prvi niz, na primer »Slovenščina (sl)«, v programu pa se uporabi drugi niz, ki vsebuje le kratico jezika, v tem primeru »sl«.

projectPropertiesScene.fxml

```
<SearchableComboBox fx:id="sourceLangSelector" editable="true" />
<!--
-->
<ListView fx:id="fileList" HBox.hgrow="ALWAYS" />
```

Metoda getLanguagesCodesAndNames()

```
public static Map<String, String> getLanguageCodesAndNames()
 throws IOException {
  Map<String, String> languages = new HashMap<>();
  String[] parts;
  BufferedReader br = new BufferedReader(
    new FileReader("src/main/resources/langAndCodes.csv")
  );
  String line;
  while ((line = br.readLine()) != null) {
    parts = line.split(";");
    String name = String.format("%s (%s)", parts[0], parts[1]);
    String code = parts[1];
    languages.put(name, code);
  }
  return languages;
}
```

Metoda, ki nastavi vrednosti na meni za izbiro jezika

```
sourceLangSelector.setItems(
  new SortedList<String>(langOptions, Collator.getInstance())
);
```

Drugi fxml zapis pa ustvari seznam, na katerega v kontrolerju nastavi datoteke, ki jih uporabnik doda v projekt. Iz seznama je mogoče tudi izbrisati datoteke, tako da uporabnik označi datoteko in pritisne gumb Remove. To omogočata dve metodi v kontrolerju in sicer `addFile()` in `removeFile()`, ki ju sprožita pritiska na primeren gumb.

V pred metodo `addFile()` je definiran oznaka `@FXML`, kar pomeni, da se metoda uporablja kot metoda za obdelavo dogodkov v grafičnem vmesniku. Nato se ustvari objekt razreda `FileChooser`, ki omogoča izbiro datotek iz sistema. Nastavijo se naslov

in izhodiščna mapa. Sledi definicija filtrov za vrste datotek, ki jih uporabnik lahko izbere. V tem primeru so to datoteke s končnicami .txt, .docx in .odt. Nato se filter dodaja v seznam filtrov v objektu `FileChooser`. Klic metode `showOpenMultipleDialog()` prikaže pogovorno okno za izbiro datotek, ki uporabniku omogoča izbiro več datotek hkrati. Če uporabnik izbere datoteke in klikne na gumb Add, se datoteke dodajo v seznam, ki je predstavljen s seznamom `fileList`.

Metoda addFile()

```
@FXML
private void addFile() {
  FileChooser fileChooser = new FileChooser();
  fileChooser.setTitle("Select Files");
  fileChooser.setInitialDirectory(Ozrpp.projectProperites.getProjectRoot());
  FileChooser.ExtensionFilter filter = new FileChooser.ExtensionFilter(
      "Text Files (*.txt), Word Files (*.docx),
       OpenDocument Text Files(*.odt)",
       "*.txt","*.docx", "*.odt");
  fileChooser.getExtensionFilters().addAll(filter);
  List<File> selectedFiles = fileChooser.showOpenMultipleDialog(new Stage());
  if (selectedFiles != null) {
    files.addAll(selectedFiles);
    fileList.setItems(files);
  }
}
```

Druga metoda je `removeFile()`, ki sprejme dogodek `ActionEvent`, ki se sproži, ko uporabnik klikne na gumb za odstranjevanje datoteke v grafičnem vmesniku. Nato se uporabi metodo `getSelectionModel()` na objektu `fileList`, da se pridobi model izbire elementov iz seznama. Metoda `getSelectedItems()` nato vrne seznam izbranih elementov, ki so tipa `ObservableList<File>.` Če seznam izbranih datotek ni prazen, se iz seznama datotek, ki so bile prej dodane s pomočjo druge metode, odstranijo izbrane datoteke s pomočjo metode `removeAll()` na objektu `files`.

Metoda removeFile()

```
@FXML
private void removeFile(ActionEvent event) {
  ObservableList<File> selectedFiles =
fileList.getSelectionModel().getSelectedItems();
  if (!selectedFiles.isEmpty()) {
    files.removeAll(selectedFiles);
  }
}
```

3.3.1 Generiranje tmx datotek

Generiranje tmx datotek poteka v več fazah. Najprej metode `parseTxtFile()`, `parseDocxFile()` in `parseOdtFile()` iz razreda `FileImporter` iz izvornih datotek pridobijo njihove segmente – povedi. Primer za metodo `parseDocxFile()` lahko vidimo spodaj. Metoda kot argument prejme datoteko, ki je formata, ki jo podpira, vrne pa seznam `Tu` elementov, ki so del tmx formata zapisa. V tem primeru je ustvarjen nov objekt razreda `XWPFDocument`, ki kot argument prejme datoteko, ki je bila prej spremenjena v `FileInputStream`. Nato se v zanki za vsak objekt `XWPFParagraph` ustvarita dva nova objekta, ki imata vsebino `paragraph.getText().`

Zelo podobno je tudi pridobivanje povedi iz txt in odt datotek, le da pri txt datotekah programsko razdelimo niz besedila, pri odt datotekah pa uporabimo razred `OdfTextDocument.`

parseDocxFile()

```
public List<Tu> parseDocxFile(File file) throws IOException {
  List<Tu> tuList = new ArrayList<>();

  FileInputStream fileInputStream = new FileInputStream(file);
  XWPFDocument document = new XWPFDocument(fileInputStream);

  for (XWPFParagraph paragraph : document.getParagraphs()) {
    if (paragraph.isEmpty()) {
      continue;
    }
    Tu tu = new Tu();
    Tuv tuv1 = createSourceSegment(paragraph.getText());
    Tuv tuv2 = createTargetSegment();
    tu.getTuv().add(tuv1);
    tu.getTuv().add(tuv2);
    tuList.add(tu);
  }
  document.close();
  fileInputStream.close();
  return tuList;
}
```

Generiranje tmx datotek se nadaljuje v metodi `importToTmx()` v kateri se s pomočjo razreda `JAXBContent` in razredov, ki smo jih generirali iz tmx sheme nastavijo značke tmx dokumenta in njihove lastnosti, kot so jezik dokumenta in ime orodja, ki je ustvarilo dokument. Polega tega se v tej metodi tudi nastavi prej ustvarjene `Tu` elemente. Na koncu metode se ustvarjen razred `Tmx` preko uporabe razreda `Marshaller` zapiše v tmx

datoteko, ki je poimenovana kot `ime_datoteke.prejsnji_format.tmx`, katere primer lahko vidimo v Prilogi 3.

3.3.2 Shranjevanje datotek in lastnosti projekta

Pri koncu ustvarjanja projekta se ustvari mapa z vsebino projekta. V njo se v podmapo `source/` kopirajo prej izbrane datoteke, v mapo `tmx/` pa se shranijo ustvarjene tmx datoteke. Poleg tega se ustvari tudi mapa `target/`, v katero bodo shranjene prevedene datoteke kar lahko vidimo na spodnjem primeru.

Primer ustvarjenih datotek projekta

```
ime_projekta/
├─ ime_projekta.project
├─ source/
│   ├─ sloveniatext.docx
│   ├─ sloveniatext.odt
│   └─ sloveniatext.txt
├─ target/
└─ tmx/
    ├─ sloveniatext.docx.tmx
    ├─ sloveniatext.odt.tmx
    └─ sloveniatext.txt.tmx
```

Poleg vseh map pa se ustvari tudi datoteka `ime_projekta.project` v katero se shranijo lastnosti, kot so ime projekta in lokacija projekta, izvorni in ciljni jezik ter seznam lokacij kopiranih datotek, primer česar lahko vidimo na spodnjem primeru.

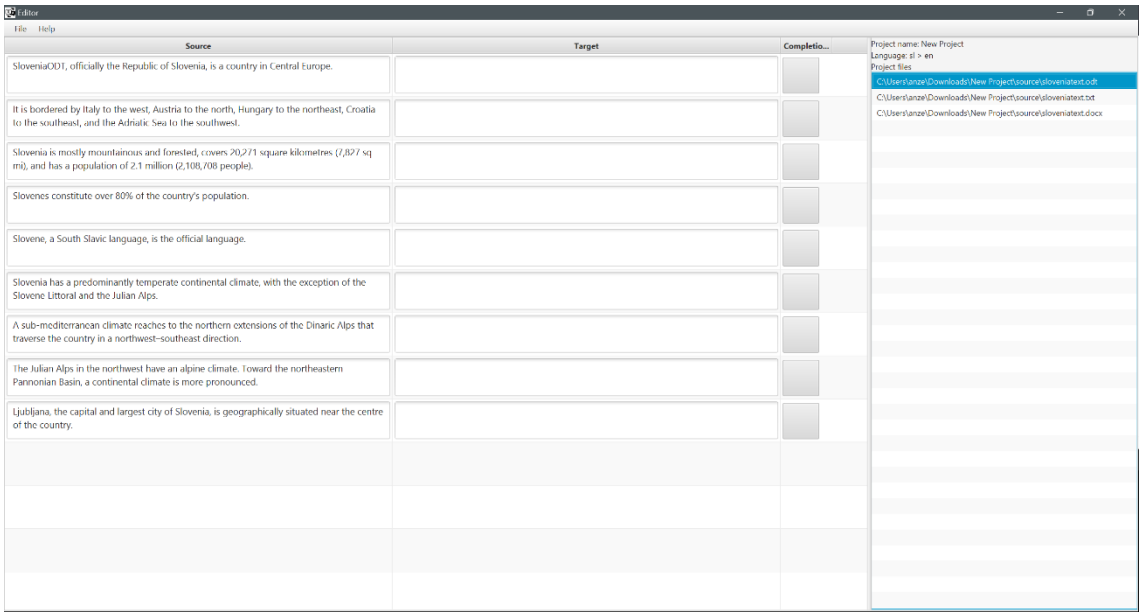
Ime_projekta.project

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<projectProperites>
  <projectRoot>C:\Users\anze\Downloads\New Project</projectRoot>
  <projectName>New Project</projectName>
  <sourceLang>sl</sourceLang>
  <targetLang>en</targetLang>
  <sourceFiles>
    <file>C:\Users\anze\Downloads\New Project\source\sloveniatext.odt</file>
    <file>C:\Users\anze\Downloads\New Project\source\sloveniatext.txt</file>
    <file>C:\Users\anze\Downloads\New Project\source\sloveniatext.docx</file>
  </sourceFiles>
</projectProperites>
```

Ko se okno pravilno shrani, program nastavi sceno na `editorScene.fxml` ter s tem odpre okno za urejanje datotek.

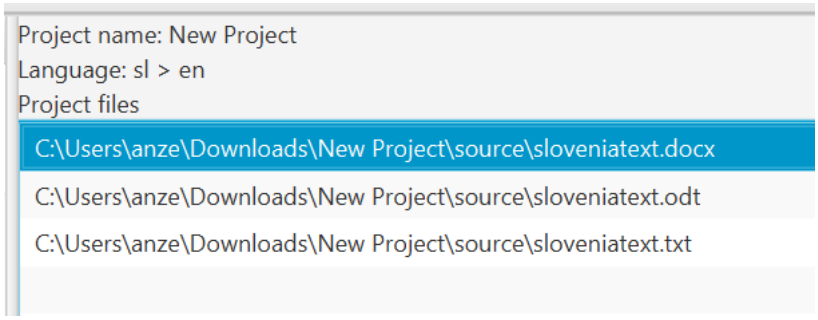
3.4 OKNO ZA UREJANJE DATOTEK

Okno za urejanje datotek je sestavljeno iz dveh objektov `SplitPane` in enega objekta `Menu`, ki delijo sceno na več delov, kar lahko vidimo na spodnji sliki (Slika 6). V levem delu okna se nahaja tabela, ki ima tri stolpce. V prvem stolpcu se nahajajo stavki iz izvirne datoteke, v drugem se nahajajo stavki, ki jih bo uporabnik vnesel kot prevode in v tretjem pa potrditveno polje `CheckBox`, ki označuje ali je segment že preveden.



Slika 6 Okno za urejanje datotek ob odprtju novega projekta

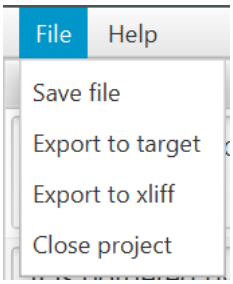
Na levi strani okna se na vrhu nahaja nekaj splošnih podatkov o projektu (Slika 7), pod njimi pa je seznam odprtih datotek. Če uporabnik klikne na datoteko, ki je tu prikazana, se le ta odpre.

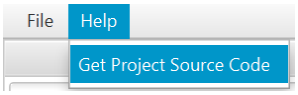


Slika 7 Desna stran okna za urejanje

Na vrhu okna je meni z dvema gumboma, ki ob pritisku pokažeta podmeni. Prvi gumb File pokaže možnost Save file, ki shrani delno prevedeno tmx datoteko, Export to target, ki shrani vse prevedene segmente dokumenta v enako datoteko, kot je bila izvorna, Export

to xliff pretvori tmx datoteko v xliff, Close project pa zapre projekt in uporabniku prikaže začetno okno (Slika 8). Drugi gumb menija pa prikaže povezavo za pomoč – povezavo do izvorne kode programa (Slika 9).

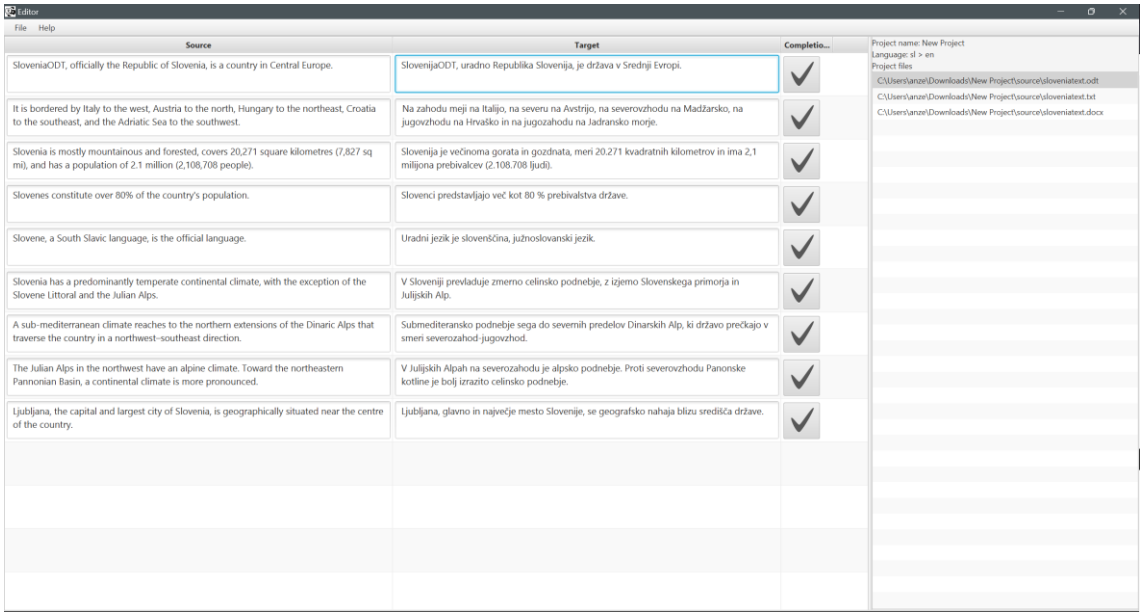




Slika 8 Gumb File prikaže štiri možnosti (levo)

Slika 9 Gumb Help prikaže povezavo (desno)

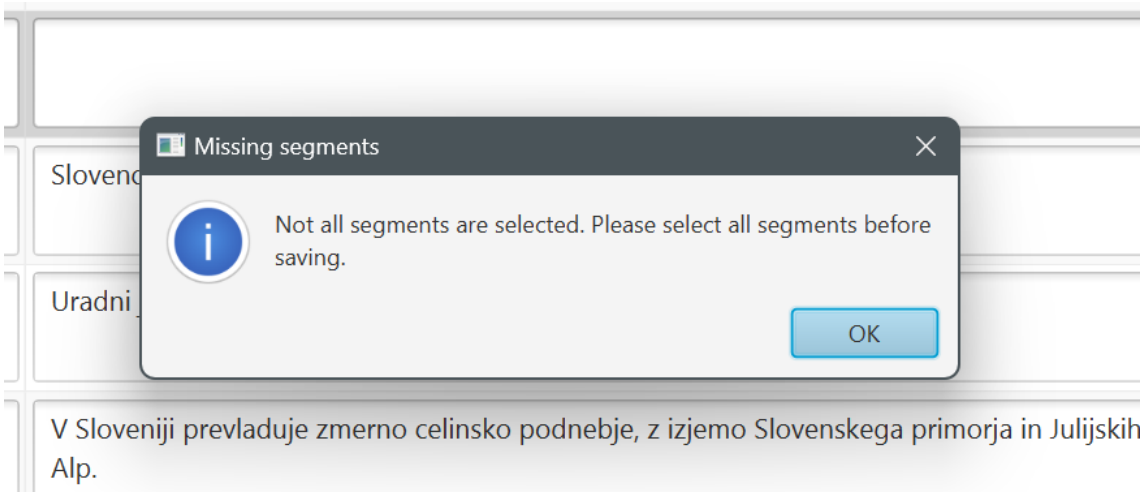
V tem oknu lahko uporabnik ureja vse dokumente projekta in označi status prevoda, tako da pritisne na sivi kvadrat. Ko uporabnik prevede vse segmente lahko te datoteke izvozi v xliff format oziroma



Slika 10 Okno za urejanje datotek po prevajanju segmentov

3.4.1 Izvažanje datotek v format vhodne datoteke

Ko uporabnik s pritiskom na gumb Export to xliff sproži izvoz datotek, program najprej preveri, če so vsi segmenti označeni kot prevedeni in uporabnika opozori, če kakšen izmed elementov ni preveden (Slika 11).



Slika 11 Primer opozorila pri neoznačenih segmentih

Če je uporabnik vse segmente označil za prevedene se izvede metoda `saveToTarget()`, ki je del razreda `FileExporter`. Ta metoda iz imena tmx datoteke ugotovi primeren format datoteke za izvoz in jo izvozi. Spodnji program prikazuje privatno metodo tega razreda, `saveAsOdt()`. Ta metoda kot argument prejme ime ciljne datoteke in seznam vseh prevedenih povedi. Nato s pomočjo razreda `OdfTextDocument` iz vsake povedi ustvari nov element odt datoteke in jo na koncu shrani v mapo `target/`

saveAsOdt()

```
private static void saveAsOdt(String fileName, List<String> strings)
throws Exception {
  OdfTextDocument doc = OdfTextDocument.newTextDocument();
  for (String str : strings) {
    TextPElement paragraph = doc.newParagraph();
    paragraph.setTextContent(str);
    doc.getContentRoot().appendChild(paragraph);
  }
  File outputFile = new File(
      Ozrpp.projectProperites.getProjectRoot().getAbsolutePath() + "/target/"
       + fileName
          + ".odt");
  outputFile.getParentFile().mkdirs();
  doc.save(outputFile);
  doc.close();
}
```

3.4.2 Pretvarjanje tmx datoteke v xliff datoteko

Če želi uporabnik shraniti svoje delo v xliff datoteko lahko to stori s pritiskom na gumb Export to xliff, ki sproži izvajanje metode `exportToXliff()`, ki ponovno preveri stanje prevoda vseh segmentov in uporabniku javi, če zazna, da kateri izmed segmentov ni označen kot preveden (Slika 11). Ta metoda nato požene metodo `saveTmxToXliff()`, ki je del razreda `TmxToXliffSaver`. Ta metoda s pomočjo razreda `DocumentBuilderFactory` ustvari nov xliff dokument in mu doda značke in lastnosti, kot so različica, vhodni in ciljni jezik, ter mu na znački »source« in «target«, s pomočjo metod getTmxSourceStrings() in getTmxTargetStrings(), nastavi segmente v vhodnem in izhodnem jeziku. Končno datoteko lahko vidite v prilogi (Priloga 4).

4 ZAKLJUČEK

Med programiranjem orodja za računalniško podprto prevajanje sem se naučil ogromno stvari o datotečnem formatu xml, na katerem temeljita tudi tmx in xliff format, ter o uporabi različnih zunanjih nestandardnih java knjižnic. Poleg tega sem spoznal programiranje, ki temelji na testih programiranja (angl. Test-driven development), saj sem se naučil pisati teste za metode, ki sem jih napisal.

Program ni primerljiv s profesionalnimi orodji za računalniško podprto prevajanje, a vsebuje vse osnovne funkcionalnosti, ki jih potrebuje tak program. Moj program bi lahko nadgradil s sprotnim analiziranjem vsebine segmentov, predlaganjem prevodov, možnostjo avtomatskega strojnega prevoda ter podporo za več vhodnih in izhodnih formatov datotek ter datotek, v katerih se shranjujejo podatki o prevajanju.

Pri izdelavi programa sem spoznal tudi, da JavaFX ni najbolj primerna knjižnica za ustvarjanje grafičnega uporabniškega vmesnika, čeprav je novejša od drugih pogosto uporabljenih knjižnic za grafični prikaz, kot sta knjižnici Java Swing in AWT (28), saj se v literaturi za JavaFX pojavi bistveno manj primerov uporabe kot za drugi dve knjižnici.

Menim, da sem s seminarsko nalogo dosegel svoje cilje ter da sem sem pri tem pridobil veščine, ki mi bodo koristile pri nadaljnjem šolanju.

5 ZAHVALA

Iskreno bi se rad zahvalil svojemu mentorju dr. Albertu Zorku, ki me je tekom šolanja naučil rednega dela, izdelave seminarskih nalog in programiranja. Prav tako se zahvaljujem profesorjem Gregorju Medetu, Simonu Vovku, Tomažu Ferbežarju in Miletu Božiću, ki so me skozi leta poučevali računalništvo.

Rad bi se zahvalil tudi svojemu bratu, ki mi je svetoval pri izdelavi aplikacije ter predlagal uporabo sprotnega testiranja pri razvijanju aplikacije ter sošolcem za njihove nasvete, pripombe in ideje pri pisanju seminarske naloge.

Za konec pa bi se rad zahvalil tudi svoji družini, ki me je čez vsa 4 leta spodbujala pri mojem pridobivanju znanja.

Anže Pintar

6 VIRI IN LITERATURA

1. **openjfx.io.** JavaFX. *openjfx.io.* [Elektronski] [Navedeno: 14. april 2023.] https://openjfx.io/.

2. **Sharan, Kishori in Späth, Peter.** *Learn JavaFX 17.* s.l. : Apress, 2022. 9781484278482.

3. **Lowe, Dough.** *JavaFX For Dummies.* s.l. : For Dummies, 2014. 9781118417430.

4. **www.gala-global.org.** TMX 1.4b Specification. *www.gala-global.org.* [Elektronski] 26. april 2005. [Navedeno: 15. april 2023.] https://www.gala-global.org/tmx-14b.

5. **docs.oasis-open.org.** XLIFF Version 2.0. *docs.oasis-open.org.* [Elektronski] [Navedeno: 17. april 2023.] http://docs.oasis-open.org/xliff/xliff-core/v2.0/xliff-core-v2.0.html.

6. **Dias, Kevin.** Unlocking the Black Box of Translation Memory Files. *www.tm-town.com.* [Elektronski] [Navedeno: 17. april 2023.] https://www.tm-town.com/blog/unlocking-black-box-translation-memory-files.

7. **abhinayagarwal.** Controlsfx: High quality UI controls to complement the core JavaFX distribution . *www.github.com.* [Elektronski] [Navedeno: 3. april 2023.] https://github.com/controlsfx/controlsfx/blame/master/readme.md.

8. **Bechtold, Stefan, in drugi.** JUnit 5 User Guide. *www.junit.org.* [Elektronski] [Navedeno: 6. april 2023.] https://junit.org/junit5/docs/current/user-guide/.

9. **www.softwaretestinghelp.com.** JUnit Tests: How To Write JUnit Test Cases With Examples. *www.softwaretestinghelp.com.* [Elektronski] 13. marec 2023. [Navedeno: 6. april 2023.] https://www.softwaretestinghelp.com/junit-tests-examples/.

10. **www.odftoolkit.org.** ODF Toolkit. *www.odftoolkit.org.* [Elektronski] [Navedeno: 7. april 2023.] http://opendocument.xml.org/.

11. **www.opendocument.xml.org.** Open Document | Online Community for the OpenDocument OASIS Standard. *www.opendocument.xml.org.* [Elektronski] [Navedeno: 7. april 2023.] http://opendocument.xml.org/.

12. **Schubert, Svante.** New ODF Toolkit from TDF (The Document Foundation). *blog.documentfoundation.org.* [Elektronski] 2022. [Navedeno: 8. april 2023.]

file:///C:/Users/anze/Downloads/2022-FOSDEM-News-from-the-ODF_Toolkit__with_Notes.pdf.

13. **poi.apache.org.** Apache POI - the Java API for Microsoft Documents. *poi.apache.org.* [Elektronski] 11. februar 2023. [Navedeno: 9. april 2023.] https://poi.apache.org/.

14. **www.javatpoint.com.** Apache POI Features - javapoint. *www.javatpoint.com.* [Elektronski] [Navedeno: 9. april 2023.] https://www.javatpoint.com/apache-poi-features.

15. **www.jakarta.ee.** Why Jakarta EE? | The Eclipse Foundation. *www.jakarta.ee.* [Elektronski] [Navedeno: 10. april 2023.] https://jakarta.ee/about/why-jakarta-ee/.

16. **jakartaee-platform-dev.** Jakarta EE Platform. *www.jakarta.ee.* [Elektronski] 26. avgust 2019. [Navedeno: 9. april 2023.] https://jakarta.ee/specifications/platform/8/platform-spec-8.html.

17. **logging.apache.org.** Log4j - Overview. *logging.apache.org.* [Elektronski] https://logging.apache.org/log4j/2.x/manual/index.html.

18. **www.tutorialspoint.com.** log4j - Logging Levels. *www.tutorialspoint.com.* [Elektronski] [Navedeno: 6. april 2023.] https://www.tutorialspoint.com/log4j/log4j_logging_levels.htm.

19. **maven.apache.org.** Maven - Introduction. *maven.apache.org.* [Elektronski] [Navedeno: 7. april 2023.] https://maven.apache.org/what-is-maven.html.

20. **baeldung.** Apache Maven Tutorial | baeldung. *www.baeldung.com.* [Elektronski] 5. julij 2022. [Navedeno: 12. april 2023.] https://www.baeldung.com/maven.

21. **www.git-scm.com.** About - Git. *www.git-scm.com.* [Elektronski] [Navedeno: 6. april 2023.] https://git-scm.com/about.

22. **Chacon, Scott in Straub, Ben.** *Pro Git.* s.l. : Apress, 2023.

23. **www.oracle.com.** JavaFX Scene Builder Information. *www.oracle.com.* [Elektronski] [Navedeno: 5. april 2023.] https://www.oracle.com/java/technologies/javase/javafxscenebuilder-info.html.

24. **www.gluonhq.com.** Scene Builder. *www.gluonhq.com.* [Elektronski] [Navedeno: 5. april 2023.] https://gluonhq.com/products/scene-builder/.

25. **Oracle+.** xjc. *docs.oracle.com.* [Elektronski] [Navedeno: 15. april 2023.] https://docs.oracle.com/javase/8/docs/technotes/tools/unix/xjc.html.

26. **IBM Corporation.** xjc command for JAXB applications - IBM Documentation. *www.ibm.com.* [Elektronski] 13. februar 2023. [Navedeno: 15. april 2023.] https://www.ibm.com/docs/en/was-zos/8.5.5?topic=SS7K4U_8.5.5/com.ibm.websphere.base.doc/ae/rwbs_xjc.htm.

27. **wikipedia.og.** Slovenia. *www.wikipedia.org.* [Elektronski] [Navedeno: 20. december 2022.] https://en.wikipedia.org/wiki/Slovenia.

28. **Singh, Japkeerat.** What is JavaFX and how is it different from Swing and AWT? *www.medium.com.* [Elektronski] 24. september 2018. [Navedeno: 18. april 2023.] https://medium.com/@japkeerat21/what-is-javafx-and-how-is-it-different-from-swing-and-awt-54de995e4869.

29. **Grobmeier, Christian.** The new log4j 2.0. *www.grobmeier.solutions.* [Elektronski] 5. december 2012. [Navedeno: 6. april 2023.] https://grobmeier.solutions/the-new-log4j-2-0-05122012.html.

7 PRILOGE

Priloga 1: Povezava do izvorne kode programa

https://github.com/anzepintar/ozrpp

Priloga 2: Struktura datotek programa

```
pom.xml
src/
├── main/
│   ├── java/
│   │   └── com/
│   │       ├── anzepintar/
│   │       │   └── ozrpp/
│   │       │       ├── Ozrpp.java
│   │       │       ├── OzrppLauncher.java
│   │       │       ├── controlers/
│   │       │       │   ├── EditorController.java
│   │       │       │   ├── LauncherController.java
│   │       │       │   └── ProjectPropertiesController.java
│   │       │       ├── customcotrols/
│   │       │       │   ├── AutoResizableTextArea.java
│   │       │       │   └── TranslationCheckBox.java
│   │       │       ├── editordata/
│   │       │       │   └── TableRow.java
│   │       │       ├── fileexport/
│   │       │       │   ├── FileExporter.java
│   │       │       │   ├── TmxSaver.java
│   │       │       │   └── TmxToXliffSaver.java
│   │       │       ├── fileimport/
│   │       │       │   ├── FileImporter.java
│   │       │       │   └── TmxLoader.java
│   │       │       └── generatedclasses/
│   │       │           ├── tmxgenerated/
│   │       │           │   ├── Body.java
│   │       │           │   ├── Bpt.java
│   │       │           │   ├── Ept.java
│   │       │           │   ├── Header.java
│   │       │           │   ├── Hi.java
│   │       │           │   ├── It.java
│   │       │           │   ├── Map.java
│   │       │           │   ├── Note.java
│   │       │           │   ├── ObjectFactory.java
│   │       │           │   ├── Ph.java
│   │       │           │   ├── Prop.java
│   │       │           │   ├── Sub.java
│   │       │           │   ├── Tmx.java
│   │       │           │   ├── Tu.java
│   │       │           │   ├── Tuv.java
│   │       │           │   ├── Ude.java
│   │       │           │   └── Ut.java
│   │       │           └── projectproperties/
│   │       │               ├── ProjectProperites.java
│   │       │               └── ProjectPropertiesManager.java
```

```
│   │       └── module-info.java
│   └── resources/
│       ├── langAndCodes.csv
│       ├── img/
│       │   ├── icon.ico
│       │   ├── icon.png
│       │   └── icon.svg
│       ├── schemas/
│       │   ├── schemasHelp.md
│       │   ├── tmx/
│       │   │   └── tmx14.dtd
│       └── ui/
│           ├── editorScene.fxml
│           ├── launcherScene.fxml
│           └── projectPropertiesScene.fxml
└── test/
    ├── java/
    │   ├── fileexport/
    │   │   └── FileExporterTest.java
    │   ├── fileimport/
    │   │   ├── FileImporterTest.java
    │   │   └── TmxLoaderTest.java
    │   └── projectproperties/
    │       └── ProjectPropertiesTest.java
    └── resources/
        ├── fileexport/
        │   ├── sloveniatext.docx.tmx
        │   ├── sloveniatext.odt.tmx
        │   └── sloveniatext.txt.tmx
        └── fileimport/
            ├── sloveniatext.docx
            ├── sloveniatext.odt
            ├── sloveniatext.tmx
            └── sloveniatext.txt
```

Priloga 3: Primer generirane tmx datoteke

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<tmx version="1.4">
  <tu>
    <tuv xml:lang="en">
      <seg>SloveniaODT, officially the Republic of Slovenia, is a country in
       Central Europe.</seg>
    </tuv>
    <tuv xml:lang="sl">
      <seg>SlovenijaODT, uradno Republika Slovenija, je država v Srednji
       Evropi.</seg>
    </tuv>
  </tu>
  <tu>
    <tuv xml:lang="en">
      <seg>It is bordered by Italy to the west, Austria to the north, Hungary
       to the northeast, Croatia to the southeast, and the Adriatic Sea to
       the southwest.</seg>
    </tuv>
    <tuv xml:lang="sl">
      <seg>Na zahodu meji na Italijo, na severu na Avstrijo, na severovzhodu
       na Madžarsko, na jugovzhodu na Hrvaško in na jugozahodu na Jadransko
       morje.</seg>
    </tuv>
  </tu>
  <tu>
    <tuv xml:lang="en">
      <seg>Slovenia is mostly mountainous and forested, covers 20,271 square
       kilometres (7,827 sq mi), and has a population of 2.1 million
       (2,108,708 people).</seg>
    </tuv>
    <tuv xml:lang="sl">
      <seg>Slovenija je večinoma gorata in gozdnata, meri 20.271 kvadratnih
       kilometrov in ima 2,1 milijona prebivalcev (2.108.708 ljudi).</seg>
    </tuv>
  </tu>
  <tu>
    <tuv xml:lang="en">
      <seg>Slovenes constitute over 80% of the country's population.</seg>
    </tuv>
    <tuv xml:lang="sl">
      <seg>Slovenci predstavljajo več kot 80 % prebivalstva države.</seg>
    </tuv>
  </tu>
  <tu>
    <tuv xml:lang="en">
      <seg>Slovene, a South Slavic language, is the official language.</seg>
    </tuv>
    <tuv xml:lang="sl">
      <seg>Uradni jezik je slovenščina, južnoslovanski jezik.</seg>
    </tuv>
  </tu>
  <tu>
    <tuv xml:lang="en">
      <seg>Slovenia has a predominantly temperate continental climate, with
       the exception of the Slovene Littoral and the Julian Alps.</seg>
    </tuv>
```

```
    <tuv xml:lang="sl">
      <seg>V Sloveniji prevladuje zmerno celinsko podnebje, z izjemo
       Slovenskega primorja in Julijskih Alp.</seg>
    </tuv>
  </tu>
  <tu>
    <tuv xml:lang="en">
      <seg>A sub-mediterranean climate reaches to the northern extensions of
       the Dinaric Alps that traverse the country in a northwest–southeast
       direction.</seg>
    </tuv>
    <tuv xml:lang="sl">
      <seg>Submediteransko podnebje sega do severnih predelov Dinarskih Alp,
       ki državo prečkajo v smeri severozahod-jugovzhod.</seg>
    </tuv>
  </tu>
  <tu>
    <tuv xml:lang="en">
      <seg>The Julian Alps in the northwest have an alpine climate. Toward
       the northeastern Pannonian Basin, a continental climate is more
       pronounced.</seg>
    </tuv>
    <tuv xml:lang="sl">
      <seg>V Julijskih Alpah na severozahodu je alpsko podnebje. Proti
       severovzhodu Panonske kotline je bolj izrazito celinsko
       podnebje.</seg>
    </tuv>
  </tu>
  <tu>
    <tuv xml:lang="en">
      <seg>Ljubljana, the capital and largest city of Slovenia, is
       geographically situated near the centre of the country.</seg>
    </tuv>
    <tuv xml:lang="sl">
      <seg>Ljubljana, glavno in največje mesto Slovenije, se geografsko
nahaja blizu središča države.</seg>
    </tuv>
  </tu>
</tmx>
```

Priloga 4: Primer generirane xliff datoteke

```
<?xml version="1.0" encoding="UTF-8"?>
<xliff xmlns="urn:oasis:names:tc:xliff:document:1.2" version="1.2">
  <file datatype="plaintext" original="tmxFile" source-language="sl" target-
language="sl">
    <body>
      <trans-unit id="1">
        <source>SloveniaODT, officially the Republic of Slovenia, is a
       country in Central Europe.</source>
        <target>SlovenijaODT, uradno Republika Slovenija, je država v Srednji
       Evropi.</target>
       </trans-unit>
       <trans-unit id="2">
        <source>It is bordered by Italy to the west, Austria to the north,
      Hungary to the northeast, Croatia to the southeast, and the Adriatic Sea
       to the southwest.</source>
        <target>Na zahodu meji na Italijo, na severu na Avstrijo, na
      severovzhodu na Madžarsko, na jugovzhodu na Hrvaško in na jugozahodu na
       Jadransko morje.</target>
       </trans-unit>
       <trans-unit id="3">
        <source>Slovenia is mostly mountainous and forested, covers 20,271
      square kilometres (7,827 sq mi), and has a population of 2.1 million
      (2,108,708 people).</source>
        <target>Slovenija je večinoma gorata in gozdnata, meri 20.271
      kvadratnih kilometrov in ima 2,1 milijona prebivalcev (2.108.708
      ljudi).</target>
       </trans-unit>
       <trans-unit id="4">
        <source>Slovenes constitute over 80% of the country's
      population.</source>
        <target>Slovenci predstavljajo več kot 80 % prebivalstva
      države.</target>
       </trans-unit>
       <trans-unit id="5">
        <source>Slovene, a South Slavic language, is the official
      language.</source>
        <target>Uradni jezik je slovenščina, južnoslovanski jezik.</target>
       </trans-unit>
       <trans-unit id="6">
        <source>Slovenia has a predominantly temperate continental climate,
       with the exception of the Slovene Littoral and the Julian
      Alps.</source>
        <target>V Sloveniji prevladuje zmerno celinsko podnebje, z izjemo
       Slovenskega primorja in Julijskih Alp.</target>
       </trans-unit>
       <trans-unit id="7">
        <source>A sub-mediterranean climate reaches to the northern
      extensions of the Dinaric Alps that traverse the country in a northwest–
      southeast direction.</source>
        <target>Submediteransko podnebje sega do severnih predelov Dinarskih
       Alp, ki državo prečkajo v smeri severozahod-jugovzhod.</target>
       </trans-unit>
       <trans-unit id="8">
        <source>The Julian Alps in the northwest have an alpine climate.
       Toward the northeastern Pannonian Basin, a continental climate is more
       pronounced.</source>
        <target>V Julijskih Alpah na severozahodu je alpsko podnebje. Proti
```

```
      severovzhodu Panonske kotline je bolj izrazito celinsko
      podnebje.</target>
      </trans-unit>
      <trans-unit id="9">
        <source>Ljubljana, the capital and largest city of Slovenia, is
      geographically situated near the centre of the country.</source>
        <target>Ljubljana, glavno in največje mesto Slovenije, se geografsko
      nahaja blizu središča države.</target>
      </trans-unit>
    </body>
  </file>
</xliff>
```