Домашно бр. 1
Jena RDF API

А) Прашања

1. **Како се изразува еден запис (еден факт) во RDF моделот?**

Со дефинирање на тројка од субјект, предикат и објект. Секој од наведените има уникатно УРИ (најчесто УРЛ). После тројката ставаме точка за да дефинираме крај на фактот.
(Објектот може да прими константа вредност – литерал или друго УРИ – Ресурс).

2. **Кои различни синтакси за RDF моделот постојат? Изразете го следниот факт во неколку различни RDF синтакси: „ВБС се предава на ФИНКИ“. Користете го префиксот @prefix finki: <http://finki.ukim.mk/resource#> за URI вредностите на ентитетите и релацијата.**

Постојат многу различни синтакси за РДФ моделот. Дел од почесто користените се: Turtle, RDF/XML, RDFa, JSON-LD.

Turtle:

```
@prefix finki: <http://finki.ukim.mk/resource#> .
finki:VBS     finki:sePredavaNa    finki:FINKI .
```

RDF/XML:

```
<?xml version="1.0" encoding='utf-8'>
<rdf:RDF finki=" http://finki.ukim.mk/resource#">
<rdf:Description
          rdf:about=" http://finki.ukim.mk/resource#VBS">
          <finki:sePredavaNa finki:FINKI />
</rdf:Description>
</rdf:RDF>
```

RDFa:

```
<html xmlns:finki="http://finki.ukim.mk/resource#">
<body>
<div about="[finki:VBS]"> VBS se predava na FINKI.
     <span rel="[finki:sePredavaNa]"
          resource="[finki:FINKI]"></span>
</div>
</body>
</html>
```

JSON-LD:

```
{"@context": {
      "finki":" http://finki.ukim.mk/resource#"
      },
"@graph": [
      "@id": "http://finki.ukim.mk/resource#VBS",
      "finki:sePredavaNa":"finki:FINKI"
      ]
}
```

3. **За што се користи RDF Schema?**
   RDF Schema се користи за да се дефинира СЕМАНТИКАТА на РДФ, односно да се дефинира вокабуларот кој ќе го користиме.
   Преку неа опишуваме ресурси и релациите меѓу нив.

4. **Дефинирајте RDFS класи за „факултет“ и „предмет“, како и една релација која ги поврзува нив, „е предмет на“. Користете го префиксот од 2. за нивните URI вредности. Користете Turtle синтакса.**

   @prefix finki: <http://finki.ukim.mk/resource#> .
   @prefix rdfs: <https://www.w3.org/2000/01/rdf-schema#> .

   finki:Fakultet rdf:type rdfs:Class .
   finki:Predmet rdf:type rdfs:Class .
   finki:ePredmetNa rdf:type rdfs:Property .

Б) Практична задача

I. Креирање едноставен RDF граф

1. Креирајте нов Java проект во IDE по ваш избор. Вклучете ги во проектот сите .jar библиотеки од lib фолдерот од Jena. Jena преземете ја директно од Jena сајтот.

2. Во main() методот на главната класа од проектот, креирајте основен Jena model, кој ќе го содржи RDF графот кој треба да го изградите во текот на вежбата.

```
public static void main(String[] args) {
        /**
         PART 1: Creating simple RDF Graph
         */

        //Create empty model
        Model model = ModelFactory.createDefaultModel();
```

Декларирање на сите променливи кои ќе се користат низ графот:

```
public class creationOfRDFGraph {
    1 usage
    public static String personalName = "Tijana Atanasovska";
    1 usage
    public static String personalURI = "https://www.linkedin.com/in/tijana-atanasovska-0b7b99201/";
    1 usage
    public static String personalEmail = "tijana.atanasovska@students.finki.ukim.mk";
    1 usage
    public static String personalAddress = "Spanec, Negotino";
    1 usage
    public static String personalTitle = "Computer Science student";
    1 usage
    public static int personalAge = 22;
    1 usage
    public static String personalBday = "2000-07-19T09:03:40";
    1 usage
    public static String schoolURI = "https://finki.ukim.mk/";
    1 usage
    public static String schoolLogoURI = "https://www.finki.ukim.mk/mk/downloads";
    1 usage
    public static String neighbourSchoolURI = "https://feit.ukim.edu.mk/en/";
```

3. Во моделот додадете нов ресурс, кој ќе ве репрезентира вас како личност. Како URI на ресурсот искористете URL адреса од некој ваш социјален профил (Facebook, Twitter, Instagram, TikTok, …), кој уникатно ве идентификува.

```
//Create resource to represent yourself
Resource personalResource = model.createResource(personalURI);
```

4. Додадете својство на вашиот ресурс, кое ќе го репрезентира вашето целосно име. Искористете го својството ‘vcard:fn’.

```
//Add property to represent your full name
personalResource.addProperty(VCARD.FN, personalName);
```

5. Додадете уште неколку својства по избор, кои ќе бидат од истата ‘vcard’ или пак од ‘foaf’ RDF шемата. Во моделот треба да имате минимум 10 RDF тројки. Притоа,

внимавајте на тоа дали range вредноста на својството кое го додавате треба да биде литерал или друг објект.

```
//Add more properties
personalResource.addProperty(VCARD.EMAIL, personalEmail);
personalResource.addProperty(VCARD.ADR, personalAddress);

//Add property as Literal
Literal title = model.createLiteral(personalTitle);
personalResource.addProperty(VCARD.TITLE, title);

//Add property as Typed Literal (int)
Literal age = model.createTypedLiteral((int)personalAge);
personalResource.addProperty(FOAF.age, age);

//Add property as Typed Literal (DateTime)
Literal bday = model.createTypedLiteral(personalBday, XSDDatatype.XSDdateTime);
personalResource.addProperty(FOAF.birthday, bday);

//Add property as other Resources
//Explanation: Link of FINKI connected with its own Logo Link and based_near Property
// to connect with FEIT's URI
personalResource.addProperty(FOAF.schoolHomepage, model.createResource(schoolURI)
                                        .addProperty(FOAF.logo, model.createResource(schoolLogoURI))
                                        .addProperty(FOAF.based_near,model.createResource(neighbourSchoolURI)
                                                                .addProperty(FOAF.name, s: "FEIT"))
                                        .addProperty(FOAF.name, s: "FINKI"));
```

II. Печатење на RDF граф

6. Со користење на model.listStatements() методот на моделот, изминете ги сите RDF записи (тројки) од графот и отпечатете ги во формат: “subject – predicate – object”. При печатењето, литералите отпечатете ги во наводници (“”). Печатењето нека биде во конзола, т.е. преку System.out.

   Напомена: Пред да ги отпечатите RDF тројките, напишете на конзола “Printing with model.listStatements():”.

```
/**
 PART 2: Printing RDF Graph
 */
System.out.println("Printing with model.listStatements(): ");
StmtIterator iter = model.listStatements();
while (iter.hasNext()){
        Statement s = iter.next();
        Resource subject = s.getSubject();
        Property predicate = s.getPredicate();
        RDFNode object = s.getObject();
        System.out.print(subject.toString() + " - " + predicate.toString() + " - " );

        if(object instanceof Resource)
                System.out.print(object.toString());
        else System.out.print("\"" + object.toString() + "\"");

        System.out.println(" .");
}
```

7. Извршете ја програмата. Може да го извршите целиот проект или само класата во која го дефинирате кодот до сега. Проверете дали излезот на конзола соодветствува со она што очекувате да се прикаже. Дали сите RDF тројки кои ги дефинирате во кодот, ги гледате отпечатени?

   Да.

```
Printing with model.listStatements():
https://www.linkedin.com/in/tijana-atanasovska-0b7b99201/ - http://xmlns.com/foaf/0.1/schoolHomepage - https://finki.ukim.mk/ .
https://www.linkedin.com/in/tijana-atanasovska-0b7b99201/ - http://xmlns.com/foaf/0.1/birthday - "2000-07-19T09:03:40^^http://www.w3.org/2001/XMLSchema#dateTime" .
https://www.linkedin.com/in/tijana-atanasovska-0b7b99201/ - http://xmlns.com/foaf/0.1/age - "22^^http://www.w3.org/2001/XMLSchema#int" .
https://www.linkedin.com/in/tijana-atanasovska-0b7b99201/ - http://www.w3.org/2001/vcard-rdf/3.0#TITLE - "Computer Science student" .
https://www.linkedin.com/in/tijana-atanasovska-0b7b99201/ - http://www.w3.org/2001/vcard-rdf/3.0#ADR - "Spanec, Negotino" .
https://www.linkedin.com/in/tijana-atanasovska-0b7b99201/ - http://www.w3.org/2001/vcard-rdf/3.0#EMAIL - "tijana.atanasovska@students.finki.ukim.mk" .
https://www.linkedin.com/in/tijana-atanasovska-0b7b99201/ - http://www.w3.org/2001/vcard-rdf/3.0#FN - "Tijana Atanasovska" .
https://feit.ukim.edu.mk/en/ - http://xmlns.com/foaf/0.1/name - "FEIT" .
https://finki.ukim.mk/ - http://xmlns.com/foaf/0.1/name - "FINKI" .
https://finki.ukim.mk/ - http://xmlns.com/foaf/0.1/based_near - https://feit.ukim.edu.mk/en/ .
https://finki.ukim.mk/ - http://xmlns.com/foaf/0.1/logo - https://www.finki.ukim.mk/mk/downloads .
```

8. Без да го бришете претходното печатење, додадете ново печатење на RDF тројките од моделот, со користење на model.write() методот. Притоа направете повеќе печатења, во следните RDF формати: RDF/XML, Pretty RDF/XML, N-Triples и Turtle.

   Напомена: Пред секое од печатењата, напишете на конзола “Printing with model.print(), in *Turtle*.”, во зависност од конкретниот формат.

```
System.out.println("Printing with model.print() in Turtle:");
model.write(System.out, s: "TURTLE");

System.out.println("Printing with model.print() in RDF/XML:");
model.write(System.out);

System.out.println("Printing with model.print() in Pretty RDF/XML:");
model.write(System.out, s: "RDF/XML-ABBREV");

System.out.println("Printing with model.print() in JSON-LD:");
model.write(System.out, s: "JSON-LD");

System.out.println("Printing with model.print() in N-Triples:");
model.write(System.out, s: "N-Triples");
```

9. Извршете ја програмата. Може да го извршите целиот проект или само класата во која го дефинирате кодот до сега. Проверете дали излезот на конзола соодветствува со она што очекувате да се прикаже. Кој од RDF форматите има најкратка (најкомпактна) содржина? Кој најлесно се „чита“ на прв поглед? Кој, пак, сметате дека најлесно би го испроцесирале во код, доколку го прочитате програмски од некаде?

   Turtle има најкомпактна содржина.
   Најлесно се чита во Pretty RDF/XML.
   Најлесно процесирање во код би било со JSON-LD.

III. Читање на RDF граф

10. Креирајте нова Java класа во проектот, во која ќе додадете и main() метод.

11. Ископирајте еден од излезите од претходните задачи (вашиот RDF граф во некоја од RDF синтаксите) и ставете го во текстуален фајл, кој ќе го снимите локално, под

произволно име и соодветна наставка: .xml за RDF/XML и Pretty RDF/XML, .ttl за Turtle, .nt за N-Triples и n3 за N3.

graphTurtle.ttl

Plugins supporting *.ttl files found.

```
<https://www.linkedin.com/in/tijana-atanasovska-0b7b99201/>
        <http://www.w3.org/2001/vcard-rdf/3.0#ADR>
                "Spanec, Negotino" ;
        <http://www.w3.org/2001/vcard-rdf/3.0#EMAIL>
                "tijana.atanasovska@students.finki.ukim.mk" ;
        <http://www.w3.org/2001/vcard-rdf/3.0#FN>
                "Tijana Atanasovska" ;
        <http://www.w3.org/2001/vcard-rdf/3.0#TITLE>
                "Computer Science student" ;
        <http://xmlns.com/foaf/0.1/age>
                "22"^^<http://www.w3.org/2001/XMLSchema#int>
        <http://xmlns.com/foaf/0.1/schoolHomepage>
                <https://finki.ukim.mk/> .

<https://feit.ukim.edu.mk/en/>
        <http://xmlns.com/foaf/0.1/name>
                "FEIT" .

<https://finki.ukim.mk/>
        <http://xmlns.com/foaf/0.1/based_near>
                <https://feit.ukim.edu.mk/en/> ;
        <http://xmlns.com/foaf/0.1/logo>
                <https://www.finki.ukim.mk/mk/downloads> ;
        <http://xmlns.com/foaf/0.1/name>
                "FINKI" .
```

12. Во main() методот на новата класа креирајте нов модел и со користење на model.read() вчитајте го RDF графот од датотеката креирана во претходниот чекор.

    Напомена: Искористете го третиот параметар на model.read() кој го означува RDF форматот на датотеката која ја читате – има исти вредности како model.write() при запишување, односно “RDF/XML”, “RDF/XML-ABBREV”, “TTL”, “N-TRIPLES”, итн.

```
public static void main(String[] args) {
    /**
     PART 3: Reading simple RDF Graph
     */
    final String modelPath = "src/main/resources/graphTurtle.ttl";

    Model model = ModelFactory.createDefaultModel();

    InputStream inp = FileManager.get().open(modelPath);
    if(inp==null){
        throw new IllegalArgumentException("File not exists");
    }

    //Third argument in .read is the format of the file from which we read the model
    model.read(inp,  s: "", s1: "TURTLE");
```

13. Напишете код за печатење на моделот (графот), за да видите дали успешно е прочитан.

```
model.write(System.out, "TURTLE");
```

14. Извршете ја програмата. Може да го извршите целиот проект или само класата во која го дефинирвте кодот до сега. Проверете дали излезот на конзола соодветствува со она што очекувате да се прикаже.

```
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder f
<https://www.linkedin.com/in/tijana-atanasovska-0b7b99201/>
        <http://www.w3.org/2001/vcard-rdf/3.0#ADR>
                "Spanec, Negotino" ;
        <http://www.w3.org/2001/vcard-rdf/3.0#EMAIL>
                "tijana.atanasovska@students.finki.ukim.mk" ;
        <http://www.w3.org/2001/vcard-rdf/3.0#FN>
                "Tijana Atanasovska" ;
        <http://www.w3.org/2001/vcard-rdf/3.0#TITLE>
                "Computer Science student" ;
        <http://xmlns.com/foaf/0.1/age>
                "22"^^<http://www.w3.org/2001/XMLSchema#int> ;
        <http://xmlns.com/foaf/0.1/schoolHomepage>
                <https://finki.ukim.mk/> .

<https://feit.ukim.edu.mk/en/>
        <http://xmlns.com/foaf/0.1/name>
                "FEIT" .
```

IV. Навигација низ RDF граф

15. Откако ќе го вчитате графот од датотека во претходниот дел од вежбата, додадете код кој ќе го селектира ресурсот од графот кој ве репрезентира вас.

```java
/* PART 4: Navigating in RDF Graph
 */
Resource personalRepresent = null;
System.out.println("Printing personal Node as a Resource:");

//One way: iterating in Statements
personalRepresent = model.listStatements(new SimpleSelector(
        subject: null,VCARD.FN,(RDFNode) null)).nextStatement().getSubject();
if(personalRepresent != null){
    System.out.println(personalRepresent);
}
else {
    System.out.println("No statements in the Model.");
}

//Another way: iterating in Resources directly
personalRepresent = model.listResourcesWithProperty(VCARD.FN).nextResource();
if(personalRepresent != null){
    System.out.println(personalRepresent);
}
else {
    System.out.println("No statements in the Model.");
}
```

16. Преку селектираниот ресурс, прочитајте ја вредноста на дел од релациите (целосно име, име, презиме, итн.), во зависност од тоа што сте креирале како RDF тројки на почетокот од вежбата.

    Напомена: Внимавајте како пристапувате до вредностите кои се ресурси, а како до вредностите кои се литерали. Постои ли разлика во начинот на пристап?

    Можеме да провериме дали пристапениот објект е Литерал или Ресурс и потоа да печатиме во однос на тоа. Сепак, ако не нагласиме за што станува збор самиот метод враќа класа RDFNode која има метод за печатење и може да печатиме без разлика од кој тип е.

```java
//Iterating in Properties of a Resource
System.out.println("Printing personal Node Properties:");
assert personalRepresent != null;
StmtIterator propIter = personalRepresent.listProperties();
while (propIter.hasNext()){
    Statement s = propIter.nextStatement();
    System.out.print("Property: " + s.getPredicate());
    RDFNode object = s.getObject();
    if(object.isLiteral())
        System.out.println(" Property value: " + object.asLiteral());
    else System.out.println(" Property value: " + object.asResource());
}
```

17. Извршете ја програмата. Може да го извршите целиот проект или само класата во која го дефинирате кодот до сега. Проверете дали излезот на конзола соодветствува со она што очекувате да се прикаже.

```
Printing personal Node as a Resource:
https://www.linkedin.com/in/tijana-atanasovska-0b7b99201/
https://www.linkedin.com/in/tijana-atanasovska-0b7b99201/
Printing personal Node Properties:
Property: http://xmlns.com/foaf/0.1/schoolHomepage Property value: https://finki.ukim.mk/
Property: http://xmlns.com/foaf/0.1/age Property value: 22^^http://www.w3.org/2001/XMLSchema#int
Property: http://www.w3.org/2001/vcard-rdf/3.0#TITLE Property value: Computer Science student
Property: http://www.w3.org/2001/vcard-rdf/3.0#FN Property value: Tijana Atanasovska
Property: http://www.w3.org/2001/vcard-rdf/3.0#EMAIL Property value: tijana.atanasovska@students.finki.ukim.mk
Property: http://www.w3.org/2001/vcard-rdf/3.0#ADR Property value: Spanec, Negotino

Process finished with exit code 0
```

V. Извлекување податоци од RDF граф

18. Креирајте нова Java класа во проектот, во која ќе додадете и main() метод.

19. Преземете ја датотеката “hifm-dataset.ttl” од Courses и снимете ја локално.

20. Во main() методот на новата класа напишете код со кој ќе ја прочитате содржината на оваа датотека. Внимавајте третиот параметар на model.read() да го поставите за вчитување на Turtle содржина.

```
/**
 PART 5: Manipulation of RDF Graph
 */
Model model = ModelFactory.createDefaultModel();
//Read the model
InputStream inputStream = FileManager.get().open(HIFMDatasetLocation);
model.read(inputStream, s: "", s1: "TTL");
//model.write(System.out,"TTL");
```

21. Проучете ја содржината на “hifm-dataset.ttl” датотеката. Станува збор за податочно множество кое содржи лекови од Фондот за здравство на РМ. За секој од лековите имаме тип (hifm-ont:Drug и drugbank:drugs), име (rdfs:label, drugbank:brandName и drugbank:genericName), цена (hifm-ont:refPriceWithVAT), релации кон други локални (hifm-ont:similarTo) и светски лекови (rdfs:seeAlso), итн.

22. Врз база на наученото од досегашниот тек на вежбата, излистајте ги имињата на сите лекови кои се наоѓаат во графот (моделот) (една од трите релации за име е доволна), по азбучен редослед.

```
//List all drugs' names sorted. Relations: rdfs:label, drugbank:brandName и drugbank:genericName
//One way: Search for the property with full URI
Property genericNameProperty = model.getProperty("http://wifo5-04.informatik.uni-mannheim.de/drugbank/res

//Another way: Use saved prefixes in the model
String drugbankPrefix = model.getNsPrefixMap().get("drugbank");
genericNameProperty = model.getProperty(drugbankPrefix+"genericName");

StmtIterator allStmtsWithProperty = model.listStatements(
        new SimpleSelector( subject: null, genericNameProperty, (RDFNode) null));
List<String> sortedStmts = allStmtsWithProperty.toList()
        .stream().map(x->x.getObject().toString()).distinct().sorted().toList();
System.out.println("List of drugs' names sorted by name: ");
sortedStmts.forEach(System.out::println);
```

23. Одберете еден лек од графот (моделот) и за него излистајте ги сите релации и вредности.

```
//Choose drug and list relations and objects:
Resource chosenDrug = model.getResource( s: "http://purl.org/net/hifm/data#988324");
String chosenDrugName = chosenDrug.listProperties(genericNameProperty).nextStatement().getString();

System.out.println("Relations and Values for " + chosenDrugName + ":");
StmtIterator propertiesIter = chosenDrug.listProperties();
List<Statement> propertiesList = propertiesIter.toList();
for (Statement s : propertiesList){
    System.out.println(s);
}
```

```
Relations and Values for Dextriferon:
[http://purl.org/net/hifm/data#988324, http://purl.org/net/hifm/ontology#similarTo, http://purl.org/net/hifm/data#988359]
[http://purl.org/net/hifm/data#988324, http://purl.org/net/hifm/ontology#dosageForm, "□□□□□□□□ □□ □□□□□□□□"]
[http://purl.org/net/hifm/data#988324, http://wifo5-04.informatik.uni-mannheim.de/drugbank/resource/drugbank/atcCode, "B03AB05"]
[http://purl.org/net/hifm/data#988324, http://wifo5-04.informatik.uni-mannheim.de/drugbank/resource/drugbank/genericName, "Dextriferon"]
[http://purl.org/net/hifm/data#988324, http://www.w3.org/2000/01/rdf-schema#label, "Dextriferon"]
[http://purl.org/net/hifm/data#988324, http://wifo5-04.informatik.uni-mannheim.de/drugbank/resource/drugbank/brandName, "FERRUM LEK □□□□□□□□ □□ □□□□□□□□ 30x100mg"]
[http://purl.org/net/hifm/data#988324, http://purl.org/net/hifm/ontology#refPriceNoVAT, "81.9"^^http://www.w3.org/2001/XMLSchema#decimal]
[http://purl.org/net/hifm/data#988324, http://purl.org/net/hifm/ontology#refPriceWithVAT, "86.0"^^http://www.w3.org/2001/XMLSchema#decimal]
[http://purl.org/net/hifm/data#988324, http://purl.org/net/hifm/ontology#id, "988324"^^http://www.w3.org/2001/XMLSchema#integer]
[http://purl.org/net/hifm/data#988324, http://purl.org/net/hifm/ontology#similarTo, http://purl.org/net/hifm/data#988332]
[http://purl.org/net/hifm/data#988324, http://purl.org/net/hifm/ontology#manufacturer, "LEK"]
[http://purl.org/net/hifm/data#988324, http://purl.org/net/hifm/ontology#similarTo, http://purl.org/net/hifm/data#988375]
[http://purl.org/net/hifm/data#988324, http://www.w3.org/1999/02/22-rdf-syntax-ns#type, http://purl.org/net/hifm/ontology#Drug]
[http://purl.org/net/hifm/data#988324, http://www.w3.org/1999/02/22-rdf-syntax-ns#type, http://wifo5-04.informatik.uni-mannheim.de/drugbank/resource/drugbank/drugs]
[http://purl.org/net/hifm/data#988324, http://purl.org/net/hifm/ontology#similarTo, http://purl.org/net/hifm/data#988391]
[http://purl.org/net/hifm/data#988324, http://purl.org/net/hifm/ontology#packaging, "30"^^http://www.w3.org/2001/XMLSchema#integer]
[http://purl.org/net/hifm/data#988324, http://purl.org/net/hifm/ontology#strength, "100mg"]
```

24. Одберете еден лек од графот (моделот) и за него излистајте ги имињата на сите лекови кои имаат иста функција како и тој, т.е. лекови со кои тој е во релација ‘hifm-ont:similarTo’. Форматирајте го печатењето за да е јасно видливо за што станува збор.

```
//Print drugs in relation SimilarTo with the chosen Drug:
String hifmOntPrefix = model.getNsPrefixMap().get("hifm-ont");
Property similarToProperty = model.getProperty(hifmOntPrefix + "similarTo");
StmtIterator similarToStatementsIter = chosenDrug.listProperties(similarToProperty);
List<Statement> similarToStatementsList = similarToStatementsIter.toList();
for (Statement s: similarToStatementsList){
    Resource o = s.getObject().asResource();
    //Get name of similar drug
    String similarName = o.listProperties(genericNameProperty).nextStatement().getObject().toString();
    System.out.printf("%-15s IS SIMILAR TO %-15s\n", chosenDrugName, similarName);
}
```

```
Dextriferon     IS SIMILAR TO Dextriferon
Dextriferon     IS SIMILAR TO Dextriferon
Dextriferon     IS SIMILAR TO Dextriferon
Dextriferon     IS SIMILAR TO Dextriferon
```

25. Одберете еден лек од графот (моделот) и за него најпрвин излистајте ја неговата цена (hifm-ont:refPriceWithVAT), а потоа излистајте ги и имињата и цените на лековите кои ја имаат истата функција како и тој (hifm-ont:similarTo). Форматирајте го печатењето за да е јасно видливо за што станува збор.

```
//Print price for a drug (hifm-ont:refPriceWithVAT),
// then print names and prices of drugs connected with relation (hifm-ont:similarTo).
Property refPriceWithVATProperty = model.getProperty(hifmOntPrefix+"refPriceWithVAT");
Double price = chosenDrug.listProperties(refPriceWithVATProperty)
               .nextStatement().getObject().asLiteral().getDouble();
System.out.println("Price of drug " + chosenDrugName + " is: " + price);
for (Statement s: similarToStatementsList){
    Resource o = s.getObject().asResource();
    String similarName = o.listProperties(genericNameProperty).nextStatement().getObject().toString();
    Double similarPrice = o.listProperties(refPriceWithVATProperty)
                        .nextStatement().getObject().asLiteral().getDouble();
    System.out.printf("%-15s with price %f IS SIMILAR TO %-15s with price %f.\n",
                        chosenDrugName,price, similarName,similarPrice);
}
```

26. Извршете ја програмата. Може да го извршите целиот проект или само класата во која го дефинирате кодот до сега. Проверете дали излезот на конзола соодветствува со она што очекувате да се прикаже.

```
Price of drug Dextriferon is: 86.0
Dextriferon     with price 86.000000 IS SIMILAR TO Dextriferon     with price 68.000000.
Dextriferon     with price 86.000000 IS SIMILAR TO Dextriferon     with price 86.000000.
Dextriferon     with price 86.000000 IS SIMILAR TO Dextriferon     with price 68.000000.
Dextriferon     with price 86.000000 IS SIMILAR TO Dextriferon     with price 136.000000.
```

Напомена: Доколку успеавте да ги завршите задачите под точка 22, 23, 24 и 25, практично напишавте код кој може да биде основа за една мобилна, веб или десктоп апликација за лекови: на корисникот му се претставуваат сите лекови (22), може да одбере некој од нив и да му се отвори приказ со сите детали за лекот (23), да ги види алтернативните лекови со иста функција кои може да ги купи наместо селектираниот (24) и да ги спореди нивните цени (25) со цел да го избере најевтиниот од таа група лекови со исто дејство.