

**Домашно бр. 1
Jena RDF API**

203235 - Димитрија Тимески

27-Oct-2023

A) Прашања

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

Тој се изразува како тројка во граф. Под тројка се подразбира на **Subject – Predicate – Object**.

Subject-от може да биде **RDF URI reference** но и **blank node**

- За него даваме информација

Predicate-от може да биде **RDF URI reference**

- Предикатот е односот меѓу субјектот и објектот

Object-от може да биде **RDF URI reference** но и **literal**, но и **blank node**

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.

Turtle, RDF/XML, RDFA, JSON-LD

Turtle:

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

RDF/XML:

```
<?xml version="1.0"?>  
<rdf:RDF  
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"  
    xmlns:finki="http://finki.ukim.mk/resource#">  
    <rdf:Description rdf:about="http://finki.ukim.mk/resource#ВБС">  
        <finki:sePredavaNa rdf:resource="http://finki.ukim.mk/resource#FINKI"/>  
    </rdf:Description>  
</rdf:RDF>
```

RDFA (HTML):

```
<!DOCTYPE html>  
<html xmlns="http://www.w3.org/1999/xhtml"  
      xmlns:finki="http://finki.ukim.mk/resource#">  
<head>  
    <title>RDFA Example</title>  
</head>  
<body>
```

```
<div typeof="finki:ВБС">
  <span property="finki:sePredavaNa" resource="finki:FINKI"></span>
</div>
</body>
</html>
```

JSON-LD:

```
{
  "@context": {
    "finki": "http://finki.ukim.mk/resource#"
  },
  "@id": "finki:ВБС",
  "finki:sePredavaNa": "finki:FINKI"
}
```

3. За што се користи RDF Schema?

RDF Schema е всушност вокабулар за описување (vocabulary description language)

Ги објаснува својствата и класите на RDF ресурсите

Нуди семантика за генерализација на хиерархиите од својства и класите.

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 rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://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 ;
  rdfs:range finki:fakultet .
```

What can exist on the left side of a Property = domain

What can exist on its right side = range

Б) Практична задача

I. Креирање едноставен RDF граф

- 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](https://jena.apache.org/download/index.cgi).

<https://jena.apache.org/download/index.cgi>

Apache Jena Release

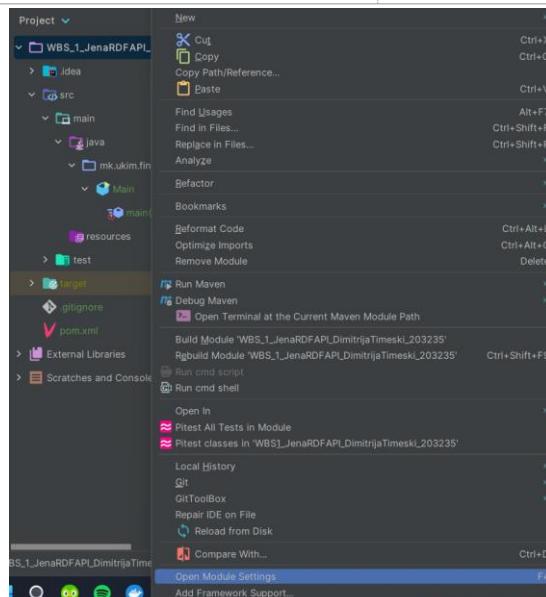
Source release: this forms the official release of Apache Jena. All binaries artifacts and maven binaries correspond to this source.

Apache Jena Release	SHA512	Signature
jena-4.9.0-source-release.zip	SHA512	PGP

Apache Jena Binary Distributions

The binary distribution of the Fuseki server:

Apache Jena Fuseki	SHA512	Signature
apache-jena-fuseki-4.9.0.tar.gz	SHA512	PGP
apache-jena-fuseki-4.9.0.zip	SHA512	PGP



- 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 class Main {
    new *
    public static void main(String[] args) {
        Model model = ModelFactory.createDefaultModel(); // 2 The model in Jena is the Graph in RDF
    }
}
```

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

```
Resource dimitrijaTimeski = model.createResource(personURI); // 3
```

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

```
String fullName = "Dimitrija Timeski";
dimitrijaTimeski.addProperty(VCARD.FN, fullName);
```

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 давате треба да биде литерал или друг објект.

```
String fullName = "Dimitrija Timeski";
dimitrijaTimeski.addProperty(VCARD.FN, fullName);

String address = "Skopje, Macedonia";
dimitrijaTimeski.addProperty(VCARD.ADR, address);

String email = "dimitrijatimeskidimitrija@gmail.com";
dimitrijaTimeski.addProperty(VCARD.EMAIL, email);

String birthday = "2000-11-29";
dimitrijaTimeski.addProperty(VCARD.BDAY, birthday);

String givenName = "Dimitrija";
dimitrijaTimeski.addProperty(VCARD.Given, givenName);

String familyName = "Timeski";
dimitrijaTimeski.addProperty(VCARD.Family, familyName);

// FOAF

dimitrijaTimeski.addProperty(FOAF.name, fullName);
dimitrijaTimeski.addProperty(FOAF.birthday, birthday);
dimitrijaTimeski.addProperty(FOAF.accountName, givenName);
dimitrijaTimeski.addProperty(FOAF.givenName, givenName);
dimitrijaTimeski.addProperty(FOAF.familyName, familyName);

String person2URI = "https://github.com/DimeJovanovski";
Resource anotherPerson = model.createResource(person2URI);

dimitrijaTimeski.addProperty(FOAF.knows, anotherPerson);
```

II. Печатење на RDF граф

6. Со користење на `model.listStatements()` методот на моделот, изминете ги сите RDF записи (тројки) од граffот 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():`”.

```

System.out.println("Printing with model.listStatements()");
StmtIterator iter = model.listStatements();
// In format Subject -> Predicate -> Object
while (iter.hasNext()) {
    Statement stmt = iter.nextStatement(); // Џаземаме тројката (SP0)
    Resource subject = stmt.getSubject(); // Го земаме S
    Property predicate = stmt.getPredicate(); // Го земаме P
    RDFNode object = stmt.getObject(); // Го земаме O

    System.out.print(subject.toString()); // Печатење на S
    System.out.print(" " + predicate.toString() + " "); // Печатење на P

    // Печатење на O (O може да биде Resource или Literal)
    if (object instanceof Resource) {
        System.out.print(object.toString());
    } else {
        // object is a literal
        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?

```

Run Main x
C:\Users\dimit\.jdks\openjdk-20.0.2\bin\java.exe ...
Printing with model.listStatements():
https://github.com/DimitrijaT http://xmlns.com/foaf/0.1/birthDay "2000-11-29" .
https://github.com/DimitrijaT http://www.w3.org/2001/vcard-rdf/3.0#ADR "Skopje, Macedonia" .
https://github.com/DimitrijaT http://www.w3.org/2001/vcard-rdf/3.0#Family "Timeski" .
https://github.com/DimitrijaT http://xmlns.com/foaf/0.1/givenName "Dimitrija" .
https://github.com/DimitrijaT http://www.w3.org/2001/vcard-rdf/3.0#Given "Dimitrija" .
https://github.com/DimitrijaT http://www.w3.org/2001/vcard-rdf/3.0#EMAIL "dimitrijatimeskidimitrija@gmail.com" .
https://github.com/DimitrijaT http://xmlns.com/foaf/0.1/familyName "Timeski" .
https://github.com/DimitrijaT http://www.w3.org/2001/vcard-rdf/3.0#BDAY "2000-11-29" .
https://github.com/DimitrijaT http://xmlns.com/foaf/0.1/accountName "Dimitrija" .
https://github.com/DimitrijaT http://xmlns.com/foaf/0.1/knows https://github.com/DimeJovanovski .
https://github.com/DimitrijaT http://xmlns.com/foaf/0.1/name "Dimitrija Timeski" .
https://github.com/DimitrijaT http://www.w3.org/2001/vcard-rdf/3.0#FN "Dimitrija Timeski" .

```

Да сите тројки кои што ги дефинирајат ги гледам отпечатени.

8. Без да го бришете претходното печатење, додадете ново печатење на RDF тројките од моделот, со користење на `model.write()` методот. При тоа 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("Print with model.write() in RDF/XML");
    .....model.write(System.out);

    .....System.out.println("Print with model.write() in pretty RDF/XML:");
    .....model.write(System.out, lang: "RDF/XML-ABBREV");

    .....System.out.println("Print with model.write() in N-TRIPLES:");
    .....model.write(System.out, lang: "N-TRIPLES");

    .....System.out.println("Print with model.write() in Turtle:");
    .....model.write(System.out, lang: "TURTLE"); // or "TTL"
```

9. Извршете ја програмата. Може да го извршите целиот проект или само класата во која го дефинираате 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?

```
Printing with model.listStatements():
https://github.com/DimitrijaT http://xmlns.com/foaf/0.1/birthday "20001129".
https://github.com/DimitrijaT http://www.w3.org/2001/vcard-rdf/3.0#ADR "Skopje, Macedonia".
https://github.com/DimitrijaT http://www.w3.org/2001/vcard-rdf/3.0#Family "Timeski".
https://github.com/DimitrijaT http://xmlns.com/foaf/0.1/givenName "Dimitrija".
https://github.com/DimitrijaT http://www.w3.org/2001/vcard-rdf/3.0#Given "Dimitrija".
https://github.com/DimitrijaT http://www.w3.org/2001/vcard-rdf/3.0#EMAIL "dimitrijatimeskidimitrija@gmail.com".
https://github.com/DimitrijaT http://xmlns.com/foaf/0.1/familyName "Timeski".
https://github.com/DimitrijaT http://www.w3.org/2001/vcard-rdf/3.0#BDAY "20001129".
https://github.com/DimitrijaT http://xmlns.com/foaf/0.1/accountName "Dimitrija".
https://github.com/DimitrijaT http://xmlns.com/foaf/0.1/knows https://github.com/DimeJovanovski .
https://github.com/DimitrijaT http://xmlns.com/foaf/0.1/name "Dimitrija Timeski".
https://github.com/DimitrijaT http://www.w3.org/2001/vcard-rdf/3.0#FN "Dimitrija Timeski".
=====
Print with model.write() in RDF/XML
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:vcard="http://www.w3.org/2001/vcard-rdf/3.0#"
  xmlns:j_0="http://xmlns.com/foaf/0.1/"
  <rdf:Description rdf:about="https://github.com/DimitrijaT">
    <j_0:birthday>20001129</j_0:birthday>
    <vcard:ADR>Skopje, Macedonia</vcard:ADR>
    <vcard:Family>Timeski</vcard:Family>
    <j_0:givenName>Dimitrija</j_0:givenName>
    <vcard:Given>Dimitrija</vcard:Given>
    <vcard:EMAIL>dimitrijatimeskidimitrija@gmail.com</vcard:EMAIL>
    <j_0:familyName>Timeski</j_0:familyName>
    <vcard:BDAY>20001129</vcard:BDAY>
    <j_0:accountName>Dimitrija</j_0:accountName>
    <j_0:knows rdf:resource="https://github.com/DimeJovanovski"/>
    <j_0:name>Dimitrija Timeski</j_0:name>
    <vcard:FN>Dimitrija Timeski</vcard:FN>
  </rdf:Description>
</rdf:RDF>
Print with model.write() in pretty RDF/XML:
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:vcard="http://www.w3.org/2001/vcard-rdf/3.0#"
  xmlns:j_0="http://xmlns.com/foaf/0.1/"
  <rdf:Description rdf:about="https://github.com/DimitrijaT">
    <j_0:birthday>20001129</j_0:birthday>
    <vcard:ADR>Skopje, Macedonia</vcard:ADR>
    <vcard:Family>Timeski</vcard:Family>
    <j_0:givenName>Dimitrija</j_0:givenName>
    <vcard:Given>Dimitrija</vcard:Given>
    <vcard:EMAIL>dimitrijatimeskidimitrija@gmail.com</vcard:EMAIL>
    <j_0:familyName>Timeski</j_0:familyName>
    <vcard:BDAY>20001129</vcard:BDAY>
    <j_0:accountName>Dimitrija</j_0:accountName>
    <j_0:knows rdf:resource="https://github.com/DimeJovanovski"/>
    <j_0:name>Dimitrija Timeski</j_0:name>
```

```
<vcard:FN>Dimitrija Timeski</vcard:FN>
</rdf:Description>
</rdf:RDF>
Print with model.write() in N-TRIPLES:
<https://github.com/DimitrijaT> <http://xmlns.com/foaf/0.1/birthday> "20001129" .
<https://github.com/DimitrijaT> <http://www.w3.org/2001/vcard-rdf/3.0#ADR> "Skopje, Macedonia" .
<https://github.com/DimitrijaT> <http://www.w3.org/2001/vcard-rdf/3.0#Family> "Timeski" .
<https://github.com/DimitrijaT> <http://xmlns.com/foaf/0.1/givenName> "Dimitrija" .
<https://github.com/DimitrijaT> <http://www.w3.org/2001/vcard-rdf/3.0#Given> "Dimitrija" .
<https://github.com/DimitrijaT> <http://www.w3.org/2001/vcard-rdf/3.0#EMAIL> "dimitrijatimeskidimitrija@gmail.com" .
<https://github.com/DimitrijaT> <http://www.w3.org/2001/vcard-rdf/3.0#FamilyName> "Timeski" .
<https://github.com/DimitrijaT> <http://www.w3.org/2001/vcard-rdf/3.0#BDAY> "20001129" .
<https://github.com/DimitrijaT> <http://xmlns.com/foaf/0.1/accountName> "Dimitrija" .
<https://github.com/DimitrijaT> <http://www.w3.org/2001/vcard-rdf/3.0#FN> "Dimitrija Timeski" .
<https://github.com/DimitrijaT> <http://www.w3.org/2001/vcard-rdf/3.0#NAME> "Dimitrija Timeski" .
Print with model.write() in Turtle:
<https://github.com/DimitrijaT>
    <http://www.w3.org/2001/vcard-rdf/3.0#ADR>
        "Skopje, Macedonia";
    <http://www.w3.org/2001/vcard-rdf/3.0#BDAY>
        "20001129";
    <http://www.w3.org/2001/vcard-rdf/3.0#EMAIL>
        "dimitrijatimeskidimitrija@gmail.com";
    <http://www.w3.org/2001/vcard-rdf/3.0#FN>
        "Dimitrija Timeski";
    <http://www.w3.org/2001/vcard-rdf/3.0#Family>
        "Timeski";
    <http://www.w3.org/2001/vcard-rdf/3.0#Given>
        "Dimitrija";
    <http://xmlns.com/foaf/0.1/accountName>
        "Dimitrija";
    <http://xmlns.com/foaf/0.1/birthday>
        "20001129";
    <http://xmlns.com/foaf/0.1/familyName>
        "Timeski";
    <http://xmlns.com/foaf/0.1/givenName>
        "Dimitrija";
    <http://xmlns.com/foaf/0.1/knows>
        <https://github.com/DimeJovanovski>;
    <http://xmlns.com/foaf/0.1/name>
        "Dimitrija Timeski".
```

Process finished with exit code 0

Излезот се поклопува со посакуваното. Најлесно читливиот код е N-Triples, па по него Turtle.

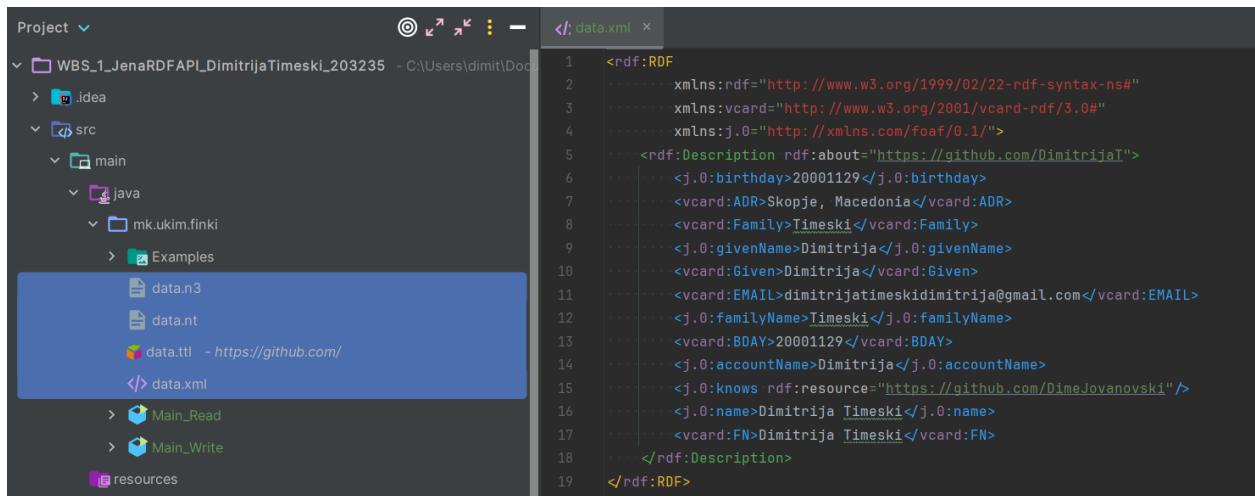
А форматот кој најлесно се процесира од машината е RDF/XML

III. Читање на RDF граф

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

```
public class Main_Read {
    new *
    public static void main(String[] args) {
        ...
    }
}
```

11. Ископирајте еден од излезите од претходните задачи (вашиот RDF граф во некоја од RDF синтаксите) и ставете го во текстуален файл, 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.



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.

```

static final String inputFileName = "C:\\\\Users\\\\dimit\\\\\" +  
    "Documents\\\\7th Semester\\\\WBS_1_JenaRDFAPI_DimitrijaTimeski_203235\\\\\" +  
    "WBS_1_JenaRDFAPI_DimitrijaTimeski_203235\\\\src\\\\main\\\\java\\\\mk\\\\\" +  
    "ukim\\\\finki\\\\data.xml";  
  
new *  
  
public static void main(String[] args) {  
    // create an empty model  
    Model model = ModelFactory.createDefaultModel();  
  
    // read the RDF/XML file  
    model.read(in, base: "");  
}

```

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

```

// read the RDF/XML file  
model.read(in, base: "");  
  
// write it  
model.write(System.out, lang: "TTL");

```

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

```
C:\Users\dimit\.jdks\openjdk-20.0.2\bin\java.exe ...
@prefix j: <http://xmlns.com/foaf/0.1/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix vcard: <http://www.w3.org/2001/vcard-rdf/3.0#> .

<https://github.com/DimitrijaT>
    vcard:ADR      "Skopje, Macedonia";
    vcard:BDAY     "20001129";
    vcard:EMAIL    "dimitrijatimeskidimitrija@gmail.com";
    vcard:FN       "Dimitrija Timeski";
    vcard:Family   "Timeski";
    vcard:Given    "Dimitrija";
    j:0:accountName "Dimitrija";
    j:0:birthday  "20001129";
    j:0:familyName "Timeski";
    j:0:givenName  "Dimitrija";
    j:0:knows     <https://github.com/DimeJovanovski>;
    j:0:name      "Dimitrija Timeski" .
```

Process finished with exit code 0

Излезот од конзолата е точен. Прочитав од RDF/XML формат а го испринтав во Turtle/Ttl формат.

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 презентира вас.

```
// Grab the resource that represents us
Resource dimitrijaTimeski = model.getResource( uri: "https://github.com/DimitrijaT" );
```

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?

```
// Literals
System.out.println("Fullname: " + dimitrijaTimeski.getProperty(VCARD.FN).getString());
System.out.println("Given name: " + dimitrijaTimeski.getProperty(VCARD.Given).getString());
System.out.println("Family name: " + dimitrijaTimeski.getProperty(VCARD.Family).getString());

// Resources
System.out.println("Knows: " + dimitrijaTimeski.getProperty(FOAF.knows).getResource().getURI());
```

Пристапот до литералот и ресурсот се изведува на различен начин.

Целото име, името и презимето се литерали, додека пак лицето кое го познава е resource.

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

```
Fullname: Dimitrija Timeski
Given name: Dimitrija
Family name: Timeski
Knows: https://github.com/DimeJovanovski
```

Програмата се соодветствува на тоа што беше очекувано.

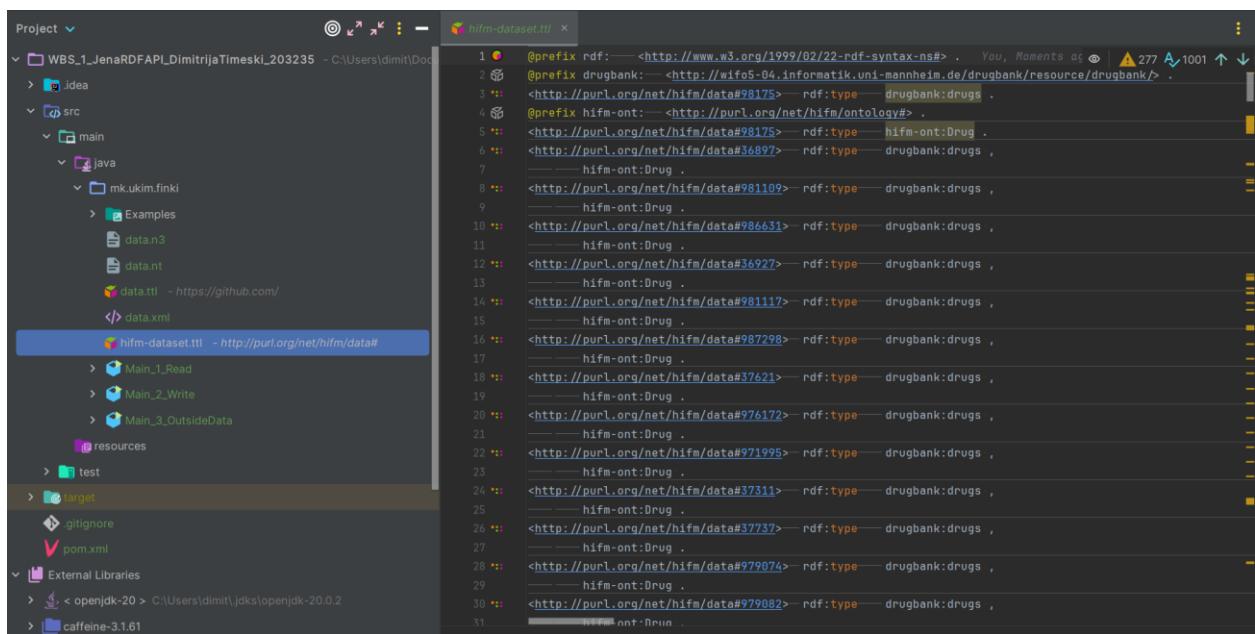
V. Извлекување податоци од RDF граф

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

```
public class Main_3_OutsideData {
    new *
    public static void main(String[] args) {
        ...
    }
}
```

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

Веб базирани системи 2023



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.

```
public static void main(String[] args) {  
    Model model = ModelFactory.createDefaultModel();  
  
    InputStream in = FileManager.get().open(inputFileName);  
    if (in == null) {  
        throw new IllegalArgumentException("File: " + inputFileName + " not found");  
    }  
  
    model.read(in, base: null, lang: "TTL"); // Third parameter tells us the type!  
  
    model.write(System.out, lang: "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.

Проучено 😊

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.

```
public static void readAllBrandName(Model model) {  
  
    Property brandNameProperty = model.createProperty(drugbankPrefix, localName: "brandName");  
    List<String> drugs = new ArrayList<>();  
  
    Selector selector = new SimpleSelector(subject: null, brandNameProperty, (String) null);  
    StmtIterator sIter = model.listStatements(selector);  
    if (sIter.hasNext()) {  
        System.out.println("The database contains these drugs:");  
        while (sIter.hasNext()) {  
            drugs.add("..." + sIter.nextStatement() Statement  
                    .getObject() RDFNode  
                    .toString());  
        }  
    } else {  
        System.out.println("No drugs were found in the database");  
    }  
    drugs.sort(String::compareTo);  
    System.out.println(drugs);  
}
```

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

```
public static void ReadOneDrug(Model model, String drug) {  
  
    Property brandNameProperty = model.createProperty(drugbankNamespace, localName: "brandName");  
  
    // List all relations and values for a drug  
    Resource drugResource = model.getResource(drug);  
    String drugName = drugResource.getProperty(brandNameProperty).getString();  
  
    StmtIterator sIter = drugResource.listProperties(); // All Relations  
    System.out.println("The drug " + drugName + " has these relations:");  
    while (sIter.hasNext()) {  
        Statement stmt = sIter.nextStatement();  
        System.out.println(stmt.getPredicate().getLocalName() + " - " + stmt.getObject());  
    }  
}
```

```

C:\Users\dimit\.jdks\openjdk-20.0.2\bin\java.exe ...
The drug ZINNAT табл. 10 x 250 mg has these relations:
dosageForm - Таблети
seeAlso - http://wifo5-04.informatik.uni-mannheim.de/drugbank/resource/drugs/DB01112
refPriceNoVAT - 100.0^^http://www.w3.org/2001/XMLSchema#decimal
refPriceWithVAT - 105.0^^http://www.w3.org/2001/XMLSchema#decimal
similarTo - http://purl.org/net/hifm/data#974935
manufacturer - GSK
similarTo - http://purl.org/net/hifm/data#978752
similarTo - http://purl.org/net/hifm/data#994774
type - http://purl.org/net/hifm/ontology#Drug
type - http://wifo5-04.informatik.uni-mannheim.de/drugbank/resource/drugbank/drugs
genericName - Cefuroxime
brandName - ZINNAT табл. 10 x 250 mg
id - 974919^^http://www.w3.org/2001/XMLSchema#integer
similarTo - http://purl.org/net/hifm/data#974927
similarTo - http://purl.org/net/hifm/data#993514
packaging - 10^^http://www.w3.org/2001/XMLSchema#integer
strength - 250mg
similarTo - http://purl.org/net/hifm/data#994766
label - Cefuroxime
similarTo - http://purl.org/net/hifm/data#978744
atcCode - J01DC02

Process finished with exit code 0

```

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.

```

public static void findSimilarDrugs(Model model, String drug) {
    Property brandNameProperty = model.createProperty(drugbankPrefix, "brandName");
    Property similarToProperty = model.createProperty(hifmOntPrefix, "similarTo");

    // List all relations and values for a drug
    Resource drugResource = model.getResource(drug);
    String drugName = drugResource.getProperty(brandNameProperty).getString();

    Selector selector = new SimpleSelector(drugResource, similarToProperty, (String) null);
    StmtIterator sIter = model.listStatements(selector);
    System.out.println("The following drugs are similar to " + drugName + ":");
    while (sIter.hasNext()) {
        Statement stmt = sIter.nextStatement();
        System.out.print(stmt.getObject() + " Name: ");
        System.out.println(stmt.getResource().getProperty(brandNameProperty).getString());
    }
}

```

```
C:\Users\dimit\.jdks\openjdk-20.0.2\bin\java.exe ...
The following drugs are similar to ZOLSANA филм обл.табл. 20 x 10mg:
http://purl.org/net/hifm/data#978108 Name: ZONADIN филм обл.табл.20 x 5mg
http://purl.org/net/hifm/data#979147 Name: LUNATA филм обл.табл. 10 x 10mg
http://purl.org/net/hifm/data#987964 Name: ZOLSANA филм обл.табл. 20 x 5mg
http://purl.org/net/hifm/data#972878 Name: SANVAL филм обл.табл. 20 x 10mg
http://purl.org/net/hifm/data#978116 Name: ZONADIN филм обл.табл.20 x 10mg
http://purl.org/net/hifm/data#979139 Name: LUNATA филм обл.табл. 10 x 5mg
```

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.

```
public static void findDrugPriceAndSimilarProducts(Model model, String drug) {
    Property brandNameProperty = model.createProperty(drugbankPrefix, localName: "brandName");
    Property refPriceWithVATProperty = model.createProperty(hifmOntPrefix, localName: "refPriceWithVAT");
    Property similarToProperty = model.createProperty(hifmOntPrefix, localName: "similarTo");

    Resource drugResource = model.getResource(drug);
    String drugPrice = drugResource.getProperty(refPriceWithVATProperty).getString();
    String drugName = drugResource.getProperty(brandNameProperty).getString();

    Selector selector = new SimpleSelector(drugResource, similarToProperty, (String) null);
    StmtIterator sIter = model.listStatements(selector);
    System.out.println("NAME: " + drugName + "\t PRICE: " + drugPrice + "\nSee Also:");

    while (sIter.hasNext()) {
        Statement stmt = sIter.nextStatement();

        String name = stmt.getProperty(brandNameProperty).getString();
        String price = stmt.getProperty(refPriceWithVATProperty).getString();

        System.out.print(stmt.getObject() + " ");
        System.out.println("NAME: " + name + "\t PRICE: " + price);
    }
}
```

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

```
C:\Users\dimit\.jdks\openjdk-20.0.2\bin\java.exe ...
NAME: ZOLSANA фильм обл.табл. 20 x 10mg — PRICE: 82.0
See Also:
http://purl.org/net/hifm/data#978108 NAME: ZONADIN фильм обл.табл.20 x 5mg — PRICE: 52.0
http://purl.org/net/hifm/data#979147 NAME: LUNATA фильм обл.табл. 10 x 10mg — PRICE: 41.0
http://purl.org/net/hifm/data#987964 NAME: ZOLSANA фильм обл.табл. 20 x 5mg — PRICE: 52.0
http://purl.org/net/hifm/data#972878 NAME: SANVAL фильм обл.табл. 20 x 10mg — PRICE: 82.0
http://purl.org/net/hifm/data#978116 NAME: ZONADIN фильм обл.табл.20 x 10mg — PRICE: 82.0
http://purl.org/net/hifm/data#979139 NAME: LUNATA фильм обл.табл. 10 x 5mg — PRICE: 26.0

Process finished with exit code 0
```

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.

Кодот е достапен на следниот линк

<https://github.com/DimitrijaT/web-based-systems-homework>