Домашно бр. 1
Jena RDF API

А) Прашања

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

Еден запис односно факт во RDF моделот се изразува како RDF тројка.
Од ресурс ⟶ својство -> ресурс или ресурс ⟶ својство -> литерал.

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:

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

finki:VBS finki:sePredavaNa finki:FINKI.
```

RDF/XML:

```
<rdf:RDF xmlns:finki=”http://finki.ukim.mk/resource#”>
    <rdf:Description rdf:about=”finki:VBS”>
            < finki:sePredavaNa  rdf:resource=”finki:FINKI”/>
    </rdf:Description>
</rdf:RDF>
```

JSON-LD:

```
{
    “@context”: “http://finki.ukim.mk/resource#”
    “id”: “VBS”
    “sePredeavaNa”: “FINKI”
}
```

3. **За што се користи RDF Schema?**

RDF шема се користи за да се додаде семантика на RDF како дефинирање на класи, подкласи, наследување, својства и подсвојства, ограничувања во рангот и доменот на својсвата и сл.

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#>

finki:Fakultet rdf:type rdfs:Class.
finki:Predmet rdf:type rdfs:Class.
finki:ePredmetNa rdf:type rdfs:Property;
rdfs:domain finki:Predmet;
rdfs:range finki:Fakultet.

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.

```
<dependencies>
    <dependency>
        <groupId>org.apache.jena</groupId>
        <artifactId>apache-jena-libs</artifactId>
        <type>pom</type>
        <version>4.9.0</version>
    </dependency>
```

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.

```
Model model = ModelFactory.createDefaultModel();
```

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.

```
String instagramId = "https://www.instagram.com/andrejbardakoski/";
Resource Andrej = model.createResource(instagramId);
```

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

```
andrej.addProperty(VCARD.FN, "Andrej Bardakoski");
```

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.

```
andrej.addProperty(VCARD.FN, "Andrej Bardakoski");
andrej.addProperty(VCARD.NICKNAME, "Jony");
andrej.addProperty(VCARD.NICKNAME, "Bardak");
andrej.addProperty(VCARD.NICKNAME, "Sashko");
andrej.addProperty(VCARD.BDAY,"20.10.2001");
andrej.addProperty(VCARD.N, model.createResource()
        .addProperty(VCARD.Given, "Andrej")
        .addProperty(VCARD.Family, "Bardakoski"));
andrej.addProperty(VCARD.ADR, model.createResource()
        .addProperty(VCARD.Country,"Macedonia")
        .addProperty(VCARD.Locality, "Skopje")
        .addProperty(VCARD.Pcode, "1000")
        .addProperty(VCARD.Street, "Krume Kepeski"));
andrej.addProperty(FOAF.age,"22");
andrej.addProperty(FOAF.accountName,"andrejbardakoski");
```

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():”.

```
public static void printModelListStatements(Model model) {
    System.out.println("Printing with model.listStatements():");
    StmtIterator iterator = model.listStatements();

    while (iterator.hasNext()) {
        Statement statement = iterator.nextStatement();
        Resource subject = statement.getSubject();
        Property predicate = statement.getPredicate();
        RDFNode object = statement.getObject();

        System.out.print(subject.toString());
        System.out.print(" - " + predicate.toString() + " - ");
        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 дефинирвте кодот до сега. Проверете дали излезот на конзола соодветствува со она што очекувате да се прикаже. Дали сите RDF тројки кои ги дефинирвте во кодот, ги гледате отпечатени?

```
Printing with model.listStatements():
e8ef1f5d-1fe7-4a42-bdba-7f7dee16ea94 - http://www.w3.org/2001/vcard-rdf/3.0#Family -  "Bardakoski"
e8ef1f5d-1fe7-4a42-bdba-7f7dee16ea94 - http://www.w3.org/2001/vcard-rdf/3.0#Given -  "Andrej"
https://www.instagram.com/andrejbardakoski/ - http://xmlns.com/foaf/0.1/accountName -  "andrejbardakoski"
https://www.instagram.com/andrejbardakoski/ - http://xmlns.com/foaf/0.1/age -  "22"
https://www.instagram.com/andrejbardakoski/ - http://www.w3.org/2001/vcard-rdf/3.0#ADR - 7683751d-e3ca-4685-8f49-322d1686f64c
https://www.instagram.com/andrejbardakoski/ - http://www.w3.org/2001/vcard-rdf/3.0#N - e8ef1f5d-1fe7-4a42-bdba-7f7dee16ea94
https://www.instagram.com/andrejbardakoski/ - http://www.w3.org/2001/vcard-rdf/3.0#BDAY -  "20.10.2001"
https://www.instagram.com/andrejbardakoski/ - http://www.w3.org/2001/vcard-rdf/3.0#NICKNAME -  "Sashko"
https://www.instagram.com/andrejbardakoski/ - http://www.w3.org/2001/vcard-rdf/3.0#NICKNAME -  "Bardak"
https://www.instagram.com/andrejbardakoski/ - http://www.w3.org/2001/vcard-rdf/3.0#NICKNAME -  "Jony"
https://www.instagram.com/andrejbardakoski/ - http://www.w3.org/2001/vcard-rdf/3.0#FN -  "Andrej Bardakoski"
7683751d-e3ca-4685-8f49-322d1686f64c - http://www.w3.org/2001/vcard-rdf/3.0#Street -  "Krume Kepeski"
7683751d-e3ca-4685-8f49-322d1686f64c - http://www.w3.org/2001/vcard-rdf/3.0#Pcode -  "1000"
7683751d-e3ca-4685-8f49-322d1686f64c - http://www.w3.org/2001/vcard-rdf/3.0#Locality -  "Skopje"
7683751d-e3ca-4685-8f49-322d1686f64c - http://www.w3.org/2001/vcard-rdf/3.0#Country -  "Macedonia"
```

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.

```
printModelListStatements(model);
System.out.println("-----------------------------------------------------------------
----------------------");
System.out.println("Printing with model.print(), in RDF/XML.");
model.write(System.out);
System.out.println("-----------------------------------------------------------------
----------------------");
System.out.println("Printing with model.print(), in Pretty RDF/XML");
model.write(System.out, "RDF/XML-ABBREV");
System.out.println("-----------------------------------------------------------------
----------------------");
System.out.println("Printing with model.print(), in Turtle");
model.write(System.out, "TURTLE");
System.out.println("-----------------------------------------------------------------
----------------------");
System.out.println("Printing with model.print(), in N-Triples");
model.write(System.out, "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?

   Најкомпактна N-Triple, најлесно се чита TTL, најлесно бо го прочитал програмски во XML

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.

Example1.java | example1_output_N_Triples.nt | Example2.java

```
_:Ba0508da7X2D5521X2D4fcfX2Db68fX2Dab10399dff68 <http://www.w3.org/2001/vcard-rdf/3.0#Family> "Bardakoski" .
_:Ba0508da7X2D5521X2D4fcfX2Db68fX2Dab10399dff68 <http://www.w3.org/2001/vcard-rdf/3.0#Given> "Andrej" .
<https://www.instagram.com/andrejbardakoski/> <http://xmlns.com/foaf/0.1/accountName> "andrejbardakoski" .
<https://www.instagram.com/andrejbardakoski/> <http://xmlns.com/foaf/0.1/age> "22" .
<https://www.instagram.com/andrejbardakoski/> <http://www.w3.org/2001/vcard-rdf/3.0#ADR> _:Bb8794f23X2D6372X2D42a4X2D
<https://www.instagram.com/andrejbardakoski/> <http://www.w3.org/2001/vcard-rdf/3.0#N> _:Ba0508da7X2D5521X2D4fcfX2Db6
<https://www.instagram.com/andrejbardakoski/> <http://www.w3.org/2001/vcard-rdf/3.0#BDAY> "20.10.2001" .
<https://www.instagram.com/andrejbardakoski/> <http://www.w3.org/2001/vcard-rdf/3.0#NICKNAME> "Sashko" .
<https://www.instagram.com/andrejbardakoski/> <http://www.w3.org/2001/vcard-rdf/3.0#NICKNAME> "Bardak" .
<https://www.instagram.com/andrejbardakoski/> <http://www.w3.org/2001/vcard-rdf/3.0#NICKNAME> "Jony" .
<https://www.instagram.com/andrejbardakoski/> <http://www.w3.org/2001/vcard-rdf/3.0#FN> "Andrej Bardakoski" .
_:Bb8794f23X2D6372X2D42a4X2Da5acX2Dfbfc2685358c <http://www.w3.org/2001/vcard-rdf/3.0#Street> "Krume Kepeski" .
_:Bb8794f23X2D6372X2D42a4X2Da5acX2Dfbfc2685358c <http://www.w3.org/2001/vcard-rdf/3.0#Pcode> "1000" .
_:Bb8794f23X2D6372X2D42a4X2Da5acX2Dfbfc2685358c <http://www.w3.org/2001/vcard-rdf/3.0#Locality> "Skopje" .
_:Bb8794f23X2D6372X2D42a4X2Da5acX2Dfbfc2685358c <http://www.w3.org/2001/vcard-rdf/3.0#Country> "Macedonia" .
```

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.

```
model.read("example1_output_N_Triples.nt","N-TRIPLES");
```

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

```
model.write(System.out);
```

14. 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.

```
<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://www.instagram.com/andrejbardakoski/">
    <vcard:FN>Andrej Bardakoski</vcard:FN>
    <vcard:NICKNAME>Jony</vcard:NICKNAME>
    <vcard:NICKNAME>Bardak</vcard:NICKNAME>
    <vcard:NICKNAME>Sashko</vcard:NICKNAME>
    <vcard:BDAY>20.10.2001</vcard:BDAY>
    <vcard:N rdf:parseType="Resource">
      <vcard:Given>Andrej</vcard:Given>
      <vcard:Family>Bardakoski</vcard:Family>
    </vcard:N>
    <vcard:ADR rdf:parseType="Resource">
      <vcard:Country>Macedonia</vcard:Country>
      <vcard:Locality>Skopje</vcard:Locality>
      <vcard:Pcode>1000</vcard:Pcode>
      <vcard:Street>Krume Kepeski</vcard:Street>
    </vcard:ADR>
    <j.0:age>22</j.0:age>
    <j.0:accountName>andrejbardakoski</j.0:accountName>
  </rdf:Description>
</rdf:RDF>
```

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.

```
Resource Andrej = model.getResource(instagramId);
```

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?

```
String formatName = Andrej.getProperty(VCARD.FN).getObject().asLiteral().toString();
Resource name = Andrej.getProperty(VCARD.N).getObject().asResource();
String givenName = name.getProperty(VCARD.Given).getObject().asLiteral().toString();
String familyName = name.getProperty(VCARD.Family).getObject().asLiteral().toString();

System.out.println("formatname: "+formatName);
System.out.println("Given name: "+givenName);
System.out.println("Family name: "+familyName);
```

17. 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.

```
formatname: Andrej Bardakoski
Given name: Andrej
Family name: Bardakoski

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.

```
model.read("hifm-dataset.ttl","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 научената 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<String> labelNames=model.listStatements(new SimpleSelector(null,
RDFS.label,(RDFNode) null))
        .mapWith(x -> x.getObject().asLiteral().toString())
        .toList().stream().sorted().toList();
```

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

```
String tramadolID = "http://purl.org/net/hifm/data#989649";
Resource tramadol = model.getResource(tramadolID);
StmtIterator tramadolStatementsItr = tramadol.listProperties();

System.out.println("All tramadol relations");
while (tramadolStatementsItr.hasNext()) {
    Statement statement = tramadolStatementsItr.nextStatement();
    System.out.println(statement.getObject().toString()
            + " -> " + statement.getPredicate().toString()
            + " -> " + statement.getObject().toString());
}
```

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.

```
String hifm_ont = "http://purl.org/net/hifm/ontology#";
String similarToID=hifm_ont+"similarTo";
tramadolStatementsItr = tramadol.listProperties(model.getProperty(similarToID));
System.out.println("names of all tramadol similar to");
while (tramadolStatementsItr.hasNext()) {
    String objID = tramadolStatementsItr.nextStatement().getObject().toString();
    Resource resource = model.getResource(objID);
    System.out.println(resource.getProperty(RDFS.label).getObject().toString());
}
```

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.

```
String priceID=hifm_ont+"refPriceWithVAT";
float tramadolPrice = tramadol.getProperty(model.getProperty(priceID)).getObject().asLiteral().getFloat();
System.out.println("Tramadol price: "+tramadolPrice);

tramadolStatementsItr = tramadol.listProperties(model.getProperty(similarToID));
System.out.println("prices of all tramadol similar to");
while (tramadolStatementsItr.hasNext()) {
    String objID = tramadolStatementsItr.nextStatement().getObject().toString();
    Resource resource = model.getResource(objID);
    String resourceName = resource.getProperty(RDFS.label).getObject().toString();
    float resourcePrice = resource.getProperty(model.getProperty(priceID)).getObject().asLiteral().getFloat();

    System.out.println(resourceName + " so cena: " + resourcePrice);
}
```

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.

```
Tramadol price: 60.0
prices of all tramadol similar to
TRAMADOL перорални капки 100mg/ml (10ml) so cena: 68.0
TRAMADOL R табл. 30 x 100mg so cena: 209.0
TRAMADOL ALKALOID капс. 20 x 50mg so cena: 60.0
TRAMADOL супп.5 x 100mg so cena: 38.0
TRAMADOL перорални капки 100mg/ml (10ml) so cena: 68.0
TRAMADOL капс. 20 x 50mg so cena: 60.0
TRAMADOL капс. 20 x 50mg so cena: 60.0
```

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.