Vježbe 6 - korištenje koda ovisnog o arhitekturi, korištenje baza podataka

Matej Mihelčić

Prirodoslovno-matematički fakultet, Sveučilište u Zagrebu

matmih@math.hr

16. studenoga, 2022.

Pozivanje *C/C++* funkcija bez parametara iz *Jave*

```
javac ProbaJNI.java -h cwd.
```

```java
public class ProbaJNI {
     public native void helloC();
     static {  System.loadLibrary("HelloWorld"); }

    public static void main(String[] args) {
         new ProbaJNI().helloC();
    }
}
```

Java kod.

```c
#include "probajni_ProbaJNI.h"
JNIEXPORT void JNICALL Java_probajni_ProbaJNI_helloC(JNIEnv
    *env, jobject javaobj)
{
  printf("Hello World: From C");
  return;
}
```

C kod.

Pozivanje *C/C++* funkcija s parametrima osnovnog tipa iz *Jave*

```
1 public class Parametri {
2   public native double ZbrojiDouble(double a, double b);
3       static {System.loadLibrary("HelloWorld"); }
4
5  public static void main(String args[]){
6 double res = new Parametri().ZbrojiDouble(45.67, 123.78);
7 System.out.println("Rezultat je: "+res);//Rezultat je:
      169.45
8 }  }
```

Java kod.

```
1 #include "probajni_Parametri.h"
2 JNIEXPORT jdouble JNICALL
      Java_probajni_Parametri_ZbrojiDouble(JNIEnv *env,
      jobject javaobj, jdouble x, jdouble y){
3      return x+y;
4 }
```

C kod.

Pozivanje *C/C++* funkcija koje imaju polja kao parametre iz *Jave*

```java
public class Polja {
   public native double sumaElemenata(double [] ulaz);
   static { System.loadLibrary("HelloWorld"); }

    public static void main (String args[]){
        double p[] = {1.0,12.0,34.5,12.67,9.4,18.98};
        double res = new Polja().sumaElemenata(p);
        System.out.println("Suma elemenata: "+res); }   }
```

Java kod.

```c
#include "probajni_Polja.h"
JNIEXPORT jdouble JNICALL Java_probajni_Polja_sumaElemenata(
    JNIEnv *env, jobject javaobj, jdoubleArray polje){
    jdouble *arr = env->GetDoubleArrayElements(polje, NULL);
    double res=0.0; int size = env->GetArrayLength(polje);
    for(int i=0;i<size;i++) res+=arr[i];
    env->ReleaseDoubleArrayElements(polje, arr, 0);
    return res; }
```

C kod.

Pozivanje *C/C++* funkcija koje imaju referencirane objekte kao parametre iz *Jave*

```
javac Klasa.java Brojac.java -h cwd.
```

```java
public class Klasa {
    public native void povecajBroj(Brojac brojaci[]);
    static { System.loadLibrary("HelloWorld"); }

    public static void main(String args[]){
        Brojac polje[] = new Brojac[10];
       for(int i=0;i<10;i++){
         polje[i] = new Brojac(); polje[i].namjesti(i); }
         for(int i=0;i<10;i++)
            System.out.print(polje[i].vrati()+" ");
        System.out.println();
        //0 1 2 3 4 5 6 7 8 9
        new Klasa().povecajBroj(polje);
        //1 2 3 4 5 6 7 8 9 10
        for(int i=0;i<10;i++)
            System.out.print(polje[i].vrati()+" ");
        System.out.println(); } }
```

Java kod.

Pozivanje *C/C++* funkcija koje imaju referencirane objekte kao parametre iz *Jave*

```java
public class Brojac {
    int broj;

    Brojac(){ broj = 0;}
    public void povecaj(){broj++;}
    public int vrati(){return broj;}
    public void namjesti(int _broj){broj = _broj;}
}
```

Java kod.

Kod dohvaćanja metoda preko pokazivača na JNI okruženje `JNIEnv *env`, treba specificirati prototip po pravilima definiranja prototipa *Java* metoda (vidi predavanja).

Pozivanje *C/C++* funkcija koje imaju referencirane objekte kao parametre iz *Jave*

```c
#include "probajni_Klasa.h"
JNIEXPORT void JNICALL Java_probajni_Klasa_povecajBroj(
    JNIEnv *env, jobject javaobj, jobjectArray polje){
    int broj, i;
    int size = env->GetArrayLength(polje);
    jobject objekt;
    for(i=0;i<size;i++){
    objekt = (jobject) env->GetObjectArrayElement(polje, i);
    jclass brojacClass = env->GetObjectClass(objekt);
    jmethodID vrati = env->GetMethodID(brojacClass, "vrati",
     "()I");
    jmethodID namjesti = env->GetMethodID(brojacClass, "
    namjesti", "(I)V");
     broj =  env->CallIntMethod(objekt, vrati);
     broj = broj+1;
     env->CallVoidMethod(objekt, namjesti,broj);
    }
}
```

C kod.

Call<Type>Method funkcije u *JNI*-u

JNI sadrži familiju funkcija oblika `Call<Type>Method`, gdje je `type` tip povratnog tipa podataka (svi osnovni tipovi iz *Jave*) i `Object`. Prototip tih funkcija je oblika:

```
<NativeType> Call<Type>Method(JNIEnv *env, jobject obj,
jmethodID methodID, jvalue *args);.
```

Call<Type>Method	<NativeType>
CallVoidMethod	void
CallObjectMethod	jobject
CallBooleanMethod	jboolean
CallByteMethod	jbyte
CallCharMethod	jchar
CallShortMethod	jshort
CallIntMethod	jint
CallLongMethod	jlong
CallFloatMethod	jfloat
CallDoubleMethod	jdouble

Kreiranje *Java* objekta iz *C/C++*-a koristeći *JNI*

```java
public class Klasa1 {
     public native void kreirajBrojace(Brojac brojaci[]);
    static {
        System.loadLibrary("HelloWorld");
    }

    public static void main(String args[]){
        Brojac polje[] = new Brojac[10];
        new Klasa1().kreirajBrojace(polje);

        for(int i=0;i<10;i++)
            System.out.print(polje[i].vrati()+" ");
        System.out.println();
        //0 1 4 9 16 25 36 49 64 81
    }
}
```

Java kod.

Kreiranje *Java* objekta iz *C/C++*-a koristeći *JNI*

```c
#include "probajni_Klasa1.h"
#include <math.h>
JNIEXPORT void JNICALL Java_probajni_Klasa1_kreirajBrojace(
    JNIEnv *env, jobject javaobj, jobjectArray polje){
      int i;
     int size = env->GetArrayLength(polje);
     jobject objekt;
     jclass brojacClass = env->FindClass("Lprobajni/Brojac;")
    ;
     for(i=0;i<size;i++){
     jmethodID konstruktor = env->GetMethodID(brojacClass,
    "<init>", "()V");
     jmethodID namjesti = env->GetMethodID(brojacClass, "
    namjesti", "(I)V");
     objekt = env->NewObject(brojacClass, konstruktor);
      env->CallVoidMethod(objekt, namjesti,(int)pow(i,2));
      env->SetObjectArrayElement(polje,i,objekt);
     }
}
```

C kod.

Korištenje C/C++ funkcija koje koriste povezane biblioteke

- Osigurati da je korištena biblioteka povezana s projektom dinamičke biblioteke `Build Options->Linker Settings->Link Libraries` te da je unutar `Search directories->Compiler/Linker` dodan direktorij koji sadrži biblioteku i odgovarajuću datoteku zaglavlja (najjednostavnije koristiti statičku biblioteku s odgovarajućom datotekom zaglavlja).
- Osigurati da korištena biblioteka nema dodatnih ovisnosti inače će doći do **Linkage error**.
- Prevesti projekt koristeći `Release` zastavicu (namijenjeno korištenju u produkciji). Za razliku od toga `Debug` služi za testiranje.

Ideja je da funkcije povezanih *C/C++* biblioteka koristimo unutar funkcija koje se pozivaju pomoću *JNI*-a, čime efektivno proširimo mogućnosti programa (iako ga učinimo ovisnoga o platformi!). Primjetimo da je *Java* kod i dalje neovisan o platformi/arhitekturi međutim, dinamičku biblioteku moramo ponovo stvoriti ovisno o platformi i arhitekturi.

Korištenje *Jave* iz *C/C++*-a

Java se može koristiti i kao **pomoćno sredstvo** izračunavanja unutar *C/C++* programa. U tom slučaju iz *C/C++*-a treba pokrenuti instancu *Java virtualnog stroja* i zatim raditi **komunikaciju** s *Javom* koristeći *JNI* pozive.

```c
#include <stdio.h>
#include <stdlib.h>
#include <jni.h>
#include <windows.h>

typedef jint (JNICALL *JNI_CreateJavaVM_func)(JavaVM **pvm,
    void **penv, void *args);

JNI_CreateJavaVM_func JNI_CreateJavaVM_ptr;

int main(int argc, char **argv)
{
  JavaVM *vm; JNIEnv *env; JavaVMInitArgs vm_args;
  jint res; jclass cls; jmethodID mid; jstring jstr;
```

C kod.

Korištenje *Jave* iz *C/C++*-a

```
1  jobjectArray main_args; JavaVMOption options[1];
2  vm_args.version  = JNI_VERSION_1_8;
3  vm_args.nOptions = 1;
4  options[0].optionString = "-Djava.class.path=putanja\\dist
     \\Proba2.jar";
5  vm_args.options = options;
6  HMODULE jvm_dll;
7  jvm_dll = LoadLibrary("C:\\Program Files\\Java\\jdk
     -15.0.1\\bin\\server\\jvm.dll");
8    if(jvm_dll == NULL) { printf("ucitavanje DLL-a
     neuspjesno\n"); exit(1); }
9   JNI_CreateJavaVM_ptr= (JNI_CreateJavaVM_func)
     GetProcAddress(jvm_dll,"JNI_CreateJavaVM");
10  if(JNI_CreateJavaVM_ptr == NULL) { printf("ne mogu
     ucitati funkciju\n"); exit(1); }
11  res = JNI_CreateJavaVM_ptr(&vm, (void**)&env, &vm_args);
12 if (res != JNI_OK) {
13   printf("Stvaranje Java VS nije uspjelo\n");
14   return 1; }
```

C kod.

Korištenje *Jave* iz *C/C++*-a

```
1   cls = (*env)->FindClass(env, "Ltocka/main;");
2   if (cls == NULL) {
3     printf("Nije uspjelo trazenje glavne klase\n");
4     return 1; }
5   mid = (*env)->GetStaticMethodID(env, cls, "main", "([Ljava
      /lang/String;)V");
6   if (mid == NULL) {
7     printf("Nije uspjelo ucitavanje funkcije main\n");
8     return 1;
9   }
10  jstr = (*env)->NewStringUTF(env, "");
11  main_args = (*env)->NewObjectArray(env, 1, (*env)->
      FindClass(env, "java/lang/String"), jstr);
12  (*env)->CallStaticVoidMethod(env, cls, mid, main_args);
13  //ispise izlaz klase main.java iz paketa tocka
14  //pazite da pokrecete koristeci 64-bitno okruzenje
15  //ili pokrenite u terminalu
16  return 0;
17 }
```

C kod.

Korištenje *Java* iz *C/C++*-a

Napomene: Ukoliko se prevođenje vrši strožim prevodiocem kao što je `gcc`, tada treba strogo razlikovati sintaksu `C`-a od `C++`-a unutar `JNI`-a. Kod `C` načina korištenja `JNI` funkcija, `JNIEnv * env` je pokazivač na pokazivač na strukturu JNIEnv, stoga na svim mjestima u kodu trebamo imati `(*env)->pozivFunkcije`. Također kod svake funkcije koju pozivamo trebamo staviti `env` kao parametar.
Npr. `(*env)->findClass(env,"LTocka/main;)"`. Kod `C++` poziva (obavezno prevoditi koristeći `g++`), ispuštamo `env` iz poziva funkcije i `JNIEnv *env` je pokazivač na strukturu tipa `JNIEnv`.
Npr. `env->findClass("LTocka/main;)"`.

Povezivanje *Java* programa s *R*-om

```
1  import org.rosuda.JRI.Rengine;
2
3  public class Kombinacija {
4    public static void main(String[] args) {
5      String vektor = "c(1,2,3,4,5)"; //Java reprezentacija
       deklaracije R vektora
6      double vektorJava[] = {10,45.3,23.7,12.8,12.6}; //obicno
        Java polje (vektor 2)
7      Rengine engine = Rengine.getMainEngine(); //dohvacamo R
       okruzenje
8      if(engine == null){
9        engine=new Rengine (new String [] {"--vanilla"}, false
       , null); //ukoliko ne postoji R instanca, stvaramo novu
10     if (!engine.waitForR())
11         {
12       System.out.println ("R se ne moze ucitati");
13           return;
14         }
15     }
```

Korištenje R-a unutar Java programa.

Povezivanje *Java* programa s *R*-om

```
1   engine.eval("rVektor<-" + vektor);//spremamo vektor u R
      vektor
2   double vekRToJava[] =  engine.eval("rVektor<-rVektor*
      rVektor").asDoubleArray();//kvadrat spremamo u Java
      polje
3
4           String newRVektor = "c(";
5           for(int i=0;i<vekRToJava.length;i++){
6               vekRToJava[i]+=vektorJava[i];
7               if(i+1<vekRToJava.length)
8                   newRVektor+=vekRToJava[i]+",";
9               else  newRVektor+=vekRToJava[i]+")";
10          } //stvaramo R reprezentaciju vektora vekRToJava
11
12  engine.eval("novirVektor<-" + newRVektor); //stvaramo novi
      vektor unutar R-a
```

Korištenje R-a unutar Java programa.

Povezivanje *Java* programa s *R*-om

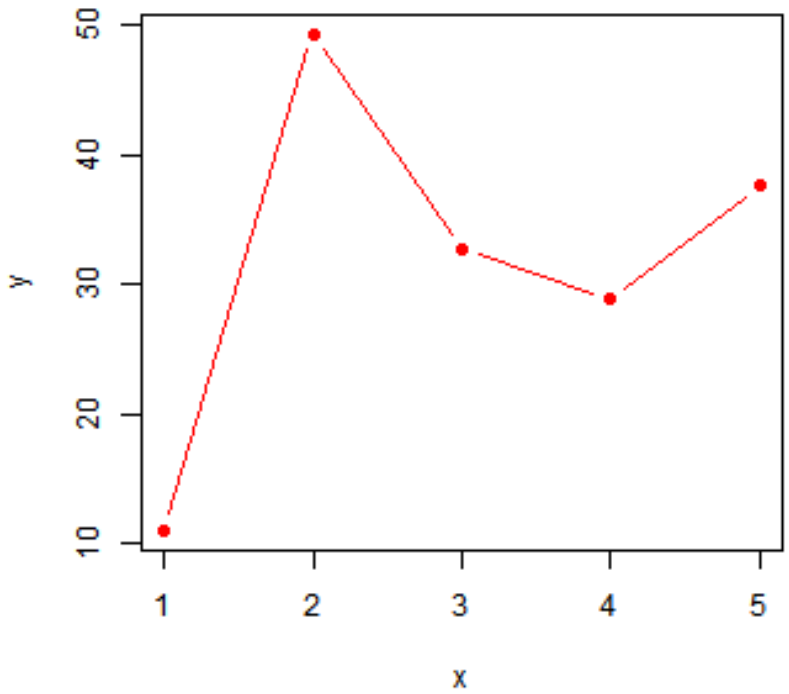
```
1  double std = engine.eval("sd(novirVektor)").asDouble();
2  double srednjaVrijednost = engine.eval("mean(novirVektor)")
       .asDouble();
3  //racunamo srednju vrijednost i standardnu devijaciju
       vektora
4   engine.end();
5
6      System.out.println("Vektor: ");
7      for(int i=0;i<vekRToJava.length;i++)
8          System.out.print(vekRToJava[i]+" ");//ispis vektora
9          //11.0 49.3 32.7 28.8 37.6
10     System.out.println();
11     System.out.println("Prosjek: "+srednjaVrijednost+"
       standardna devijacija: "+std);
12     //Prosjek: 31.88 standardna devijacija:
       13.984884697415277
13     }
14 }
```

Korištenje R-a unutar Java programa.

Povezivanje *Java* programa s *R*-om

```
1 engine.eval("png(\"rplot.png\", width = 350, height = 350)"
    ); //definiramo sliku u .png formatu dimenzija 350x350
2 engine.eval("plot(novirVektor, type = \"b\", pch = 19, col =
    \"red\", xlab = \"x\", ylab = \"y\")"); //napravimo
    line plot
3 engine.eval("dev.off()");//ugasimo uredaj za crtanje/prikaz
    slika
```

Korištenje R-a unutar Java programa.



Povezivanje *Java* programa s *R*-om

```
1  import java.util.Random;
2  import org.rosuda.JRI.Rengine;
3
4  public class Funkcija {
5    public static void main(String args[]){
6      Rengine engine = Rengine.getMainEngine();
7        if(engine == null){
8          engine=new Rengine (new String [] {"--vanilla"},
       false, null);
9        if (!engine.waitForR())
10         {
11             System.out.println ("Cannot load R");
12             return;
13         }
14     }
15
16  String rFunkcija = "C:/f1/f2/f3/crtajHistogram.R"; //
      primijetite drugaciji separator!
```

Poziv R funkcije unutar Java programa.

Povezivanje *Java* programa s *R*-om

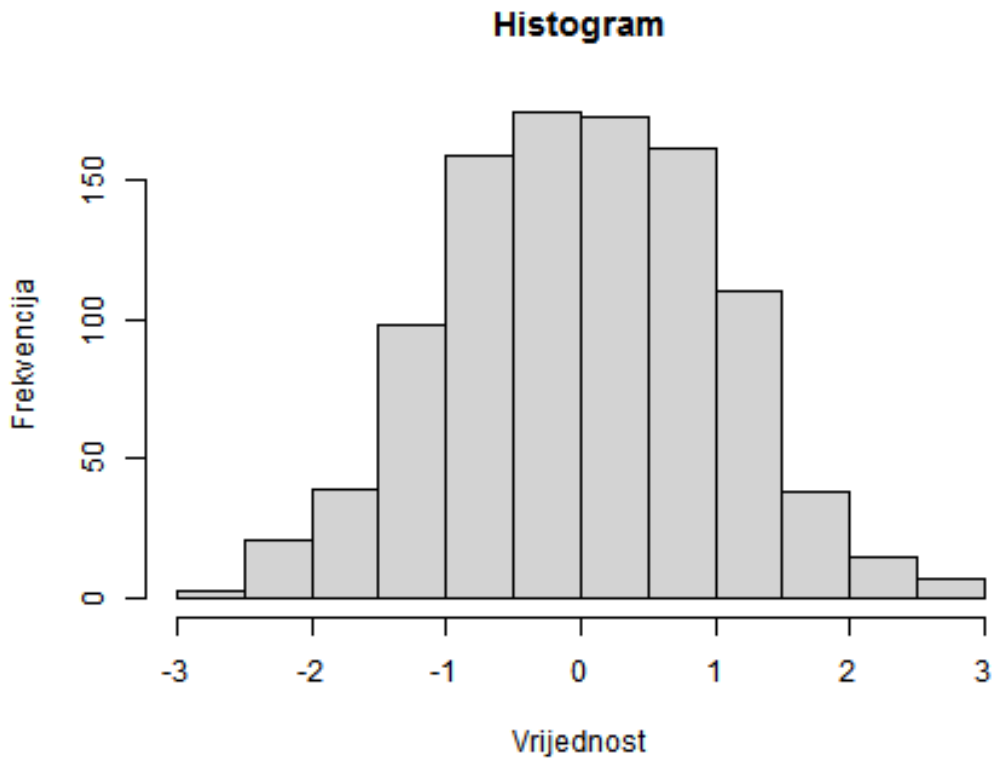
```
1  engine.eval("source('" +rFunkcija + "')");//jako bitno,
     koristiti ' '
2     double brojevi[] = new double[1000];
3     Random rand = new Random();
4     for(int i=0;i<1000;i++)
5       brojevi[i]=rand.nextGaussian();//generiramo u Javi
     1000 brojeva iz N(0,1)
6     String newRVektor = "c(";
7         for(int i=0;i<brojevi.length;i++){
8             if(i+1<brojevi.length)
9                 newRVektor+=brojevi[i]+",";
10            else  newRVektor+=brojevi[i]+")";
11        }//stvorimo Java string naredbe kreiranja R vektora
12 engine.eval("x<-"+newRVektor);//spremimo u x unutar R-a
13 engine.eval("crtajHistogram(x,\"normalni.png\")");
14 //funkcija crta histogram i spremi u datoteku "normalni.
     png" unutar radnog direktorija projekta
15    engine.end();
16  } }
```

Poziv R funkcije unutar Java programa.

Povezivanje *Java* programa s *R*-om

```
1 crtajHistogram<-function(ulaz, putanja){
2   png(putanja,width = 450, height = 350);
3   hist(ulaz,xlab = c("Vrijednost"),ylab = c("Frekvencija"),
      main = "Histogram");
4   dev.off();
5 }
```

R funkcija.



Povezivanje *Java* programa s *R*-om

Ukoliko nisu postavljene sistemske varijable R-a na UNIX-u:
jedan način da se pokrenu programi je:

- Postaviti R_HOME u terminalu (`export R_HOME=putanja`)
- Pokrenuti programsko okruženje u tom istom terminalu
- Postaviti `-Djava.library.path` varijablu ili na direktorij projekta (pa kopirati odgovarajuće biblioteke) ili direktno na direktorij koji sadrži biblioteke `jri` i `rJava` (treba dodati `lib` prije imena biblioteka).
- Uključiti dinamičke biblioteke unutar izvornog koda *Java* programa koristeći naredbu `System.loadLibrary("ime")`.

Rad s *SQLite* bazom podataka

```
1  import java.sql.Connection;
2  import java.sql.DatabaseMetaData;
3  import java.sql.DriverManager;
4  import java.sql.SQLException;
5
6   public static void main(String[] args) {
7        String imeBaze = "test.db";
8        String url = "jdbc:sqlite:" + imeBaze;
9
10     try (Connection conn = DriverManager.getConnection(url)){
11        if (conn != null) {
12           DatabaseMetaData meta = conn.getMetaData();
13           System.out.println("Ime biblioteke za rad s bazom
       podataka " + meta.getDriverName());
14          System.out.println("Stvorena je nova baza.");
15               }
16           } catch (SQLException e) {
17               System.out.println(e.getMessage());
18           } }    }
```

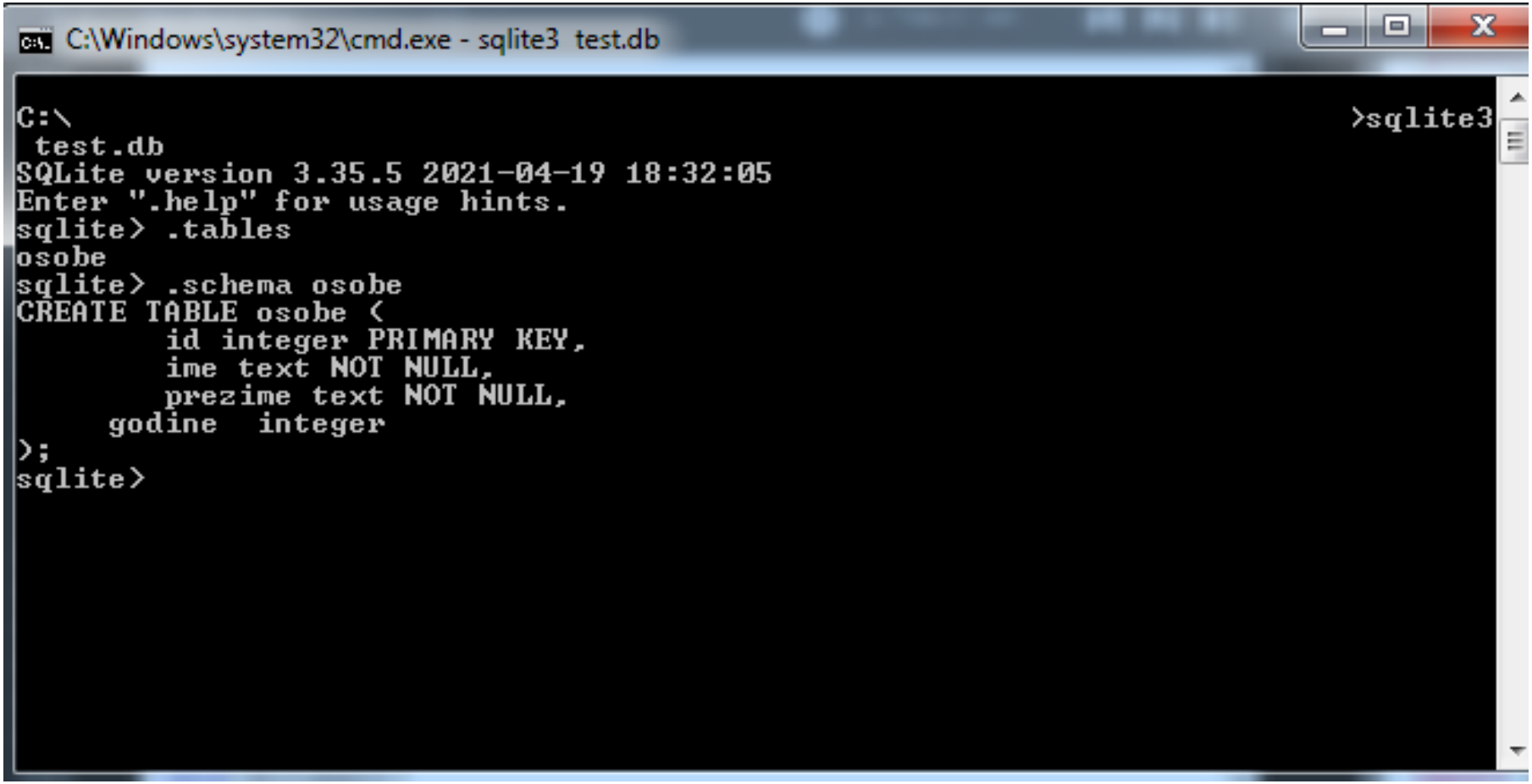
Stvaranje sqlite baze podataka.

Rad s *SQLite* bazom podataka

```
1  import java.sql.Connection;
2  import java.sql.DatabaseMetaData;
3  import java.sql.DriverManager;
4  import java.sql.SQLException;
5
6   public static void main(String[] args) {
7          String fileName = "test.db";
8          String url = "jdbc:sqlite:" + fileName;
9     String sql = "CREATE TABLE IF NOT EXISTS osobe (\n"
10                  + " id integer PRIMARY KEY,\n"
11                  + " ime text NOT NULL,\n"
12                  + " prezime text NOT NULL,\n"
13                  + "     godine  integer\n"
14                  + ");";
15    try (Connection conn = DriverManager.getConnection(url);
      Statement stmt = conn.createStatement()) {
16     if (conn != null) {stmt.execute(sql);}
17          } catch (SQLException e) {
18              System.out.println(e.getMessage()); } }
```

Stvaranje tablice sqlite baze podataka.

Rad s *SQLite* bazom podataka

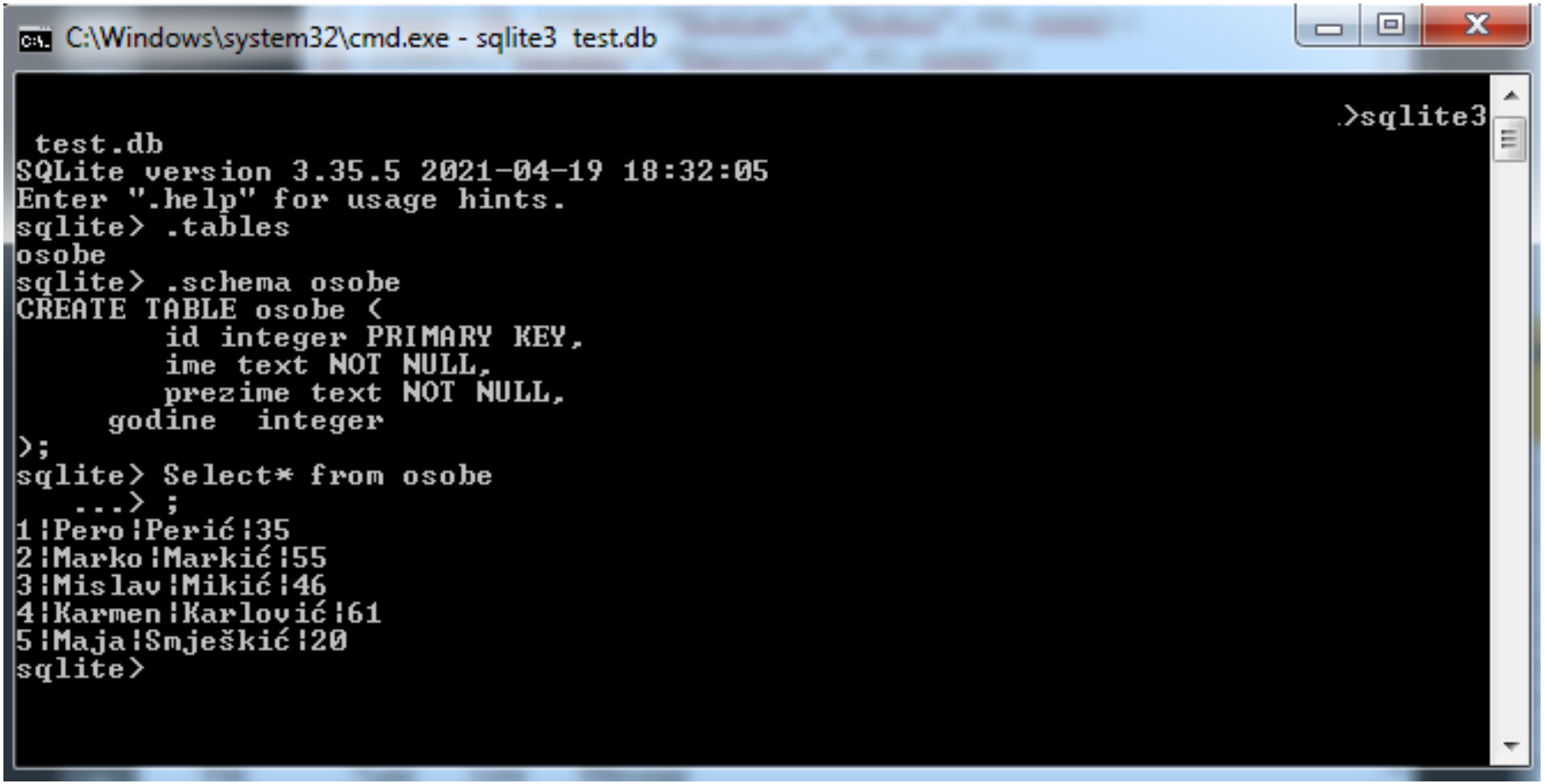


Rad s *SQLite* bazom podataka

```
1  public static void insert(String ime, String prezime, int
       godine, Connection conn) {
2       String sql = "INSERT INTO osobe(ime,prezime,godine)
       VALUES(?,?,?)";
3
4    try{
5    PreparedStatement pstmt = conn.prepareStatement(sql);
6              pstmt.setString(1, ime);
7              pstmt.setString(2, prezime);
8               pstmt.setInt(3, godine);
9              pstmt.executeUpdate();
10          }
11          catch(SQLException e){ } }
12
13           //poziv u programu
14   Ub.insert("Pero","Peric",35,conn);Ub.insert("Marko","
       Markic",55,conn);Ub.insert("Mislav","Mikic",46,conn); Ub
       .insert("Karmen","Karlovic",61,conn); Ub.insert("Maja","
       Smjeskic",20,conn);
```

Ubacivanje podataka u tablicu.

Rad s *SQLite* bazom podataka



Rad s *SQLite* bazom podataka

```
1  public static void selectAll(Connection conn){
2       String sql = "SELECT id, ime, prezime, godine FROM
       osobe";
3
4      try{
5        Statement stmt  = conn.createStatement();
6        ResultSet rs    = stmt.executeQuery(sql);
7        while (rs.next()) {
8          System.out.println(rs.getInt("id") +  "\t" +
9          rs.getString("ime") + "\t" +
10         rs.getString("prezime")+"\t"+
11         rs.getInt("godine"));
12             }
13         } catch (SQLException e) {
14             System.out.println(e.getMessage());
15         }
16     }
```

Selekcija elemenata iz tablice.

Rad s *SQLite* bazom podataka

```
1  public static void delete(int id, Connection conn) {
2          String sql = "DELETE FROM osobe WHERE id = ?";
3
4          try{
5
6     PreparedStatement pstmt = conn.prepareStatement(sql);
7              pstmt.setInt(1, id);
8              pstmt.executeUpdate();
9          } catch (SQLException e) {
10             System.out.println(e.getMessage());
11         }
12      }
```

Brisanje elemenata iz tablice.

Rad s *SQLite* bazom podataka

```
1  public static void update(int id, String ime, String
       prezime, int godine, Connection conn) {
2          String sql = "UPDATE osobe SET ime = ? , "
3                  + "prezime = ? , "
4                  + "godine = ? "
5                  + "WHERE id = ?";
6
7          try{
8          PreparedStatement pstmt = conn.prepareStatement(sql);
9              pstmt.setString(1, ime);
10             pstmt.setString(2, prezime);
11             pstmt.setInt(3, godine);
12             pstmt.setInt(4, id);
13             pstmt.executeUpdate();
14         } catch (SQLException e) {
15             System.out.println(e.getMessage());
16         }
17     }
```

Mijenjanje postojećih elemenata iz tablice.