Sveučilište u Splitu

Sveučilišni odjel za stručne studije

Odsjek za informacijsku tehnologiju

Praktikum za laboratorijske vježbe kolegija

Programiranje u Javi

Izradio:

pred. Josip Vrlić, dipl. ing. rač.

Rujan, 2018.

1 Sadržaj

2 Uvod

Laboratorijske vježbe se izvode sukladno Syllabusu kolegija Programiranje u Javi, a koji je objavljen na web stranicama Sveučilišnog odjela za stručne studije: https://moodle.oss.unist.hr.

Kolegij Programiranje u Javi ima definirane sljedeće ciljeve:

- programiranje u programskom jeziku Java,
- poznavanje objektno orijentirane paradigme u programskom jeziku Java,
- korištenje programskog jezika Java u različitim tehnologijama i na različitim platformama.

Očekivani ishodi učenja na razini kolegija su:

- poznavati strukturu i model programskog jezika Java, (*znanje*)
- koristiti programski jezik Java za različite programske tehnologije, (*razumijevanje*)
- izraditi programsku podršku u programskom jeziku Java, (*primjena*)
- ocijeniti korisničkih zahtjeva za funkcionalnostima tražene programske podrške kako bi se odlučilo da li programski jezik Java može ispuniti korisničke zahtjeve, (*analiza*)
- predložiti korištenje određenih tehnologija implementirajući ih u programskom jeziku Java za rješavanje zadanih problema, (*sinteza*)
- izabrati inženjerski pristup u rješavanju problema, polazeći od usvojenih znanja iz programiranja i poznavanja rada operativnih sustava. (*vrednovanje*)

Sukladno definiranim ishodima učenja u ovom praktikumu je opisano ukupno 8 vježbi sa sljedećom tematikom:

1. Upoznavanje NetBeans razvojnog okruženja (1 tjedan)
2. Klase i objekti (1 tjedan)
3. Nasljeđivanje i apstraktne klase (3 tjedna)
4. I/O – datoteke (1 tjedan)
5. GUI (2 tjedna)
6. Dretve (1 tjedan)
7. Upoznavanje Intellij IDEA Android razvojnog okruženja (1 tjedan)
8. Android aplikacija (3 tjedna)

Svaka vježba ima opciju sadržaja A i B, čime su definirane različite vježbe za istu temu.

Kao uvjet za pristupanje ispitu nužno je riješiti, odnosno izraditi svih 8. laboratorijskih vježbi.

3 Prva (1.) vježba

Tema: Upoznavanje NetBeans razvojnog okruženja

Trajanje izrade: 1 tjedan

Opis:

NetBeans IDE je službeni IDE (engl. Integrated Development Environment) za Java 8. Koristeći uređivač koda, analizator koda, konvertere moguće je jednostavno nadograditi aplikaciju kako bi se koristili nove Java 8 mogućnosti, kao što su lambda izrazi, funkcijski operatori i reference na metode.

NetBeans IDE je integrirano razvojno okruženje i omogućuje razvoj svih vrsta Java aplikacija (Java SE, JavaFX, Java ME, web, EJB i mobilne aplikacije). Među ostalim značajkama je projektni sustav baziran na Ant, Maven podršci, ugrađenim konverterima izvornog koda, kao i različitim kontrolama verzije aplikacija.

IDE je puno više od uređivača teksta, te je moguće koristiti različite alate specifične za rad sa izvornim kodom. Vrlo jednostavno je koristiti uvlake koda, uparivati ključne riječi i zagrade, kao i označavati izvorni kod u sintaksnom i semantičkom smislu. Uređivač koda ima podršku za više programskih i skriptnih jezika, kao što su Java, C/C++, XML, HTML, PGP, Groovy, Javadoc, JavaScript i JSP. S obzirom kako je uređivač koda proširiv, moguće je dodati podršku i za druge programske i skriptne jezike.

Olakšan je rad s velikim projektima koji sadrže mnogo datoteka, mapa i izvornog koda. NetBeans IDE omogućava različit pogled na podatke, od više prozora za pregled projekata, do pomoćnih alata za podešavanje i upravljanje aplikacijama. Također je podržan rad s verzijama integrirajući vanjske alate za Subversion, Mercurial ili Git.

Podržane su i sve funkcionalnosti vezane za kompajliranje, debugiranje, buildanje i pokretanje projekata.

Sadržaj A:

1. Zadatak

S web stranice https://netbeans.org skinite zadnju verziju *NetBeans IDE for Java SE* razvojnog okruženja. Prije instalacije provjeriti da su ispunjeni svi traženi zahtjevi (engl. system requirements). Instalirati i pokrenuti razvojno okruženje NetBeans.

2. Zadatak

Prepišite/kopirajte donji izvorni kod, te ga prevedite i pokrenite. Analizirajte funkcionalnosti.

Testirajte debugiranje u razvojnom okruženju.

```
import java.awt.Color;

public class ColorConverter {

	public static void main(String[] args) {

		String hexColor = "0x1FF0FF";

		Color c = Color.decode(hexColor);
```

```
            float[] hsbCode = new float[3];

            Color.RGBtoHSB(c.getRed(), c.getGreen(), c.getBlue(),
            hsbCode);

            System.out.println("Boja u HEX formatu: 0x" +
            Integer.toHexString(c.getRGB() & 0x00FFFFFF));

            System.out.println("Boja u RGB formatu: " + c.getRed() +
            ", " + c.getGreen() + ", " + c.getBlue());

            System.out.println("Boja u HSB formatu: " + hsbCode[0] *
            360 + "°, " + hsbCode[1] * 100 + "%, " + hsbCode[2] * 100
            + "%");

        }

}
```

3. Zadatak

Napišite svoju klasu Color tako da isti izvorni kod iz 2. zadatka radi s vašom klasom, a ne s klasom iz biblioteke java.awt.

Nadogradite svoju klasu Color tako da može konvertirati boju u dodatne HSL i CMYK formate boja.

Sadržaj B:

1. Zadatak

S web stranice https://netbeans.org skinite zadnju verziju *NetBeans IDE for Java SE* razvojnog okruženja. Prije instalacije provjeriti da su ispunjeni svi traženi zahtjevi (engl. system requirements). Instalirati i pokrenuti NetBeans razvojno okruženje.

2. Zadatak

Prepišite/kopirajte donji kod, te ga prevedite i pokrenite. Analizirajte funkcionalnosti.

Testirajte debugiranje u razvojnom okruženju.

```
1.  /*
2.          Generate CRC32 Checksum For Byte Array Example
3.          This Java example shows how to get the CRC32 checksum
    value for
4.          array of bytes using CRC32 Java class.
5.  */
6.
7.  import java.util.zip.CRC32;
8.  import java.util.zip.Checksum;
9.
10.       public class CalculateCRC32ForByteArray {
11.
12.               public static void main(String args[]){
13.
14.                       String str = "Generate CRC32 Checksum For
    Byte Array Example";
15.
16.                       //Convert string to bytes
17.                       byte bytes[] = str.getBytes();
```

```
18.
19.                     Checksum checksum = new CRC32();
20.
21.                     /*
22.                      * To compute the CRC32 checksum for byte
   array, use
23.                      *
24.                      * void update(bytes[] b, int start, int
   length)
25.                      * method of CRC32 class.
26.                      */
27.
28.                     checksum.update(bytes,0,bytes.length);
29.
30.                     /*
31.                      * Get the generated checksum using
32.                      * getValue method of CRC32 class.
33.                      */
34.                     long lngChecksum = checksum.getValue();
35.
36.                     System.out.println("CRC32 checksum for
   byte array is :" + lngChecksum);
37.             }
38.
39.     }
40.
41.     /*
42.     Output of this program would be
43.     CRC32 checksum for byte array is :3510043186
44.     */
```

3. Zadatak

Napišite svoje klase Checksum i CRC32 tako da isti izvorni kod iz 2. zadatka radi s vašim klasama, a ne klasama iz biblioteka java.util.zip.

4 Druga (2.) vježba

Tema: Klase i objekti

Trajanje izrade: 1 tjedan

Opis:

Kao i u svim drugim objektno orijentiranim jezicima, a što je jedna od osnovnih karakteristika, u Javi je omogućen rad s objektima i klasama. Klasa predstavlja predložak za objekt i sadrži podatkovne i funkcijske članove. Objekt je instanca ili primjerak klase, te predstavlja model stvarnog ili apstraktnog objekta.

Java podržava i ostale karakteristike objektno orijentiranih jezika: enkapsulaciju, nasljeđivanje i polimorfizam.

Klasa u Javi

Klasa je skup objekata koji imaju zajedničke osobine definirane metodama (funkcijskim članovima) i podacima (podatkovnim članovima). Klasa u Javi se definira ključnom riječju *class* i može sadržati:

- podatak,
- metodu,
- konstruktor,
- druge dijelove.

Konstruktor predstavlja posebnu metodu (bez povratne vrijednosti istog naziva kao i naziv klase) koja se automatski izvršava prilikom kreiranja objekta. Ukoliko u klasi nije definiran niti jedan kontruktor, prilikom kompjaliranja automatski će se generirati zadani konstruktor bez ulaznih parametara.

Primjer klase: Automobil, Vatrogasac, Žarulja, …

Objekt u Javi

Objekt predstavlja instancu određene klase, te je svaki objekt jedinstven jer kreiranjem objekt dobiva svoj memorijski prostor u kojem se nalaze podatkovni članovi. Podatkovni članovi određuju stanje, a funkcijski članovi ponašanje objekta te mogu mijenjati stanje objekta.

U Javi se objekt kreira koristeći ključnu riječ (operator) *new*.

Primjer kreiranja objekta klase Automobil:

```
Automobil merc = new Automobil();
```

Sadržaj A:

1. Zadatak

Kao nastavak 3. zadatka iz 1. laboratorijske vježbe implementirajte u svojoj klasi Color isto sučelje i punu funkcionalnost kao i u originalnoj klasi java.awt.Color (https://docs.oracle.com/javase/7/docs/api/java/awt/Color.html).

2. Zadatak

Napraviti „tester klasu“ koja testira napravljenu klasu Color iz prethodnog zadatka.

Sadržaj B:

1. Zadatak
Kao nastavak 3. zadatka iz 1. laboratorijske vježbe implementirajte u svojoj klasi CRC32 isto sučelje i punu funkcionalnost kao i u originalnoj klasi java.util.zip.CRC32 (https://docs.oracle.com/javase/7/docs/api/java/util/zip/CRC32.html)

2. Zadatak
Napraviti „tester klasu“ koja testira napravljenu klasu CRC32 iz prethodnog zadatka.

5 Treća (3.) vježba

Tema: Nasljeđivanje, apstraktne klase i polimorfizam

Trajanje izrade: 3 tjedna

Opis:

U Javi je podržana mogućnost nasljeđivanja klasa. Nasljeđivanje je mogućnost definiranja nove klase koja ima svojstva već postojeće Java klase. Nasljeđivanje predstavlja IS-A relaciju između dvije klase. Nasljeđivanje se koristi zbog ponovnog korištenja postojećeg izvornog koda (engl. reusability) uz mogućnost proširivanja funkcionalnosti (dodavanje novih podatkovnih i funkcijskih članova), ili izmjene postojećih funkcionalnosti (preklapanje podatkovnih u funkcijskih članova).

Postojeća klasa se zove nadklasa, roditelj (engl. superclass), a novo definirana klasa koja je naslijedila postojeću klasu se zove podklasa, dijete (engl. subclass).

Ključna riječ u Javi za korištenje nasljeđivanja je *extends*.

Primjer:

```
class B extends A {

}
```

Klasa A je nadklasa, a klasa B je podklasa.

Podklasa nasljeđuje ona svojstva nadklase sukladno definiranoj enkapsulaciji nadklase.

Kada ne možemo implementirati neki funkcijski član klase (tijelo metode) moramo ga definirati kao apstraktni funkcijski član. Ukoliko klasa sadrži barem jedan funkcijski član definiran kao apstraktan, i sama klasa mora biti definirana kao apstraktna.

Ključna riječ u Javi za definiranje klase ili funkcijskog člana klase kao apstraktnog je *abstract*.

Abstraktna klasa može za razliku od sučelja definirati podatkovne članove i implementirane funkcijske članove.

S obzirom da nije moguće kreirati/instancirati objekt apstraktne klase, iste nemaju smisla, osim da se iste naslijedi, te u naslijeđenoj klasi izvrši implementacija apstraktnih metoda. Ukoliko je naslijeđena klasa također definirana kao apstraktna, tada nije nužno implementirati apstraktne funkcijske članove nadklase.

Polimorfizam je koncept kojim možemo jednu funkciju (ponašanje) izvršiti na različite načine. U Javi se polimorfizam implementira korištenjem nasljeđivanje i implementacijom apstraktnog ili preopterećenjem (engl. override) funkcijskog člana u podklasi, nakon čega je moguće preko reference tipa nadklase koja pokazuje na objekt tipa podklase pozivati funkcijski član podklase.

Primjer:

```
class A {
      abstract void foo();
}
```

```
class B extends A{
      void foo(){
            System.out.println("foo");
      }
}
A a = new B();
a.foo();
```

Sadržaj A:

1. Zadatak

Proučiti MQTT komunikacijski protokol koji je standard za povezivanje raznih uređaja i senzora, te se koristi u IoT mrežama:

- https://en.wikipedia.org/wiki/MQTT

Za izradu vježbe koristiti neki od dostupnih MQTT Broker-a (npr. https://mosquitto.org/) i MQTT klijent simulatora (npr. http://www.jensd.de/apps/mqttfx/ ili http://kamilfb.github.io/mqtt-spy/). Isto tako moguće je koristiti i neki od dostupnih web servisa (npr. https://ubidots.com/).

Koristeći projekt Eclipse Paho (https://www.eclipse.org/paho/) definirati klasu(e) za apstrakciju mjerača protoka vode. Sve vrijednosti koji se objavljuju iz uređaja trebaju biti generirani kao slučajne vrijednosti u zadanim rasponima.

Sljedeće podatke generira mjerač protoka vode:

1. **Trenutna temperatura vode**

Tip podatka: int16
Faktor: 10
Raspon: -3266.8 do 3266.8
Jedinica: °C

2. **Trenutni tlak vode**

Tip podatka: uint16
Faktor: 1000
Raspon: 0 do 65.336
Jedinica: Bar

3. **Potrošnja u zadnjih 1 min, 10 min, 1 sat, 1 dan**

Tip podatka: uint16
Faktor: 0
Raspon: 0 do 65336
Jedinica: l

4. **Potrošnja u zadnjih 1 tjedan, 1 mjesec, 1 godinu**

Tip podatka: uint16
Faktor: 10
Raspon: 0 do 6533.6
Jedinica: m^3

Sadržaj B:

1. Zadatak

Proučiti Modbus TCP/IP komunikacijski protokol koji je standard za povezivanje industrijskih elektroničkih uređaja:

- https://en.wikipedia.org/wiki/Modbus
- http://www.rtaautomation.com/technologies/modbus-tcpip/
- http://www.simplymodbus.ca/TCP.htm

Koristeći Java Modbus Library (http://jamod.sourceforge.net/) definirati klasu(e) za apstrakciju Modbus TCP Slave uređaja. Sve vrijednosti koji se čitaju iz uređaja trebaju biti generirani kao slučajne vrijednosti.

Podržani ModbusTCP funkcijski kodovi:

- ReadHoldingRegisters = 3
- ReadInputRegisters = 4
- WriteSingleRegister = 6
- WriteMultipleRegisters = 16

Kodovi grešaka:

Code	Name	Description
0x01	IllegalFunction	The requested function is not supported
0x02	IllegalDataAddress	The unid it is available, but one or more of the requested register(s) do not exist
0x03	IllegalDataValue	The requested quantity of registers is invalid. See the modbus specs, http://www.modbus.org/specs.php, for the limits per function code
0x0A	GatewayPathUnavailable	Unit id is defined in the mapping list, but there is no device found on the mapped port. double check the unit id mapping list in the excel sheet, and make sure that the device is properly connected to the CCGX, switched on, and that its data is available on the CCGX display.
0x0B	GatewayTargetDeviceFailedToRespond	Requested unit id not found in the mapping list, double check the unit id mapping list in the excel sheet

Registri:

Popis za Victron regulatora punjenja:
https://victronenergy.volts.ca/media/attachment/file/c/c/ccgx-modbus-tcp-register-list-2.09_1_.xlsx

Description	Address	Type	Scale factor	Range	Writable	Unit
Battery voltage	771	uint16	100	0 to 653.36	no	V DC
Battery current	772	int16	10	-3266.8 to 3266.8	no	A DC
Battery temperature	773	int16	10	-3266.8 to 3266.8	no	Degrees celsius
Charger on/off	774	uint16	1	0 to 65336	yes	0=Off;1=On;2=Error; 3=Unavailable- Unknown
Charge state	775	uint16	1	0 to 65336	no	0=Off;2=Fault;3=Bulk; 4=Absorption;5=Float; 6=Storage;7=Equalize; 11=Other (Hub-1);252=Hub-1
PV voltage	776	uint16	100	0 to 653.36	no	V DC
PV current	777	int16	10	-3266.8 to 3266.8	no	A DC
Equalization pending	778	uint16	1	0 to 65336	no	0=No;1=Yes;2=Error; 3=Unavailable- Unknown
Equalization time remaining	779	uint16	10	0 to 6533.6	no	seconds
Relay on the charger	780	uint16	1	0 to 65336	no	0=Open;1=Closed
Alarm	781	uint16	1	0 to 65336	no	0=No alarm;2=Alarm
Low batt. voltage alarm	782	uint16	1	0 to 65336	no	0=No alarm;2=Alarm
High batt. voltage alarm	783	uint16	1	0 to 65336	no	0=No alarm;2=Alarm
Yield today	784	uint16	10	0 to 6533.6	no	kWh
Maximum charge power today	785	uint16	1	0 to 65336	no	W

Yield yesterday	786	uint16	10	0 to 6533.6	no	kWh
Maximum charge power yesterday	787	uint16	1	0 to 65336	no	W
Error code	788	uint16	1	0 to 65336	no	0=No error;1=Battery temperature too high;2=Battery voltage too high;3=Battery temperature sensor miswired (+);4=Battery temperature sensor miswired (-);5=Battery temperature sensor disconnected;6=Battery voltage sense miswired (+);7=Battery voltage sense miswired (-);8=Battery voltage sense disconnected;9=Battery voltage wire losses too high;17=Charger temperature too high;18=Charger over-current;19=Charger current polarity reversed;20=Bulk time limit reached;22=Charger temperature sensor miswired;23=Charger temperature sensor disconnected;34=Input current too high
Power	789	uint16	10	0 to 6533.6	no	W
User yield	790	uint16	10	0 to 6533.6	no	kWh

6 Četvrta (4.) vježba

Tema: I/O - datoteke

Trajanje izrade: 1 tjedan

Opis:

Sve potrebne klase za rad s datotekama i mapama, odnosno datotečnim u Javi se nalaze u standardnom paketu java.io. Koristeći klase iz navedenog paketa moguće je kreirati, brisati, preimenovati datoteke i mape, kao i čitati iz datoteke, odnosno pisati u datoteku.

Za čitanje/pisanje binarnih datoteka koristimo klase koje proširuju apstraktnu klasu java.io.InputStream, odnosno java.io.OutputStream.

Pisanje i čitanje tekstualnih datoteka, odnosno općenito stream-ova postavlja problem konverzije znakova između Unicode formata, koji Java koristi interno, i kodiranja teksta koji koristi operativni sustav. Za rješenje tog problema postoji drugi skup klasa koje ne rade s bitovnim streamovima, nego znakovnim streamovima, a koje proširuju apstraktne klase Reader i Writer.

JSON (engl. JavaScript Object Notation) je otvoreni standardni format datoteke koja koristi čitljiv tekst za prijenos podataka objekta. Sastoji se od parova atributa i vrijednosti. Uobičajeni je format podataka koji se koristi za asinkronu komunikaciju klijenta i poslužitelja, uključujući i zamjenu za XML format.

Primjer JSONa:

```
{
	"ime": "Maja",
	"prezime": "Majić",
	"godine": 22,
	"adresa": {
		"ulica": "Majina 33",
		"grad": "Split",
		"drzava": "Hrvatska",
		"postanskiBroj": "21000"
	}
}
```

Serijalizacija objekta služi da bi objekt slali kroz kanal (npr. putem interneta) za što ga je potrebno sve njegove podatkovne članove preusmjeriti u stream, poslati i onda na drugom kraju kanala objekt ponovno stvoriti na osnovu njegove klase i primljenih podatkovnih članova iz streama.

Sadržaj A:

1. Zadatak

Nadograditi rješenje prethodnog zadatka na način da se konfiguracija uređaja i pripadajućih senzora učitava iz JSON konfiguracijske datoteke.

Osmisliti format JSON datoteke.

Koristiti org.json biblioteku (https://github.com/stleary/JSON-java). Istu uključiti u svoj projekt koristeći MAVEN repozitorij.

Sadržaj B:

1. Zadatak

Koristeći Java Modbus Library (http://jamod.sourceforge.net/) definirati apstrakciju klasa za simulator različitih Modbus TCP slave uređaja. Svaki uređaj mora svakih 5 sekunda mijenjati slučajne vrijednosti u svim registrima. Koristiti isključivo Holding registre.

Informacije o registrima pojedinog uređaja učitavaju se iz JSON stringa. O svakom registru zadaju se sljedeći parametri:

- adresa,
- raspon,
- moguće vrijednosti.

Korisnik apstrakcije prije pokretanja simulatora postavlja konfiguraciju uz pomoć JSON stringa.

Pomoć:

Koristiti sučelje Register: http://jamod.sourceforge.net/api/index.html

7 Peta (5.) vježba

Tema: GUI

Trajanje izrade: 2 tjedna

Opis:

Java klase za rad sa GUI (engl. Graphical User Interface) se nalazi se u standarnim paketima java.awt (engl. Abstract Window Toolkit) i javax.swing.

java.awt je paket Java klasa koji se inicijalno koristio do verzije Jave 1.2, a koji je samo tanak omotač oko odgovarajućih grafičkih komponenti koje postoje na operacijskom sustavu na kojem se program izvršava.

javax.swing je paket Java klas dostupan od verzije Jave 1.2, a koje nisu implementirane na osnovu dostupnih grafičkih komponenti operativnog sustava već izrađene u potpunosti u Javi. Swing koristi tek minimalno grafičke metode platforme na kojoj se program izvršava i stoga daje programskom sučelju sličan izgled na različitim platformama (engl. look&feel). Također, Swing ima znatno bogatiji izbor grafičkih komponenti i danas je standardni paket za kreiranje korisničkog sučelja.

Glavni prozor u aplikaciji (prozor koji nije sadržan u drugom prozoru) naziva se *frame* u Javinoj terminologiji. U AWT biblioteci kreira se uz pomoć klase Frame, a u Swing biblioteci klasa JFrame, koja nasljeđuje klasu Frame. JFrame je jedna od rijetkih Swing komponenti koju iscrtavaju grafičke biblioteke platforme na kojoj se program izvršava.

Ostale grafičke komponente su predstavljene klasama JPanel, JLabel, JButton, …

Prilikom rada s Java GUI komponentama treba paziti u kojoj dretvi se obavlja promjena izgleda komponenti, jer se isto treba raditi koristeći posebnu dretvu zaduženju za distribuciju događaja prema grafičkim komponentama (Event Dispatch Thread). Za to se koristi posebna statička metoda:

```
SwingUtilities.invokeLater();
```

Sadržaj A:

1. Zadatak

Izraditi Java GUI aplikaciju koja će služiti kao simulator MQTT klijenta (engl. subscriber) na informacije koje objavljuje MQTT klijent (engl. publisher) iz prethodnog zadatka.

Sadržaj GUI elemenata i MQTT servera u aplikaciji mora biti konfigurabilan na osnovu JSON datoteke.

Osmisliti format JSON datoteke.

Koristiti org.json biblioteku (https://github.com/stleary/JSON-java). Isti uključiti u svoj projekt koristeći MAVEN repozitorij.

Sadržaj B:

1. Zadatak

Koristeći Java Modbus Library (http://jamod.sourceforge.net/) napraviti GUI aplikaciju koja će biti apstrakcija Modbus TCP Master-a. Aplikacija treba omogućiti paralelne upite u više Modbus TCP Slave uređaja.

Popis Slave uređaja s njihovim karakteristikama (ip adresa, port) i registrima koji će se čitati (adresa) zadaju se preko JSON string-a. Osmisliti format JSON stringa.

Omogućiti promjenu vrijednosti registra dvostrukim klikom na prikazanu trenutno očitanu vrijednost i upisom nove vrijednosti.

Napomena:

- za svaki Slave uređaj koristiti posebni panel.

8 Šesta (6.) vježba

Tema: Dretve

Trajanje izrade: 1 tjedan

Opis:

Dretve predstavljaju oblik paralelizacije na razini procesa. Svaki se proces može podijeliti u više dretvi koje se izvršavaju paralelno, na isti način kao i procesi. To znači da je izvorni kod procesa podijeljen u više podskupova koji se izvršavaju paralelno. Za razliku od procesa, sve dretve unutar jednog procesa dijele zajedničku memoriju što omogućava efikasnu komunikaciju između različitih dretvi. Paralelnost izvršavanja procesa, odnosno dretvi ovisi prvenstveno o operativnom sustavu i dostupnom hardware-u na kojem je isti pokrenut.

Kada želimo izvorni kod izvršiti u zasebnoj dretvi potrebno ga je smjestiti u run() metodu klase koja implementira java.lang.Runnable sučelje. Novi dretvu kreiramo instanciranjem objekta klase java.lang.Thread s referencom na objekt klase koja je implementirala sučelje java.lang.Runnable i pozivom metode start().

Primjer:

```
class A implements Runnable{
	public void run(){
		// izvorni kod koji se izvršava paralelno
	}
}
Thread t = new Thread(new A());
t.start();
```

Sadržaj A:

1. Zadatak

Nadograditi simulator MQTT publishera na način da svaki senzor u posebnoj dretvi generira slučajne vrijednosti.

Sadržaj B:

1. Zadatak

Koristeći Java Firebase Library (http://search.maven.org/remotecontent?filepath=com/google/firebase/firebase-admin/4.0.3/firebase-admin-4.0.3.jar) napraviti GUI aplikaciju koja će biti apstrakcija simulatora „pametne“ kuće. Pametna kuća se sastoji od više uređaja:

- jednih ili više ulaznih vrata:
- čitač pametne RFID kartice,
- električna brava,
- senzor otvorenih/zatvorenih vrata,

- jedne ili više klima uređaja
- IR predajnik za upravljanje klimom (pali/gasi)
- senzora temperature
- senzor vlage

Na osnovu konfiguracijske JSON datoteke koja određuje od kojih uređaja se sastoji „pametna“ kuća, potrebno je omogućiti automatski:

- potvrdu otključavanja vrata
- nasumično generiranje podataka o otvaranju i zatvaranju vrata
- nasumično generiranje podataka o otključavanju i zaključavanju vrata
- potvrdu o upravljanju klimom
- vrijednost temperature
- vrijednost vlage

Svi podaci se nalaze i u bazi podataka u stvarnom vremenu (engl. realtime) na servisu https://labjava-9864c.firebaseio.com/, te su dostupni s anonimnom autentifikacijom.

Struktura podataka se nalazi na sljedećoj stranici.

Više informacija o bazi podataka u stvarnom vremenu se može pronaći na linku: https://www.firebase.com/

Ukoliko želite možete kreirati i svoj besplatni servis za potrebe testiranja vaše aplikacije.

Napomena:

- ispod se nalazi i demo primjer zapisa u bazu podataka u stvarnom vremenu.

```
{
  "SmartHomeLabJava" : {
    "Climates" : {
      "Klima Panasonic" : "HomeClimate1"
    },
    "Doors" : {
      "Glavna vrata" : "HomeDoorMain",
      "Vrata stana" : "HomeDoor1"
    },
    "name" : "Pametna kuća za Javu"
  },
  "HomeClimate1" : {
    "availability" : 0,
    "cmdClimate" : 0,
    "huminidity" : 50,
    "temperature" : 25
  },
  "HomeDoor1" : {
    "Cmd" : {
      "cmdUnlock" : 0
    },
    "Keys" : [ 1, null, 3 ],
    "LogDoorOpening" : {
      "-KLkPw1P4ySTamTvz5uK" : {
            "closed" : 1467543710330,
            "opened" : 1467543699610
        },
```

```
        "-KLkTh9drOXVxhFgxeI6" : {
            "closed" : 1467544700016,
            "opened" : 1467544687273
        }
    },
    "LogDoorUnlocking" : {
      "-KLlmUdUPQ2FE_xqj7mY" : {
            "locked" : 1467566666803,
            "unlocked" : 1467566651999
      },
        "-KLm0vlFlFCFGrPPPJf7" : {
            "locked" : 1467570708023,
            "unlocked" : 1467570699344
      }
    },
    "resetCount" : 0
  },
  "HomeDoorMain" : {
    "Cmd" : {
      "cmdUnlock" : 0
    },
    "Keys" : [ 1, 2, 3 ],
    "LogDoorOpening" : {
      "-KLuDIfBgcYDmfTrB_sB" : {
            "closed" : 1467708195404,
            "opened" : 1467708160717
      }
    },
    "LogDoorUnlocking" : {
      "-KLutEKGMBudiNcq7BZn" : {
            "locked" : 1467719416665,
            "unlocked" : 1467719415122
      }
    },
    "resetCount" : 0
  }
}
```

Demo primjer:

```
import java.io.FileInputStream;
import com.google.firebase.*;
import com.google.firebase.database.*;

public class glavni {

      public static void main(String[] args) throws Exception{
            FirebaseOptions options = new FirebaseOptions.Builder()
                        .setServiceAccount(new
FileInputStream("C:\\Users\\jvrlic\\Dropbox\\labjava-9864c-firebase-
adminsdk-evw0q-c3b8590894.json"))
                        .setDatabaseUrl("https://labjava-
9864c.firebaseio.com/")
                        .build();
            FirebaseApp.initializeApp(options);
```

```
            // As an admin, the app has access to read and write all
data, regardless of Security Rules
            DatabaseReference ref = FirebaseDatabase
                .getInstance()
                .getReference("test");
            ref.addListenerForSingleValueEvent(new
ValueEventListener() {
                @Override
                public void onDataChange(DataSnapshot dataSnapshot) {
                    Object document = dataSnapshot.getValue();
                    System.out.println(document);
                }

                  @Override
                  public void onCancelled(DatabaseError arg0) {
                        // TODO Auto-generated method stub

                  }
            });
      }
}
```

9 Sedma (7.) vježba

Tema: Upoznavanje Intellij IDEA Android razvojnog okruženja

Trajanje izrade: 1 tjedan

Opis:

IntelliJ IDEA je Java IDE (engl. Integrated Development Environment) za razvoj računalnog softvera i podrškom svih standardnih funkcija koje moderna razvojna okruženja omogućavaju. Pogodno je za razvoj Android aplikacija jer je manje zahtjevno (engl. system requirements) za razliku od Googleovog Android Studio.

Sadržaj A i B:

1. Zadatak

Ukoliko je potrebno instalirati Android razvojno okruženje Intellij IDEA (https://www.jetbrains.com/idea/). Neće se koristiti „standardni“ Android Studio zbog hardwareskih ograničenja računala na kojima se izbode vježbe.

Prije instalacije razvojnog okruženja provjeriti je li instaliran JDK (http://www.oracle.com/technetwork/java/javase/downloads/index.html) i Android SDK Tools (http://filehippo.com/download_android_sdk/).

Nakon instalacije razvojnog okruženja podesiti operativni sustav (varijable okruženja JAVA_HOME, IDEA_JDK, …) kao i potrebne aplikacije ukoliko već nisu instalirane (HAXM, …).

Uz pomoć Android SDK Managera instalirati SDK platform API24 (može i neka druga verzija u ovisnosti koji Android uređaj želite simulirati).

Testirati razvojno okruženje s nekim jednostavnim programom iz predloška.

10 Osma (8.) vježba

Tema: Android aplikacija

Trajanje izrade: 3 tjedna

Opis:

Operativni sustav Android prvenstveno je kreiran za uređaje koji imaju zaslon na dodir (smartphone, tablet), te je baziran na jezgri Linux operativnog sustava, ali koja je namjenski izmijenjena kako bi se bolje prilagodila Android-u. Inicijalni razvoj je obavila tvrtka Android Inc., koju je 2005. preuzeo Google. 2007. g. Android je predstavljen kao platforma otvorenog koda.

Prednosti programiranja za Android su:

- platforma otvorenog koda (engl. open source),
- programiranje u Javi,
- jednostavna i brza objava novih aplikacija na Google Play-u,
- mnogo dostupne dokumentacije i foruma za razvojne programere.,
- veliki broj korisnika.

Android arhitektura se sastoji od više slojeva. Aplikacije se nalaze na vrhu, na dnu se nalazi Linux jezgra, a između njih Android aplikacijski framework, biblioteke, Android runtime i sl.. Android pokreće virtualni uređaj ART (engl. Android RunTime) koji je od verzije 4.4. zamijenio Dalvik Virtual Machine, i biblioteke jezgre (Core Libraries). Svaka Android aplikacija dobiva svoj vlastiti Linux proces kao instancu ART-a. ART pretvara Java kod u DEX format koji je zapakiran u aplikacijski paket.
Aplikacija je zapravo skup raznih komponenti: aktivnosti, servisi, content provideri, broadcast receiveri.

Aktivnost (engl. activity) je osnovna komponenta za prikaz korisničkog sučelja na način da korisnik može imati interakciju s aplikacijom. Na njemu se nalaze različite grafičke komponente od kojih su mnoge specifične za Android.

Servis (engl. service) je komponenta koja je pokrenuta kontinuirano u pozadini i nema sučelje prema korisniku. Postoje dvije vrste servisa: lokalni (engl. local service), koji su pokrenuti u istom procesu kao i ostatak aplikacije i udaljeni (engl. remote service), koji su pokrenuti u posebnom procesu.

Broadcast Receiver je komponenta koja prima emitirajuće signale (engl. broadcast) drugih Android komponenti.

Content Provider služi za razmjenu podataka između različitih aplikacija. Podaci mogu biti spremljeni u Android datotečni sustav, SQLite database, itd..

Intent je poruka između komponenti koja opisuje operaciju koju će izvršiti komponenta primatelj poruke. Gotovo sve komponente u Androidu koriste Intent pa je moguće zamijeniti postojeće komponente vlastitima.

Sadržaj A:

1. Zadatak

Izraditi Android aplikaciju koja će služiti kao simulator MQTT klijenta (engl. subscriber) na informacije koje objavljuje MQTT klijent (engl. publisher) iz 6. zadatka.

Za bibilioteku mqqt klijenta možete koristiti projekt Paho:

https://github.com/eclipse/paho.mqtt.android

MQQT server treba biti u lokalnoj mreži Android aplikacije.

Sadržaj B:

1. Zadatak

Napraviti Android aplikaciju za nadzor i upravljanje pametnom kućom koristeći istu Firebase bazu podataka u stvarnom vremenu (engl. realtime) iz prošlog zadatka.

Aplikacija treba omogućiti:

- prikaz svih vrata i klima uređaja u pametnoj kući
- za svaka vrata
  - otključavanje (upravljanje)
  - logove s vremenima otključavanja i zaključavanja (nadzor)
  - logove s vremenima otvaranja i zatvaranja (nadzor)
- za svaki klima uređaj
  - paljenje klime (upravljanje)
  - gašenje klime (upravljanje)
  - dostupnost (nadzor)
  - temperaturu prostorije (nadzor)
  - vlažnost prostorije (nadzor)
  -

Sve o korištenju Firebase na Androidu:

https://firebase.google.com/docs/database/android/start/