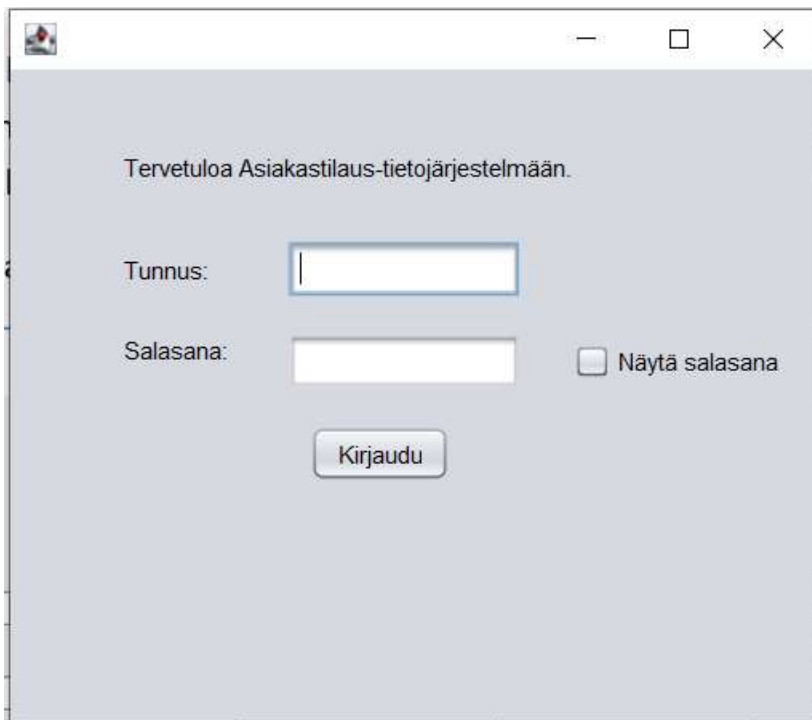


Graafisen asiakasrekisterikäyttöliittymän ohjelmointi Javalla hyödyntäen Amazon AWS pilvipalvelussa sijaitsevaa MarianDB-tietokantaa

Tehtävän tavoitteena on luoda graafinen käyttöliittymä Javalla ja NetBeans-ohjelmistolla. Tehtävässä hyödynnetään aikaisemmin Amazon AWS pilvipalveluun luotua MarianDB ASIAKASTILAUS-tietokantaa ja siihen kuuluvaa ASIAKAS-taulukkoa, jonka ylläpitoon tässä tehtävässä luotava Asiakasrekisteri-käyttöliittymä tulee. Tekemällä tehtävät opit mm. käyttöönottamaan MariaDB connectorin (ajuri), yhteyden muodostamisen käyttöliittymän ja tietokannan välille sekä keskeisten SQL-luokkien käytön tietokantaohjelmoinnissa (connection, statement, preparedstatement, resultset jne.).

Graafisen käyttöliittymän lomakkeet osalta:



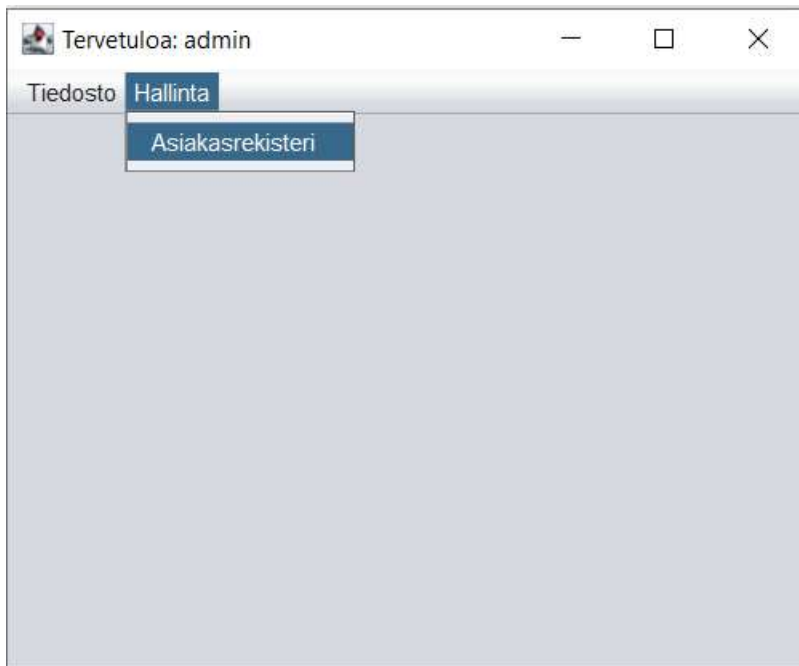
Tervetuloa Asiakastilaus-tietojärjestelmään.

Tunnus:

Salasana: ☐ Näytä salasana

Asiakastilaus-tietojärjestelmään kirjaudutaan tunnuksella ja salasanalla, jotka on tarkoitus tallentaa ja salata MariaDB-tietokantaan omaan taulukkoon ja kenttiin.

Kirjautumisen jälkeen avautuu pääkäyttöliittymä, jossa on navigointivalikko, johon toteutetaan ainakin Asiakasrekisteri-käyttöliittymän avaus.



Asiakasrekisteri-valinnan jälkeen käyttäjä ohjataan Asiakasrekisteri-käyttöliittymään, joka on seuraavan näköinen.

Opiskelija saa luotua ohjeiden avulla ao. käyttöliittymän, mutta kuten kuvasta näkee, kaikkia ASIAKAS-tilin kenttiä (Katuosoite, Postinumero, Postitoimipaikka jne.) ei ole käyttöliittymässä. Puuttuvien kenttien toteutus ja ohjelmakoodin täydennys ja muokaus jää opiskelijalle tehtäväksi.

ASIAKASNUMERO	ETUNIMI	SUKUNIMI	YRITYS
1	Kukkonen XXXX	Jukka Kukkonen OyXXXX	JukkaXXX
2	Jukka	Kukkonen	Jukka 2 Oy
3	Teemu	Heikkinen	Teemu Oy
4	Tiina	Heikkinen	Tiina Ky
5	Leena	Turunen	Leena Oy
6	Harri	Heikkinen	Harrin Asennus Ky
7	Jukka	Kukkonen	Jukka Kukkonen Oy
8	Jukka	Kukkonen	Jukka Kukkonen Oy
9	Tiina2	Heikkinen2	Tiina Ky2

Tehtävässä käytettävät ohjelmistot ja ajurit (muuta kuin mainittuja ohjelmistoversioita ei ole testattu, mutta toimintakuntoon saamiseksi uudemmillakin versioilla ei pitäisi olla vaikeaa):

- NetBeans 8.02

- Amazon Linux 2 AMI Amazon palvelin instanssi, uusin saatavilla versio Amazonissa
- MariaDB v. 10.2.10 - MariaDB Server
- Java MariaDB Connector v.2.7.2
- phpMyAdmin v. 5.0.4

Tarvitset myös käyttöliittymissä muita apulomakkeita mm. kirjautuminen.java (kirjautumista varten) ja GUI.java (pääkäyttöliittymä, jossa on valikot ym.) -lomakkeet sekä Asiakasrekisterihallinta.java-lomake. Niiden toteutukseen voit katsoa myös mallia aiemmin tehdystä Java harjoitustyöstä ja sen ohjeistuksesta.

Amazon Linux 2 AMI Amazon palvelin instanssin palomuurin konfigurointi graafisen käyttöliittymän tietokantaliikennettä varten

Oletuksena AWS instanssien palomuurisäännöt ovat varsin tiukat ja palomuuriin on sallittava MYSQL/Aurora -protokollalle tietoliikenne kaikkialta portissa 3306 ao. kuvan mukaisesti. Etsi ensiksi oma palvelin instanssisi AWS-palvelussa NETWORK & SECURITY-kohdassa ja muokkaa (Edit) sisään tulevaa liikennettä (Inbound) seuraavasti:

Edit inbound rules

Type	Protocol	Port Range	Source	Description	
HTTP	TCP	80	Custom 0.0.0.0/0	e.g. SSH for Admin Desktop	✕
HTTP	TCP	80	Custom ::/0	e.g. SSH for Admin Desktop	✕
SSH	TCP	22	Custom 0.0.0.0/0	e.g. SSH for Admin Desktop	✕
MYSQL/Aurora	TCP	3306	Custom 0.0.0.0/0	e.g. SSH for Admin Desktop	✕
MYSQL/Aurora	TCP	3306	Custom ::/0	e.g. SSH for Admin Desktop	✕

Add Rule

NOTE: Any edits made on existing rules will result in the edited rule being deleted and a new rule created with the new details. This will cause traffic that depends on that rule to be dropped for a very brief period of time until the new rule can be created.

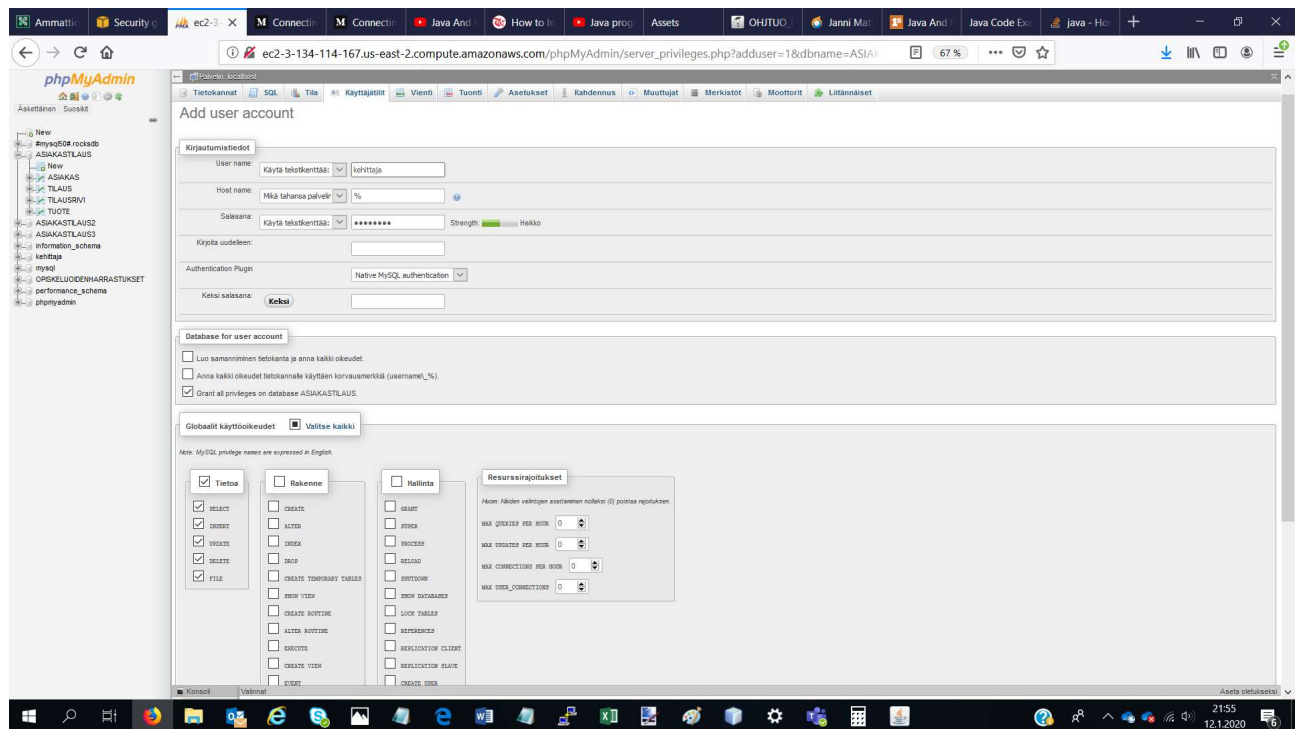
Cancel Save

Tallenna lopuksi. MariaDB-tietokantapalvelimellesi pitäisi olla nyt pääsy portin 3306 kautta.

Käyttäjätunnuksen luominen tietokantaan phpMyAdmin-ohjelmistolla

Graafinen käyttöliittymä tarvitsee yhteyden MarianDB-tietokantaan ja yhteyden muodostamiseen tarvitaan luoda käyttäjätunnus ja määrittellä salasana sekä antaa ko. tunnukselle käyttöoikeudet (lukeminen, tallentaminen ja poistaminen) tietokantaan. Edellisellä sivulla sallittiin palomuurin MSAURORA protokollalle liikennöinti portissa 3306.

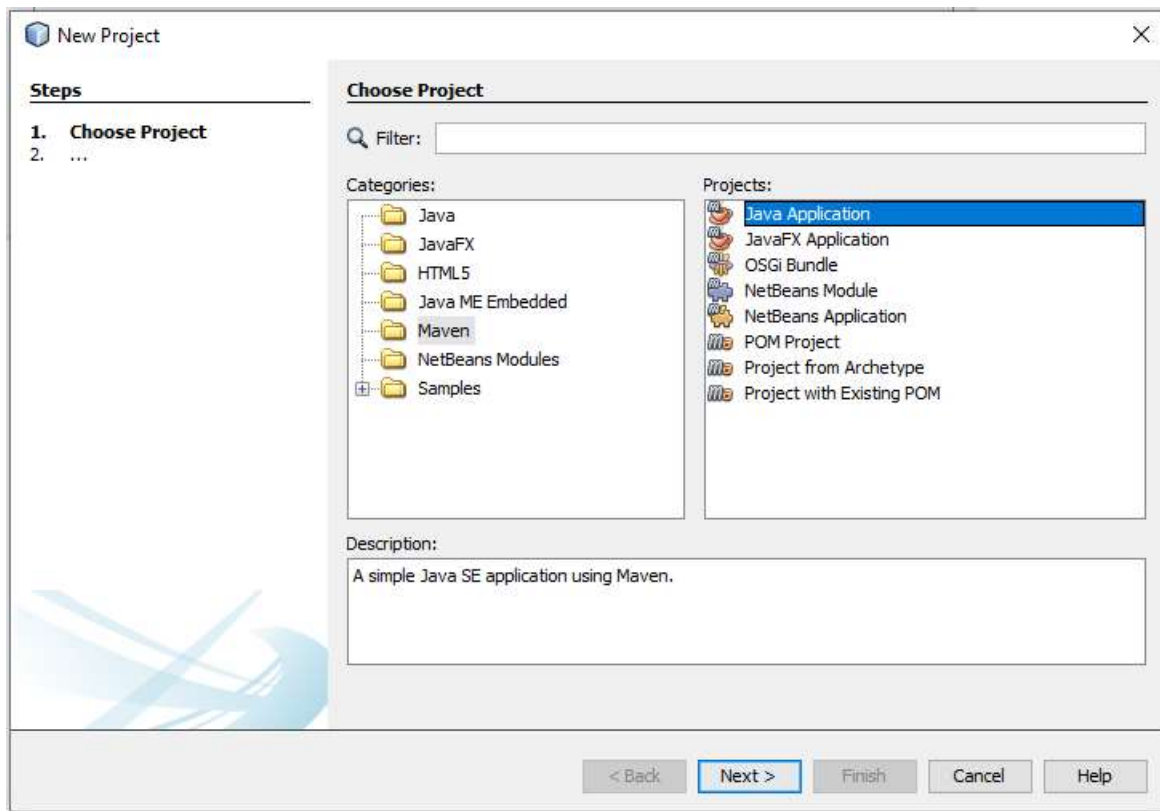
1. Avaa phpMyAdmin-ohjelmisto root-tunnuksella ja Koira123 salasanalla. Mikäli olet määritellyt tunnuksen ja salasanan ohjeista poiketen joksikin muuksi, käytä niitä.
2. Avaa ASIAKASTILAUS-tietokanta ja valitse Käyttäjätilit, Add user account: luo uusi tunnus **kehittaja** ja anna salasanaksi **Koira123!** Huomaa ottaa huutomerkki mukaan, niin tässä ohjeessa oleva koodi toimii tai jos käytät muuta salasanaa, korjaa koodia siltä osin. Anna käyttäjätunnukselle ainakin seuraavat oikeudet (SELECT, INSERT, UPDATE, DELETE ja FILE) ja harkitse tarvitseeko antaa rakenteeseen ja hallintaan liittyviä oikeuksia. Periaatteena on, että ei turhia ja liian suuria oikeuksia ei pidä tietoturvasyistä antaa.
3. Paina lopuksi Siirry-painiketta.



Tietoturvamielessä Hostname -kohta on tärkeä. % vai localhost? Ja miksi annetaan tunnuksiksi **kehittaja** eikä **admin** tai **administrator**?

Käyttöliittymäprojektin luominen

1. Luo NetBeans-sovelluksessa uusi projekti **File – New Project**-valikon kautta, valitse sieltä Java with Maven, ja Projects kohdasta valitse Java Application, lopuksi klikkaa Finish-painiketta.



Anna projektille nimeksi **Asiakastilaus** ao. kuvan mukaisesti:

New Java Application

Steps

1. Choose Project
2. **Name and Location**

Name and Location

Project Name:

Project Location:

Project Folder:

Artifact Id:

Group Id:

Version:

Package: (Optional)

< Back Next > **Finish** Cancel Help

Paina lopuksi **Finish**-painiketta.

Jotta sovellus voi keskustella tietokannan kanssa, täytyy määritellä Java Connector (suomennettuna ajuri) MariaDB -tietokantayhteyksiä varten. Lisätietoja: <https://mariadb.com/kb/en/java-connector-using-maven/>

Avaa NetBeansissä luomasi projektin Project Files -kansion sisältä pom.xml asetustiedosto ja lisää sinne kuvassa näkyvät rivit 8-14. Saman koodin voi kopioida yo. osoitteesta. Huomaa versionumero 2.7.2.

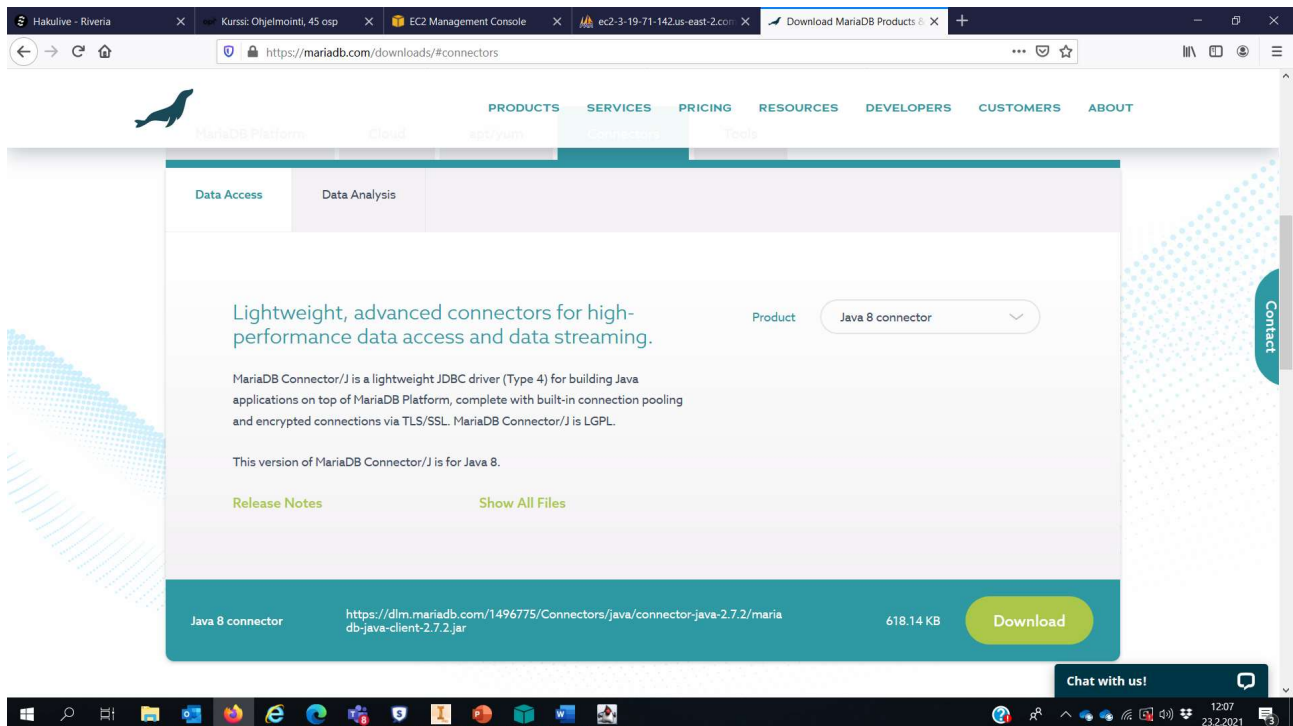
```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
3   <modelVersion>4.0.0</modelVersion>
4   <groupId>com.mycompany</groupId>
5   <artifactId>kirjautuminen</artifactId>
6   <version>1.0-SNAPSHOT</version>
7   <packaging>jar</packaging>
8   <dependencies>
9     <dependency>
10       <groupId>org.mariadb.jdbc</groupId>
11       <artifactId>mariadb-java-client</artifactId>
12       <version>2.7.2</version>
13     </dependency>
14   </dependencies>
15   <properties>
16     <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
17     <maven.compiler.source>11</maven.compiler.source>
18     <maven.compiler.target>11</maven.compiler.target>
19   </properties>
20 </project>
  
```

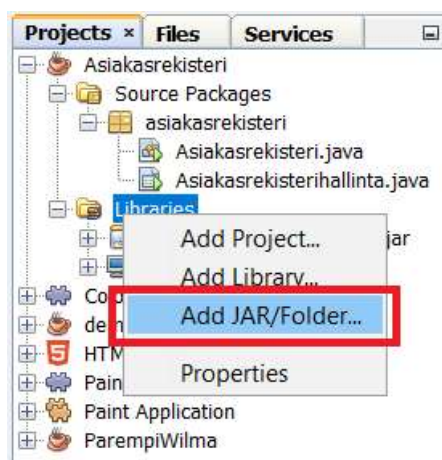
Seuraavat sivut 8-10 koskee VAIN Java ANT -projekteja, jos siis teit Maven projektin aikaisemman sivun ohjeen mukaan, ohita seuraavat ohjeet ja siirry suoraan sivulle 11 Käyttäjät taulukon luomiseen.

MariaDB Connectorin asentaminen NetBeansiin (Ant Java Application)

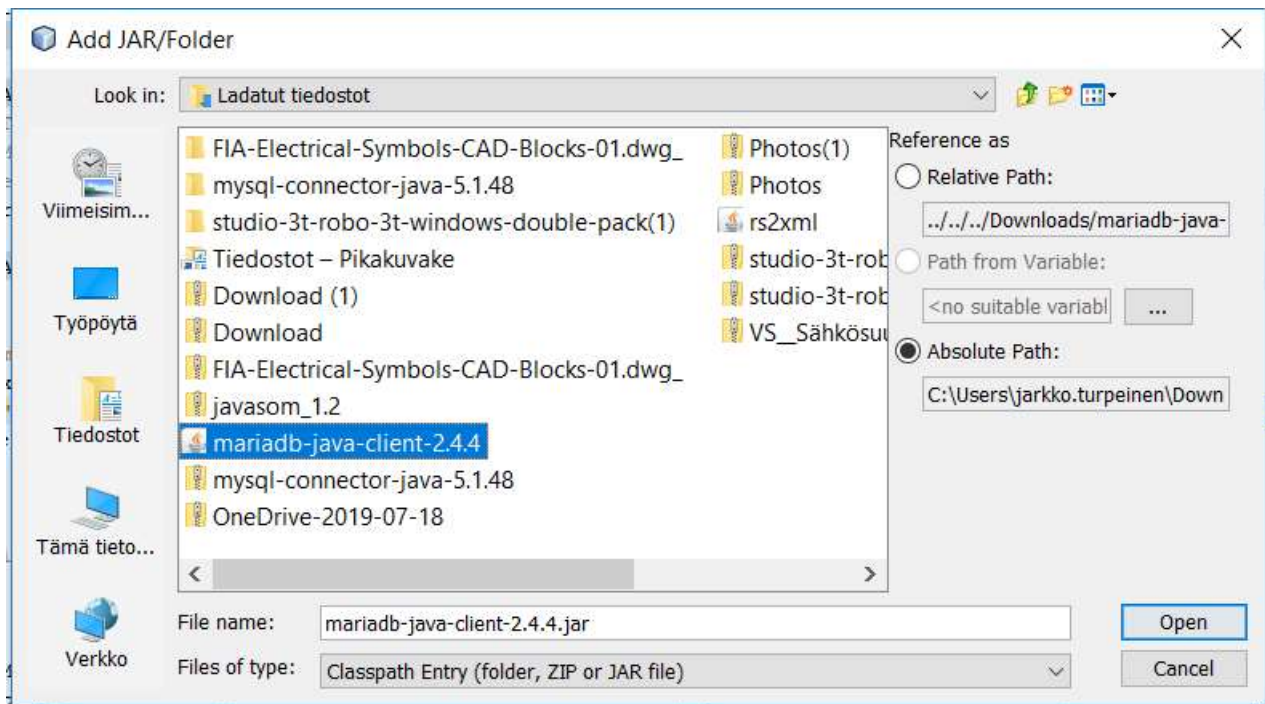
Jotta tietokantayhteys onnistuisi käyttöliittymälomakkeen ja MariaDB-tietokannan välillä, täytyy NetBeansiin asentaa MariaDB Connector (ajuri) **mariadb-java-client-2.7.2.jar** Java 8:lle, jonka voi ladata esim. seuraavasta osoitteesta: <https://mariadb.com/downloads/#connectors> työpöydällesi. Voit ladata tarvittaessa uudemmankin version, mikäli sellainen on saatavilla, mutta tämä ohje ja tehtävä on tehty ja testattu tällä versiolla.



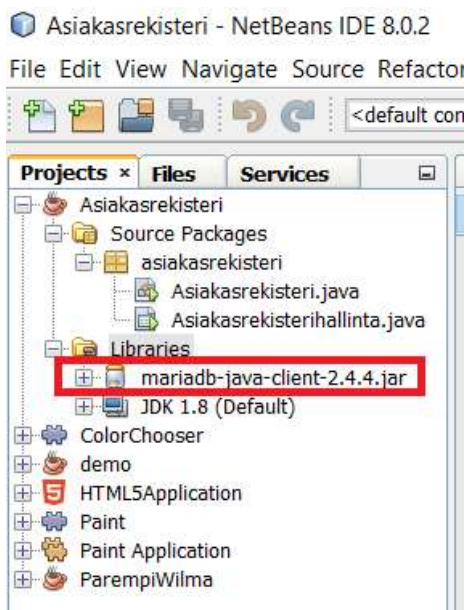
Seuraavaksi voit lisätä **mariadb-java-client.jar**-tiedoston **NetBeans**-ohjelmistossa Project-ikkunassa Asiakasrekisteri-projektin **Libraries**-kansion päällä valitsemalla hiiren kakkosnäppäimen ja sieltä Add JAR/Folder-valikon ao. kuvan mukaisesti:



Lisää **mariadb-java-client-2.7.2.jar**-ajuri lataamastasi sijainnista Open-painikkeella ao. kuvan mukaisesti.



Lisäämisen jälkeen mariadb-java-client-2.4.4.jar-ajuri näky NetBeans-ohjelmistossa Libraries-kohdassa:



Käyttäjät taulukon luominen ASIAKASTILAUS-tietokantaan

Seuraavaksi toteutetaan MariaDB-palvelimessa olevaan ASIAKASTILAUS-tietokantaan KAYTTAJA-taulukko, johon tallennetaan käyttäjätunnukset ja salasanat salatussa muodossa. Käytetään phpMyAdmin-ohjelmistoa. Suorita seuraava komento SQL-näkymässä. Huomaa SALASANA-kentän BLOB-tietotyyppi.

```
CREATE TABLE `ASIAKASTILAUS`.`KAYTTAJA` ( `TUNNUS` VARCHAR(30) NOT NULL ,  
`SALASANA` BLOB NOT NULL , PRIMARY KEY (`TUNNUS`)) ENGINE = InnoDB;
```

Taulukon rakenteessa olisi kehitettävää, mutta palataan siihen myöhemmin.

Aja SQL-näkymässä seuraavista parempi komento, jossa on tarkoitus luoda pääkäyttäjätasoinen käyttäjätunnus. Paremmassakin versiossa on edelleen parannettavaa:

```
INSERT INTO `KAYTTAJA`(`TUNNUS`,`SALASANA`) VALUES ('admin',DES_ENCRYPT('Kissa123','salainenavain'))
```

Parempi versio:

```
INSERT INTO `KAYTTAJA`(`TUNNUS`,`SALASANA`) VALUES ('HeikkiHuoltohenkilo',DES_ENCRYPT('KiSsa123#!',  
'salainenavain'))
```

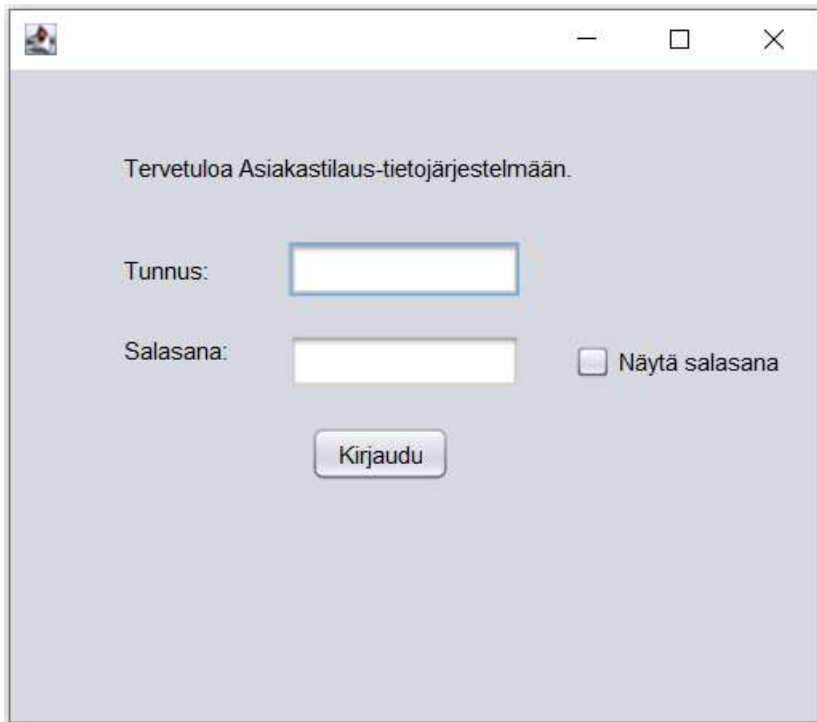
Testaa käyttäjätunnuksen hakemista tietokannasta seuraavalla SQL-lauseella:

```
SELECT TUNNUS, DES_DECRYPT(SALASANA,'salainenavain') AS SALASANA FROM KAYTTAJA  
WHERE TUNNUS='HeikkiHuoltoHenkilo'
```

Miten kehittäisit tunnistautumista ja KAYTTAJA-taulukon rakennetta? Toteuta parannukset.

Kirjautuminen.java-lomakkeen toteutus

Aikaisemmissa harjoituksissa on laadittu sisäänkirjautumislomake, jossa käyttäjältä kysytään tunnus ja salasana. Voit kopioida ja liittää ko. lomakkeen NetBeansissä aikaisemmasta projektistasi tähän Asiakastilausjärjestelmä-projektiin tai vaihtoehtoisesti luoda uuden valitsemalla Projects-ikkunassa Asiakastilausjärjestelmä ja siellä <default package> ja siellä klikataan hiiren kakkospainiketta ja valitaan New – JFrame Form. Määritä lomakkeelle nimeksi Kirjautuminen.java. Piirretään seuraavan näköinen lomake.



Tervetuloa Asiakastilaus-tietojärjestelmään.

Tunnus:

Salasana: ☐ Näytä salasana

Nimetään käyttöliittymäkontrollit nimeämiskäytännön mukaisesti. Seuraavassa on listattu käytetyt nimet.

Tekstiruutu(textbox)	jtxtTunnus
Salasanaruutu(password)	jtxtSalana
Painike (button)	jbtnKirjaudu
Valintaruutu (checkbox)	chkSalasana

Avaa Kirjautuminen-lomake koodinäkyymässä ja lisää luokkaan seuraava luoYhteys-metodi:

```

24 public class Kirjautuminen extends javax.swing.JFrame {
25
26     public Kirjautuminen() {
27         initComponents();
28     }
29     public Connection luoYhteys()
30     {
31         Connection cn=null;
32         // Muuta Oman Amazon palvelimesi osoite, tietokannan nimi sekä käyttäjätunnus ja salasana, jolla yhteys muodostetaan
33         try {
34             cn = DriverManager.
35                 getConnection("jdbc:mariadb://" + "ec2-13-53-96-100.eu-north-1.compute.amazonaws.com" + ":3306/ASIAKASTILAUS"+"?socketTimeout=2000", "kehittaja", "Koirai23!");
36             return cn;
37         } catch (SQLException e) {
38             System.out.println("Yhteyden luominen epäonnistui!\n" + e.getMessage());
39             e.printStackTrace();
40             return null;
41         }
42     }
43     public void sulje() {
44
45         WindowEvent winClosingEvent = new WindowEvent(this, WindowEvent.WINDOW_CLOSING);
46         Toolkit.getDefaultToolkit().getSystemEventQueue().postEvent(winClosingEvent);
47     }
48 }

```

Koodissa rivinumeroinnin kohdalla on todennäköisesti punaisella näkyviä virheilmoituksia, napsauta hiirellä ja valitse import, jos sellainen on tarjolla. Näin saadaan tietokantaohjelmoinnissa tarvittavat luokat importattua sovelluksen käyttöön.

Toteuta myös kuvassa näkyvä lomakkeen sulkeva Sulje-metodi.

Siirry Design-näkymään ja tuplaklikkaa Kirjautu-painiketta. Kirjoita sinne seuraava koodi:

```

144 private void jbtnKirjautuActionPerformed(java.awt.event.ActionEvent evt) {
145     // Käyttöliittymästä haettu tunnus ja salasana
146     String tunnus = txtTunnus.getText();
147     String oikeasalasana = "";
148     // Miksi salasanojen käsittely char taulukkoina eikä String-muuttujina?
149     char annettusalasana[] = jpswSalasana.getPassword();
150     char [] oikeasalasana = new char[annettusalasana.length];
151     //Muodostetaan tietokantaan yhteys
152     Connection cn = luoYhteys();
153     // Kahdessa seuraavassa rivissä piilee heikkoja lenkkejä, mitä ja missä kohti?
154     String sqlKysely = "SELECT TUNNUS, DES_DECRYPT(SALASANA, 'salainenavain') AS SALASANA FROM KAYTTAJA WHERE TUNNUS='"+tunnus+"'";
155     jdbcStatement.setString(1, sqlKysely);
156     try {
157         PreparedStatement stm = cn.prepareStatement(sqlKysely);
158         ResultSet tulos = stm.executeQuery();
159         // Jos tulos.next(), tunnus annettu oikein
160         if(tulos.next()) {
161             // Selvitetään ja haetaan resultset-objektista salasana ja trimataan lopuksi
162             oikeasalasana = tulos.getString("SALASANA").trim();
163             // muunnetaan string muotoon salasana taulukoksi
164             oikeasalasana = oikeasalasana.toCharArray();
165             // Jos tallennettu salasana ja annettu salasana täsmää..
166             if(Arrays.equals(annettusalasana, oikeasalasana)){
167                 // Valitetaan käyttäjätunnus GUI-luokan muodostimelle ja tuodaan lomake-esille
168                 GUI g = new GUI(tunnus);
169                 g.setVisible(true);
170                 // suljetaan kirjautumislomake
171                 sulje();
172             } else {
173                 // Miksi ei pidä kertoa kumpi ollut väärin: tunnus vai salasana?
174                 JOptionPane.showMessageDialog(this, "Tunnus tai salasana on väärin. Yritä uudelleen.");
175                 txtTunnus.requestFocus();
176                 return;
177             }
178         }
179     } catch (SQLException ex) {
180         Logger.getLogger(Kirjautuminen.class.getName()).log(Level.SEVERE, null, ex);
181     }
182     // Tyhjennetään tietoturvasyistä salasanojen käyttöön liittyvät muuttujat täyttämällä taulukko nolamerkeillä ja ""
183     Arrays.fill(oikeasalasana, '0');
184     oikeasalasana = "";
185 }

```

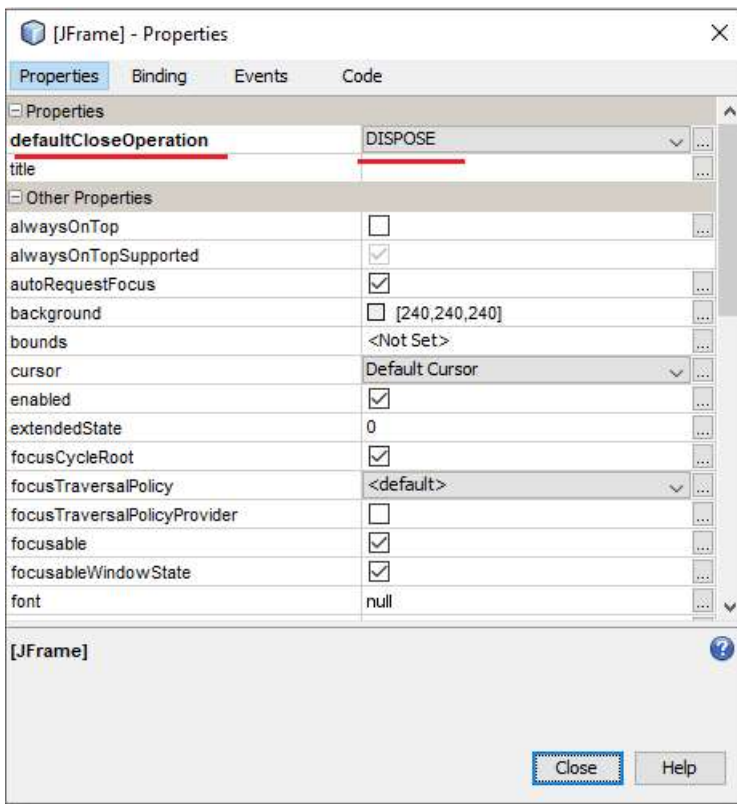
Toteuta myös salasanan näkymiseen liittyvä koodi:

```

187 private void jchkSalasanaActionPerformed(java.awt.event.ActionEvent evt) {
188     if(jchkSalasana.isSelected()){
189         jpswSalasana.setEchoChar((char)0);
190         jchkSalasana.setSelected(true);
191     } else {
192         jpswSalasana.setEchoChar('+');
193         jchkSalasana.setSelected(false);
194     }
195 }

```

Tarkista lopuksi ja aseta defaultCloseOperation-kohdassa arvo DISPOSE, joka sulkee lomakkeen.



Testaa tietokantayhteys Amazon MarianDB-palvelimeen -painikkeen ohjelmointi

Ohjelmointityön alussa on syytä toteuttaa yhteyden testaaminen luomalla painike, esimerkiksi Kirjautuminen.java-lomakkeelle, jossa tarkistetaan **org.mariadb.jdbc.Driver** löytyminen (Ant projektit) sekä yhteyden testaaminen muutenkin (Maven projektit). Luo käyttöliittymään painike ja nimeä se **jbtnTestaa-**nimiseksi. Tuplaklikkaa ko. painiketta kopioi ja liitä seuraava koodi try-koodista alkaen. Klikkaa mahdollisesti punaisella merkittyjen koodirivien kohtaa ja paina Alt + F4, jolloin NetBeans ehdottaa mitä riville voisi tehdä. Yleisin virhe on import-rivin puuttuminen lähdekooditiedoston alusta. Käy läpi kaikki punaisella merkityt rivit.

```
private void jbtnTestaaActionPerformed(java.awt.event.ActionEvent evt) {

    // Kopioi seuraava koodi ja liitä se painikkeesi tapahtuman käsittelijään.

    // Huomioi, että koodirivien rivitys ja muuta koodiin palvelimesi julkinen DNS osoite, mariaDB tietokannan
// tunnus ja salasana

    try {

        Class.forName("org.mariadb.jdbc.Driver");

    } catch (ClassNotFoundException e) {

        System.out.println("Missä on MariaDB JDBC ajuri? Oletko ladannut mariadb connectorin osoitteesta:
https://mariadb.com/downloads/#connectors ja lisännyt sen Netbeansissä Asiakasrekisteri-Libraries-Add
JAR/Folder kohdassa? ");

        e.printStackTrace();

        return;

    }

    System.out.println("Mariadb JDBC Driver rekisteröity!");

    Connection connection = null;

    try {

        connection = DriverManager.

            getConnection("jdbc:mariadb://\" + "ec2-3-134-114-167.us-east-2.compute.amazonaws.com\" +
":3306/ASIAKASTILAUS", "kehittaja", "Koira123!");

    } catch (SQLException e) {

        System.out.println("Yhteyden luominen epäonnistui!\n" + e.getMessage());
```

```

}

if (connection != null) {

    System.out.println("Hienoa ja onnittelut! Sait luotua yhteyden tietokantaasi. Voit aloittaa käyttöliittymän
koodaamisen!");

} else {

    System.out.println("Pahus, tarkista vielä, että kaikki tarvittava on tehty ja virheitä ei ole!");

}

}

```

Yo. koodin lisäyksen jälkeen osa koodista voi olla punaisena. Siirrä hiiren kohdistin ko. rivin päälle ja valitse **Alt + Enter**, joka ehdottaa **import**-rivien lisäämistä lähdekoodiin. Hyväksy **import**-lisäys.

Testaa painikkeella yhteyden luominen.

Etkö saa yhteyttä toimimaan edelleenkaan? Tässä on tarkistuslista, joka kannattaa vielä käydä läpi.

- Onko Amazon palvelimesi käynnissä ja onko yhteysmerkkijonosi oikein Java-koodissa? <http://> liitettä ei saa olla yhteysmerkkijonossa.
- Jos olet luonut projektisi Maven Java-projektina, oletko lisännyt dependencies -rivit pom.xml-tiedostoon? Katso ohje tämän tiedoston alkupuolelta.
- Onko käyttämäsi ja phpMyAdmin-ohjelmassa luomasi **kehittäjä**-tunnus ja salasana **Koira123!** oikein ja onko tunnuksella riittävät oikeudet tietokantaan?
- Onko tietokantayhteyksiä varten tarvittavat luokat importattu .java-lomakkeiden lähdekoodissa?

```
import java.sql.Connection;
```

```
import java.sql.DriverManager;
```

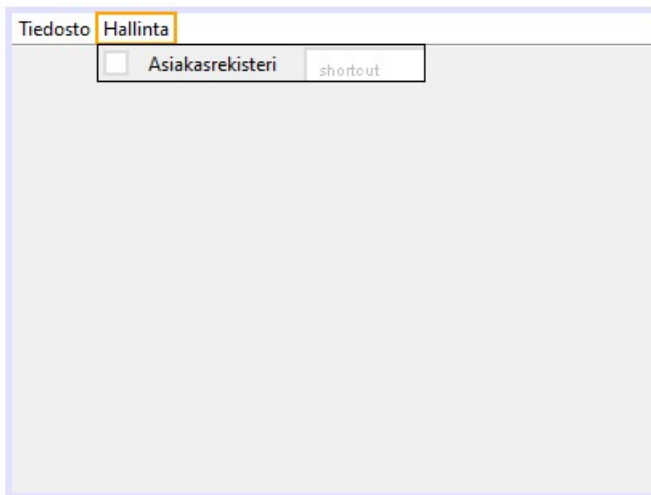
```
import java.sql.SQLException;
```

```
import java.sql.*;
```

- Onko Amazon palvelemisesi palomuurisäännöissä MySQL / Aurora sääntö luotu ohjeen mukaisesti?
- Jos lähdekoodissa on paljon ”punaista”, tarkista, että aaltosulkuja on parillinen määrä ja ne ovat ainapareittain { }.

Testaa Kirjautumislomakkeen toimintaa HeikkiHuoltoHenkilo-tunnuksella ja KiSsa123#! salasanalla.

Lisää projektiin uusi lomake, nimeltään GUI.java, johon rakennetaan navigointivalikot. Luo ensiksi päävalikko Swing Menu -kohdasta Menu Bar, ja lisää Tiedosto ja Hallinta-valikot kohdasta Menu Item. Lisää Tiedosto-valikkoon Lopeta-valikkoja Hallinta-valikkoon Asiakasrekisteri-valikko kuvan mukaisesti.



Nimeä valikot nimikäytännön mukaisesti esim. jmn-etuiliitteellä. Tuplaklikkaa Asiakasrekisteri-valikkoa ja ohjelmoi seuraavassa kuvassa oleva koodi ActionPerformed-metodiin.

```

84 private void jmnAsiakasrekisteriActionPerformed(java.awt.event.ActionEvent evt) {
85     Asiakasrekisterihallinta as = new Asiakasrekisterihallinta();
86     as.setVisible(true);
87 }

```

Lisää GUI-luokan muodostimeen käyttäjätunnuksen välittämistä varten String tunnus -koodi ja lisää myös kuvassa rivillä 18 oleva koodi:

```

17 public GUI(String tunnus) {
18     initComponents();
19     this.setTitle("Tervetuloa: " + tunnus);

```

Lisää myös seuraavassa kuvassa rivillä 119 kaksi laubausmerkkiä "" sulkujen sisään eli välitetään luokan muodostimelle aluksi tyhjä tunnus.

```

116 /* Create and display the form */
117 java.awt.EventQueue.invokeLater(new Runnable() {
118     public void run() {
119         new GUI("").setVisible(true);
120     }
121 });
122 }

```

Toteuta Tiedosto-valikossa oleva Lopeta-valikon toiminta aikaisemmin opitulla tavalla, jossa käyttäjältä kysytään haluaako hän lopettaa ohjelman käyttämisen. Vastausvaihtoehtoina Kyllä tai Ei-painikkeet.

Navigoinnin pitäisi nyt toimia.

Asiakasrekisteri-lomakkeen toteuttaminen

Lisää projektiin uusi lomake, **Asiakasrekisterihallinta.java**-niminen JFrame Form -lomake. Piirretään seuraavanlainen käyttöliittymä.

Käytä käyttöliittymässä seuraavia objektien nimiä ja tekstejä:

Selitteet: **jlblAsiakasrekisteri**, **jlblAsiakasnumero**, **jlblEtunimi**, **jlblSukunimi**, **jlblYritys** (tekstit näkyvät ao. kuvassa)

Tekstiruudut: **jtxtAasiakasnumero**, **jtxtEtunimi**, **jtxtSukunimi**, **jtxtYritys** (tekstit näkyvät ao. kuvassa)

Taulukko (jTable): **jtblAsiakkaat**

Painikkeet: **jbtnUusi**, **jbtnPaivita**, sekä **jbtnPoista** (tekstit näkyvät ao. kuvassa)

Asiakasrekisteri

Asiakasnumero:

Etunimi:

Sukunimi:

Yritys:

Uusi Päivitä Poista

Title 1	Title 2	Title 3	Title 4

MongoDB yhteyden testaus Testaa tietokantayhteys Amazon MarianDB-palvelimeen

Kuvassa näkyvät yhteyden testauspainikkeet voi halutessa jättää toteuttamatta. Seuraavassa kuvassa sama lomake on ajotilassa ja klikkaamalla taulukossa näkyviä ASIAKAS-riviä, hakee ohjelma tietokannasta tiedot ja laittaa tiedot tekstiruutuihin muokattavaksi ja uudelleen tietokantaan tallennettavaksi Päivitä-painikkeella.

ASIAKASNUMERO	ETUNIMI	SUKUNIMI	YRITYS
1	Jukka	Kukkonen	Jukka Oy
2	Jukka	Kukkonen	Jukka 2 Oy
3	Teemu	Heikkinen	Teemu Oy
4	Tiina	Heikkinen	Tiina Ky
5	Leena	Turunen	Leena Ky
6	Harri	Heikkinen	Harrin Asennus Ky

Asiakasrekisterin jTable-kontrollin ohjelmointi

JTable-kontrolli muistuttaa Excel-taulukkoa, ja siinä voidaan esittää SQL-kyselyn tuloksia. JTable, kuten kaikki muutkin Swing-luokkakirjastoon kuuluvat luokat voidaan ohjelmoida reagoimaan erilaisiin tapahtumiin mitä käyttäjä voi näppäimistöllä tai hiirellä tehdä. Tässä asiakasrekisterin ohjelmointitehtävässä tarvitaan toimintoja, jotka täyttävät jTable-kontrollin asiakkaiden tiedoilla ja kun käyttäjä klikkaa jotakin riviä, näytetään käyttöliittymässä olevissa tekstiruutu (textField) kentissä asiakkaan tiedot, jotka ovat siten muokattavissa ja uudelleen tallennettavissa Päivitä-painikkeella.

Siirry Asiakasrekisterinhallinta.java-luokan Source-näkymään. Täytä jTable-kontrollin täyttämiseen tarvitaan Java-luokkaa, joka voidaan nimetä esim. **Asiakas**. Tämä luokka voidaan toteuttaa **Asiakasrekisterinhallinta**-luokan sisälle seuraavasti:

```

29
30 public class Asiakasrekisterinhallinta extends javax.swing.JFrame {
31
32     /**
33      * Creates new form Asiakasrekisterinhallinta
34      */
35     class Asiakas{
36
37         private int ASIAKASNUMERO;
38         private String ETUNIMI;
39         private String SUKUNIMI;
40         private String YRITYS;
41
42         public Asiakas(int id, String etunimi, String sukunimi, String yritys){
43             this.ASIAKASNUMERO = id;
44             this.ETUNIMI = etunimi;
45             this.SUKUNIMI = sukunimi;
46             this.YRITYS= yritys;
47         }
48
49         public int HaeAsiakasnumero(){
50             return this.ASIAKASNUMERO;
51         }
52
53         public String HaeEtunimi(){
54             return this.ETUNIMI;
55         }
56
57         public String HaeSukunimi(){
58             return this.SUKUNIMI;
59         }
60
61         public String HaeYritys(){
62             return this.YRITYS;
63         }
64     }
65

```

Ohjelmoidaan yhteyden luomiseen tarkoitettu metodi **Asiakas**-luokan jälkeen, muokkaa oheisesta koodista palvelimesi DNS-osoite, käyttäjätunnus sekä salasana omaa palvelintasi ja tietokantaasi vastaavaksi:

```

67 // Tietokantayhteyden luomisen metodi
68 public Connection luoYhteys()
69 {
70     Connection cn=null;
71     // Muuta oman Amazon palvelimesi osoite, tietokannan nimi sekä käyttäjätunnus ja salasana, jolla yhteys muodostetaan
72     try {
73         cn = DriverManager.
74             getConnection("jdbc:mariadb://" + "ec2-3-134-114-167.us-east-2.compute.amazonaws.com" + ":3306/ASIAKASTILAUS", "kehittaja", "Koirael23");
75         return cn;
76     } catch (SQLException e) {
77         System.out.println("Yhteyden luominen epäonnistui:\n" + e.getMessage());
78         e.printStackTrace();
79         return null;
80     }
81 }

```

Ohjelmoidaan uusi **HaeAsiakastaulukko**-niminen metodi:

```

81 // Metodi, joka täyttää Asiakas-arraylist objektin
82 public ArrayList<Asiakas> HaeAsiakastaulukko()
83 {
84     ArrayList<Asiakas> Asiakastaulukko = new ArrayList<Asiakas>();
85
86     Connection yhteys = luoYhteys();
87
88     String query = "SELECT ASIAKASNUMERO, ETUNIMI, SUKUNIMI, YRITYS FROM ASIAKAS";
89     Statement st;
90     ResultSet rs;
91
92     try {
93         st = yhteys.createStatement();
94         rs = st.executeQuery(query);
95
96         Asiakas a;
97         // Lisätään kaikki asiakkaat taulukkoon
98         while(rs.next())
99         {
100             a = new Asiakas(rs.getInt("ASIAKASNUMERO"), rs.getString("ETUNIMI"), rs.getString("SUKUNIMI"), rs.getString("YRITYS"));
101             Asiakastaulukko.add(a);
102         }
103     }
104     catch (Exception e) {
105         e.printStackTrace();
106     }
107     return Asiakastaulukko;
108 }
109
110

```

Luodaan taas aikaisemmin luodun metodin alapuolelle **Naytaasiakkaat**-metodi, joka määrittää taulukon sarakeotsikot sekä täyttää datalla taulukon.

```

112 public void Naytaasiakkaat()
113 {
114     ArrayList<Asiakas> list = HaeAsiakastaulukko();
115     DefaultTableModel model = (DefaultTableModel)jtblAsiakkaat.getModel();
116     // Luodaan sarakeotsikot
117     model.setColumnIdentifiers(new Object[]{"ASIAKASNUMERO", "ETUNIMI", "SUKUNIMI", "YRITYS"});
118     Object[] row = new Object[4];
119
120     // Putsataan taulukko ennen täyttämistä
121     for (int i = jtblAsiakkaat.getRowCount() - 1; i >= 0; i--) {
122         model.removeRow(i);
123     }
124     // Täytetään taulukko uudella datalla
125     for(int i = 0; i < list.size(); i++)
126     {
127         row[0] = list.get(i).HaeAsiakasnumero();
128         row[1] = list.get(i).HaeEtunimi();
129         row[2] = list.get(i).HaeSukunimi();
130         row[3] = list.get(i).HaeYritys();
131         model.addRow(row);
132     }
133 }

```

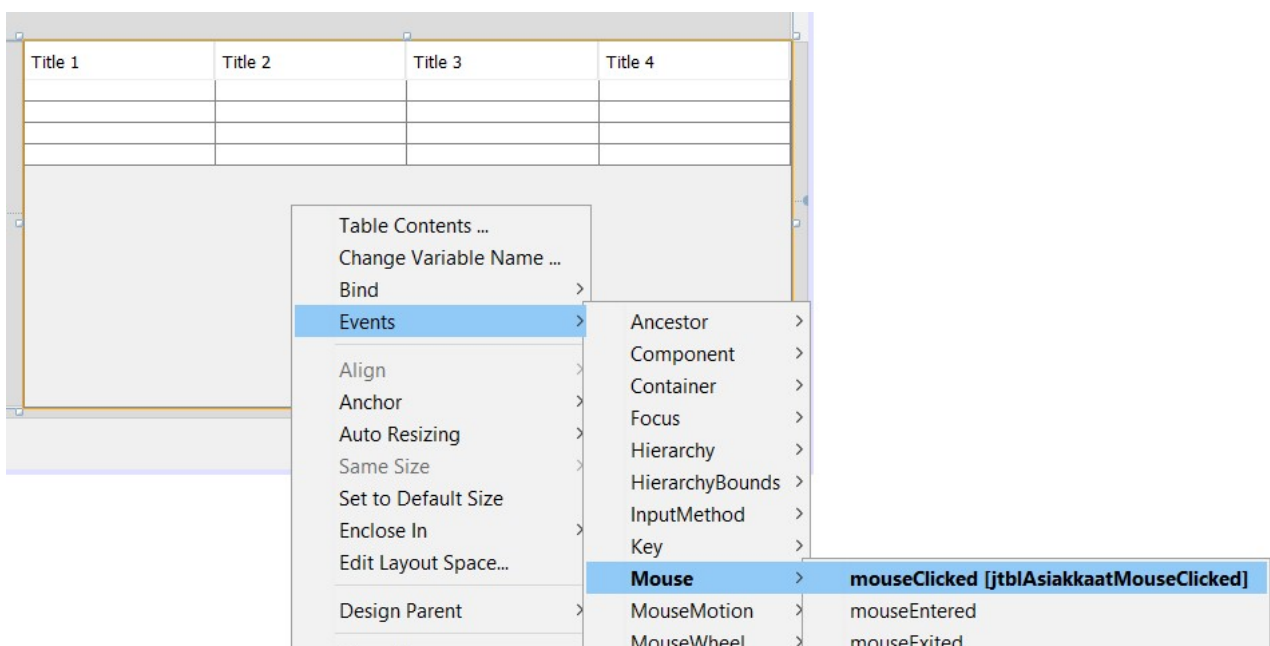
Lisätään **Naytaasiakkaat**-metodin kutsu Asiakasrekisterihallinta-luokan muodostimessa kts. rivi 163 ao. kuvassa. Jos kutsua ei määritellä, ei taulukkoa luonnollisesti täytetä asiakastiedoilla.

```

159
160 public Asiakasrekisterihallinta() {
161
162     initComponents();
163     Naytaasiakkaat();
164 }

```

Ohjelmoidaan jtblAsiakkaat-tilin MouseClicked-metodi klikkaamalla Design-näkymässä taulukkoa, valitsemalla hiiren kakkospainikkeella Events-Mouse-MouseClicked kuten kuvassa näkyy:



Kirjoitetaan seuraava koodi, joka tuo asiakastiedot esille käyttöliittymään tekstiruutu-kenttiin.

```

375 private void jTableAsiakkaatMouseClicked(java.awt.event.MouseEvent evt) {
376     // Haetaan käyttöliittymästä klikattu rivinumero
377     int i = jTableAsiakkaat.getSelectedRow();
378     TableModel model = jTableAsiakkaat.getModel();
379     // Asetetaan käyttöliittymään tiedot
380     jTextFieldAsiakasnumero.setText((model.getValueAt(i,0).toString()));
381     jTextFieldEtunimi.setText((model.getValueAt(i,1).toString()));
382     jTextFieldSukunimi.setText((model.getValueAt(i,2).toString()));
383     jTextFieldYritys.setText((model.getValueAt(i,3).toString()));
384 }

```

Tässä vaiheessa on hyvä testata Asiakasrekisteri-lomakkeen toimintaa. Kirjaudu järjestelmään HeikkiHuoltohenkilö tunnuksella ja KiSsa123#! salasanalla ja testaa.

Luodaan SQL-kyselyitä (INSERT, UPDATE ja DELETE) varten oma metodi, jolle välitetään haluttu kysely parametrinä (query):

```

134 // SQL-kyselyt
135 public void suoritaSQLKysely(String query, String message)
136 {
137     Connection yhteys = luoYhteys();
138     Statement st;
139     try{
140         st = yhteys.createStatement();
141         if((st.executeUpdate(query)) == 1)
142         {
143             // päivitetään jTableAsiakkaat taulukko
144             DefaultTableModel model = (DefaultTableModel) jTableAsiakkaat.getModel();
145             model.setRowCount(0);
146             Naytaasiakkaat();
147
148             JOptionPane.showMessageDialog(null, "Data "+message+" onnistuneesti");
149         }else{
150             JOptionPane.showMessageDialog(null, "Data ei "+message);
151         }
152         // Suljetaan tietokantayhteys
153         yhteys.close();
154     }catch(Exception ex){
155         ex.printStackTrace();
156     }
157 }

```


Siirry Asiakasrekisterihallinta.java-lomakkeen Design näkymään ja tuplaklikkaa Uusi-painiketta. Ohjelmoidaan Uusi-painikkeen ActionPerformed-metodiin koodi, joka lisää ASIAKAS-tilukkeen uuden asiakkaan tiedot hyödyntäen suoritaSQLKysely-metodia:

```

356 private void jbtnUusiActionPerformed(java.awt.event.ActionEvent evt) {
357     // Muotoillaan päivämäärä muotoon yyyy/MM/dd
358     DateFormat dateFormat = new SimpleDateFormat("yyyy/MM/dd");
359     // Hasteen järjestelmän päiväys
360     Date tamapaiva = new Date();
361     // Muodostetaan INSERT lause, huomaa + ja ' merkkien käyttäminen
362     String query = "INSERT INTO 'ASIAKAS' ('ASIAKASNUMERO', 'YRITYS', 'ETUNIMI', 'SUKUNIMI', 'KATUOSOITE', 'POSTINUMERO', 'POSTITOIMIPAIKKA', 'PUHELIN', 'EMAIL')";
363     query = query + " VALUES('" + dateFormat.format(tamapaiva) + "','" + jtxtYritys.getText() + "','" + jtxtEtunimi.getText() + "','" + jtxtSukunimi.getText() + "','"
364     query = query + "','" + jtxtKatuosoite.getText() + "','" + jtxtPostinumero.getText() + "','" + jtxtPostitoimipaikka.getText() + "','" + jtxtPuhelin.getText() + "','"
365     // JOptionPane.showMessageDialog(null, query);
366     // Luomistavaiheessa SQL merkkijonoa kannattaa testata ja kehittää phpMyAdmin-ohjelmassa, välitulkia, joita voi kopioida, kannattaa tulostaa sovelluksen käyttöliittymään tai Debug-ikkunaan
367     // jtxtSQL.setText(query);
368     suoritaSQLKysely(query, "Lisäetty");
369 }

```

Siirry Asiakasrekisterihallinta.java-lomakkeen Design näkymään ja tuplaklikkaa Päivitä-painiketta. Ohjelmoidaan Päivitä-painikkeen ActionPerformed-metodiin koodi, joka tallentaa ASIAKAS-tilukkeen käyttöliittymään määritetyt asiakastiedot hyödyntäen suoritaSQLKysely-metodia:

```

370 private void jbtnPäivitäActionPerformed(java.awt.event.ActionEvent evt) {
371     String query = "UPDATE ASIAKAS SET ETUNIMI='"+jtxtEtunimi.getText()+"', SUKUNIMI='"+jtxtSukunimi.getText()+"', YRITYS='"+jtxtYritys.getText()+"' WHERE ASIAKASNUMERO = '"+jtxtAsiakasnumero.getText()+"'";
372     JOptionPane.showMessageDialog(null, query);
373     // jtxtSQL.setText(query);
374     suoritaSQLKysely(query, "Päivitetty");
375 }

```

Siirry Asiakasrekisterihallinta.java-lomakkeen Design näkymään ja tuplaklikkaa Poista-painiketta. Ohjelmoidaan Poista-painikkeen ActionPerformed-metodiin koodi, joka poistaa valitun asiakkaan (jonka asiakasnumero on jtxtAsiakasnumero-kontrollissa) hyödyntäen suoritaSQLKysely-metodia:

```

378 private void jbtnPoistaActionPerformed(java.awt.event.ActionEvent evt) {
379     String query = "DELETE FROM ASIAKAS WHERE ASIAKASNUMERO='"+jtxtAsiakasnumero.getText()+"'";
380     JOptionPane.showMessageDialog(null, query);
381     // jtxtSQL.setText(query);
382     suoritaSQLKysely(query, "Poistettu");
}

```

Huomaa etu, joka syntyy ohjelmakoodin uudelleenkäyttämömahdollisuudesta suoritaSQLKysely-metodin kohdalla.

Käyttöliittymän testaaminen

Aja sovellus F6 Run project -painikkeella. Jos lähdekoodissa on punaisia rivejä, tarkista vielä ko. rivin kohdalla **Alt + Enter**-painikkeilla, puuttuko ko. metodista tms. **import**-lauseke lähdekooditiedoston alusta ja voiko lisätä sen **Add Import** -komennolla.

Testattavat kohdat:

- a) Klikkaa taulukkoa, näyttääkö tekstiruuduissa ko. asiakkaan tiedot?
- b) Lisää uusi asiakas, toimiiko?
- c) Päivitä jonkun olemassa olevan asiakkaan tietoja, toimiiko?
- d) Poista jokin asiakas, toimiiko?

Lisätehtävät:

- a) Lisää Uusi-, Päivitä- ja Poista-painikkeisiin JoptionPane-ikkunalla (Kyllä- tai Ei-painikkeet) varmistuskysymys haluaanko suorittaa asiakkaan lisäys-, päivitys tai poisto.
- b) Lisää käyttöliittymään kaikki ASIAKAS-tilukossa näkyvät kentät mm. **KATUOSOITE, POSTINUMERO, POSTITOIMIPAIKKA, PUHELIN JA EMAIL**. Muokkaa ja täydennä lähdekoodia siten, että käyttöliittymä huolehtii kaikista ASIAKAS-tilukon kentistä eli kenttiä voi päivittää ja JTable-tilukko näyttää kaikki kentät.
- c) Lisää ohjelmistoon tarkistukset, että tekstikentät eivät ole tyhjiä, kun asiakkaan tietoja yritetään tallentaa päivitettäessä tai uutta asiakasta lisätessä.
- d) Lisää ohjelmistoon poikkeuskäsittelijät sellaisiin kohtiin mitkä ovat kriittisiä toiminnan kannalta.
- e) Kommentoi ja dokumentoi lähdekoodi.
- f) Ota kuvakaappaus toimivasta käyttöliittymästä, jossa on b) kohdan mukaiset lisäkentät.
- g) Palauta pakattuna zip-tiedostona projektisi sisältäen lähdekoodin ja kuvakaappauksen f) kohdan mukaisesti esim. Word-asiakirjaan liitettynä tai erillisenä kuvana.

Lähdekoodi kokonaisuudessaan Asiakasrekisterihallinta.java-tiedostoon

```

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package asiakasrekisteri;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.*;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.logging.Level;
import java.util.logging.Logger;
import java.util.Date;
import javax.swing.DefaultListModel;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.TableModel;

/**
 *
 * @author jarkko.turpeinen
 */

public class Asiakasrekisterihallinta extends javax.swing.JFrame {

    /**
     * Creates new form Asiakasrekisterihallinta
     */
    // Asiakas-luokka
    class Asiakas{
        // attribuutit
        private int ASIAKASNUMERO;
        private String ETUNIMI;
        private String SUKUNIMI;
        private String YRITYS;
        // Luokan muodostin

```

```

public Asiakas(int id, String etunimi, String sukunimi, String yritys){
    this.ASIAKASNUMERO = id;
    this.ETUNIMI = etunimi;
    this.SUKUNIMI = sukunimi;
    this.YRITYS= yritys;
}

public int HaeAsiakasnumero(){
    return this.ASIAKASNUMERO;
}

public String HaeEtunimi(){
    return this.ETUNIMI;
}

public String HaeSukunimi(){
    return this.SUKUNIMI;
}

public String HaeYritys(){
    return this.YRITYS;
}
}

// Tietokantayhteyden luomisen metodi
public Connection luoYhteys()
{
    Connection cn=null;
    // Muuta oman Amazon palvelimesi osoite, tietokannan nimi sekä käyttäjätunnus ja salasana, jolla
yhteys muodostetaan
    try {
        cn = DriverManager.
            getConnection("jdbc:mariadb://" + "ec2-3-134-114-167.us-east-2.compute.amazonaws.com" +
":3306/ASIAKASTILAUS", "kehittaja", "Koira123");
        return cn;
    } catch (SQLException e) {
        System.out.println("Yhteyden luominen epäonnistui!\n" + e.getMessage());
        e.printStackTrace();
        return null;
    }
}

// Metodi, joka täyttää Asiakas-arraylist objektin
public ArrayList<Asiakas> HaeAsiakastaulukko()

```

```

{
    ArrayList<Asiakas> Asiakastaulukko = new ArrayList<Asiakas>();

    Connection yhteys = luoYhteys();
    // SQL kysely, joka hakeaa tarvittavat kentät
    String query = "SELECT ASIAKASNUMERO, ETUNIMI, SUKUNIMI, YRITYS FROM ASIAKAS";
    Statement st;
    ResultSet rs;

    try {
        st = yhteys.createStatement();
        rs = st.executeQuery(query);

        Asiakas a;
        // Lisätään kaikki asiakkaat taulukkoon
        while(rs.next())
        {
            a = new
Asiakas(rs.getInt("ASIAKASNUMERO"),rs.getString("ETUNIMI"),rs.getString("SUKUNIMI"),rs.getString("Y
RITYS"));
            Asiakastaulukko.add(a);
        }

    }
    catch (Exception e) {
        e.printStackTrace();
    }
    return Asiakastaulukko;
}

public void Naytaasiakkaat()
{
    ArrayList<Asiakas> list = HaeAsiakastaulukko();
    DefaultTableModel model = (DefaultTableModel)jtblAsiakkaat.getModel();
    // Luodaan sarakeotsikot
    model.setColumnIdentifiers(new Object[]{"ASIAKASNUMERO", "ETUNIMI", "SUKUNIMI", "YRITYS"});
    Object[] row = new Object[4];

    // Putsataan taulukko ennen täyttämistä
    for (int i = jtblAsiakkaat.getRowCount() - 1; i >= 0; i--) {
        model.removeRow(i);
    }
}

```

```

// Täytetään taulukko uudella datalla
for(int i = 0; i < list.size(); i++)
{
    row[0] = list.get(i).HaeAsiakasnumero();
    row[1] = list.get(i).HaeEtunimi();
    row[2] = list.get(i).HaeSukunimi();
    row[3] = list.get(i).HaeYritys();
    model.addRow(row);
}
}
// SQL-kyselyt
public void suoritaSQLKysely(String query, String message)
{ // Luodaan yhteys tietokantaan
    Connection yhteys = luoYhteys();
    Statement st;
    try{
        st = yhteys.createStatement();
        if((st.executeUpdate(query)) == 1)
        {
            // päivitetään jtblAsiakkaat taulukko
            DefaultTableModel model = (DefaultTableModel)jtblAsiakkaat.getModel();
            model.setRowCount(0);
            Naytaasiakkaat();

            JOptionPane.showMessageDialog(null, "Data "+message+" onnistuneesti");
        }else{
            JOptionPane.showMessageDialog(null, "Data ei "+message);
        }
        // Suljetaan tietokantayhteys
        yhteys.close();

    }catch(Exception ex){
        ex.printStackTrace();
    }
}

public Asiakasrekisterihallinta() {

    initComponents();
    Naytaasiakkaat();
}

/**

```

```

* This method is called from within the constructor to initialize the form.
* WARNING: Do NOT modify this code. The content of this method is always
* regenerated by the Form Editor.
*/
@SuppressWarnings("unchecked")

```

```

// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

    jbtnUusi = new javax.swing.JButton();
    jtxtAsiakasnumero = new javax.swing.JTextField();
    jScrollPane2 = new javax.swing.JScrollPane();
    jtblAsiakkaat = new javax.swing.JTable();
    jtxtYritys = new javax.swing.JTextField();
    jtxtEtunimi = new javax.swing.JTextField();
    jtxtSukunimi = new javax.swing.JTextField();
    lblAsiakasnumero = new javax.swing.JLabel();
    lblEtunimi = new javax.swing.JLabel();
    lblSukunimi = new javax.swing.JLabel();
    lblYritys = new javax.swing.JLabel();
    jbtnPaivita = new javax.swing.JButton();
    jbtnPoista = new javax.swing.JButton();
    jLabel2 = new javax.swing.JLabel();
    jbtnTestaa = new javax.swing.JButton();

    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

    jbtnUusi.setText("Uusi");
    jbtnUusi.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jbtnUusiActionPerformed(evt);
        }
    });

    jtblAsiakkaat.setModel(new javax.swing.table.DefaultTableModel(
        new Object [][] {
            {null, null, null, null},
            {null, null, null, null},
            {null, null, null, null},
            {null, null, null, null}
        },
        new String [] {

```

```

        "Title 1", "Title 2", "Title 3", "Title 4"
    }
});
jtblAsiakkaat.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(java.awt.event.MouseEvent evt) {
        jtblAsiakkaatMouseClicked(evt);
    }
});
jScrollPane2.setViewportViewView(jtblAsiakkaat);

lblAsiakasnumero.setText("Asiakasnumero:");

jlblEtunimi.setText("Etunimi:");

jlblSukunimi.setText("Sukunimi:");

jlblYritys.setText("Yritys:");

jbtnPaivita.setText("Päivitä");
jbtnPaivita.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jbtnPaivitaActionPerformed(evt);
    }
});

jbtnPoista.setText("Poista");
jbtnPoista.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jbtnPoistaActionPerformed(evt);
    }
});

jLabel2.setText("Asiakasrekisteri");

jbtnTestaa.setText("Testaa tietokantayhteys Amazon MarianDB-palvelimeen");
jbtnTestaa.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        jbtnTestaaActionPerformed(evt);
    }
});

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);

```

```

layout.setHorizontalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup())
    .addContainerGap()
    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup())
    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addComponent(lblAsiakasnumero)
    .addComponent(jlblEtunimi)
    .addComponent(jlblSukunimi)
    .addComponent(jlblYritys)
    .addComponent(jbtnUusi))
    .addGap(18, 18, 18)
    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)
    .addGroup(layout.createSequentialGroup())
    .addComponent(jbtnPaivita)
    .addGap(37, 37, 37)
    .addComponent(jbtnPoista))
    .addComponent(jtxtAsiakasnumero, javax.swing.GroupLayout.PREFERRED_SIZE, 85,
javax.swing.GroupLayout.PREFERRED_SIZE)
    .addComponent(jtxtYritys, javax.swing.GroupLayout.DEFAULT_SIZE, 193,
Short.MAX_VALUE)
    .addComponent(jtxtEtunimi)
    .addComponent(jtxtSukunimi))
    .addGap(18, 18, 18)
    .addComponent(jScrollPane2, javax.swing.GroupLayout.DEFAULT_SIZE, 575,
Short.MAX_VALUE))
    .addGroup(layout.createSequentialGroup())
    .addComponent(jLabel2)
    .addGap(0, 0, Short.MAX_VALUE))
    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup())
    .addGap(0, 0, Short.MAX_VALUE)
    .addComponent(jbtnTestaa)))
    .addContainerGap())
);
layout.setVerticalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup())
    .addContainerGap()
    .addComponent(jLabel2)
    .addGap(52, 52, 52)
    .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)

```



```

        .addGroup(layout.createSequentialGroup())
        .addGap(28, 28, 28)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(jtxtAsiakasnumero, javax.swing.GroupLayout.PREFERRED_SIZE,
                javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addComponent(lblAsiakasnumero))
        .addGap(27, 27, 27)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(jlblEtunimi)
            .addComponent(jtxtEtunimi, javax.swing.GroupLayout.PREFERRED_SIZE,
                javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGap(21, 21, 21)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(jlblSukunimi)
            .addComponent(jtxtSukunimi, javax.swing.GroupLayout.PREFERRED_SIZE,
                javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGap(18, 18, 18)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
            .addComponent(jtxtYritys, javax.swing.GroupLayout.PREFERRED_SIZE,
                javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
            .addComponent(jlblYritys))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
            javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
            .addComponent(jbtnUusi)
            .addComponent(jbtnPaivita)
            .addComponent(jbtnPoista)))
        .addComponent(jScrollPane2, javax.swing.GroupLayout.PREFERRED_SIZE, 275,
            javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(8, 8, 8)
        .addComponent(jbtnTestaa)
        .addContainerGap()
    );

```

```

    pack();
} // </editor-fold>

```

```

private void jbtnUusiActionPerformed(java.awt.event.ActionEvent evt) {
    // Muotoillaan päivämäärä muotoon yyyy/MM/dd
    DateFormat dateFormat = new SimpleDateFormat("yyyy/MM/dd");
    // Haetaan järjestelmän päiväys
    Date tamapaiva = new Date();
    // Muodostetaan INSERT lause, huomaa + ja ' merkkien käyttäminen

```

```

String query = "INSERT INTO `ASIAKAS`(`ASIAKKAAKSITULOPAIVA`, `YRITYS`, `ETUNIMI`, `SUKUNIMI`,
`KATUOSOITE`, `POSTINUMERO`, `POSTITOIMIPAikka`, `PUHELIN`, `EMAIL`)";
query = query + " VALUES('" + dateFormat.format(tamapaiva) + "','" +
jtxtYritys.getText()+"','"+jtxtEtunimi.getText()+"','"+jtxtSukunimi.getText()+"','";
//JOptionPane.showMessageDialog(null, query);
// luomisvaiheessa SQL merkkijonoa kannattaa testata ja kehittää phpMyAdmin-ohjelmassa,
välituloksia, joita voi kopioida, kannattaa tulostaa sovelluksen käyttöliittymän tai Debug-ikkunaan
//jtxtSQL.setText(query);
suoritaSQLKysely(query, "lisätty"); }

private void jbtnPaivitaActionPerformed(java.awt.event.ActionEvent evt) {
String query = "UPDATE ASIAKAS SET ETUNIMI='"+jtxtEtunimi.getText()+"',
SUKUNIMI='"+jtxtSukunimi.getText()+"', YRITYS='"+jtxtYritys.getText()+"' WHERE ASIAKASNUMERO =
"+jtxtAsiakasnumero.getText();
JOptionPane.showMessageDialog(null, query);
//jtxtSQL.setText(query);
suoritaSQLKysely(query, "Päivitetty");
}

private void jbtnPoistaActionPerformed(java.awt.event.ActionEvent evt) {
String query = "DELETE FROM ASIAKAS WHERE ASIAKASNUMERO="+jtxtAsiakasnumero.getText();
//JOptionPane.showMessageDialog(null, query);
//jtxtSQL.setText(query);
suoritaSQLKysely(query, "Poistettu"); }

private void jbtnTestaaActionPerformed(java.awt.event.ActionEvent evt) {
// Tietokantayhteyden testaus-painike
try {
Class.forName("org.mariadb.jdbc.Driver");
} catch (ClassNotFoundException e) {
System.out.println("Missä on MariaDB JDBC ajuri? Oletko ladannut mariadb connectorin
osoitteesta: https://mariadb.com/downloads/#connectors ja lisännyt sen Netbeansissä Asiakasrekisteri-
Libraries-Add JAR/Folder kohdassa? ");
e.printStackTrace();
return;
}

System.out.println("Mariadb JDBC Driver rekisteröity!");
Connection connection = null;
// Muokkaa palvelimen osoite, tietokannan nimi ja läyttäjätunnus sekä salasana vastaamaan omaa
ympäristöäsi
try {
connection = DriverManager.

```

```

        getConnection("jdbc:mariadb://" + "ec2-3-134-114-167.us-east-2.compute.amazonaws.com" +
":3306/ASIAKASTILAUS", "kehittaja", "Koiria123");
    } catch (SQLException e) {
        System.out.println("Yhteyden luominen epäonnistui!\n" + e.getMessage());
    }

    if (connection != null) {
        System.out.println("Hienoa ja onnittelut! Sait luotua yhteyden tietokantaasi. Voit aloittaa
käyttöliittymän koodaamisen!");
    } else {
        System.out.println("Pahus, tarkista vielä, että kaikki tarvittava on tehty ja virheitä ei ole!");
    }

}

private void jtblAsiakkaatMouseClicked(java.awt.event.MouseEvent evt) {
    // Haetaan käyttöliittymästä klikattu rivinumero
    int i = jtblAsiakkaat.getSelectedRow();
    TableModel model = jtblAsiakkaat.getModel();
    // Asetetaan käyttöliittymään tiedot
    jtxtAsiakasnumero.setText((model.getValueAt(i,0).toString()));
    jtxtEtunimi.setText((model.getValueAt(i,1).toString()));
    jtxtSukunimi.setText((model.getValueAt(i,2).toString()));
    jtxtYritys.setText((model.getValueAt(i,3).toString()));
}

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
    /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
     * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
     */
    try {
        for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {
                javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    }
}

```

```

    } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(Asiakasrekisterihallinta.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(Asiakasrekisterihallinta.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(Asiakasrekisterihallinta.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(Asiakasrekisterihallinta.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);
    }
//</editor-fold>

/* Create and display the form */
java.awt.EventQueue.invokeLater(new Runnable() {
    public void run() {
        new Asiakasrekisterihallinta().setVisible(true);
    }
});

}

// Variables declaration - do not modify
private javax.swing.JLabel jLabel2;
private javax.swing.JScrollPane jScrollPane2;
private javax.swing.JButton jbtnPaivita;
private javax.swing.JButton jbtnPoista;
private javax.swing.JButton jbtnTestaa;
private javax.swing.JButton jbtnUusi;
private javax.swing.JLabel jlblEtunimi;
private javax.swing.JLabel jlblSukunimi;
private javax.swing.JLabel jlblYritys;
private javax.swing.JTable jtblAsiakkaat;
private javax.swing.JTextField txtAsiakasnumero;
private javax.swing.JTextField txtEtunimi;
private javax.swing.JTextField txtSukunimi;
private javax.swing.JTextField txtYritys;

```

```
private javax.swing.JLabel lblAsiakasnumero;  
// End of variables declaration  
}
```