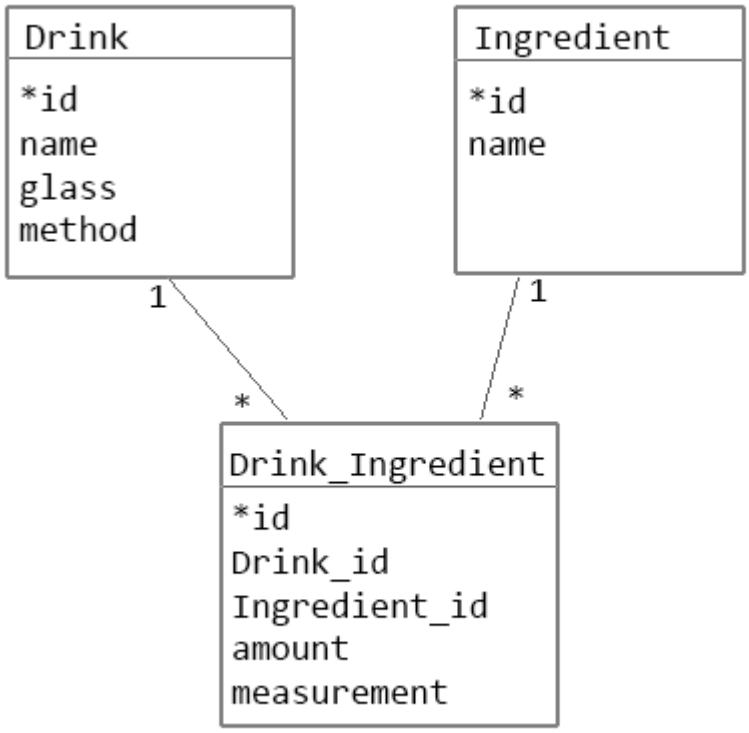
Baarisovellus

1. Ongelman kuvaus

Baarissa on drinkkejä ja drinkeissä taas ainesosia. Sovelluksen pitäisi tietokannasta haetun drinkin ainesosat ja näiden suhteet. Lisäksi drinkeille pitää toteuttaa lisäys- ja poisto-operaatiot.

2. Tietokannan rakenne

Tietokannassa on kolme taulua; Drink, Ingredient ja DrinkIngredient.



3. Hibernate-konfiguraatiotiedosto

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
        "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
    <session-factory>
        <property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
        <property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
        <property name="hibernate.connection.url">jdbc:mysql://localhost:3306/baari</property>
        <property name="hibernate.connection.username">root</property>
        <property name="hibernate.connection.password"></property>
        <property name="hbm2ddl.auto">validate</property>
        <property name="show_sql">true</property>
        <mapping class="baarisovellus.Drink"/>
        <mapping class="baarisovellus.Ingredient"/>
        <mapping class="baarisovellus.DrinkIngredient"/>
    </session-factory>
</hibernate-configuration>
```

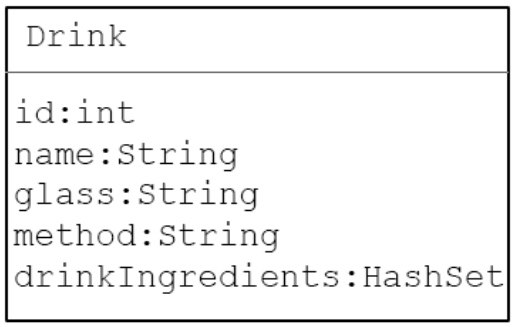
4. Java-lähdekooditiedostot

Sovellus koostuu yhteensä seitsemästä java-tiedostosta, joista kolme on hibernate-tiedostoja, kolme MVC-mallin mukaisesti koottuja käyttöliittymästä vastaavia tiedostoja sekä yksi DAO-tiedosto, joka huolehtii kaikista tietokantaan yhteyttä ottavista operaatioista.

Hibernate

Hibernate-tiedostot Drink, Ingredient ja DrinkIngredient huolehtivat tietokantaan talletettavista olioista.

Drink tiedosto vastaa tietokantatauluun DRINK talletettavien juomien ominaisuuksista.



Drink-luokassa on ominaisuuksina kokonaisluku id, joka samalla on taulun perusavain, sekä merkkijonot nimi, lasin tyyppi ja tekotapa. Jokaiselle drinkille on myös olemassa HashSet, johon talletetaan tiedot drinkin käyttämistä ainesosista.

Luokan luomisessa käytetyt annotaatiot

```
@Entity
@Table(name="DRINK")
public class Drink {
```

Perusavaimen luontiin tarkoitetut annotaatiot

```
@Id
@GeneratedValue
@Column(name = "DRINK_ID")
public int getId() {
    return id;
}
```

Nimi kentän annotaatiot

```
@Column(name="NAME")
public String getName(){
    return name;
}
```

Lasin annotaatiot

```
@Column(name="GLASS")
public String getGlass(){
    return glass;
}
```

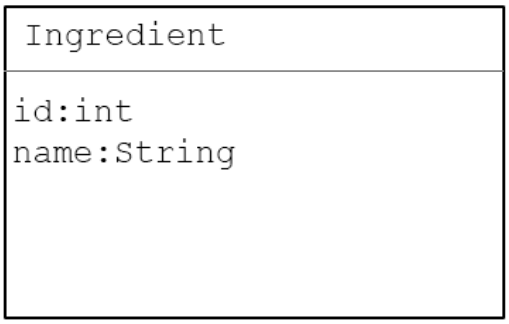
Tekotavan annotaatiot

```
@Column(name="METHOD")
public String getMethod(){
     return method;
}
```

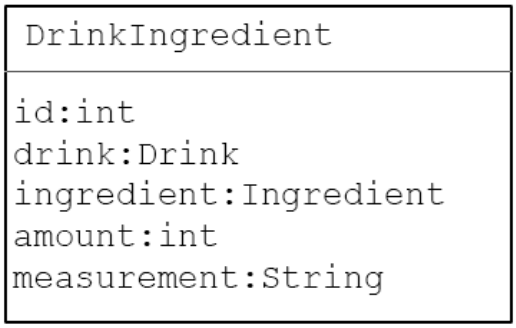
Monen suhdetta moneen ylläpitävän luokan HashSet

```
@OneToMany(mappedBy = "drink", cascade = CascadeType.ALL)
public Set<DrinkIngredient> getDrinkIngredients(){
      return drinkIngredients;
}
```

Ingredient luokka vastaa ainesosista, jotka talletetaan tietokantaan.



Monta moneen suhteesta huoltapitävä DrinkIngredient-luokka.



Tietokantaliittymä

BaariDAO-luokka huolehtii kaikista tietokantaan yhteyttä ottavista operaatioista.

```
public List<Drink> getDrinkList(){
    String hql = "from Drink";
    List<Drink> drinks = session.createQuery(hql).list();
    return drinks;
}
```

getDrinkList() palauttaa tietokannasta kaikki siihen talletetut Drink-oliot.

```
public Drink getDrink(String drink){
    String hql = "from Drink where name = \'"+drink+"\'";
    List<Drink> drinks = session.createQuery(hql).list();
    Drink d = (Drink) drinks.get(0);
    return d;
}
```

getDrink() –palauttaa sille annetusta merkkijonosta, sitä vastaavan Drink-olion.

```
public Ingredient getIngredient(String ingredient){
    String hql = "from Ingredient where name = \'"+ingredient+"\'";
    List<Ingredient> ingredients = session.createQuery(hql).list();
    Ingredient i = (Ingredient) ingredients.get(0);
    return i;
}
```

Hibernate luokille on kaikkille omat lisäysoperaatiot, joissa funktio tallettaa sille annetun olion tietokantaan.

```
public void addDrink(Drink d){
    t = session.beginTransaction();
    session.save(d);
    t.commit();
}

public void addIngredient(Ingredient i){
    t = session.beginTransaction();
    session.save(i);
    t.commit();
}

public void addDrinkIngredient(DrinkIngredient di){
```

```
    t = session.beginTransaction();
    session.save(di);
    t.commit();
}
```

Drinkin poisto-operaatiossa poistetaan tietokannasta ensin eheyssyistä kaikki drinkkiin viittaavat DrinkIngredient-ilmentymät, jonka jälkeen poistetaan itse drinkki. Ensin haetaan nimellä poistettava drinkki, jonka jälkeen otetaan talteen tämän id, jonka avulla poisto suoritetaan.

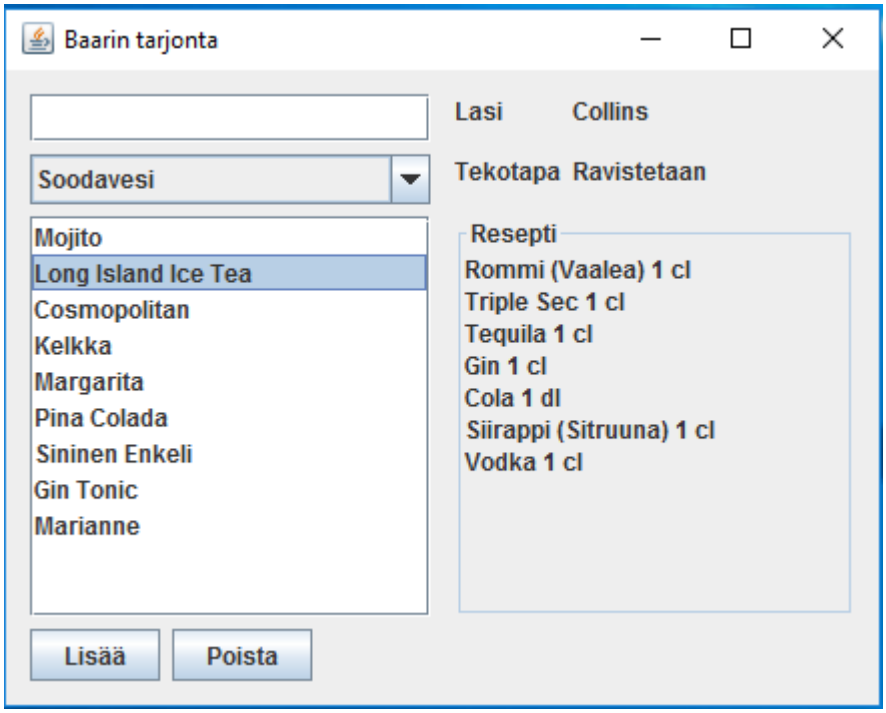
```
public void deleteDrink(String name){
    t = session.beginTransaction();
    Drink d = getDrink(name);
    int id = d.getId();
    String hql1 = "delete from DrinkIngredient where drink_id="+id;
    Query query1 = session.createQuery(hql1);
    query1.executeUpdate();
    String hql = "delete from Drink where name = \'"+name+"\'";
    Query query = session.createQuery(hql);
    int kpl = query.executeUpdate();
    t.commit();
}
```

Lisäksi luokassa on ainesosan olemassa olon tarkistukseen liittyvä luokka, joka palauttaa annetun ainesosan nimen tuottaman count(*) lauseen tuloksen.

```
public Long countIngredients(String name){
    t = session.beginTransaction();
    String hql = "select count(*) from Ingredient where name=\'"+name+"\'";
    return ((Number)session.createQuery(hql).iterate().next()).longValue();
}
```

Käyttöliittymä

MVC-mallin mukaisia tiedostoja on yhteensä kolme BaariModel, BaariView ja BaariController.



Drinkkilista on JList-komponentti, johon on haettu edellämainitulla getDrinkList()-funktiolla tietokannasta kaikkien siellä olemassa olevien drinkkien nimet. Drinkin nimeä klikkaamalla saa esille drinkille suositellun lasin, tekotavan ja reseptin.

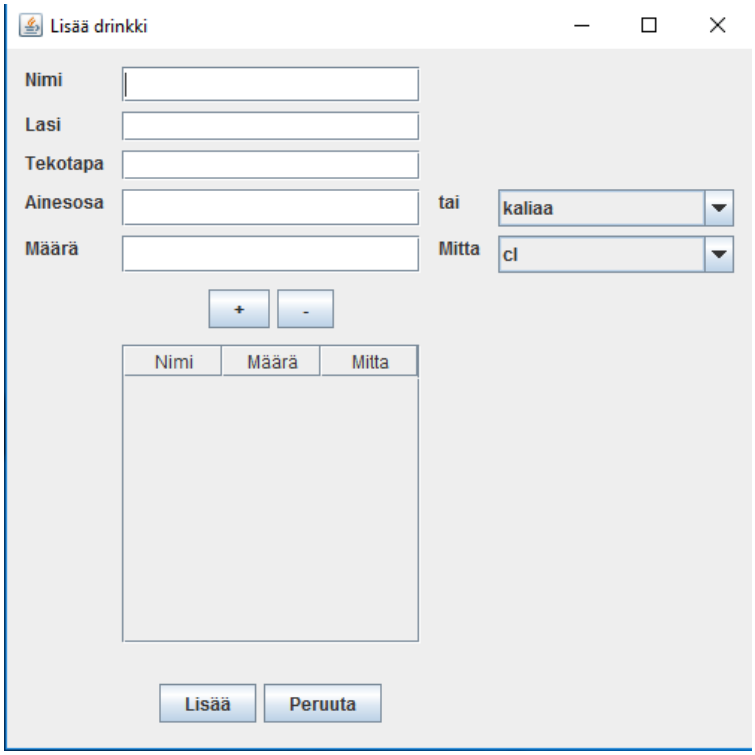
```
private JList drinkit;
private JPanel ainekset;

drinkit.addListSelectionListener(new ListSelectionListener(){
       @Override
       public void valueChanged(ListSelectionEvent arg0){
           if(!arg0.getValueIsAdjusting()){
              ainekset.removeAll(); //tyhjentää paneelin
              ohjain.aseta(getDrink()); //etsii ja asettaa paneeliin arrayn
              ainekset.repaint();
              ainekset.revalidate(); //piirtää paneelin uudelleen
           }
        }
 });
```

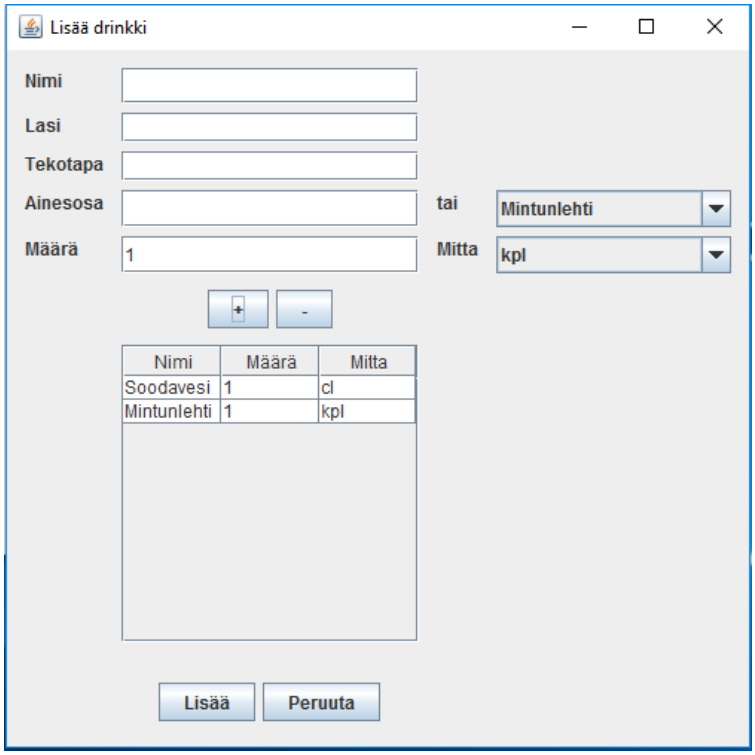
ListSelectionListenerissa käytetty aseta()-funktio etsii Drink-olion tietokannasta, minkä jälkeen se asettaa drinkin tiedot niille varattuihin JLabel-komponentteihin.

```
public void aseta(String d){
        Drink drink = model.getDrink(d); //etsii tietokannasta drinkin
        String[] ingreds =
model.getDrinkIngredients(drink.getDrinkIngredients()); //etsii
tietokannasta ainesosat
        String glass = drink.getGlass();
        String method = drink.getMethod();
        view.setIngredients(ingreds);
        view.setGlass(glass);
        view.setMethod(method);
    }
```

Drinkin lisääminen tapahtuu lisää napista, joka tekee seuraavanlaisen JFrame-komponentin.



Tekstikenttiin syötetään drinkin tiedot ja ainesosat syötetään JTable-komponenttiin.



Lisäys-operaatio tapahtuu Lisää-painikkeesta.

```
JButton add = new JButton("Lisää");
                add.addActionListener(new ActionListener(){
                    @Override
                    public void actionPerformed(ActionEvent e){
```

```
if(nimi.getText()==null||nimi.getText().equals("")){
                            warningText.setText("Anna drinkin nimi!");
                        }else{
                            ohjain.addNewDrink();
                            listModel.addElement(getNewDrinkName());
                            frame.dispose();
                        }
                    }

        });
```

Controllerin addNewDrink() saa parametreinaan drinkin tiedot ja kutsuu seuraavaa Model-luokan addNewDrink()-funktiota, joka lisää tiedot kantaan. Ainesosa taulukosta saadaan yksi merkkijono ja kaksi kokonaisluku taulukkoa. Ainesosan nimi taulukosta tarkistetaan ensin, onko kyseessä oleva ainesosa jo olemassa. Jos on, haetaan tietokannasta kyseinen Ingredient-olio ja lisätään se uuden drinkin ainesosiin. Jos taas ainesosaa ei ole entuudestaan olemassa, luodaan uusi Ingredient-olio ja viedään se tietokantaan uudeksi tietueeksi. Loput määreet lisätään drinkin tietoihin asiaan kuuluvin set-metodein.

```
public void addNewDrink(String name, String glass, String method, String[]
ingreds, int[] amount, String[] measurements){
        Drink d = new Drink(0, name, glass, method);
        dao.addDrink(d);
        Ingredient ing;

        for(int i=0;i<ingreds.length;i++){
            if(existing(ingreds[i])){
                ing = dao.getIngredient(ingreds[i]);
            }else{
                ing = new Ingredient();
                ing.setName(ingreds[i]);
                dao.addIngredient(ing);
            }
            DrinkIngredient di = new DrinkIngredient();
            di.setAmount(amount[i]);
            di.setDrink(d);
            di.setIngredient(ing);
            di.setMeasurement(measurements[i]);
            dao.addDrinkIngredient(di);
            d.getDrinkIngredients().add(di);
        }

    }
```

Drinkin poistaminen tapahtuu pääikkunassa olevasta Poista-painikkeesta.

```
poista = new JButton("Poista");
        poista.addActionListener(new ActionListener(){
            @Override
            public void actionPerformed(ActionEvent e){
                ohjain.deleteDrink();
                int position = drinkit.getSelectedIndex();
                drinkit.setSelectedIndex(position-1);
                listModel.removeElementAt(position);
                //drinkit.setSelectedIndex(0);
```

```
            }
        });
```

Nappi kutsuu Model-luokan deleteDrink() funktiota, joka taas kutsuu DAO-luokan edellä esiteltyä deleteDrink()-funktiota.

5. Sovelluksen rakenteen kuvaus

MVC-mallin mukaisesti View-luokassa on ainoastaan Swing-komponentteihin liittyvät funktiot, Model luokassa taas sovelluksen loogiikkaan liittyvät funktiot. Controller luokka taas yhdistää nämä toisiinsa. Model luokka taas ottaa yhteyttä DAO-luokkaan, joka huolehtii tietokantaan liittyvistä operaatioista.

6. Esille tulleet ideat ja ongelmat

Käyttöliittymässä on käyttämätön JComboBox-komponentti, jossa on vaihtoehtoina kaikki kannasta löytyvät ainesosat.