Hibernate –harjoitustyön raportti

# Pysäköinninvalvontajärjestelmä

Tuomas Toivonen

24.11.2015

## Ongelman kuvaus

Pysäköinninvalvontafirman on merkittävä tiedot jokaisesta sakotuksesta kirjanpitoonsa. Paperinen kirjanpito osoittautuu liian raskaaksi ja virhealttiiksi. Pysäköinninvalvontajärjestelmä tarjoaa sähköisen rajapinnan liiketoiminnallisen tiedonkäsittelyn automatisoimiseksi. Järjestelmä tallentaa siihen syötetyt tiedot sakotustapahtumasta pysyväissäilytykseen tietokantaan ja tarjoaa rajapinnan, jonka kautta tietokannassa olevaa tietoa on helppo etsiä, muokata ja poistaa.

## Tietokannan rakenne

Hibernate –kehys luo tietokantarelaatiot vastaamaan ohjelman luokkien välisiä assosiaatioita automaattisesti, ottaen olio-relaatiomuunnokseen liittyvät haasteet omille harteilleen. Näin ohjelman arkkitehtuuri voidaan suunnitella täysin businesslogiikan ehdoilla ilman tietokannan määräämiä rajoituksia. Hibernate generoi dokumenttiin sisällytetystä java-koodista seuraavanlaiset tietokantataulut

```
mysql> describe Sakko;
+-------------------------+-------------+------+-----+---------+-------------
--+
| Field                   | Type        | Null | Key | Default | Extra
  |
+-------------------------+-------------+------+-----+---------+-------------
--+
| id                      | int(11)     | NO   | PRI | NULL    | auto_incremen
t |
| paivamaara              | date        | YES  |     | NULL    |
  |
| summa                   | double      | NO   |     | NULL    |
  |
| ajoneuvo_rekisteritunnus | varchar(255) | YES  | MUL | NULL    |
  |
| tyontekija_id           | int(11)     | YES  | MUL | NULL    |
  |
+-------------------------+-------------+------+-----+---------+-------------
```

```
mysql> describe Ajoneuvo;
+----------------------+-------------+------+-----+---------+-------+
| Field                | Type        | Null | Key | Default | Extra |
+----------------------+-------------+------+-----+---------+-------+
| rekisteritunnus      | varchar(255) | NO   | PRI | NULL    |       |
| haltija_henkilotunnus | varchar(255) | YES  | MUL | NULL    |       |
+----------------------+-------------+------+-----+---------+-------+
```

```
mysql> describe Haltija;
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| henkilotunnus | varchar(255) | NO   | PRI | NULL    |       |
| etunimi       | varchar(255) | YES  |     | NULL    |       |
| sukunimi      | varchar(255) | YES  |     | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
```

```
mysql> describe Tyontekija;
+----------+-------------+------+-----+---------+----------------+
| Field    | Type        | Null | Key | Default | Extra          |
+----------+-------------+------+-----+---------+----------------+
| id       | int(11)     | NO   | PRI | NULL    | auto_increment |
| etunimi  | varchar(255) | YES  |     | NULL    |                |
| sukunimi | varchar(255) | YES  |     | NULL    |                |
+----------+-------------+------+-----+---------+----------------+
```

# Hibernate-konfiguraatiotiedosto

Konfiguraatiotiedosto sisältää ohjelman rakennetta, tietokanta-ajuria ja muuta vaadittavaa tietoa kuvaavaa metadataa. Ohessa esimerkkiohjelman käyttämä 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://mysql.metropolia.fi/tuomavt?zeroDateTimeBehavior=conve
rtToNull</property>
    <property name="hibernate.connection.username">tuomavt</property>
    <property name="hibernate.connection.password">*****</property>
    <property name="hibernate.show_sql">true</property>
    <property name="hibernate.hbm2ddl.auto">create-drop</property>

    <mapping class="com.tuoppi.pysakointi.model.Ajoneuvo" />
    <mapping class="com.tuoppi.pysakointi.model.Haltija" />
    <mapping class="com.tuoppi.pysakointi.model.Tyontekija" />
    <mapping class="com.tuoppi.pysakointi.model.Sakko" />
  </session-factory>
</hibernate-configuration>
```

# Java-lähdekooditiedostot

## Liiketoimintakomponentit

```
@Entity
public class Ajoneuvo implements Serializable {

    private String rekisteritunnus; // id
    private Haltija haltija;


    public Ajoneuvo() {
    }

    public Ajoneuvo(String rekisteritunnus, Haltija haltija) {
        this.rekisteritunnus = rekisteritunnus;
        this.haltija = haltija;
    }

    @Id
    public String getRekisteritunnus() {
        return rekisteritunnus;
```

```
    }

    @ManyToOne
    public Haltija getHaltija() {
        return haltija;
    }

    public void setRekisteritunnus(String rekisteritunnus) {
        this.rekisteritunnus = rekisteritunnus;
    }

    public void setHaltija(Haltija haltija) {
        this.haltija = haltija;
    }

}

@Entity
public class Haltija implements Serializable {

    private String henkilotunnus; // id
    private String etunimi;
    private String sukunimi;


    public Haltija() {
    }

    public Haltija(String henkilotunnus, String etunimi, String sukunimi) {
        this.henkilotunnus = henkilotunnus;
        this.etunimi = etunimi;
        this.sukunimi = sukunimi;
    }

    @Id
    public String getHenkilotunnus() {
        return henkilotunnus;
    }

    public String getEtunimi() {
        return etunimi;
    }

    public String getSukunimi() {
        return sukunimi;
    }

    public void setHenkilotunnus(String henkilotunnus) {
        this.henkilotunnus = henkilotunnus;
    }

    public void setEtunimi(String etunimi) {
```

```
        this.etunimi = etunimi;
    }

    public void setSukunimi(String sukunimi) {
        this.sukunimi = sukunimi;
    }

}

@Entity
public class Sakko implements Serializable {

    private int id;
    private Ajoneuvo ajoneuvo;
    private double summa;
    private Tyontekija tyontekija;
    private Date paivamaara;


    public Sakko() {
    }

    public Sakko(Ajoneuvo ajoneuvo, double summa, Tyontekija tyontekija, Date paivamaara) {
        this.ajoneuvo = ajoneuvo;
        this.summa = summa;
        this.tyontekija = tyontekija;
        this.paivamaara = paivamaara;
    }

    @Id @GeneratedValue (strategy = GenerationType.AUTO)
    public int getId() {
        return id;
    }

    @ManyToOne
    public Ajoneuvo getAjoneuvo() {
        return ajoneuvo;
    }

    public double getSumma() {
        return summa;
    }

    @ManyToOne
    public Tyontekija getTyontekija() {
        return tyontekija;
    }

    @Temporal (TemporalType.DATE)
    public Date getPaivamaara() {
        return paivamaara;
    }
```

```
    public void setId(int id) {
        this.id = id;
    }

    public void setAjoneuvo(Ajoneuvo ajoneuvo) {
        this.ajoneuvo = ajoneuvo;
    }

    public void setSumma(double summa) {
        this.summa = summa;
    }

    public void setTyontekija(Tyontekija tyontekija) {
        this.tyontekija = tyontekija;
    }

    public void setPaivamaara(Date paivamaara) {
        this.paivamaara = paivamaara;
    }

}

@Entity
public class Tyontekija implements Serializable {

    private int id;
    private String etunimi;
    private String sukunimi;


    public Tyontekija() {
    }

    public Tyontekija(String etunimi, String sukunimi) {
        this.etunimi = etunimi;
        this.sukunimi = sukunimi;
    }

    @Id @GeneratedValue (strategy = GenerationType.AUTO)
    public int getId() {
        return id;
    }

    public String getEtunimi() {
        return etunimi;
    }

    public String getSukunimi() {
        return sukunimi;
    }
```

```
    public void setId(int id) {
        this.id = id;
    }

    public void setEtunimi(String etunimi) {
        this.etunimi = etunimi;
    }

    public void setSukunimi(String sukunimi) {
        this.sukunimi = sukunimi;
    }

}
```

## Tietokantakerros

```
public class HibernateUtils {

    private static final SessionFactory sessionFactory = new Configuration()
            .configure().buildSessionFactory();


    public static Session getSession() {
        return sessionFactory.openSession();
    }

}

/* Handles database persitence releated tasks and manages domain object model
 * by mapping all persisted/detached entities to their primary keys in-memory */
public class PysakointiPersistence {

    private final Map<Integer, Sakko> sakot;
    private final Map<Integer, Tyontekija> tyontekijat;
    private final Map<String, Ajoneuvo> ajoneuvot;
    private final Map<String, Haltija> haltijat;

    // Error messages
    public static final String VIRHEELLINEN_SAKKO_ID = "Virheellinen sakko ID";
    public static final String VIRHEELLINEN_TYONTEKIJA_ID = "Virheellinen tyontekija ID";


    public PysakointiPersistence() {

        haltijat = new HashMap<>();
        ajoneuvot = new HashMap<>();
        tyontekijat = new HashMap<>();
        sakot = new HashMap<>();

        lisaaTyontekijat();
    }
```

```
public int lisaaSakko(String rekisteritunnus, double summa,
        String henkilotunnus, String etunimi, String sukunimi,
        int tyontekijaId) throws Exception {

    Tyontekija tyontekija = tyontekijat.get(tyontekijaId);
    if (tyontekija == null) {
        System.err.println(VIRHEELLINEN_TYONTEKIJA_ID);
        throw new Exception(VIRHEELLINEN_TYONTEKIJA_ID);
    }

    Haltija haltija = haltijat.get(henkilotunnus);
    if (haltija == null) {
        haltija = new Haltija(henkilotunnus, etunimi, sukunimi);
        haltijat.put(henkilotunnus, haltija);
    }

    Ajoneuvo ajoneuvo = ajoneuvot.get(rekisteritunnus);
    if (ajoneuvo == null) {
        ajoneuvo = new Ajoneuvo(rekisteritunnus, haltija);
        ajoneuvot.put(rekisteritunnus, ajoneuvo);
    }

    Sakko sakko = new Sakko(ajoneuvo, summa, tyontekija, new Date());

    Session session = HibernateUtils.getSession();
    Transaction tx = null;
    try {
        tx = session.beginTransaction();

        session.saveOrUpdate(haltija);
        session.saveOrUpdate(tyontekija);
        session.saveOrUpdate(ajoneuvo);
        int id = (int) session.save(sakko);

        tx.commit();

        sakot.put(id, sakko);
        System.out.println("LISAYS OK");
        return id;
    }
    catch (Exception e) {
        System.err.println(e.getMessage());
        if (tx != null && tx.isActive())
            tx.rollback();
        throw e;
    }
    finally {
        session.close();
    }
}
```

```
public void poistaSakko(int id) throws Exception {

    Sakko sakko = sakot.get(id);
    if (sakko == null) {
        System.err.println(VIRHEELLINEN_SAKKO_ID);
        throw new Exception(VIRHEELLINEN_SAKKO_ID);
    }

    Session session = HibernateUtils.getSession();
    Transaction tx = null;
    try {
        tx = session.beginTransaction();
        session.delete(sakko);
        System.out.println("OK");
        tx.commit();
        sakot.remove(id);
        System.out.println("POISTO OK");
    }
    catch (Exception e) {
        System.err.println(e.getMessage());
        if (tx != null && tx.isActive())
            tx.rollback();
        throw e;
    }
    finally {
        session.close();
    }
}

public Sakko getSakko(int id) throws Exception {

    Sakko sakko = sakot.get(id);
    if (sakko == null)
        throw new Exception(VIRHEELLINEN_SAKKO_ID);

    return sakko;
}

public void paivitaSakko(int id) throws Exception {

    Sakko sakko = sakot.get(id);
    if (sakko == null) {
        System.err.println(VIRHEELLINEN_SAKKO_ID);
        throw new Exception(VIRHEELLINEN_SAKKO_ID);
    }

    Session session = HibernateUtils.getSession();
    Transaction tx = null;
    try {
        tx = session.beginTransaction();
        session.update(sakko);
        tx.commit();
```

```
    }
    catch (Exception e) {
        System.err.println(e.getMessage());
        if (tx != null && tx.isActive())
            tx.rollback();
        throw e;
    }
    finally {
        session.close();
    }
}

private void lisaaTyontekijat() {

    Tyontekija tyontekija1 = new Tyontekija("Pirkko", "Parkkinen");
    Tyontekija tyontekija2 = new Tyontekija("Liisa", "Lappunen");
    Tyontekija tyontekija3 = new Tyontekija("Sakari", "Sakkonen");

    Session session = HibernateUtils.getSession();
    Transaction tx = null;
    try {
        tx = session.beginTransaction();

        int id1 = (int) session.save(tyontekija1);
        int id2 = (int) session.save(tyontekija2);
        int id3 = (int) session.save(tyontekija3);

        tx.commit();

        tyontekijat.put(id1, tyontekija1);
        tyontekijat.put(id2, tyontekija2);
        tyontekijat.put(id3, tyontekija3);
    }
    catch (Exception e) {
        System.err.println(e.getMessage());
        if (tx != null && tx.isActive())
            tx.rollback();
    }
    finally {
        session.close();
    }
}

public Map<Integer, Tyontekija> getTyontekijat() {
    return tyontekijat;
}

public Map<Integer, Sakko> getSakot() {
    return sakot;
}

public Map<String, Ajoneuvo> getAjoneuvot() {
```

```
        return ajoneuvot;
    }

    public Map<String, Haltija> getHaltijat() {
        return haltijat;
    }

    public void insertTestValues() {

        try {
            lisaaSakko("abc-123", 100, "123456-7890", "Erkki", "Esimerkki", 1);
        } catch (Exception ex) {
            Logger.getLogger(PysakointiPersistence.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

}
```

## Käyttöliittymä

```
public class Main {

    public static void main(String[] args) {

        PysakointiPersistence model = new PysakointiPersistence();
        model.insertTestValues();

        PysakointiGui gui = new PysakointiGui();
        PysakointiController controller = new PysakointiController(model, gui);

        gui.setController(controller);
        gui.activate();
    }

}

/* Bridge between user interface and persistence layer */
public class PysakointiController {

    private final PysakointiPersistence persister;
    private final SakkoTableModel sakkoTableModel;
    private final PysakointiGui gui;


    public PysakointiController(PysakointiPersistence persister, PysakointiGui gui) {

        this.persister = persister;
        this.gui = gui;
        sakkoTableModel = new SakkoTableModel(persister);
    }
```

```
    public void lisaaSakko(String rekisteritunnus, double summa,String henkilotunnus,
            String etunimi, String sukunimi, int tyontekijaId) {

        try {
            persister.lisaaSakko(rekisteritunnus, summa, henkilotunnus,
                    etunimi, sukunimi, tyontekijaId);
            sakkoTableModel.fireTableDataChanged();

        } catch (Exception ex) {
            System.err.println(ex.getMessage());
        }
    }

    public void poistaSakko(int id) throws Exception {
        persister.poistaSakko(id);
        sakkoTableModel.fireTableDataChanged();
    }

    public Sakko getSakko(int id) throws Exception {
        return persister.getSakko(id);
    }

    public void paivitaSakko(int id) throws Exception {
        persister.paivitaSakko(id);
    }

    public Map<Integer, Tyontekija> getTyontekijat() {
        return persister.getTyontekijat();
    }

    public Map<Integer, Sakko> getSakot() {
        return persister.getSakot();
    }

     public Map<String, Ajoneuvo> getAjoneuvot() {
         return persister.getAjoneuvot();
     }

     public Map<String, Haltija> getHaltijat() {
         return persister.getHaltijat();
     }

     public SakkoTableModel getSakkoTableModel() {
         return sakkoTableModel;
     }

}

public class SakkoTableModel extends AbstractTableModel {

    private final PysakointiPersistence perister;
    private final Map<Integer, Sakko> sakot;
```

```
private final String[] columns = { "ID", "Rekisteritunnus", "Summa",
    "Paivamaara"};


public SakkoTableModel(PysakointiPersistence perister) {
    this.perister = perister;
    sakot = perister.getSakot();
}

@Override
public int getRowCount() {
    return sakot.size();
}

@Override
public int getColumnCount() {
    return columns.length;
}

@Override
public Object getValueAt(int rowIndex, int columnIndex) {

    Sakko sakko = sakot.get(indexToId(rowIndex));
    if (sakko == null) {
        System.err.println("NULL");
        return null;
    }

    switch (columnIndex) {
        case 0: return sakko.getId();
        case 1: return sakko.getAjoneuvo().getRekisteritunnus();
        case 2: return sakko.getSumma();
        case 3: return sakko.getPaivamaara();
        default: return null;
    }
}

@Override
public Class<?> getColumnClass(int columnIndex) {
    switch (columnIndex) {
        case 0: return Integer.class;
        case 1: return String.class;
        case 2: return Double.class;
        case 3: return Date.class;
        default: return null;
    }
}

@Override
public void setValueAt(Object aValue, int rowIndex, int columnIndex) {

    Sakko sakko = sakot.get(indexToId(rowIndex));
```

```
        switch (columnIndex) {
            case 2: sakko.setSumma((Double) aValue);
                fireTableRowsUpdated(rowIndex, rowIndex);
                break;
            default: break;
        }
    }

    public static int indexToId(int rowIndex) {
        return rowIndex + 1;
    }

    @Override
    public void fireTableRowsUpdated(int firstRow, int lastRow) {

        for (int i=firstRow; i<=lastRow; i++) {
            int id = indexToId(i);
            try {
                perister.paivitaSakko(id);
            } catch (Exception ex) {
                Logger.getLogger(SakkoTableModel.class.getName()).log(Level.SEVERE, null, ex);
            }
        }
    }

    @Override
    public boolean isCellEditable(int rowIndex, int columnIndex) {
        return columnIndex == 2;
    }

    @Override
    public String getColumnName(int column) {
        return columns[column];
    }

}

public class PysakointiGui extends JFrame {

    private PysakointiController controller;
    private final JPanel content = new JPanel();
    private JScrollPane tablePanel;
    private JTable sakkoTable;

    private final JButton lisaaSakkoButton = new JButton("lisaa");
    private final JButton poistaSakkoButton = new JButton("poista");


    public PysakointiGui() {

        setContentPane(content);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
```

```
    addActionListeners();

    content.add(lisaaSakkoButton);
    content.add(poistaSakkoButton);
}

public void setController(PysakointiController controller) {

    this.controller = controller;
    sakkoTable = new JTable(controller.getSakkoTableModel());
    tablePanel = new JScrollPane(sakkoTable);
    content.add(tablePanel);
}

private void addActionListeners() {

    lisaaSakkoButton.addActionListener(e -> {
        new LisaaSakkoWindow();
    });

    poistaSakkoButton.addActionListener(e -> {

        int rowIndex = sakkoTable.getSelectedRow();

        if (rowIndex != -1) {
            int id = SakkoTableModel.indexToId(rowIndex);
            try {
                controller.poistaSakko(id);
            } catch (Exception ex) {
                Logger.getLogger(PysakointiGui.class.getName()).log(Level.SEVERE, null, ex);
            }
        }
    });
}

void activate() {
    pack();
    setVisible(true);
}

private class LisaaSakkoWindow extends JFrame {

    private final JPanel content = new JPanel();

    private final JTextField rekisteritunnus = new JTextField("rekisteritunnus");
    private final JTextField summa = new JTextField("summa");
    private final JTextField henkilotunnus = new JTextField("henkilotunnus");
    private final JTextField etunimi = new JTextField("etunimi");
    private final JTextField sukunimi = new JTextField("sukunimi");
    private final JTextField id = new JTextField("tyontekija ID");

    private final JButton lisaaButton = new JButton("Lisaa");
```

```
    public LisaaSakkoWindow() {

        setContentPane(content);

        content.add(rekisteritunnus);
        content.add(summa);
        content.add(henkilotunnus);
        content.add(etunimi);
        content.add(sukunimi);
        content.add(id);

        content.add(lisaaButton);
        lisaaButton.addActionListener(e -> {
            try {
                String rekStr = rekisteritunnus.getText();
                double sum = Double.parseDouble(summa.getText());
                String hetu = henkilotunnus.getText();
                String etunimiStr = etunimi.getText();
                String sukunimiStr = sukunimi.getText();
                int idx = Integer.parseInt(id.getText());
                controller.lisaaSakko(rekStr, sum, hetu, etunimiStr, sukunimiStr, idx);
            }
            catch (Exception ex) {}
        });

        pack();
        setDefaultCloseOperation(DISPOSE_ON_CLOSE);
        setVisible(true);
    }
  }

}
```

## Sovelluksen rakenteen kuvaus

**Sakko**, **Ajoneuvo**, **Haltija** ja **Tyontekija** ovat ohjelman liiketoiminnallisia komponentteja, joiden välisiä suhteita halutaan kuvata tietokantaan. Niinpä edellä mainitut luokat on merkattu entiteeteiksi, jolloin Hibernate luo jokaiselle oman tietokantataulun, johon olion sisäinen rakenne tallentuu, ja kuvaa olioiden väliset assosiaatiot viiteavaimiksi eri taulujen välillä.

Nykyisessä toiminnallisuudessaan ohjelmalla voidaan lisätä, muokata ja poistaa sakkoja tietokannasta. Sakko koostuu Hibernaten asettamasta int-tyyppisestä id-kentästä (perusavain), sekä ajoneuvoviittauksesta, työntekijäviittauksesta, summasta ja päivämäärästä. Vastaavasti myös muut ohjelman käyttämät entiteetit koostuvat yksilöivästä kentästä, perustietotyyppisistä kentistä ja mahdollisista olioviitteistä.

**PysakointiPersistence** suorittaa CRUD-tietokantaoperaatiot **HibernateUtils-apuluokalta** pyytämällään Session-objektilla käyttämällä tämän tarjoamia save, update ja delete –metodeja. PysakointiPersistence lisää save-operaation yhteydessä entieetit myös ajoympäristön muistissa olevaan map-tietorakenteeseen käyttäen avaimena save-operaation palauttamaa perusavainta. Näin saavutetaan se etu, että kun tiettyä entiteettiä halutaan päivittää, tarvitsee se vain hakea map-rakenteesta perusavaimellaan, päivittää sen sisäinen tila, ja kutsua Session-objektin update-metodia kyseisellä oliolla, jolloin kerran tallennetun olion tila päivittyy tietokantaan.

**PysakointiGui** toteuttaa yksinkertaisen käyttöliittymän CRUD-operaatioille. JTable, joka esittää tietokannan tiedot taulukkomuodossa, saa datan kontrollerin hallinnoimalta **SakkoTableModel –tyyppiseltä** AbstractTableModel-oliolta, joka puolestaan pyytää dataa ja metadataa suoraan PysakointiPersistencen hallinnoimista map-rakenteista, esittäen sen JTablelle tämän ymmärtämien rajapintojen kautta. Modulaarisuuden (**PysakointiController** toteuttaa MVC-arkkitehtuurin) ansiosta käyttöliittymä tai sen sisäinen esitystapa on helppo muuttaa.

## Esille tulleet ideat ja ongelmat

Käyttöliittymää voisi laajentaa myös paljon laajempaan kirjanpidollisiin tehtäviin, kuin sen nykyinen toteutus mahdollistaa. Itse asiassa tiedonmallinnuskerros tarjoaa metodit jo nyt viitteen saamiseksi sellaisiin komponentteihin, jota käyttöliittymässä ei hyödynnetä. Tiedon oikeellisuuden tarkistamista varten osaksi ohjelmaa voitaisiin integroida oma moduuli, tai ideaalisemmin rakentaa koko sovellus Spring –ohjelmistokehyksen varaan, jolloin laajennettavuuden vaivattomuus tulevaisuudessa saataisiin maksimoitua.

Käyttöliittymään sisältyy vakava bugi. JTable ei tietyissä tilanteissa hallitse rivien poistoa oikein sisäisessä esityksessään, vaikka operaatio suoriutuukin tietokannan tasolla virheettömästi. Ongelma on yksinomaan käyttölittymäkerroksessa, eikä se vaikuta datakerrokseen.