```java
package information_server;

import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.ServerSocket;
import java.net.Socket;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.Calendar;
import java.util.GregorianCalendar;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.data.jdbc.JDBCCategoryDataset;
import org.jfree.data.jdbc.JDBCPieDataset;
import org.jfree.data.jdbc.JDBCXYDataset;
import org.jfree.data.statistics.Statistics;

/**
 *
 * @author rapha
 */
public class FreetaxStatsServer {
    public static void main(String args[])
            throws IOException, ClassNotFoundException, SQLException,
            InstantiationException, IllegalAccessException
    {
        Class.forName("com.mysql.jdbc.Driver").newInstance();
        String url = "jdbc:mysql://"+Config.MYSQL_HOST+"/freetax";
        Connection con = DriverManager.getConnection(url, "ferryinpres", "pass");

        ServerSocket server_sock = new ServerSocket(Config.FREETAXSTATS_PORT);

        for (;;) {
            System.out.println("En attente d'un nouveau client");
            Socket sock = server_sock.accept();

            System.out.println("Nouveau client");
            ObjectOutputStream obj_out = new ObjectOutputStream(
                sock.getOutputStream()
            );
            ObjectInputStream obj_in = new ObjectInputStream(
                sock.getInputStream()
            );

            // Vérifie les identifiants de connexion
            FreetaxStatsLogin login = (FreetaxStatsLogin) obj_in.readObject();
            login.getLogin();

            System.out.println("Pass: "+ login.getHashedPassword());

            // Extrait le mot de passe de la base de données
            PreparedStatement instruc = con.prepareStatement(
                "SELECT COUNT(*) as existe " +
                "FROM utilisateurs " +
                "WHERE login = ? AND hashMotDePasse = ?"
            );
            instruc.setString(1, login.getLogin());
            instruc.setString(2, login.getHashedPassword());
```

```
            ResultSet rs = instruc.executeQuery();
            rs.next();

            if (rs.getInt("existe") != 0) {
                // Identifiants valides
                obj_out.writeObject(new FreetaxStatsAck());
                obj_out.flush();

                for (;;) {
                    FreetaxStatsProtocol query
                            = (FreetaxStatsProtocol) obj_in.readObject();

                    if (query instanceof FreetaxStatsDesc) {
                        statsDesc(
                            con, obj_out, obj_in, (FreetaxStatsDesc) query
                        );
                    } else if (query instanceof FreetaxStatsGraph) {
                        sendGraph(
                            con, obj_out, obj_in, (FreetaxStatsGraph) query
                        );
                    } else if (query instanceof FreetaxStatsTestComp) {
                        sendTestComp(
                            con, obj_out, obj_in, (FreetaxStatsTestComp) query
                        );
                    }
                }
            } else {
                // Identifiants non valides
                obj_out.writeObject(new FreetaxStatsFail());
                obj_out.flush();
            }

            sock.close();
        }
    }

    private static int moisPrec()
    {
        // Sélectionne le mois précédent si aucun mois n'est renseigné
        Calendar cal = new GregorianCalendar();
        // cal.add(Calendar.MONTH, -1);
        return cal.get(Calendar.MONTH);
    }

    private static void statsDesc(Connection con, ObjectOutputStream obj_out,
            ObjectInputStream obj_in, FreetaxStatsDesc freetaxStatsDesc)
            throws SQLException, IOException
    {
        PreparedStatement instruc;
        int nbJours;

        if (freetaxStatsDesc.getSemaine() != null) {
            // Recherche par semaine
            instruc = con.prepareStatement(
                "SELECT jour_semaine AS jour, count " +
                "FROM stats_desc " +
                "WHERE categorie = ? " +
                "  AND semaine = ?"
            );
            instruc.setInt(2, freetaxStatsDesc.getSemaine().intValue());
            nbJours = 7;
        } else {
            // Recherche par mois
```

```
            if (freetaxStatsDesc.getMois() == null) {
                // Sélectionne le mois précédent si aucun mois n'est renseigné
                freetaxStatsDesc.setMois(moisPrec());
            }

            instruc = con.prepareStatement(
                "SELECT jour_mois AS jour, count " +
                "FROM stats_desc " +
                "WHERE categorie = ? " +
                "  AND mois = ?"
            );
            instruc.setInt(2, freetaxStatsDesc.getMois().intValue());

            Calendar cal = new GregorianCalendar();
            cal.set(Calendar.MONTH, freetaxStatsDesc.getMois());
            nbJours = cal.getActualMaximum(Calendar.DAY_OF_MONTH);
        }
        instruc.setString(1, freetaxStatsDesc.getCategorie());
        ResultSet rs = instruc.executeQuery();

        int[] ventes = new int[nbJours];
        while (rs.next()) {
            ventes[rs.getInt("jour")-1] = rs.getInt("count");
        }

        Number[] ventesInteger = new Integer[nbJours];
        int i = 0;
        for (int v : ventes) {
            ventesInteger[i] = v;
            i++;
        }

        double moyenne = Statistics.calculateMean(ventesInteger);
        double ecartType = Statistics.getStdDev(ventesInteger);

        Integer mode = null;
        for (int v : ventes) {
            if (mode == null || v > mode.intValue())
                mode = v;
        }

        obj_out.writeObject(new FreetaxStatsDescReponse(
            ventes, moyenne, mode.intValue(), ecartType
        ));
        obj_out.flush();
    }

    private static void sendGraph(Connection con, ObjectOutputStream obj_out,
            ObjectInputStream obj_in, FreetaxStatsGraph query)
            throws SQLException, IOException
    {
        JFreeChart graph;

        if (query.getType() == FreetaxStatsGraph.SECTORIEL
            || query.getType() == FreetaxStatsGraph.HISTOGRAMME) {
            graph = graph1D(con, query);
        } else if (query.getType() == FreetaxStatsGraph.HISTOGRAMME_COMPARE) {
            graph = graphHistogrammeCompare(con, query);
        } else if (query.getType() == FreetaxStatsGraph.CHRONOLOGIE) {
            graph = graphChronologie(con, query);
        } else if (query.getType() == FreetaxStatsGraph.CHRONOLOGIE_2D) {
            graph = graphChronologie2D(con, query);
        } else {
```

```
            graph = null;
        }

        obj_out.writeObject(graph);
        obj_out.flush();
    }

    private static JFreeChart graph1D(Connection con, FreetaxStatsGraph query)
            throws SQLException
    {
        String instruc;

        if (query.getSemaine() != null) {
            // Recherche par semaine
            instruc =
                "SELECT jour_semaine AS jour, count " +
                "FROM stats_desc " +
                "WHERE categorie = '"+ query.getCategorie() +"' " +
                "  AND semaine = " + query.getSemaine();
        } else {
            // Recherche par mois
            if (query.getMois() == null) {
                // Sélectionne le mois précédent si aucun mois n'est renseigné
                query.setMois(moisPrec());
            }

            instruc =
                "SELECT jour_mois AS jour, count " +
                "FROM stats_desc " +
                "WHERE categorie = '" + query.getCategorie() + "'" +
                "  AND mois = "+ query.getMois();
        }

        if (query.getType() == FreetaxStatsGraph.SECTORIEL) {
            JDBCPieDataset ds = new JDBCPieDataset(con);
            ds.executeQuery(instruc);

            return ChartFactory.createPieChart(
                "Vente de produits par jour", ds, true, true, true
            );
        } else {
            JDBCCategoryDataset ds = new JDBCCategoryDataset(con);
            ds.executeQuery(instruc);

            return ChartFactory.createBarChart(
                "Vente de produits par jour", "Jour", "Ventes", ds,
                PlotOrientation.VERTICAL, true, true, true
            );
        }
    }

    private static JFreeChart graphHistogrammeCompare(Connection con,
            FreetaxStatsGraph query) throws SQLException
    {
        String instruc;

        if (query.getSemaine() != null) {
            // Recherche par semaine
            instruc =
                "SELECT marque, SUM(count) AS quantite " +
                "FROM stats_ventes_marque " +
                "WHERE categorie = '"+ query.getCategorie() +"' " +
                "  AND semaine = " + query.getSemaine() + " " +
```

```
                "GROUP BY marque";
        } else {
            // Recherche par mois
            if (query.getMois() == null) {
                // Sélectionne le mois précédent si aucun mois n'est renseigné
                query.setMois(moisPrec());
            }

            instruc =
                "SELECT marque, SUM(count) AS quantite " +
                "FROM stats_ventes_marque " +
                "WHERE categorie = '"+ query.getCategorie() +"' " +
                "  AND mois = " + query.getMois() + " " +
                "GROUP BY marque";
        }

        JDBCCategoryDataset ds = new JDBCCategoryDataset(con);
        ds.executeQuery(instruc);

        return ChartFactory.createBarChart(
            "Vente de produits par marque", "Marque", "Ventes", ds,
            PlotOrientation.VERTICAL, true, true, true
        );
    }

    private static JFreeChart graphChronologie(Connection con,
            FreetaxStatsGraph query) throws SQLException
    {
        String instruc;

        if (query.getSemaine() != null) {
            // Recherche par semaine
            instruc =
                "SELECT date, count AS quantite " +
                "FROM stats_desc " +
                "WHERE categorie = '"+ query.getCategorie() +"' " +
                "  AND semaine = " + query.getSemaine();
        } else {
            // Recherche par mois
            if (query.getMois() == null) {
                // Sélectionne le mois précédent si aucun mois n'est renseigné
                query.setMois(moisPrec());
            }

            instruc =
                "SELECT date, count AS quantite " +
                "FROM stats_desc " +
                "WHERE categorie = '" + query.getCategorie() + "'" +
                "  AND mois = "+ query.getMois();
        }

        JDBCXYDataset ds = new JDBCXYDataset(con);
        ds.executeQuery(instruc);

        return ChartFactory.createTimeSeriesChart(
            "Vente de produits par jour", "Jour", "Ventes", ds,
            true, true, true
        );
    }

    private static JFreeChart graphChronologie2D(Connection con,
            FreetaxStatsGraph query) throws SQLException
    {
```

```
        String instruc;

        if (query.getSemaine() != null) {
            // Recherche par semaine
            instruc =
                "SELECT age, SUM(count) AS quantite " +
                "FROM stats_ventes_age " +
                "WHERE categorie = '"+ query.getCategorie() +"' " +
                "  AND semaine = " + query.getSemaine() + " " +
                "GROUP BY age";
        } else {
            // Recherche par mois
            if (query.getMois() == null) {
                // Sélectionne le mois précédent si aucun mois n'est renseigné
                query.setMois(moisPrec());
            }

            instruc =
                "SELECT age, SUM(count) AS quantite " +
                "FROM stats_ventes_age " +
                "WHERE categorie = '" + query.getCategorie() + "'" +
                "  AND mois = "+ query.getMois() + " " +
                "GROUP BY age";
        }

        JDBCXYDataset ds = new JDBCXYDataset(con);
        ds.executeQuery(instruc);

        return ChartFactory.createXYLineChart(
            "Vente de produits par age", "Age", "Ventes", ds,
            PlotOrientation.VERTICAL, true, true, true
        );
    }

    private static void sendTestComp(Connection con, ObjectOutputStream obj_out,
            ObjectInputStream obj_in, FreetaxStatsTestComp query)
            throws SQLException, IOException
    {
        PreparedStatement instruc;

        instruc = con.prepareStatement(
            "SELECT AVG(quantite_semaine) AS moyenne " +
            "FROM stats_ventes_nationalite " +
            "WHERE categorie = ? AND nationalite IN (?, ?)" +
            "  AND marque = ?" +
            "GROUP BY nationalite"
        );

        instruc.setString(1, query.getCategorie());
        instruc.setString(2, query.getNationalite1());
        instruc.setString(3, query.getNationalite2());
        instruc.setString(4, query.getMarque());

        ResultSet rs = instruc.executeQuery();

        rs.next();
        float moyenne1 = rs.getFloat("moyenne");
        rs.next();
        float moyenne2 = rs.getFloat("moyenne");

        if (Math.abs(moyenne1 - moyenne2) <= query.getSeuil())
            obj_out.writeObject(new FreetaxStatsAck());
        else
```

```
            obj_out.writeObject(new FreetaxStatsFail());

        obj_out.flush();
    }
}
```